

### Assignment 3

**Name :** Omkar Deshpande

**Roll No :** 43212

**Batch -** Q10

**Problem statement** -To develop Microservices framework based distributed application.

#### CODE -

Publisher.java

```
package jmspublisher;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.JMSEException;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.jms.Topic;
import java.util.*;
import org.apache.activemq.ActiveMQConnection;
import org.apache.activemq.ActiveMQConnectionFactory;

public class Publisher
{

    private String clientId;
    private Connection connection;
    private Session session;
    private MessageProducer messageProducer;

    public void create(String clientId, String topicName) throws
JMSEException {
        this.setClientId(clientId);

        // create a Connection Factory
        ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory(ActiveMQConnection.DEFAULT_BROKER_URL);
```

```

        // create a Connection
        connection = connectionFactory.createConnection();
        connection.setClientID(clientId);

        // create a Session
        session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

        // create the Topic to which messages will be sent
        Topic topic = session.createTopic(topicName);

        // create a MessageProducer for sending messages
        messageProducer = session.createProducer(topic);
    }

    public void closeConnection() throws JMSEException
    {
        connection.close();
    }

    public void sendName(String firstName, String lastName) throws
JMSEException
    {
        String text = firstName + " " + lastName;

        // create a JMS TextMessage
        TextMessage textMessage = session.createTextMessage(text);

        // send the message to the topic destination
        messageProducer.send(textMessage);
    }

    public static void main(String[] args) throws JMSEException
    {
        Publisher publisher=new Publisher();
        publisher.create("1", "topic1");
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter a string message : ");
        String str= sc.nextLine();
    }

```

```

        publisher.sendName(str, "");
        sc.close();
        publisher.closeConnection();
    }

    public String getClientId() {
        return clientId;
    }

    public void setClientId(String clientId) {
        this.clientId = clientId;
    }
}

```

#### Subscriber.java

```

package jmspublisher;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.JMSEException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.jms.Topic;

import org.apache.activemq.ActiveMQConnection;
import org.apache.activemq.ActiveMQConnectionFactory;

public class Subscriber {

    private static final String NO_GREETING = "no greeting";
}

```

```

private String clientId;
private Connection connection;
private MessageConsumer messageConsumer;

public void create(String clientId, String topicName) throws
JMSEException {
    this.clientId = clientId;

    // create a Connection Factory
    ConnectionFactory connectionFactory = new
ActiveMQConnectionFactory(ActiveMQConnection.DEFAULT_BROKER_URL);

    // create a Connection
    connection = connectionFactory.createConnection();
    connection.setClientID(clientId);

    // create a Session
    Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

    // create the Topic from which messages will be received
    Topic topic = session.createTopic(topicName);

    // create a MessageConsumer for receiving messages
    messageConsumer = session.createConsumer(topic);

    // start the connection in order to receive messages
    connection.start();
}

public void closeConnection() throws JMSEException {
    connection.close();
}

public String getGreeting(int timeout) throws JMSEException {

    String greeting = NO_GREETING;

```

```

        // read a message from the topic destination
        Message message = messageConsumer.receive(timeout);

        // check if a message was received
        if (message != null) {
            // cast the message to the correct type
            TextMessage textMessage = (TextMessage) message;

            // retrieve the message content
            String text = textMessage.getText();
            System.out.println(clientId + ": received message with
text='{}' : "+ text);

            // create greeting
            greeting = "Hello " + text + "!";
        } else {
            System.out.println(clientId + ": no message received");
        }

        System.out.println("greeting={} : "+ greeting);
        return greeting;
    }

    public static void main(String[] args) throws JMSEException {
        Subscriber subscriber1=new Subscriber();
        Subscriber subscriber2=new Subscriber();

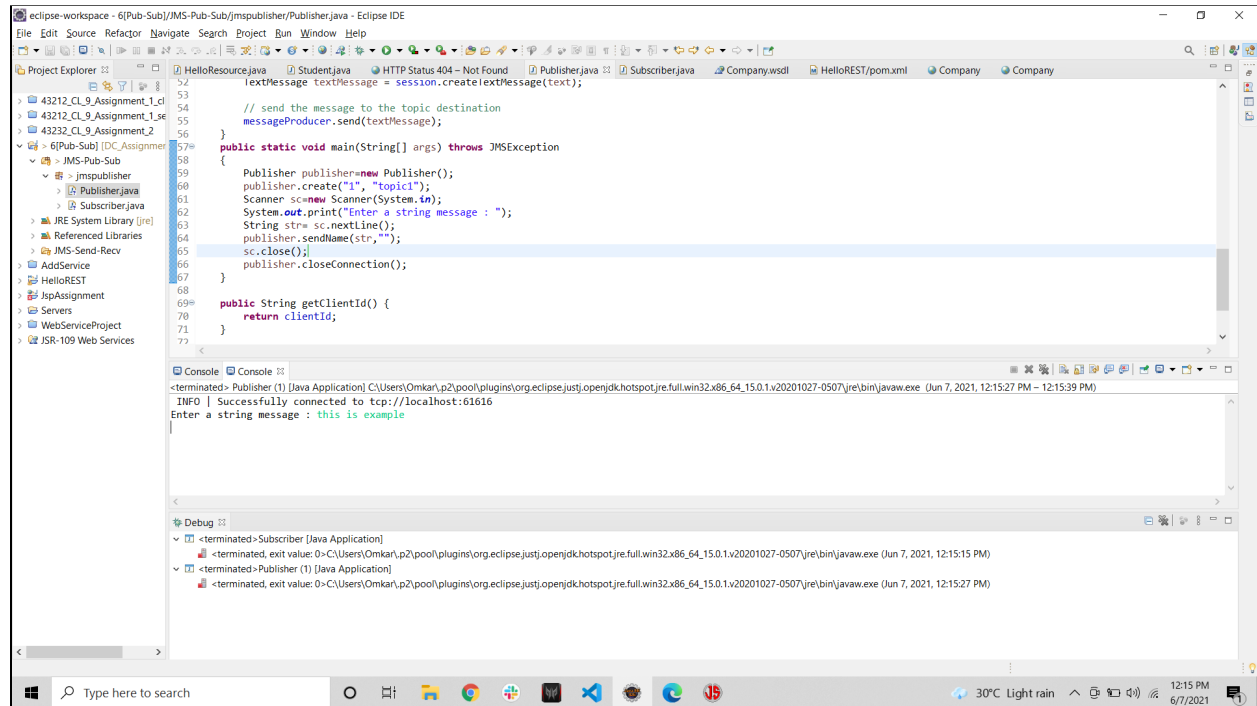
        subscriber1.create("2", "topic1");    //New client name should be
used
        subscriber2.create("3", "topic1");    //and same topic name

        subscriber1.getGreeting(30000);
        subscriber2.getGreeting(30000);

        subscriber1.closeConnection();
        subscriber2.closeConnection();
    }
}

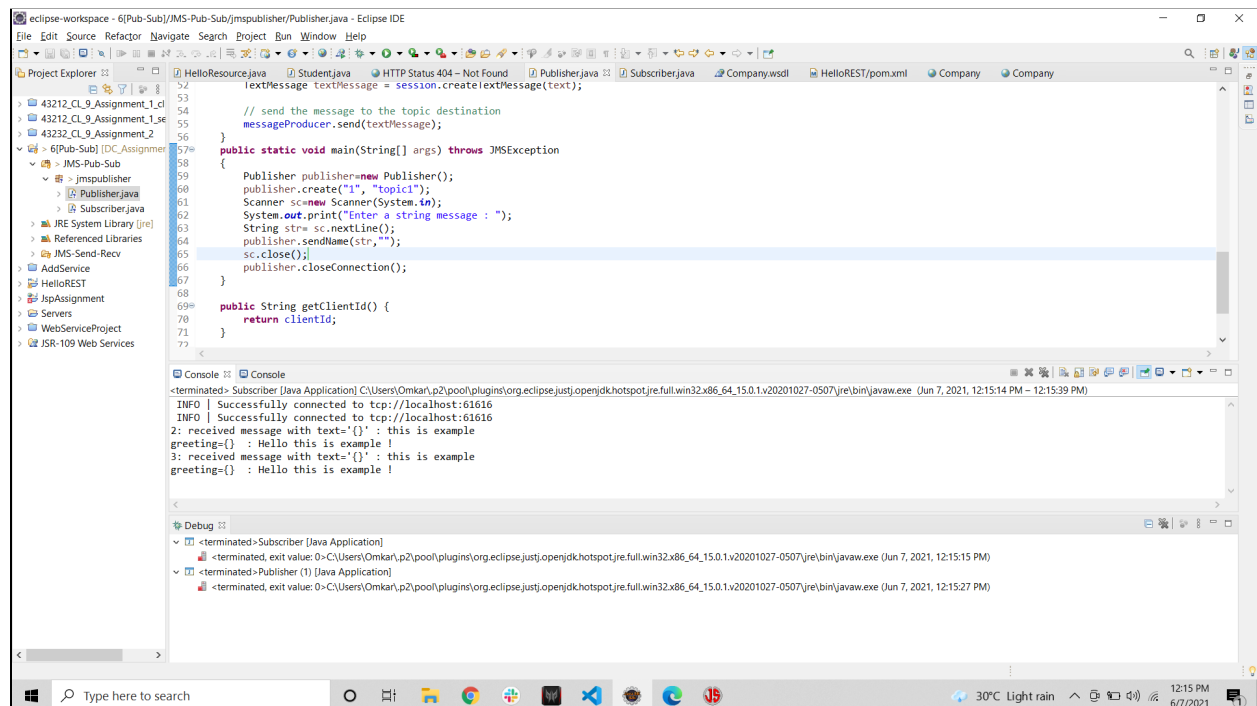
```

## OUTPUT :



The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows a project named 'JMS-Pub-Sub' with sub-projects 'JMS-Publisher' and 'JMS-Subscriber'. The 'Publisher.java' file is selected.
- Editor:** Displays the code for 'Publisher.java'. The code includes a 'main' method that creates a 'Publisher' object, sets a topic, and sends a message. It also has a 'getClientId' method.
- Console:** Shows the output of the application. It indicates that the application successfully connected to the JMS server at 'tcp://localhost:61616' and received a message: 'this is example'.
- Debug Console:** Shows the termination of the application with the message: '<terminated> Publisher (1) [Java Application]'. It also shows the exit value of the application.



The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows the same project structure as the first screenshot.
- Editor:** Displays the same code for 'Publisher.java'.
- Console:** Shows the output of the application. It indicates that the application successfully connected to the JMS server at 'tcp://localhost:61616' and received a message: 'Hello this is example'.
- Debug Console:** Shows the termination of the application with the message: '<terminated> Publisher (1) [Java Application]'. It also shows the exit value of the application.