

IE321326 / IF421326

Pengembangan Aplikasi Sistem Terdistribusi

Minggu	12
Sesi	1
Topik	RabbitMQ

Sebagai awal, anda akan menciptakan sebuah HelloWorld menggunakan RabbitMq. Model yang akan anda gunakan seperti gambar berikut ini.



Anda dapat mengakses dokumen yang sama pada official RabbitMQ untuk aplikasi Hello World ini

(<https://www.rabbitmq.com/tutorials/tutorial-one-java.html>).

1. Download RabbitMq Server pada website official RabbitMq pada link <https://www.rabbitmq.com/download.html>
2. Anda akan membutuhkan RabbitMq client. Karena anda akan menciptakan aplikasi consumer dan produser pada Java, anda download Java Client untuk RabiitMq pada link berikut <https://www.rabbitmq.com/java-client.html> . Versi yang anda gunakan adalah **rabbitmq-java-client-bin-3.6.1.zip** . Ekstrak file tersebut.
3. RabbitMq diciptakan menggunakan bahasa Erlang. Karena itu dibutuhkan compiler Erlang. Anda dapat download pada official website pada URL <https://www.erlang-solutions.com/resources/download.html>
Pada saat tutorial ini ditulis, versi Erlang yang ada alah versi 18.3
4. Instal compiler Erlang.
5. Setelah anda install compiler Erlang, install RabbitMq Server.
6. Ciptakan sebuah project Java Application, beri nama HelloProducer. Pada folder Libraries, tambahkan library RabbitMq yaitu **rabbit mq-client.jar**

7. Ciptakan sebuah kelas HelloProducer dan masukkan pada package org.rabbit.hello.

Tuliskan snippet berikut ini.

```
package org.rabbit.hello;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import java.io.IOException;
import java.util.concurrent.TimeoutException;

public class HelloProducer
{
    private final static String QUEUE_NAME = "hello";

    public static void main(String[] args) throws IOException,
    TimeoutException
    {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();

        channel.queueDeclare(QUEUE_NAME, false, false, false, null);
        String message = "Hello World!";
        channel.basicPublish("", QUEUE_NAME, null, message.getBytes("UTF-8"));
        System.out.println(" [x] Kirim Pesan '" + message + "'");

        channel.close();
        connection.close();
    }
}
```

8. Kemudian ciptakan sebuah project Java Application baru dan beri nama HelloConsumer.
9. Tambahkan library client rabbitmq yaitu rabbitmq-client.jar
10. Pada project HelloConsumer, ciptakan sebuah file java, beri nama HelloConsumer dan masukkan pada package org.rabbit.helloclient. Kemudian tuliskan snippet berikut ini.

```
package org.rabbit.helloclient;

import com.rabbitmq.client.*;
import java.io.IOException;
```

```

public class HelloConsumer
{
    private final static String QUEUE_NAME = "hello";

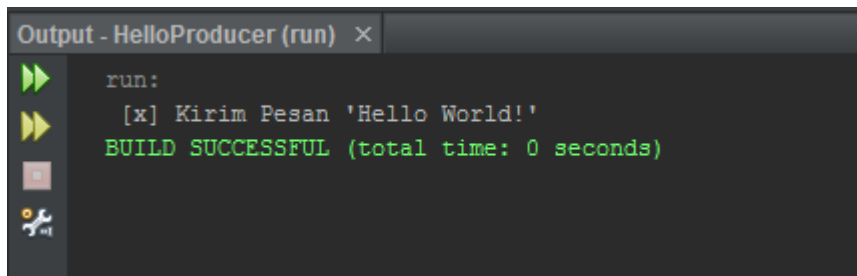
    public static void main(String[] argv) throws Exception
    {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();

        channel.queueDeclare(QUEUE_NAME, false, false, false, null);
        System.out.println(" [*] Menunggu Pesan dari Producer.....");

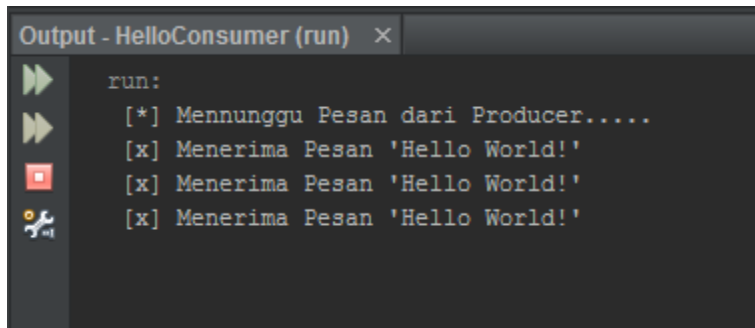
        Consumer consumer = new DefaultConsumer(channel) {
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
                AMQP.BasicProperties properties, byte[] body)
                throws IOException
            {
                String message = new String(body, "UTF-8");
                System.out.println(" [x] Menerima Pesan '" + message + "'");
            }
        };
        channel.basicConsume(QUEUE_NAME, true, consumer);
    }
}

```

11. Setelah itu, run tiga kali aplikasi HelloProducer seperti gambar berikut.



12. Setelah itu, coba Anda menjalankan HelloConsumer. Maka akan tampil seperti gambar berikut.

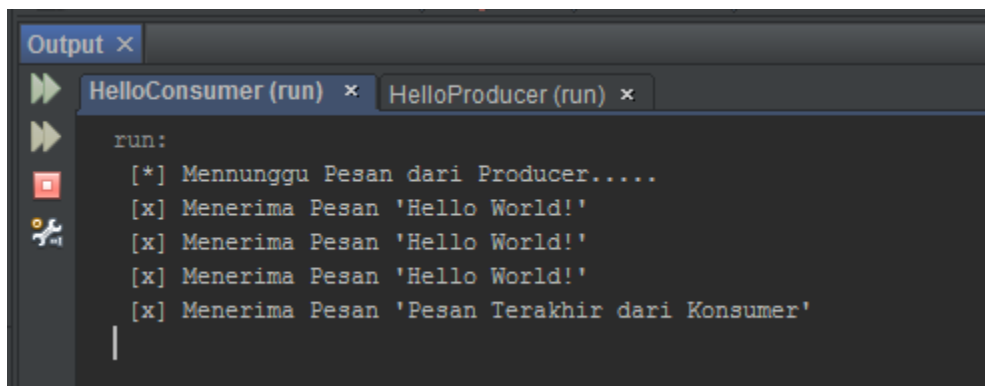


```
Output - HelloConsumer (run) x
run:
[*] Menunggu Pesan dari Producer.....
[x] Menerima Pesan 'Hello World!'
[x] Menerima Pesan 'Hello World!'
[x] Menerima Pesan 'Hello World!'
```

13. Apabila anda rubah message pada HelloProducer seperti pada snippet berikut

```
String message = "Pesan Terakhir dari Konsumer";
```

Kemudian menjalankan HelloProducer kembali, maka secara otomatis, aplikasi HelloConsumer juga akan meng-konsumsi pesan tersebut seperti gambar berikut ini.



```
Output x
HelloConsumer (run) x HelloProducer (run) x
run:
[*] Menunggu Pesan dari Producer.....
[x] Menerima Pesan 'Hello World!'
[x] Menerima Pesan 'Hello World!'
[x] Menerima Pesan 'Hello World!'
[x] Menerima Pesan 'Pesan Terakhir dari Konsumer'
```

Hal ini, dikarenakan HelloProducer akan selalu menunggu datangnya pesan dari Producer.

Anda dapat mencoba sampel yang lain pada website official RabbitMq pada link <https://www.rabbitmq.com/getstarted.html> . Semisal anda mencoba menggunakan Bahasa pemrograman lainnya.

Selanjutnya anda akan mencoba tipe message fanout. Terdapat sebuah producer yang akan mem-publish pesan, dan beberapa client yang akan men-subscribe.

1. Ciptakan sebuah Java Application, beri nama AkademikConsumer.
2. Tambahkan library commons-lang3-3.4.jar dan rabbitmq-client.jar pada project.
3. Buat sebuah file java, beri nama Mahasiswa dan masukkan pada package org.akademik.model. Tuliskan snippet berikut.

```

package org.akademik.model;

import java.io.Serializable;

public class Mahasiswa implements Serializable
{
    private String nim;
    private String nama;

    public Mahasiswa()
    {
    }

    public Mahasiswa(String nim, String nama)
    {
        this.nim = nim;
        this.nama = nama;
    }

    public String getNim()
    {
        return nim;
    }

    public void setNim(String nim)
    {
        this.nim = nim;
    }

    public String getNama()
    {
        return nama;
    }

    public void setNama(String nama)
    {
        this.nama = nama;
    }

    @Override
    public String toString()
    {
        return "NIM : " + nim + "\tNAMA : " + nama;
    }
}

```

4. Kemudian ciptakan sebuah file bernama EndPoint dan masukkan pada package org.akademik.endpoint. Tuliskan snippet berikut ini.

```

package org.akademik.endpoint;

```

```

import java.io.IOException;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import java.util.concurrent.TimeoutException;

public abstract class EndPoint
{
    protected Channel channel;
    protected Connection connection;
    protected String exchangeName;
    protected String queueName;

    public EndPoint(String exchangeName) throws IOException, TimeoutException
    {
        this.exchangeName = exchangeName;
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost("localhost");
        connection = factory.newConnection();
        channel = connection.createChannel();
        channel.exchangeDeclare(exchangeName, "fanout");
        queueName = channel.queueDeclare().getQueue();
        channel.queueBind(queueName, exchangeName, "");
    }

    public void close() throws IOException, TimeoutException{
        this.channel.close();
        this.connection.close();
    }
}

```

5. Selanjutnya ciptakan sebuah kelas bernama QueueConsumer, dan masukkan pada package org.akademik.consumer. Tuliskan snippet berikut ini.

```

package org.akademik.consumer;

import java.io.IOException;
import com.rabbitmq.client.AMQP.BasicProperties;
import com.rabbitmq.client.Consumer;
import com.rabbitmq.client.Envelope;
import com.rabbitmq.client.ShutdownSignalException;
import java.util.concurrent.TimeoutException;
import org.akademik.endpoint.EndPoint;
import org.akademik.model.Mahasiswa;
import org.apache.commons.lang3.SerializationUtils;

public class QueueConsumer extends EndPoint implements Runnable, Consumer
{

```

```

    public QueueConsumer(String exchangeName) throws IOException,
    TimeoutException
    {
        super(exchangeName);
    }
    public void run()
    {
        try
        {
            channel.basicConsume(queueName, true, this);
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    @Override
    public void handleConsumeOk(String consumerTag)
    {
        System.out.println("Consumer " + consumerTag + " registered");
    }

    @Override
    public void handleDelivery(String consumerTag, Envelope env, BasicProperties
    props, byte[] body) throws IOException
    {
        Mahasiswa m = (Mahasiswa)SerializationUtils.deserialize(body);
        System.out.println(m);
    }
    @Override
    public void handleCancel(String consumerTag) { }

    @Override
    public void handleRecoverOk(String consumerTag) { }

    @Override
    public void handleShutdownSignal(String string, ShutdownSignalException sse)
    { }

    @Override
    public void handleCancelOk(String string) { }
}

```

6. Dan yang terakhir adalah Driver. Ciptakan sebuah kelas Driver dan masukkan ke package org.akademik.driver. Tuliskan snippet berikut.

```

package org.akademik.driver;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

```

```
import org.akademik.consumer.QueueConsumer;

public class Driver
{
    public final static String exchangeName = "akademik";
    public static void main(String[] args) throws IOException, TimeoutException
    {

        QueueConsumer consumer = new QueueConsumer(exchangeName);
        Thread consumerThread = new Thread(consumer);
        consumerThread.start();
    }
}
```

Kemudian anda akan membuat aplikasi pada sisi produsennya.

1. Ciptakan sebuah Java Application, beri nama AkademikConsumer.
2. Tambahkan library commons-lang3-3.4.jar dan rabbitmq-client.jar pada project.
3. Kopikan package org.akademik.mahasiswa dan org.akademik.endpoint yang ada pada prohect sebelumnya beserta dengan isi file javanya.
4. Kemudian, ciptakan sebuah kelas Java, beri nama Server dan masukkan pada package org.akademik.server. Kemudian tuliskan snippet berikut ini.

```
package org.akademik.server;

import java.io.IOException;
import java.io.Serializable;
import java.util.concurrent.TimeoutException;
import org.akademik.endpoint.EndPoint;
import org.apache.commons.lang3.SerializationUtils;

public class MyServer extends EndPoint
{
    public MyServer(String exchangeName) throws IOException, TimeoutException
    {
        super(exchangeName);
    }

    public void sendMessage(Serializable object) throws IOException
    {
        channel.basicPublish(exchangeName, "", null,
        SerializationUtils.serialize(object));
    }
}
```



```
}
```

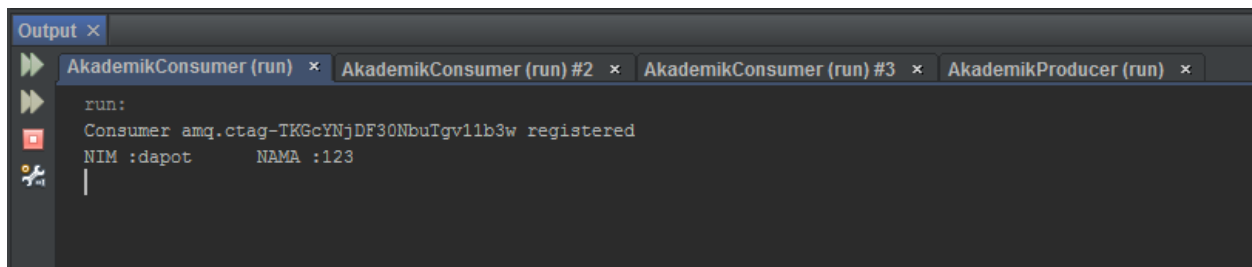
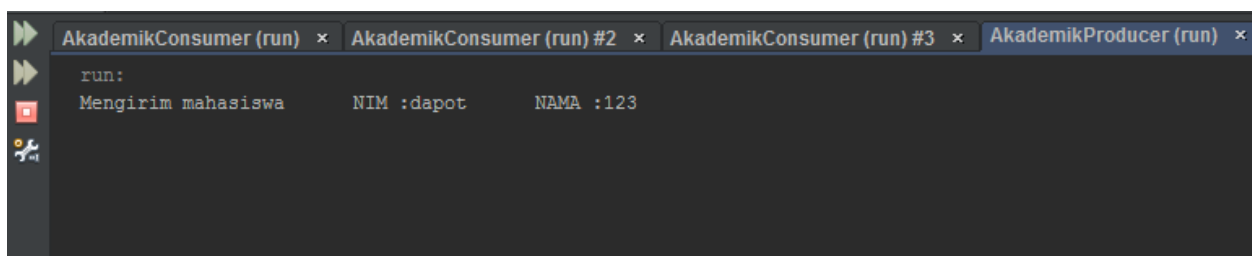
5. Kemudian ciptakan Driver dan tuliskan snippet berikut.

```
package org.akademik.driver;

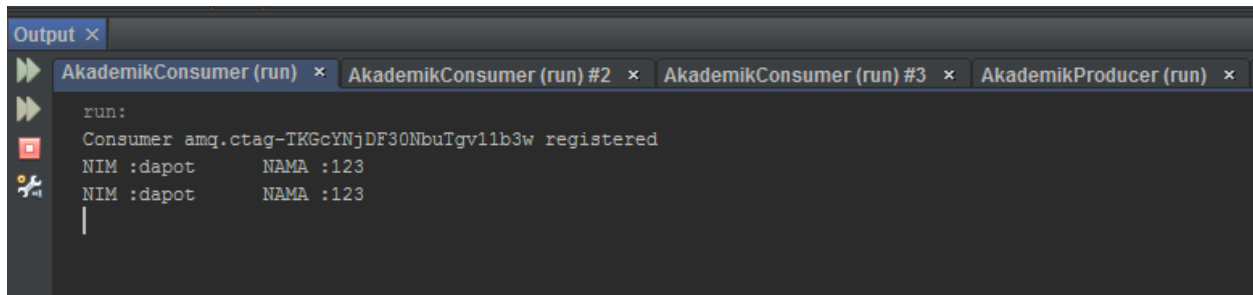
import java.io.IOException;
import java.util.concurrent.TimeoutException;
import org.akademik.server.MyServer;
import org.akademik.model.Mahasiswa;

public class Driver
{
    public final static String exchangeName = "akademik";
    public static void main(String[] args) throws IOException, TimeoutException
    {
        MyServer myServer = new MyServer(exchangeName);
        Mahasiswa m = new Mahasiswa("dapot", "123");
        myServer.sendMessage(m);
        System.out.println("Mengirim mahasiswa \t"+m);
    }
}
```

Setelah semua kelas tercipta, jalankan aplikasi AkademikConsumer sebanyak 3 buah. Setelah itu, jalankan sebuah aplikasi AkademikProdusen. Maka aplikasi akan berjalan seperti gambar dibawah ini.



Ketika Anda menjalankan sekali lagi produsen, maka semua consumer akan mendapatkan data yang sama, seperti gambar berikut ini.



```
Output ×
AkademikConsumer (run) × AkademikConsumer (run) #2 × AkademikConsumer (run) #3 × AkademikProducer (run) ×
run:
Consumer amq.ctag-TKGcYNjDF30NbuTgv11b3w registered
NIM :dapot NAMA :123
NIM :dapot NAMA :123
|
```

Tugas

Ciptakan sebuah Aplikasi Sederhana Perpustakaan. Ketika admin perpustakaan mengupdate data buku baru, semua anggota akan dikirimkan semua list data buku yang terbaru. Dan ketika seorang anggota meminjam buku ke admin, maka semua anggota akan diberitahukan bahwa buku telah dipinjam seorang anggota.

Selamat mengerjakan ! 😊😊