



Message Oriented Programming

Assignment 3

COEN 6731: Distributed Software Systems

Rishi Murugesan Gopalakrishnan - 40200594
Pritam Sethuraman - 40230509

Department of Electrical and Computer Engineering

Gina Cody School of Engineering and Computer Science

Winter 2023

Introduction

The main objective of this assignment is to use Pub/Sub message oriented programming. We are going to use the collections that we created in assignment 2 for this purpose. In assignment 2 we created Data Access Objects to query data from a collection called EduCostStat. These are the following collections that we created from assignment 2.

- EduCostStatQueryOne
- EduCostStatQueryTwo
- EduCostStatQueryThree
- EduCostStatQueryFour
- EduCostStatQueryFive

We have to use message oriented programming to pass these collections as messages. To achieve this we can use RabbitMq.

RabbitMQ

RabbitMQ is an open-source message broker software that incorporates the Advanced Message Queueing Protocol (AMQP). It is used to provide a communication interface between various software nodes and components in a distributed architecture. With the help of RabbitMQ systems can exchange messages, communicate asynchronously and reliably.

- **Producer**

In RabbitMQ, a producer is an application that dispatches messages to the server for distribution. Producers can be anything from a simple script to a complex application. Upon sending messages, the producer also includes a **routing key** along with the messages. This routing key is used to identify the message target queue. The producer is unaware of the specific queue to which the message will eventually be directed. It just sends the messages.

- **Consumer**

A consumer is a type of program or application that retrieves messages from RabbitMQ. The consumer can subscribe for one or more queues and awaits the delivery of messages. Once the message is received by the consumer, the consumer processes the message and sends a response back to the RabbitMQ server to acknowledge that the message has been dealt with correctly.

- **Exchange**

In RabbitMQ, exchanges are responsible for collecting messages from the producer and directing them to one or more queues depending upon the rules defined by the type of exchange. RabbitMQ offers 4 types of exchange.

- **Direct**: Routes messages to queues based on exact matches between routing key and queue binding keys.
- **Topic** : Routes messages to multiple queues on a pattern match between the message routing key and the queue binding key.
- **Headers** : Routes messages to the queue based on matching the message header values to the bindings defined in the queues.
- **Fanout** : Routes messages to all queues that are bound to it, regardless of any routing keys or patterns.

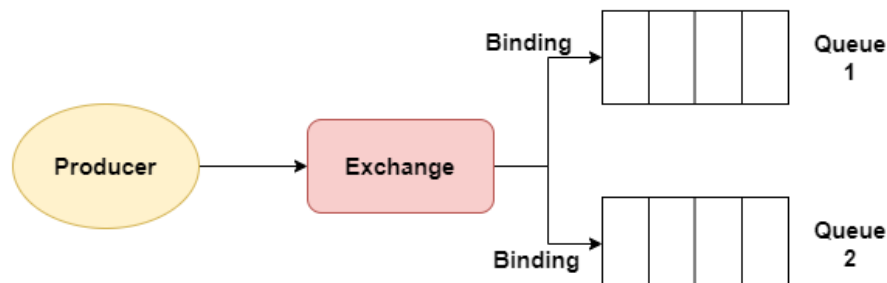
The manner in which the messages are dispatched to queues depends on the type of exchange used.

- **Queues**

In RabbitMQ, messages from producers are temporarily stored in queues until they are retrieved by the consumers. Messages from the producer goes to the Exchange and it will route it to the queue based on **bindings**. Binding establishes a relationship between a queue and an exchange, dictating the routing protocol for messages.

- **Topic Exchange**

Topic exchanges in RabbitMQ distribute messages to multiple queues based on a specific set of criteria, utilizing the key pattern. A topic routing key typically contains several words separated by dots. **Example: stock.us.tech** could be used to route messages about technology stocks in the United States to a specific queue.



Task 1 : Installing RabbitMQ Server

RabbitMQ server manages the routing and delivery of messages between producers and consumers. RabbitMQ server is open-source and can be installed on various operating systems.

We installed the RabbitMQ server in our Oracle Cloud instance (155.248.234.61) and the rabbitmq server is running on port 15672.

We performed the following steps to install rabbitMQ server:

- We installed rabbitmq as a docker container. So firstly we installed docker in Oracle VM using the following command.

```
sudo yum install -y docker
```

- Start the docker service

```
sudo systemctl start docker
```

- Download RabbitMQ docker image

```
sudo docker pull rabbitmq:3.9.9-management
```

- Creating a RabbitMQ container

```
sudo docker run -d --name myrabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.9.9-management
```

This command sets the port numbers for the instance

- Now the Docker container is live. We can check the if the container is running using the following command

```
sudo docker ps
```

```
opc@instance-20230221-1508:~  
[opc@instance-20230221-1508 ~]$ sudo docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED  
STATUS        PORTS  
NAMES  
89239c294907   rabbitmq:3.9.9-management          "docker-entrypoint.s..." 5 days ago  
Up 5 days      4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/tcp  
myrabbitmq  
[opc@instance-20230221-1508 ~]$
```

We can see the containerID, and the image is up and running the Oracle VM instance.

- Now we can check the status of our rabbitmq server using the following command

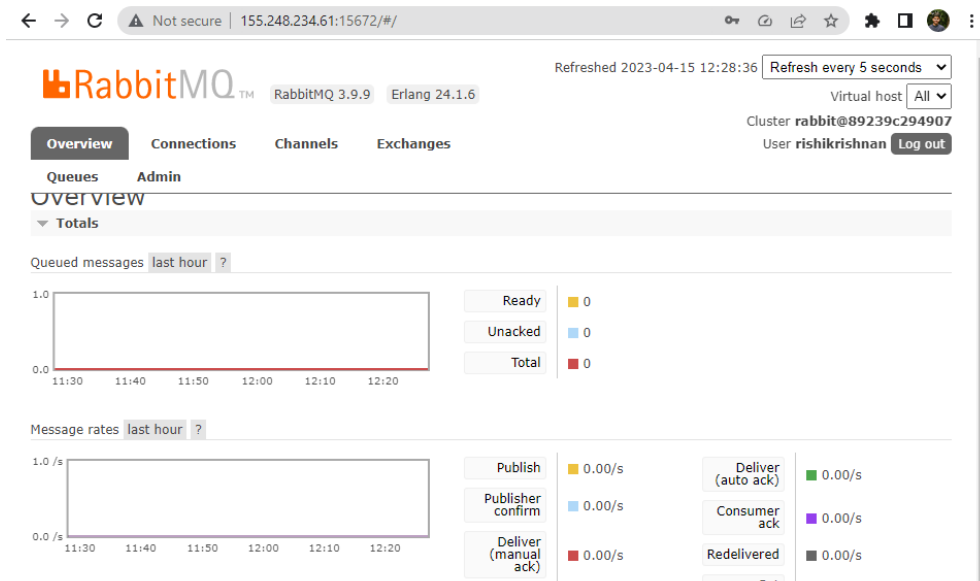
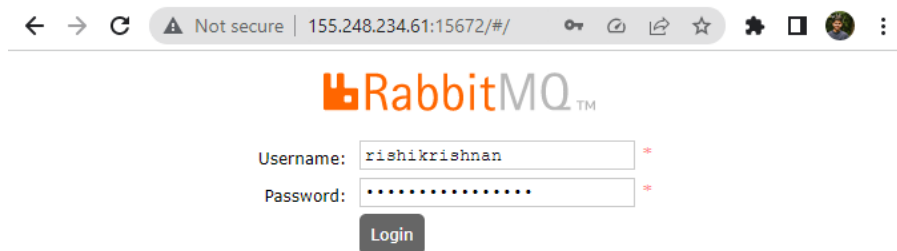
```
sudo docker exec 89239c294907 rabbitmqctl status
```

```
opc@instance-20230221-1508:~  
[opc@instance-20230221-1508 ~]$ sudo docker exec 89239c294907 rabbitmqctl status  
Status of node rabbit@89239c294907 ...  
Runtime  
  
OS PID: 20  
OS: Linux  
Uptime (seconds): 471365  
Is under maintenance?: false  
RabbitMQ version: 3.9.9  
Node name: rabbit@89239c294907  
Erlang configuration: Erlang/OTP 24 [erts-12.1.5] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1]  
Erlang processes: 394 used, 1048576 limit  
Scheduler run queue: 1  
Cluster heartbeat timeout (net_ticktime): 60  
  
Plugins  
  
Enabled plugin file: /etc/rabbitmq/enabled_plugins  
Enabled plugins:  
  
* rabbitmq_prometheus  
* prometheus  
* accept
```

- Now we can access the server dashboard using the IP address **155.248.234.61:15672**
- But to access it we have to create a user to access the RabbitMQ dashboard. To create a user, we used the following commands to add user and set permissions

```
rabbitmqctl add_user <username> <password>
rabbitmqctl set_permissions -p / <username> ".*" ".*" ".*"
```

- Now we can access the RabbitMQ Server



Task 2 : Programming Producer and Consumer

There are several libraries available in different programming languages which can be used to interact with RabbitMQ and its topic exchange feature. For this assignment, we are using RabbitMQ client libraries available in Java.

<https://www.rabbitmq.com/java-client.html>

RabbitMQ libraries provide a flexible interface to,

- Interact with RabbitMQ server
- Create Publishers and Consumers
- Create Exchanges
- Consume Messages from exchanges based on routing keys

We created java maven projects, one for producers and one for consumers and added maven dependency for rabbitmq in pom.xml to include RabbitMQ libraries.

```
<dependency>
<groupId>com.rabbitmq</groupId>
<artifactId>amqp-client</artifactId>
<version>5.15.0</version>
</dependency>
```

- Created send.java to act as the producer
- Created Recv.java to code consumers (we can create multiple consumers)

Producer (send.java)

We programmed the producer with the following steps.

- In send.java first we established a connection to our rabbitmq server hosted in the oracle machine using channels.

```
// ***** SETTING CONNECTION FOR RABBITMQ *****
ConnectionFactory factory = new ConnectionFactory();

// setting up IP address for the rabbitmq server.. It is hosted in oracle cloud instance
factory.setHost("155.248.234.61");
try(Connection connection = factory.newConnection();
    Channel channel = connection.createChannel()){
```

- Created an **exchange** which will hold the messages and routes it to the queues.
 - We declared an exchange name called **topic-mongodb**

```
private final static String EXCHANGE_NAME = "topic-mongodb";
```

```
//creating exchange interface to handle topics, here the data transferring method is "topic"
channel.exchangeDeclare(EXCHANGE_NAME, "topic");
```

- Here the **exchangeDeclare ()** method is used to declare the exchange and “topic” keyword is used to let rabbitmq server know that the exchange is following the topic exchange mechanism.
- After which we created a MongoDB client connection to extract the collection data from the MongoDB cloud.

```
// *** CREATING MONGODB CLIENT ***
// establishing connection to my mongodb instance
ConnectionString connectionString = new ConnectionString("mongodb+srv://rishik:
MongoClientSettings settings = MongoClientSettings.builder()
    .applyConnectionString(connectionString)
    .build();
MongoClient mongoClient = MongoClient.create(settings);

// Access the MongoDB database
MongoDatabase database = mongoClient.getDatabase("MyDatabase");
Thread.sleep(1000);
```

- Now we started creating topics for each query collection in the mongoDB

```
// FOR TOPIC ONE *****
//Access MongoDB collection
MongoCollection<Document> collection_one = database.getCollection("EduCostStatQueryOne");

//Retrieve documents from the MongoDB collection
Document query_one = new Document("State", "Alabama");
MongoCursor<Document> cursor_one = collection_one.find(query_one).iterator();

while(cursor_one.hasNext())
{
    Document doc = cursor_one.next();
    String routingKey = createRoutingKey_one(doc);
    String message = getMessage_one(doc);

    //Publish the message to exchange
    channel.basicPublish(EXCHANGE_NAME, routingKey, null, message.getBytes("UTF-8"));
    System.out.println("[x] Sent " + routingKey + ":" + message);
}

// *****
```

- We accessed the collection and queried the data and used to data to create a topic routing key as per the requirement.

- The **createRoutingKey_one ()** creates the topic routing key and sends it through the **basicPublish ()** method which will publish the data to the exchange.

```
private static String createRoutingKey_one(Document doc)
{
    //Extracting the required fields from the document
    int Year = doc.getInteger("Year");
    String State = doc.getString("State");
    String Type = doc.getString("Type");
    String Length = doc.getString("Length");

    //create routing key using the required fields
    String routingKey = "Cost-"+Year+"-"+State+"-"+Type+"-"+Length;

    return routingKey;
}
```

- The **getMessage ()** method is used to display the result value for the particular query

```
private static String getMessage_one(Document doc)
{
    //Extracting the required result value from the document
    int Value = doc.getInteger("Value");
    String Expense = doc.getString("Expense");

    String message = "The Result for the Query Parameters are: Value: " +Integer.toString(Value) +"For the expense type : "+Expense;

    return message;
}
```

- Similarly we created topics for all the MongoDB collections and published it using the **basicPublish ()** method. The following table shows the collection name, topic name, method used to create routing key and method used to fetch the result of the query.

Collection Name	Topic Name	Routing Key	Message
EduCostStatQueryOne	Cost-[Year]-[State]-[Type]- [Length]	<i>createRoutingKey_one ()</i>	<i>getMessage_one ()</i>
EduCostStatQueryTwo	Top5-Expensive-[Year]-[Type]-[Length]	<i>createRoutingKey_two ()</i>	<i>getMessage_two ()</i>
EduCostStatQueryThree	Top5-Economic-[Year]-[Type]- [Length]	<i>createRoutingKey_three ()</i>	<i>getMessage_three ()</i>
EduCostStatQueryFour	Top5-HighestGrow-[Years]	<i>createRoutingKey_four ()</i>	<i>getMessage_four ()</i>

EduCostStatQueryFive	AverageExpense-[Year]-[Type]-[Length]	<i>createRoutingKey_five ()</i>	<i>getMessage_five ()</i>
----------------------	---------------------------------------	----------------------------------	----------------------------

Consumer (recv.java)

We programmed the consumer with the following steps.

- Same as the producer we created a channel to establish connection to the rabbitmq server.

```
// ***** SETTING CONNECTION FOR RABBITMQ *****
ConnectionFactory factory = new ConnectionFactory();

// setting up IP address for the rabbitmq server.. It is hosted in oracle cloud instance
factory.setHost("155.248.234.61");
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();
```

- We have to declare the exchange here as well

```
private final static String EXCHANGE_NAME = "topic_mongodb";

//creating exchange interface to handle topics, here the data transferring method is "topic"
channel.exchangeDeclare(EXCHANGE_NAME, "topic");
```

- Next we have to set the queues, we are using the topic passed to the consumer as the queue names

```
//creating queues.. queues will be assigned with random names by the rabbitmq
String queueName = "";
for(String qn:args)
{
    queueName = channel.queueDeclare(qn,true,false,false,null).getQueue();
}
```

The **queueDeclare ()** method is used to declare the queue.

- Now, we are going to use the received arguments to bind it as the routing key so that it can accept messages from the exchange module. This is done using **queueBind ()** method

```
if(args.length < 1)
{
    System.err.println("Usage: Arguments not passed properly.");
    System.exit(1);
}

for(String bindingKey:args)
{
    channel.queueBind(queueName, EXCHANGE_NAME, bindingKey);
}
```

- Finally we are consuming the messages that are routed to the consumer using the **basicConsume ()** method. To display the published messages we have to use **DeliverCallback** class

```
DeliverCallback deliverCallback = (consumerTag, delivery) -> {
    String message = new String(delivery.getBody(), "UTF-8");
    System.out.println(" [x] Received '" + delivery.getEnvelope().getRoutingKey()+"':"+message + "'");
};
channel.basicConsume(queueName, true, deliverCallback, consumerTag -> { });
```

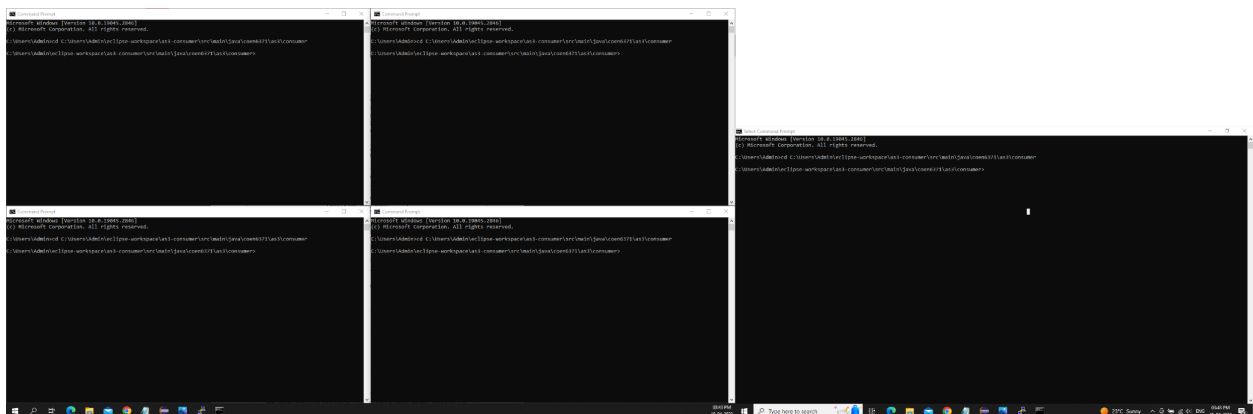
Program Execution

We are going to run the producer in Eclipse IDE and also run five consumers in five separate command line windows, to make sure that the producer and consumers are not multi-threaded in the same application.

We are going to pass the following topics as arguments to the consumers.

1. For EduCostStatQueryOne - Cost-2013-Alabama-Private-4-year
2. For EduCostStatQueryTwo - Top5-Expensive-2013-Private-4-year
3. For EduCostStatQueryThree - Top5-Economic-2013-Private-4-year
4. For EduCostStatQueryFour - Top5-HighestGrowth-3-years
5. For EduCostStatQueryFive - AverageExpense-2013-Private-4-year

Notice: In the below image i have opened 5 terminals in split screen to represent 5 consumers.

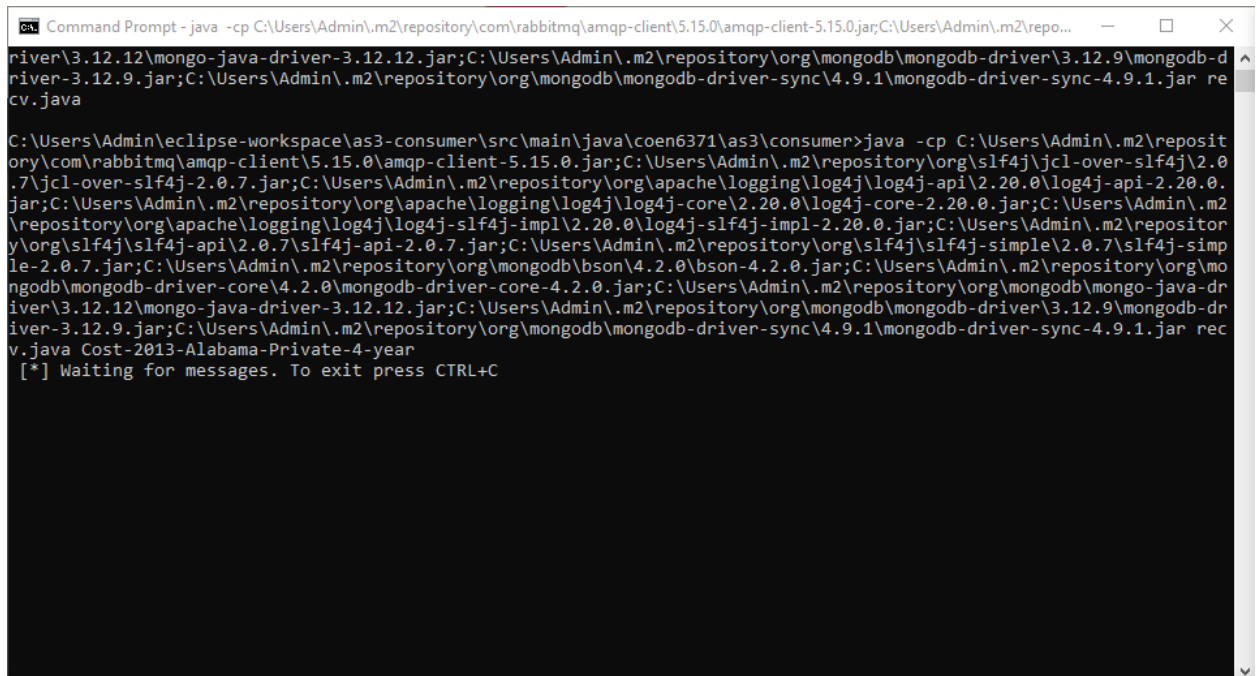


Steps to execute the producer and consumer

1. Compile consumer code using javac command
2. Run consumers using java and pass the topic as args parameter

```
java -cp <path to all jar files> recv.java Cost-2013-Alabama-Private-4-year
java -cp <path to all jar files> recv.java Top5-Expensive-2013-Private-4-year
java -cp <path to all jar files> recv.java Top5-Economic-2013-Private-4-year
java -cp <path to all jar files> recv.java Top5-HighestGrowth-3-years
java -cp <path to all jar files> recv.java AverageExpense-2013-Private-4-year
```

3. I ran all the consumers first with the above commands. The below screenshot is for the consumer one.



```
Command Prompt - java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repo...
river\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-d
river-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar re
cv.java

C:\Users\Admin\eclipse-workspace\as3-consumer\src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\.m2\reposit
ory\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\jcl-over-slf4j\2.0
.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0
.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\.m2
\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\.m2\repositor
y\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simp
le-2.0.7.jar;C:\Users\Admin\.m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mo
ngodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongo-java-dr
iver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-dr
iver-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar re
cv.java Cost-2013-Alabama-Private-4-year
[*] Waiting for messages. To exit press CTRL+C
```

We can see the message that it is waiting for messages to be passed by the exchange. Similarly the other 4 consumers are waiting for the message from the exchange module.

4. Now we ran the send.java producer code in eclipse IDE. We can see the message published output in the publisher console.

```
Problems Javadoc Declaration Console Terminal Git Staging History Error Log Synchronize Coverage
<terminated> send [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\javaw.exe (15-Apr-2023, 4:03:00 pm - 4:03:03 pm)
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
[x] Sent 'Cost-2013-Alabama-Private-4-year': 'The Result for the Query Parameters are: Value: 13983For the expense type : Fees/Tuition'
[x] Sent 'Top5-Expensive-2013-Private-4-year': 'State : Connecticut Overall Expense: 48262'
[x] Sent 'Top5-Expensive-2013-Private-4-year': 'State : District of Columbia Overall Expense: 48440'
[x] Sent 'Top5-Expensive-2013-Private-4-year': 'State : Massachusetts Overall Expense: 49871'
[x] Sent 'Top5-Expensive-2013-Private-4-year': 'State : Rhode Island Overall Expense: 46114'
[x] Sent 'Top5-Expensive-2013-Private-4-year': 'State : Vermont Overall Expense: 46255'
[x] Sent 'Top5-Economic-2013-Private-4-year': 'State : Idaho Overall Expense: 11544'
[x] Sent 'Top5-Economic-2013-Private-4-year': 'State : North Dakota Overall Expense: 17742'
[x] Sent 'Top5-Economic-2013-Private-4-year': 'State : Utah Overall Expense: 15330'
[x] Sent 'Top5-Economic-2013-Private-4-year': 'State : West Virginia Overall Expense: 19120'
[x] Sent 'Top5-Economic-2013-Private-4-year': 'State : Wyoming Overall Expense: 13562'
[x] Sent 'Top5-HighestGrowth-3-years': 'State : New Mexico Growth Rate over 3 years: 1.8921741925116504'
[x] Sent 'Top5-HighestGrowth-3-years': 'State : West Virginia Growth Rate over 3 years: 2.094578264722468'
[x] Sent 'Top5-HighestGrowth-3-years': 'State : Delaware Growth Rate over 3 years: 2.4618991793669402'
[x] Sent 'Top5-HighestGrowth-3-years': 'State : Colorado Growth Rate over 3 years: 2.8666551366309236'
[x] Sent 'Top5-HighestGrowth-3-years': 'State : New Hampshire Growth Rate over 3 years: 123.95875184143566'
[x] Sent 'AverageExpense-2013-Private-4-year': 'Region : South West Region value: 109629'
[x] Sent 'AverageExpense-2013-Private-4-year': 'Region : West Region value: 273494'
[x] Sent 'AverageExpense-2013-Private-4-year': 'Region : Mid West Region value: 353969'
[x] Sent 'AverageExpense-2013-Private-4-year': 'Region : South East Region value: 417915'
[x] Sent 'AverageExpense-2013-Private-4-year': 'Region : North East Region value: 452952'
```

5. After running the IDE. The Consumers received the messages at the same time.

```
Command Prompt - java -cp C:\Users\Admin\m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\m2\repo...
C:\Users\Admin\workspace\as3-consumer>src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\m2\repository\org\slf4j\jcl-over-slf4j\2.0.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0.jar;C:\Users\Admin\m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\m2\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\m2\repository\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simple-2.0.7.jar;C:\Users\Admin\m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\m2\repository\org\mongodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\m2\repository\org\mongodb\mongo-java-driver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-driver-3.12.9.jar;C:\Users\Admin\m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar rec
v.java Cost-2013-Alabama-Private-4-year
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Cost-2013-Alabama-Private-4-year': 'The Result for the Query Parameters are: Value: 13983For the expense type : Fees/Tuition'
```

CONSUMER 1

```
Command Prompt - java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repo...
C:\Users\Admin\workspace\as3-consumer\src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\jcl-over-slf4j\2.0.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simple-2.0.7.jar;C:\Users\Admin\.m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongo-java-driver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-driver-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar rec
v.java Top5-Expensive-2013-Private-4-year
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Top5-Expensive-2013-Private-4-year': 'State : Connecticut Overall Expense: 48262'
[x] Received 'Top5-Expensive-2013-Private-4-year': 'State : District of Columbia Overall Expense: 48440'
[x] Received 'Top5-Expensive-2013-Private-4-year': 'State : Massachusetts Overall Expense: 49871'
[x] Received 'Top5-Expensive-2013-Private-4-year': 'State : Rhode Island Overall Expense: 46114'
[x] Received 'Top5-Expensive-2013-Private-4-year': 'State : Vermont Overall Expense: 46255'
```

CONSUMER 2

```
Command Prompt - java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repo...
C:\Users\Admin\workspace\as3-consumer\src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\jcl-over-slf4j\2.0.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simple-2.0.7.jar;C:\Users\Admin\.m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongo-java-driver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-driver-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar rec
v.java Top5-Economic-2013-Private-4-year
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Top5-Economic-2013-Private-4-year': 'State : Idaho Overall Expense: 11544'
[x] Received 'Top5-Economic-2013-Private-4-year': 'State : North Dakota Overall Expense: 17742'
[x] Received 'Top5-Economic-2013-Private-4-year': 'State : Utah Overall Expense: 15330'
[x] Received 'Top5-Economic-2013-Private-4-year': 'State : West Virginia Overall Expense: 19120'
[x] Received 'Top5-Economic-2013-Private-4-year': 'State : Wyoming Overall Expense: 13562'
```

CONSUMER 3


```
Command Prompt - java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repo...
C:\Users\Admin\workspace\as3-consumer\src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\jcl-over-slf4j\2.0.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simple-2.0.7.jar;C:\Users\Admin\.m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongo-java-driver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-driver-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar recv.java Top5-HighestGrowth-3-years
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Top5-HighestGrowth-3-years': 'State : New Mexico Growth Rate over 3 years: 1.8921741925116504'
[x] Received 'Top5-HighestGrowth-3-years': 'State : West Virginia Growth Rate over 3 years: 2.094578264722468'
[x] Received 'Top5-HighestGrowth-3-years': 'State : Delaware Growth Rate over 3 years: 2.4618991793669402'
[x] Received 'Top5-HighestGrowth-3-years': 'State : Colorado Growth Rate over 3 years: 2.8666551366309236'
[x] Received 'Top5-HighestGrowth-3-years': 'State : New Hampshire Growth Rate over 3 years: 123.95875184143566'
```

CONSUMER 4

```
Command Prompt - java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository...
C:\Users\Admin\workspace\as3-consumer\src\main\java\coen6371\as3\consumer>java -cp C:\Users\Admin\.m2\repository\com\rabbitmq\amqp-client\5.15.0\amqp-client-5.15.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\jcl-over-slf4j\2.0.7\jcl-over-slf4j-2.0.7.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-api\2.20.0\log4j-api-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-core\2.20.0\log4j-core-2.20.0.jar;C:\Users\Admin\.m2\repository\org\apache\logging\log4j\log4j-slf4j-impl\2.20.0\log4j-slf4j-impl-2.20.0.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-api\2.0.7\slf4j-api-2.0.7.jar;C:\Users\Admin\.m2\repository\org\slf4j\slf4j-simple\2.0.7\slf4j-simple-2.0.7.jar;C:\Users\Admin\.m2\repository\org\mongodb\bson\4.2.0\bson-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-core\4.2.0\mongodb-driver-core-4.2.0.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongo-java-driver\3.12.12\mongo-java-driver-3.12.12.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver\3.12.9\mongodb-driver-3.12.9.jar;C:\Users\Admin\.m2\repository\org\mongodb\mongodb-driver-sync\4.9.1\mongodb-driver-sync-4.9.1.jar recv.java AverageExpense-2013-Private-4-year
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'AverageExpense-2013-Private-4-year': 'Region : South West Region value: 109629'
[x] Received 'AverageExpense-2013-Private-4-year': 'Region : West Region value: 273494'
[x] Received 'AverageExpense-2013-Private-4-year': 'Region : Mid West Region value: 353969'
[x] Received 'AverageExpense-2013-Private-4-year': 'Region : South East Region value: 417915'
[x] Received 'AverageExpense-2013-Private-4-year': 'Region : North East Region value: 452952'
```

CONSUMER 5

- 6. We can also verify the results in the rabbitMQ server.

```
rabbitmqctl list_queues
```

- The above command will list all the queues created in the server.

```
[opc@instance-20230221-1508 ~]$ sudo rabbitmqctl list_queues
Timeout: 60.0 seconds ...
Listing queues for vhost / ...
name      messages
Top5-Economic-2013-Private-4-year
Top5-HighestGrowth-3-years      0
Cost-2013-Alabama-Private-4-year
Top5-Expensive-2013-Private-4-year
AverageExpense-2013-Private-4-year
```

- We can see that in the rabbitmq dashboard as well.

RabbitMQ™ RabbitMQ 3.9.9 Erlang 24.1.6

Overview Connections Channels Exchanges **Queues**

Queues

▼ All queues (5)

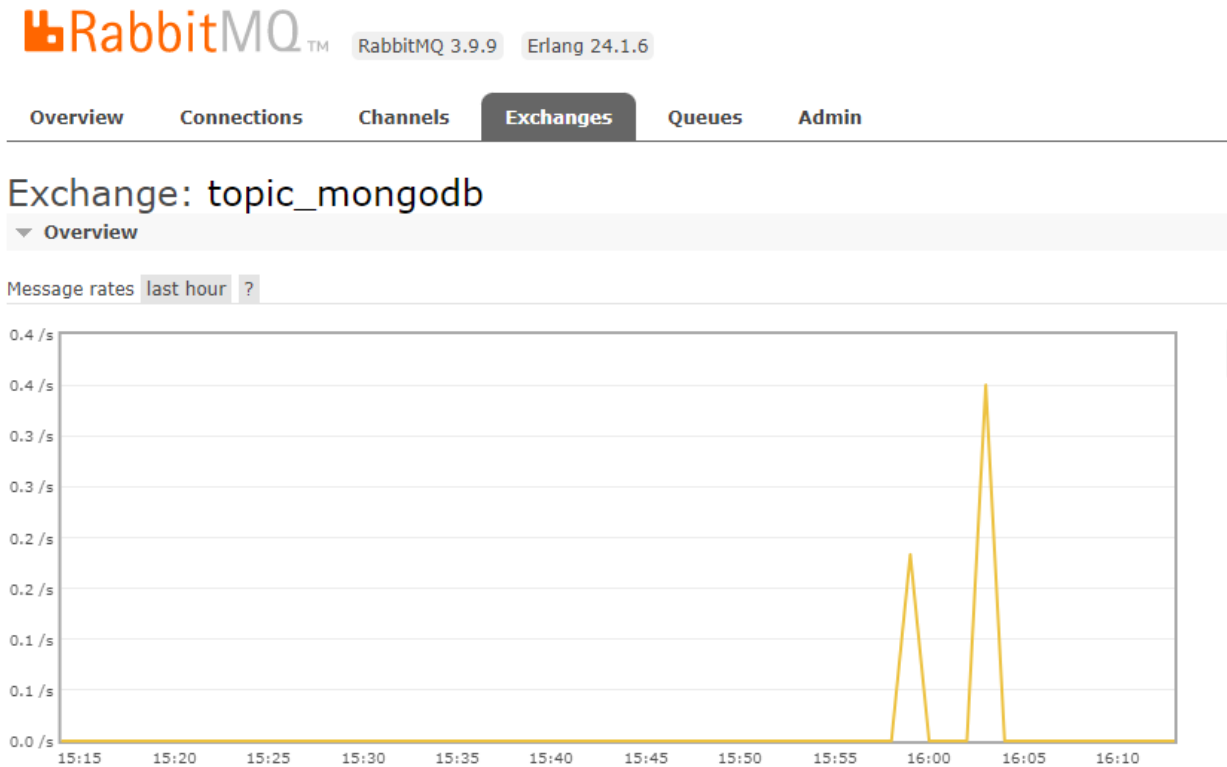
Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Overview				Mess
Name	Type	Features	State	Reac
AverageExpense-2013-Private-4-year	classic	D	idle	
Cost-2013-Alabama-Private-4-year	classic	D	idle	
Top5-Economic-2013-Private-4-year	classic	D	idle	
Top5-Expensive-2013-Private-4-year	classic	D	idle	
Top5-HighestGrowth-3-years	classic	D	idle	

► Add a new queue

- We can also verify the exchange module created



Github Link - <https://github.com/Rishi-M-G/coen6371-as3>

Conclusion

We have implemented a Producer and Consumers using Advanced Message Queueing Protocol in Java with the help of RabbitMQ messaging server. The producer retrieves the datasets from each collection from the mongodb cloud service for each topic as per the requirement. The producer publishes the data to the exchange topics with a routing key that matches to the topic for each queue. The consumer receives the data from the queue. We have successfully executed the server and clients and produced the results as well.