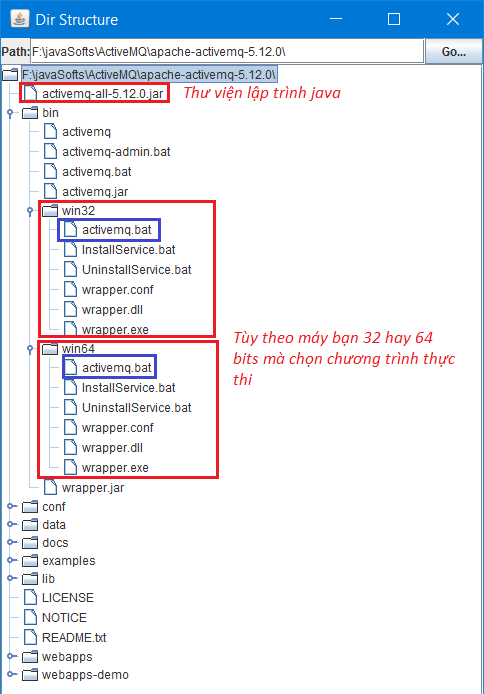
# Phần 1: MESSAGE-ORIENTED ARCHITECTURE

1. **APACHE ACTIVEMQ**
2. **Tài liệu tham khảo**

[1] Bruce Snyder, Dejan Bosanac, Rob Davies; *ActiveMQ in Action*; ©2011 by Manning Publications Co.

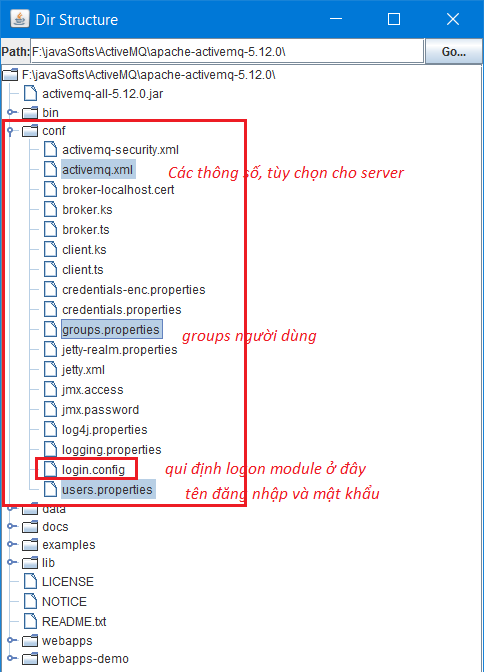
[2] Mark Richards, Richard Monson-Haefel, David A. Chappell; *Java Message Service*, *2nd Edition*; © 2009 by O’Reilly Media, Inc.

1. **Cài đặt và khởi động Apache ActiveMQ trên môi trường Windows**
   1. Download <http://activemq.apache.org> 🡪 Giải nén ở thư mục nào đó.



* 1. Thiết lập biến môi trường JDK
  2. Cấu hình cho ActiveMQ Server

Các file cấu hình của ActiveMQ trong thư mục conf



Cấu hình login module (bắt buộc login khi kết nối)

File **login.config**

|  |
| --- |
| activemq {  org.apache.activemq.jaas.PropertiesLoginModule required  org.apache.activemq.jaas.properties.user="**users.properties**"  org.apache.activemq.jaas.properties.group="**groups.properties**"  reload=true;  }; |

Cấu hình groups, file **groups.properties**

|  |
| --- |
| *#group\_name=users\_list*  admins=admin,teo  users=ty  dhcn=men  guests=guest |

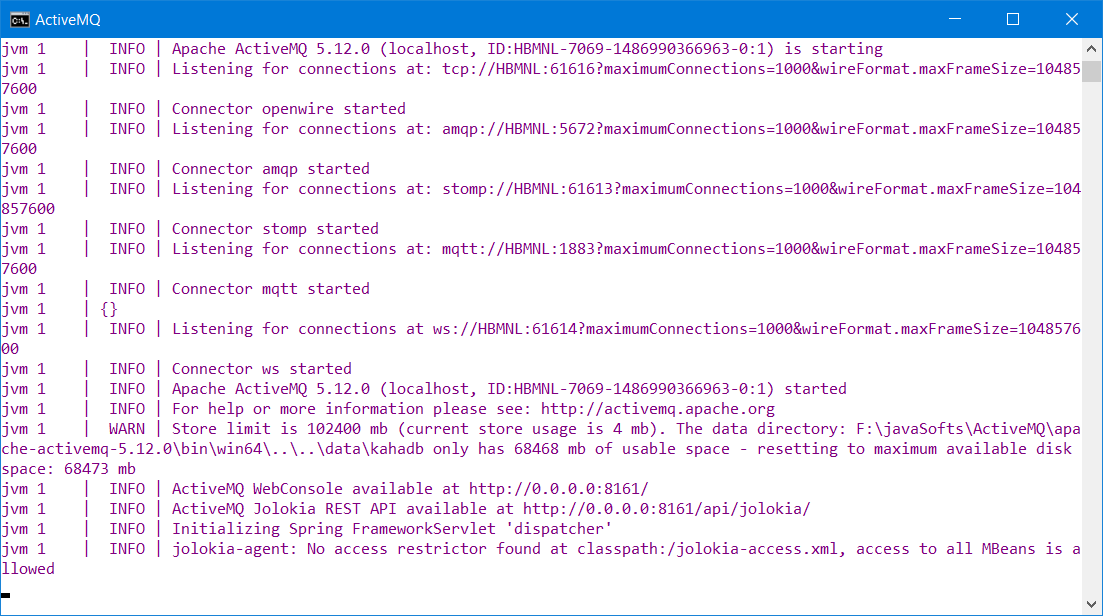
Cấu hình người dùng, file **users.properties**

|  |
| --- |
| *#username=password*  admin=admin  teo=123  ty=456  men=123 |

Đọc thêm phần security ở đây: <http://activemq.apache.org/security.html>

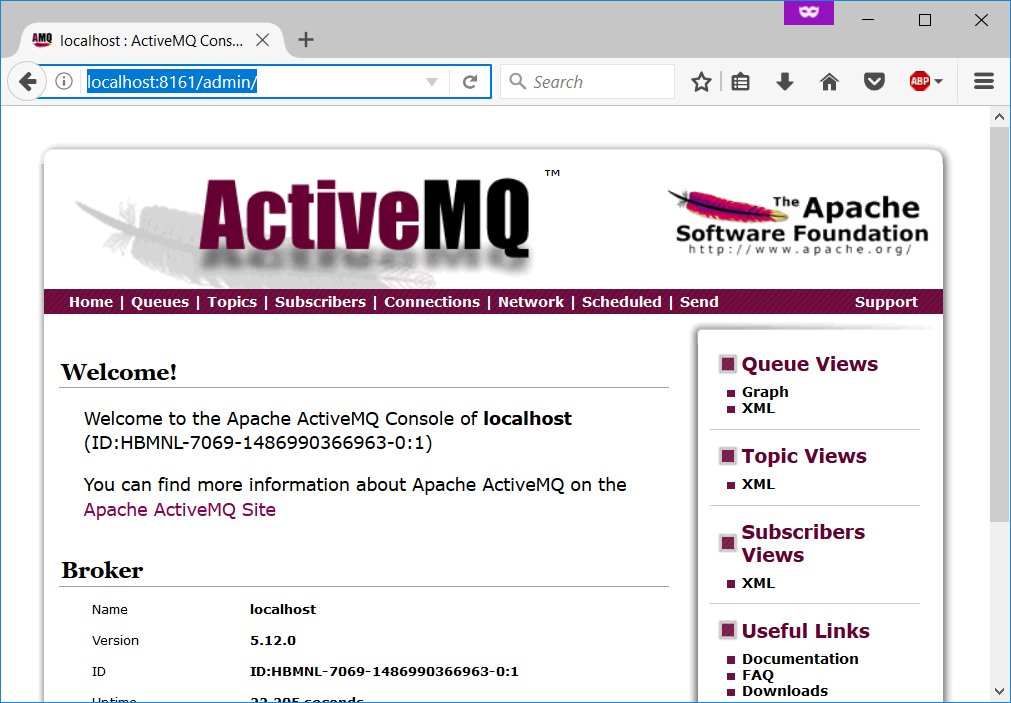
* 1. Khởi động ActiveMQ Server

Chạy file activemq.bat trong thư mục Win32 hoặc Win64 tùy theo máy 32 hay 64 bits.



*Lưu ý: không đóng của sổ này, ActiveMQ đã được khởi động và sẳn sàng cho các client truy cập thông qua giao thức TCP trên cổng 61616*

Mở cửa sổ browser, gõ địa chỉ: <http://localhost:8161/admin/>. Nếu có yêu cầu đăng nhập gõ username và password đều là admin



1. **Đọc thêm tài liệu cấu hình** <http://activemq.apache.org/getting-started.html>
2. **NGUYÊN LÝ LẬP TRÌNH**

|  |
| --- |
| Một ứng dụng JMS đơn giản sẽ gồm các bước: [1] trang 36  1. Acquire a JMS connection factory  2. Create a JMS connection using the connection factory  3. Start the JMS connection  4. Create a JMS session from the connection  5. Acquire a JMS destination  6. Create a JMS producer, OR  a. Create a JMS producer  b. Create a JMS message and address it to a destination  7. Create a JMS consumer  a. Create a JMS consumer  b. Optionally register a JMS message listener  8. Send or receive JMS message(s)  9. Close all JMS resources (connection, session, producer, consumer, and so forth) |

**Ví dụ minh họa Gửi và nhận message**

**Code gửi và nhận message bằng Java**

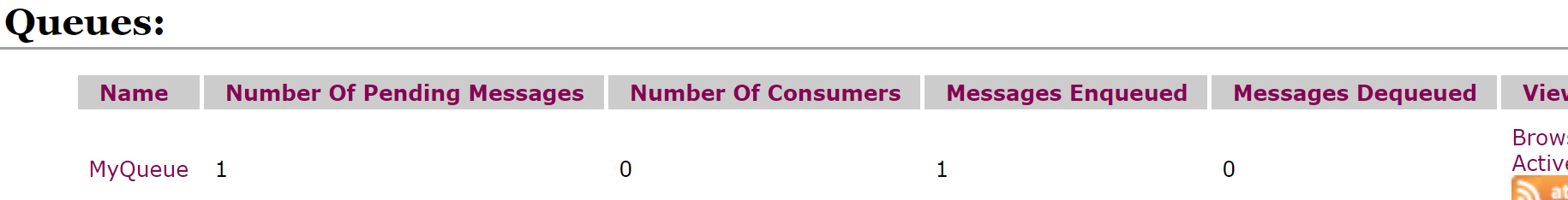
**Bước 1:** Tạo file jndi.properties trong thư mục src với nội dung bên dưới *(xem chi tiết* [*http://activemq.apache.org/jndi-support.html*](http://activemq.apache.org/jndi-support.html) *)*

|  |
| --- |
| src/jndi.properties |
| java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory  java.naming.provider.url=tcp://localhost:61616  connectionFactoryNames=connectionFactoryName  queue.MyQueue=jms.MyQueue |

**Bước 2: Viết code cho bên gửi**

|  |
| --- |
| **[1] 2.8 Sending a JMS message, trang 36** |
| **MyMessageProducer.java** |
| **public** **class** MyMessageProducer {  **public** **static** **void** main(String[] args) {  ConnectionFactory connectionFactory = **null**;  Connection connection = **null**;  Session session = **null**;  Destination destination = **null**;  MessageProducer producer = **null**;  Message message = **null**;  **boolean** useTransaction = **false**;  **try** {  //Thiết lập kết nối tới JMS provider,  //thông tin cần thiết trong file jndi.properties  Context context = **new** InitialContext();  connectionFactory = (ConnectionFactory) context.lookup("connectionFactoryName");  connection = connectionFactory.createConnection();  connection.start();  session = connection.createSession(useTransaction, Session.***AUTO\_ACKNOWLEDGE***);  destination = session.createQueue("MyQueue");  producer = session.createProducer(destination);  message = session.createTextMessage("This is a test");  producer.send(message);  }**catch** (JMSException e) {  e.printStackTrace();  }  **catch** (Exception e) {  e.printStackTrace();  }**finally** {  **try** {  producer.close();  session.close();  connection.close();  }**catch** (JMSException e) {  e.printStackTrace();  }  **catch** (Exception e) {  e.printStackTrace();  }  }  }  } |

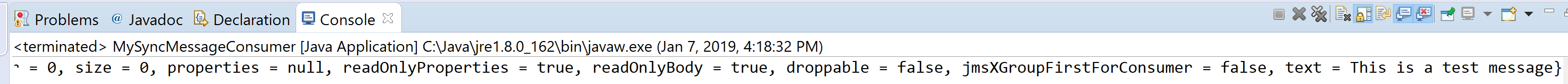
Thực thi và xem kết quả

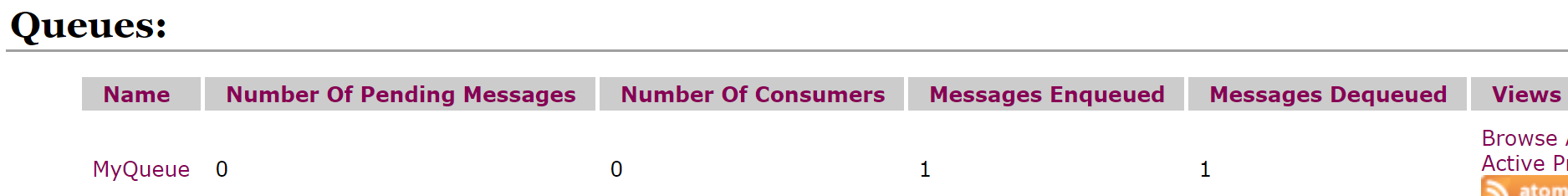


**Bước 3: Viết code cho bên nhận**

|  |
| --- |
| **[1] Listing 2.9 Receiving a JMS message synchronously** |
| **MySyncMessageConsumer.java** |
| **public** **class** MySyncMessageConsumer {  **public** **static** **void** main(String[] args) {  ConnectionFactory connectionFactory = **null**;  Connection connection = **null**;  Session session = **null**;  Destination destination =**null**;  MessageConsumer consumer = **null**;  Message message = **null**;  **boolean** useTransaction = **false**;  **try** {  Context context = **new** InitialContext();  connectionFactory = (ConnectionFactory) context.lookup("connectionFactoryName");  connection = connectionFactory.createConnection();  connection.start();  session=connection.createSession(useTransaction, Session.***AUTO\_ACKNOWLEDGE***);  destination=session.createQueue("MyQueue");  consumer=session.createConsumer(destination);  message=(TextMessage)consumer.receive();  System.***out***.println("Received message: " + message);  } **catch** (NamingException e) {  e.printStackTrace();  } **catch** (JMSException e) {  e.printStackTrace();  }**finally** {  **try** {  consumer.close();  session.close();  connection.close();  } **catch** (JMSException e) {  e.printStackTrace();  }  }  }  } |

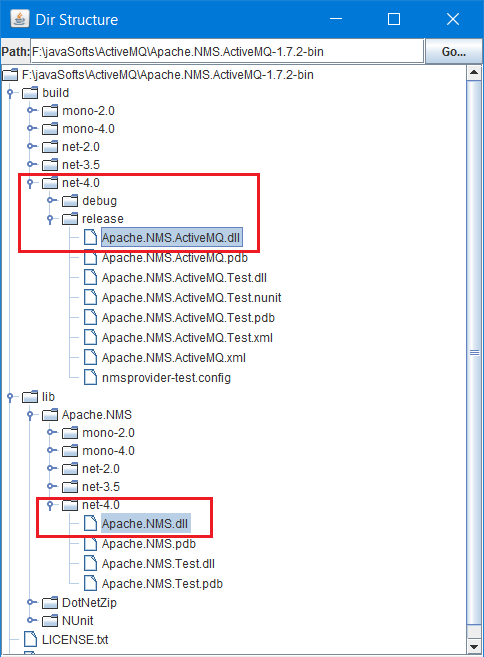
**Thực thi và xem kết quả**



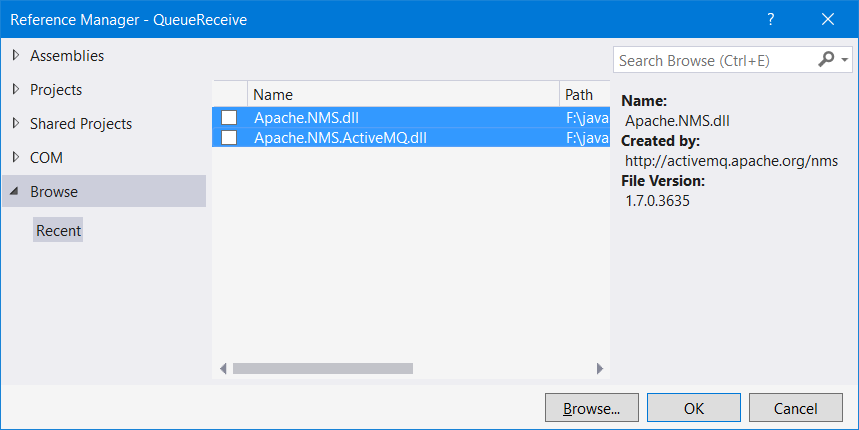


**Code minh họa gửi và nhận message bằng C#**

1. Thư viện dùng cho lập trình .NET tại [**http://activemq.apache.org/nms/activemq-downloads.html**](http://activemq.apache.org/nms/activemq-downloads.html)
2. Giải nén ta có cấu trúc thư mục như hình sau. Chú ý các chỗ đánh dấu là 2 thư viện chính ta cần làm việc.



1. Mở Visual Studio, tạo project mới, thêm tham chiếu như hình



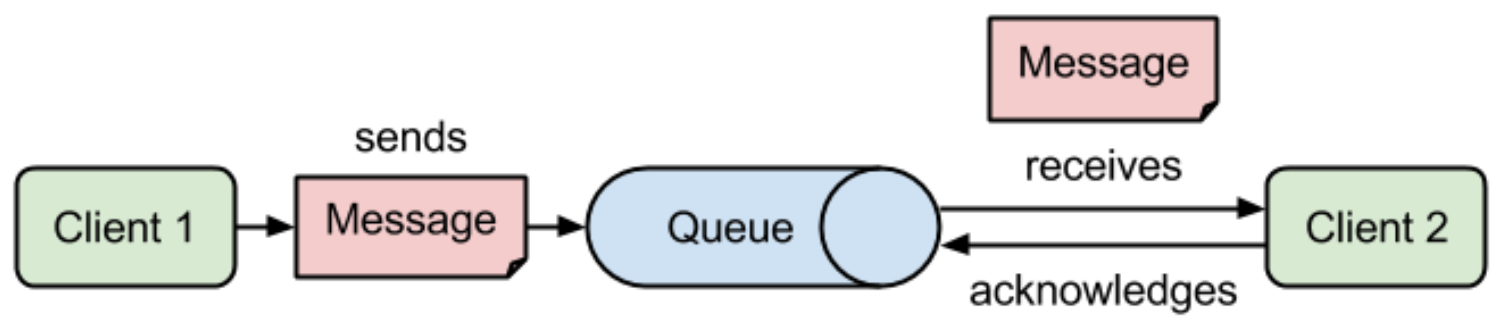
1. Code cho bên gửi

|  |
| --- |
| using Apache.NMS;  using Apache.NMS.ActiveMQ;  using Apache.NMS.ActiveMQ.Commands;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ActiveMessageSender  {  class Program  {  static void Main(string[] args)  {  IConnectionFactory factory = new ConnectionFactory("tcp://localhost:61616");  IConnection con = factory.CreateConnection("admin", "admin");  con.Start();  ISession session = con.CreateSession(AcknowledgementMode.AutoAcknowledge);  ActiveMQQueue destination = new ActiveMQQueue("MyQueue");  IMessageProducer producer = session.CreateProducer(destination);  IMessage msg = new ActiveMQTextMessage("This is a test");  producer.Send(msg);  Console.WriteLine("Finished");  session.Close();  con.Close();    }  }  } |

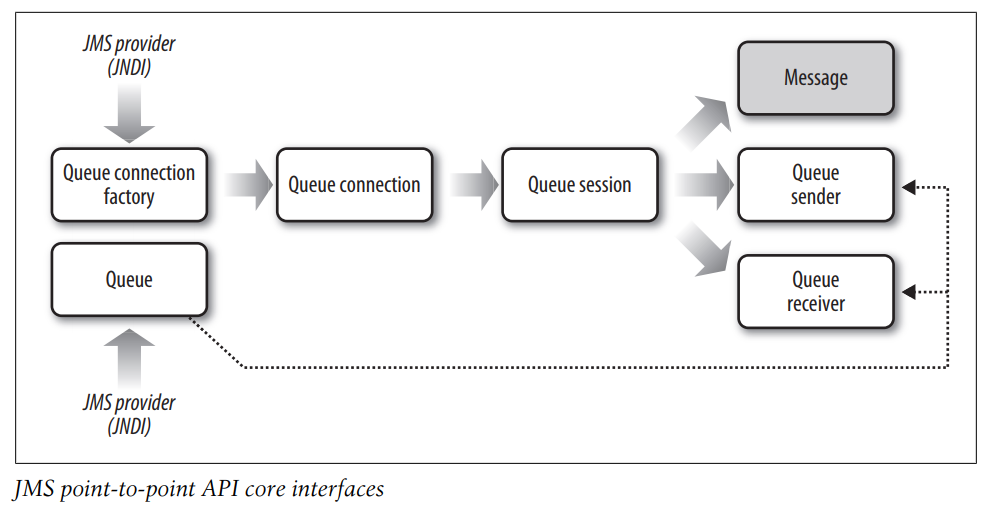
1. Code cho bên nhận

|  |
| --- |
| using Apache.NMS;  using Apache.NMS.ActiveMQ;  using Apache.NMS.ActiveMQ.Commands;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ActiveMessageReceiver  {  class Program  {  static void Main(string[] args)  {  IConnectionFactory factory = new ConnectionFactory("tcp://localhost:61616");  IConnection con = factory.CreateConnection("admin", "admin");  con.Start();  ISession session = con.CreateSession(AcknowledgementMode.AutoAcknowledge);  ActiveMQQueue destination = new ActiveMQQueue("MyQueue");  IMessageConsumer consumer = session.CreateConsumer(destination);  IMessage message = consumer.Receive();  if (message is ITextMessage)  {  ITextMessage msg = message as ITextMessage;  Console.WriteLine("Receive: " + msg.Text);  }  else  {  Console.WriteLine("Unexpected message type: " + message.GetType().Name);  }  Console.ReadKey();  session.Close();  con.Close();  }  }  } |

* 1. **Mô hình Point-to-Point**



|  |  |
| --- | --- |
| **JMS API** | **Point-to-Point API** |
| ConnectionFactory  Destination  Connection  Session  Message  MessageProducer  MessageConsumer | QueueConnectionFactory  Queue  QueueConnection  QueueSession  Message  QueueSender  QueueReceiver |
| *Note: ConnectionFactory và Destination phải được lấy từ Provider dùng JNDI* | |



**Các đặc điểm quan trọng của mô hình p2p [2] trang 63**

* Các messages được trao đổi thông qua 1 queue. Producer *(sender)* gửi các message tới queue và Consumer *(receiver)* sẽ nhận và xử lý các message từ queue.
* Mỗi message chỉ được gửi tới một receiver. Có thể có nhiều receiver cùng lắng nghe trên cùng 1 queue, nhưng 1 message chỉ được gửi tới duy nhất một receiver đang lắng nghe trên queue.
* Các message là có thứ tự. Thứ tự FIFO, thứ tự phân bổ tới các receiver là thứ tự tin nhắn được đưa vào queue. Khi 1 message được tiêu thụ, chúng sẽ được loại ra khỏi queue.
* Không có sự ràng buộc giữa các producer và các consumer. Bên gửi và bên nhận có thể thêm tự động tại thời điểm runtime, điều này cho phép hệ thống tăng hoặc giảm độ phức tạp theo thời gian.

**Khi nào nên dùng Point-to-Point Messaging [2] trang 66**

* Point-to-point được dùng khi một message chỉ được duy nhất một receiver nhận và xử lý (is used when you want one receiver to process any given message once and only once)
* Trường hợp khác là ta muốn giao tiếp đồng bộ giữa các component mà các component này viết bởi các ngôn ngữ lập trình khác nhau hoặc khác nền tảng. (Another use case for point-to-point messaging is when you need synchronous communication between components, but those components are written in different programming languages or implemented in different technology platforms (e.g., J2EE and .NET).)

**Ví dụ minh họa Gửi và nhận đối tượng Sinhvien**

*Lưu ý: Việc sử dụng ObjectMessage thường không được khuyến khích, chi tiết* [*http://activemq.apache.org/objectmessage.html*](http://activemq.apache.org/objectmessage.html)

**Code minh họa gửi và nhận message bằng Java**

Tạo file jndi.properties

|  |
| --- |
| src/jndi.properties |
| java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory  java.naming.provider.url=tcp://localhost:61616  trustAllPackages=true  connectionFactoryNames=cfName  queue.MyQueue=jms.MyQueue |

Tạo lớp đối tượng:

Sinh viên gồm: Mã sinh viên, họ tên, tuổi

Code cho bên gửi

|  |
| --- |
| MyQSender.java |
| **public** **class** MyQSender {  **private** QueueConnection qConnect = **null**;  **private** QueueSession qSession = **null**;  **private** Queue queue = **null**;  **public** MyQSender() **throws** NamingException, JMSException {    Context ctx = **new** InitialContext();  QueueConnectionFactory qFactory = (QueueConnectionFactory) ctx.lookup("cfName");  qConnect = qFactory.createQueueConnection();  qSession = qConnect.createQueueSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  queue = (Queue) ctx.lookup("MyQueue");  qConnect.start();  }  **public** **static** **void** main(String[] args) **throws** NamingException, JMSException {  Sinhvien sv = **new** Sinhvien("01420011", "Nguyễn Văn Lành", 30);  MyQSender sender = **new** MyQSender();  sender.sendObjectMessage(sv);  sender.exit();  }  **private** **void** exit() **throws** JMSException {  qSession.close();  qConnect.close();  }  **private** **void** sendObjectMessage(Sinhvien sv) **throws** JMSException {  ObjectMessage msg = qSession.createObjectMessage();  QueueSender qSender = qSession.createSender(queue);  msg.setObject(sv);  qSender.send(msg);  }  } |

Code cho bên nhận

|  |
| --- |
| MyAsyncReceiver.java |
| **public** **class** MyAsyncReceiver **implements** MessageListener {  **private** QueueConnection qConnect = **null**;  **private** QueueSession qSession = **null**;  **private** Queue queue = **null**;  **public** MyAsyncReceiver() **throws** NamingException, JMSException {  Context ctx = **new** InitialContext();  QueueConnectionFactory qFactory = (QueueConnectionFactory)ctx.lookup("cfName");  qConnect = qFactory.createQueueConnection();  qSession = qConnect.createQueueSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  queue = (Queue)ctx.lookup("MyQueue");  qConnect.start();  QueueReceiver qReceiver = qSession.createReceiver(queue);  qReceiver.setMessageListener(**this**);  System.***out***.println("Waiting ...");  }  **public** **static** **void** main(String[] args) **throws** NamingException, JMSException {  **new** MyAsyncReceiver();  }  @Override  **public** **void** onMessage(Message msg) {  **if**(msg **instanceof** ObjectMessage) {  **try** {  ObjectMessage objmsg = (ObjectMessage) msg;  Sinhvien sv = (Sinhvien) objmsg.getObject();  System.***out***.println(sv);  } **catch** (JMSException e) {  e.printStackTrace();  }  }  }  } |

**Code minh họa gửi và nhận message bằng C#**

Tạo lớp đối tượng: Sinh viên gồm: Mã sinh viên, họ tên, tuổi

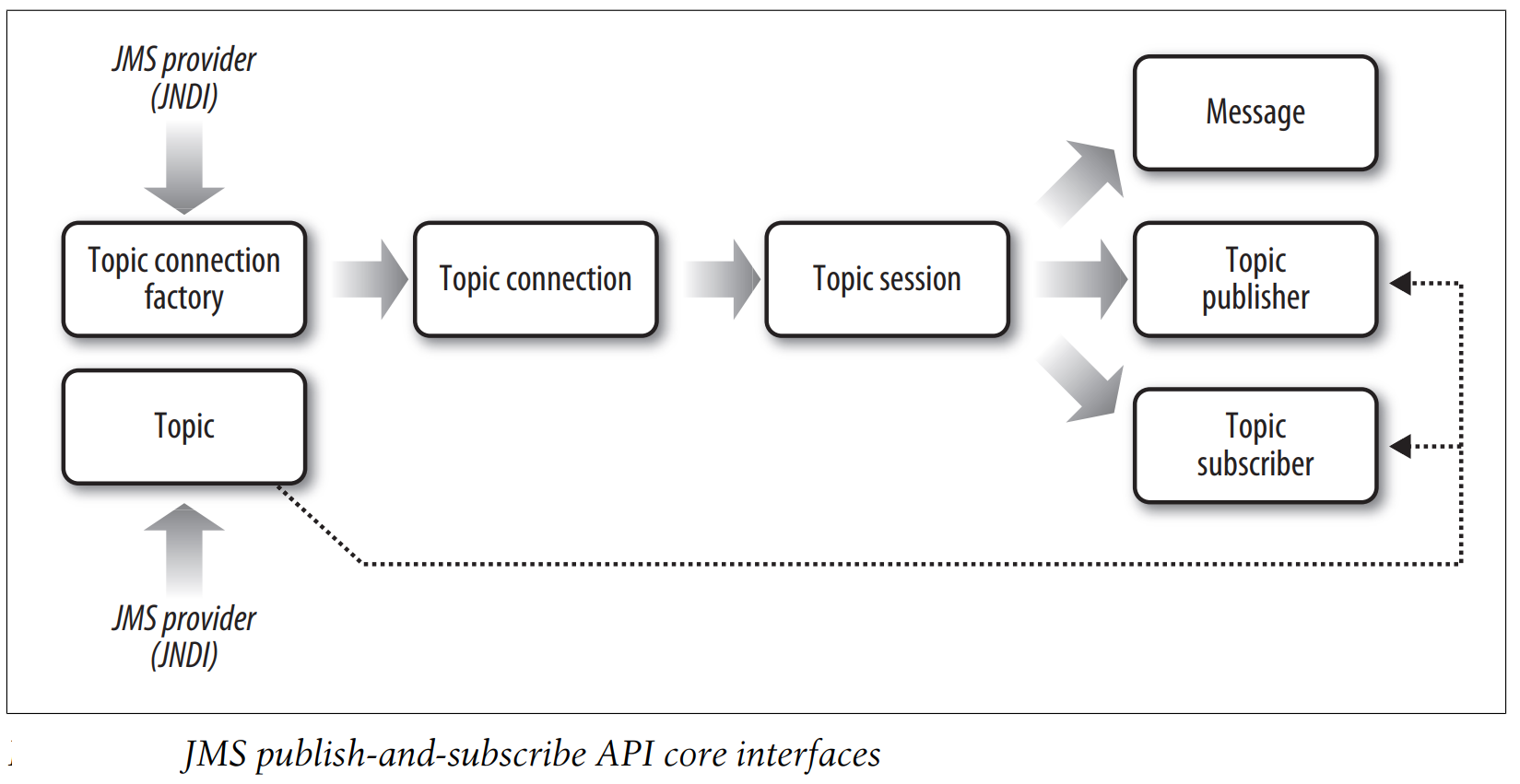
Code bên gửi

|  |
| --- |
| using Apache.NMS;  using Apache.NMS.ActiveMQ;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using BusinessObject;  namespace P2PMOM  {  class MyQSender  {  private IConnection con = null;  private ISession session = null;  public MyQSender()  {  IConnectionFactory factory = new ConnectionFactory("tcp://localhost:61616");  con = factory.CreateConnection("admin", "admin");  con.Start();  session = con.CreateSession(AcknowledgementMode.AutoAcknowledge);  }  internal void sendObjectMessage(Sinhvien sv)  {  IDestination destination = session.GetQueue("ObjectQueue");  IMessageProducer producer = session.CreateProducer(destination);  IObjectMessage msg = session.CreateObjectMessage(sv);  producer.Send(msg);    Console.WriteLine($"Sent object messages");  Console.ReadKey();  }  internal void exit()  {  session.Close();  con.Close();  }  }  } |
| using BusinessObject;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace P2PMOM  {  class Program  {  static void Main(string[] args)  {  Sinhvien sv = new Sinhvien("01420011", "Nguyễn Văn Lành", 30);  MyQSender sender = new MyQSender();  sender.sendObjectMessage(sv);  sender.exit();  }  }  } |

Code bên nhận:

* 1. **Mô hình Publish/Subscribe**

|  |  |
| --- | --- |
| **JMS API** | **Publish-and-Subscribe API** |
| ConnectionFactory  Destination  Connection  Session  Message  MessageProducer  MessageConsumer | TopicConnectionFactory  Topic  TopicConnection  TopicSession  Message  TopicPublisher  TopicSubscriber |
| *Note: ConnectionFactory và Destination phải được lấy từ Provider dùng JNDI* | |

****

**Các đặt điểm của mô hình Pub/Sub**

* Các message được trao đổi thông qua một kênh ảo gọi là Topic;
* Mỗi message được gửi đến nhiều subscriber. Có nhiều loại subscriber gồm durable, nondurable, và dynamic;
* Publisher thường thì không biết và không nhận thức được subscriber nào đang nhận message;
* Các message được đưa tới các consumer một cách tự động. Các message được trao đổi thông qua một kênh gọi là một topic. Topic là một đích nơi các producer có thể đưa các meassage vào, và các subscriber có thể tiêu thụ các message. Các message được phân phát tự động tới tất cả các consumer;
* Đối với enterprise messaging nói chung, không có sự kết dính giữa producer và consumer. Các Subscriber và publisher có thể thêm tự động ở thời gian runtime, làm cho hệ thống tăng giảm độ phức tạp theo thời gian;
* Mỗi client sẽ nhận một bản sao của message trong topic. Một message được đưa lên topic bởi một publisher có thể tạo ra nhiều bản sao và phân phối tới hàng nghìn subscriber;

**Khi nào nên dùng Publish-and-Subscribe Messaging**

* Được dùng khi ta muốn phát (broadcast) một message tới nhiều consumer.

**Ví dụ Minh họa gửi và nhận đối tượng dạng tài liệu json**

**Code minh họa gửi và nhận message bằng Java**

Tạo file jndi.properties

|  |
| --- |
| src/jndi.properties |
| java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory  java.naming.provider.url=tcp://localhost:61616  connectionFactoryNames=cfName  topic.MyTopic=jms.MyTopic |

Tạo lớp đối tượng:

Sinh viên gồm: Mã sinh viên, họ tên, tuổi

Code cho bên gửi

|  |
| --- |
| MyTPublisher.java |
| **public** **class** MyTPublisher {  **private** TopicConnection tConnect = **null**;  **private** TopicSession qSession = **null**;  **private** Topic topic = **null**;  **public** MyTPublisher() **throws** NamingException, JMSException {    Context ctx = **new** InitialContext();  TopicConnectionFactory qFactory = (TopicConnectionFactory) ctx.lookup("cfName");  tConnect = qFactory.createTopicConnection();  qSession = tConnect.createTopicSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  topic = (Topic) ctx.lookup("MyTopic");  tConnect.start();  }  **public** **static** **void** main(String[] args) **throws** NamingException, JMSException {  Sinhvien sv = **new** Sinhvien("01420011", "Nguyễn Văn Lành", 30);  MyTPublisher sender = **new** MyTPublisher();  sender.sendMessage(sv);  sender.exit();  }  **private** **void** exit() **throws** JMSException {  qSession.close();  tConnect.close();  }  **private** **void** sendMessage(Sinhvien sv) **throws** JMSException {  Gson gson = **new** Gson();  String json = gson.toJson(sv);  TopicPublisher tPublisher = qSession.createPublisher(topic);  Message msg = qSession.createTextMessage(json);  tPublisher.send(msg);  }  } |

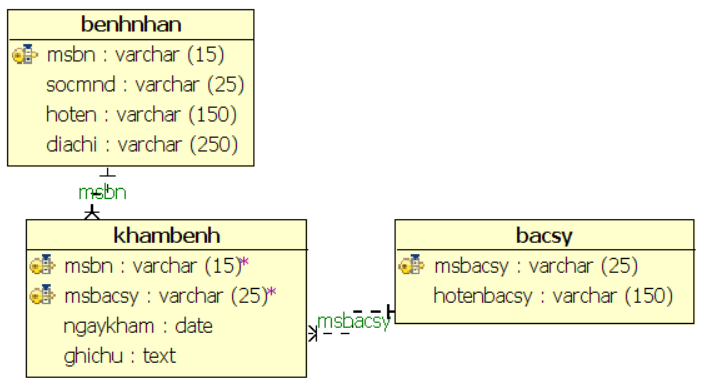
Code cho bên nhận

|  |
| --- |
| MyTSubscriber.java |
| **public** **class** MyTSubscriber **implements** MessageListener {  **private** TopicConnection tConnect = **null**;  **private** TopicSession tSession = **null**;  **private** Topic topic = **null**;  **public** MyTSubscriber() **throws** NamingException, JMSException {  Context ctx = **new** InitialContext();  TopicConnectionFactory qFactory = (TopicConnectionFactory)ctx.lookup("cfName");  tConnect = qFactory.createTopicConnection();  tSession = tConnect.createTopicSession(**false**, Session.***AUTO\_ACKNOWLEDGE***);  topic = (Topic)ctx.lookup("MyTopic");  tConnect.start();  TopicSubscriber tSubscriber = tSession.createSubscriber(topic);  tSubscriber.setMessageListener(**this**);  System.***out***.println("Subscriber\_1 is waiting ...");  }  **public** **static** **void** main(String[] args) **throws** NamingException, JMSException {  **new** MyTSubscriber();  }  @Override  **public** **void** onMessage(Message msg) {  Gson gson = **new** Gson();  **if**(msg **instanceof** TextMessage) {  **try** {  TextMessage tmsg = (TextMessage) msg;  String json = tmsg.getText();  System.out.println(json);  Sinhvien sv = gson.fromJson(json, Sinhvien.**class**);  System.***out***.println(sv);  } **catch** (JMSException e) {  e.printStackTrace();  }  }  }  } |

Thực thi:

Sao chép file MyTSubscriber.java thành nhiều file, thực thi cùng lúc. Sau đó, thực thi file MyTPublisher.java để gửi 1 đối tượng. Xem kết quả có gửi tới tất cả Subscriber không?

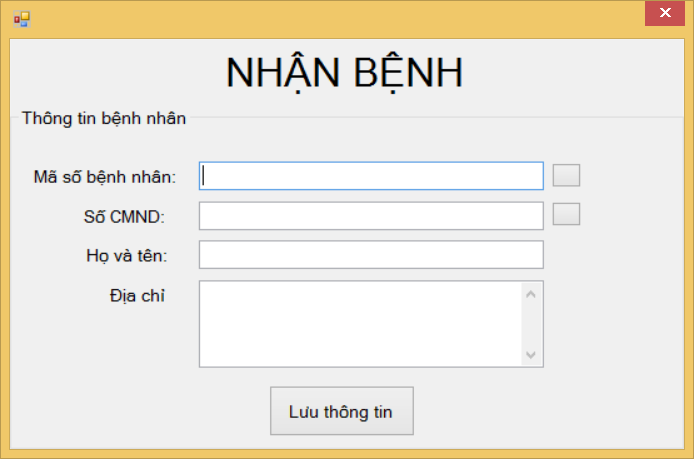
1. **BÀI TẬP**
2. Cho cơ sở dữ liệu quản lý thông tin bệnh nhân được cho như sau



**Yêu cầu:**

Chương trình được viết cho 2 loại nhân sự của phòng khám đặt tả như sau

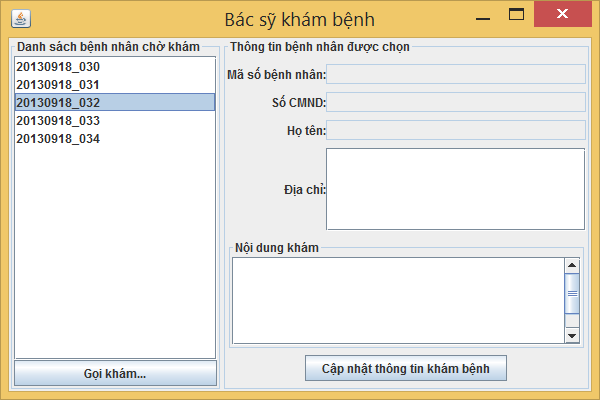
**Tại máy tính của Nhân viên lễ tân:**



Khi bệnh nhân vào khám thì nhân viên lễ tân sẽ nhập thông tin của bệnh nhân nếu đến khám lần đầu hoặc tìm thông tin bệnh nhân nếu tái khám (tìm theo mã số bệnh nhân hoặc theo số CMND). Sau khi tìm thấy hoặc nhập vào, nhấn nút “Lưu thông tin” thì chương trình sẽ notify cho máy tính đặc trong phòng của bác sỹ khám (giả sử có n bác sỹ, mỗi bác sỹ có một máy chạy độc lập).

**Tại máy tính trong phòng bác sỹ**

Khi có 1 bệnh nhân cần khám, chương trình sẽ đưa vào danh sách đợi để bác sỹ gọi vào.



Giả sử ta có một công việc cần phân chia cho N máy tính làm riêng rẽ sau đó ta tổng hợp kết quả lại. Yêu cầu mô phỏng cho hệ thống này: lập tài liệu thiết kế, hiện thực chương trình.

1. Viết chương trình groups chat với chat client được viết bằng Java và C#

|  |  |
| --- | --- |
|  |  |

1. \*\*\* Viết chương trình bằng java thực hiện công việc như sau:

Chương trình chia sẻ màn hình giữa một máy server và một tập các máy client.

1. Máy server có thể trong thấy màn hình của tất cả các máy clients kết nối
2. Khi máy server chọn chế độ trình diễn thì tất cả các máy client sẽ thấy được màn hình chia sẻ của máy server.
3. Hãy viết tài liệu thiết kế sau đó cài đặt hệ thống này

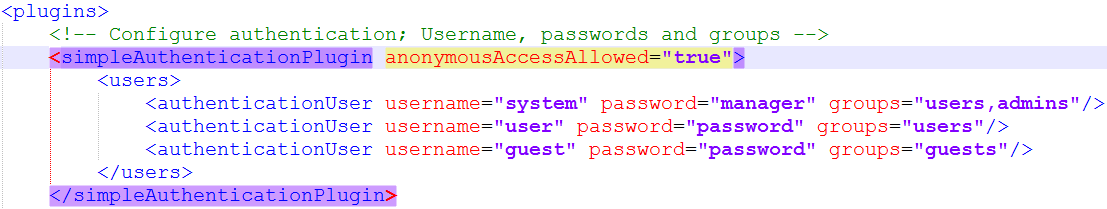
### SECURITY

#### CẤU HÌNH

##### Authetication

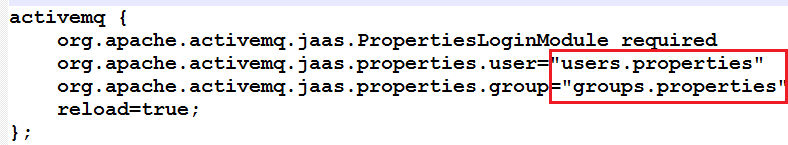
###### Simple Authentication plugin

Những gì bạn cần làm để thêm các groups và các users là mở file activemq.xml lên và thêm phân đoạn sau vào trong thẻ plugins



###### JAAS Authetication Plugin

Mở file %ACTIVEMQ\_HOME%\conf\**login.config**, hiệu chỉnh file sẽ lưu user và groups như hình



Ta có thể tạo nhiều configuration khác nhau. Ở trên là một cấu hình có tên mặc định activemq.

Sau đó ta thêm một số nhóm và một số user thuộc nhóm.

|  |  |
| --- | --- |
| groups.properties  Mỗi dòng chỉ một tên nhóm gồm những user nào | users.properties  mỗi dòng có dạng: tên\_user=mật\_khẩu |

Thêm phân đoạn sau vào trong file activemq.xml trong thẻ plugins

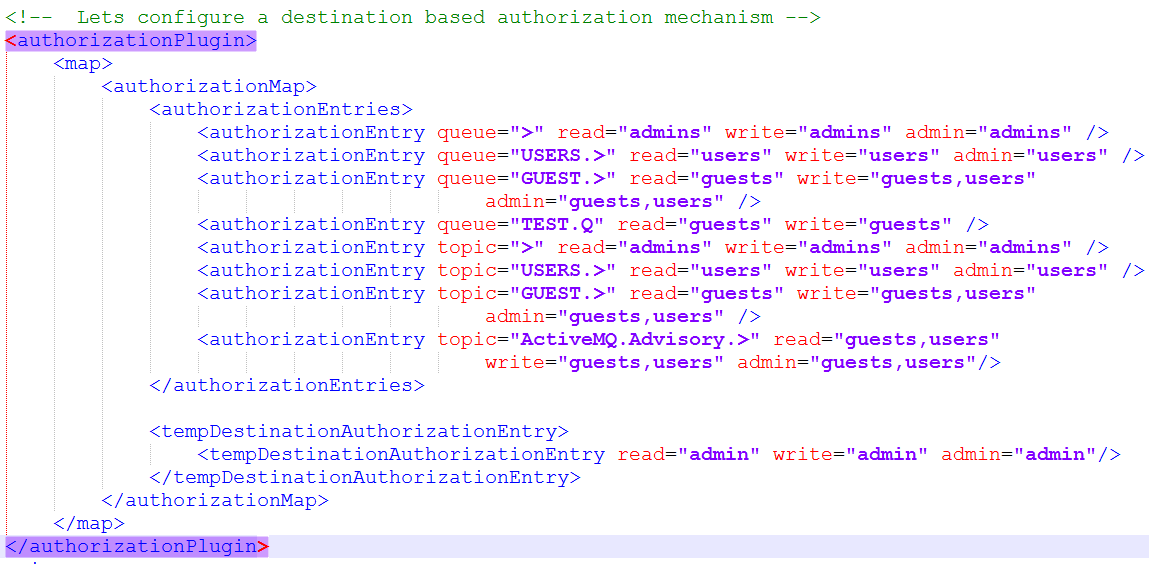
|  |
| --- |
| <jaasAuthenticationPlugin configuration=**"activemq"** /> |

##### AUTHORIZATION

Trong ActiveMQ, ta có thể gán quyền cho các groups (tên hoặc các ký tự thay thế (wild-card)) bằng một trong các quyền sau

|  |  |
| --- | --- |
| read | Có thể browse và consume từ một destination (queue/topic) |
| write | Có thể gửi message đến một destination |
| admin | Quyền admin |

Xem ví dụ ở phân đoạn sau:

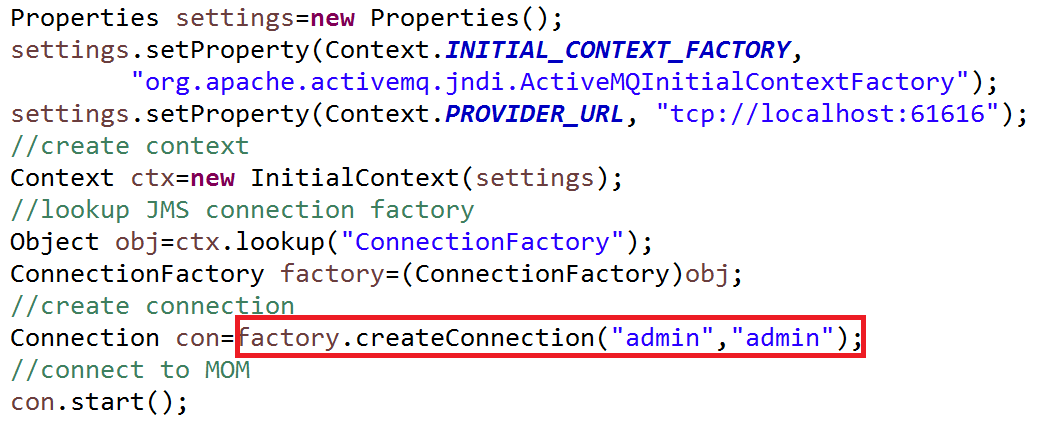


Ví dụ:

|  |
| --- |
| <authorizationEntry queue=**">"** read=**"admins"** write=**"admins"** admin=**"admins"** /> |

điều đó có nghĩa: những users thuộc group admins có thể toàn quyền nhận, gửi và quản lý trên tất cả các queue có trong hệ thống.

Trong quá trình tạo ra một Connection, ta phải cung cấp 2 tham số <username, password> để có thể đăng nhập vào destination



Nếu bạn cung cấp sai thông tin username-password thì bạn không thể authenticate

Nếu bạn đã authenticate mà không được cấp quyền trên destination thì bạn không được authorize

#### BÀI TẬP

1. Với bài tập 1 phần mô hình P2P,

* Tạo 2 users có tên **LanHT** va **HungNV** với mật khẩu tương ứng là 123Cba và 321Abc thuộc vào 2 nhóm có tên **bacsy** và **letan**.
* Cấu hình security sao cho nhân viên thuộc nhóm letan chỉ được phép gửi và nhân viên thuộc nhóm bacsy chỉ có quyền đọc queue **KhamBenh**.

1. Với bài 2 phần mô hình publisher-subcriber,

* Tạo 3 nhóm có tên và mục đích như sau:
  + Teacher – dùng cho máy server publish, các máy client subcribe
  + Student – dùng cho các máy client publisher và máy server subcribe
* Tạo Topic Presentation chỉ cho nhóm Teacher write, admin
* Tạo
* Cấu hình sao cho các máy nhóm client chỉ được quyền read cho topic có tên trình chiếu