

# MQ 接口使用和开发培训

技术组 Z.X.T

V 1.0

内部使用版本（不正确地方请指正）

|  |    |
|--|----|
| 税库银接口使用和开发培训.....                      | 1  |
| 一、 介绍 websphere MQ.....                | 1  |
| 1. MQSeries 和消息队列排队.....               | 1  |
| a. MQI - 一种公共应用程序编程接口.....             | 2  |
| b. 时间无关的应用程序.....                      | 2  |
| c. 驱动处理.....                           | 2  |
| d. 数据完整性和资源保护.....                     | 2  |
| 2. 传输示意图.....                          | 2  |
| 3. 消息和队列.....                          | 3  |
| a. 消息.....                             | 3  |
| b. 队列.....                             | 4  |
| c. 消息的特点.....                          | 4  |
| d. 队列的特点.....                          | 4  |
| 二、 WebSphere MQ server for WIN 安装..... | 4  |
| 1. 安装流程.....                           | 4  |
| 一、 安装 IBM JDK.....                     | 4  |
| 二、 安装 IES.....                         | 6  |
| 三、 安装服务.....                           | 7  |
| 2. 补充和关注点.....                         | 15 |
| 三、 WebSphere MQ for java 的简单例子.....    | 15 |
| 1..需要的 java 包.....                     | 15 |
| 2.调用的两种方式.....                         | 15 |

## 一、 介绍 websphere MQ

### 1. MQSeries 和消息队列排队

MQSeries 产品使应用程序可使用消息队列排队来参与*消息驱动处理*。通过消息驱动处理,应用程序可以使用适当的消息排队软件产品在相同或不同平台上进行应用程序彼此之间的通信

### **a. MQI - 一种公共应用程序编程接口**

MQSeries 产品实现了公共应用程序程序设计接口, 即 *消息队列接口 (MQI)*, 它可用于应用程序运行的任何平台。由应用程序发出的调用和它们交换的消息是通用的(**common**)。这使得编写和维护应用程序比使用传统方法更容易。它也方便了消息排队应用程序从一个平台向另一平台的移植

### **b. 时间无关的应用程序**

使用消息排队, 发送和接收程序之间的消息交换是时间无关的。这表示发送和接收应用程序是独立的, 这样发送程序可继续处理而不必等待接收程序收到消息后发出的确认信息。可能在发送程序发出消息时, 接收应用程序正处于忙的状态。其实, 接收应用程序即便没有运行也没有关系。MQSeries 一直将消息保存在队列中直到它可被处理。

### **c. 驱动处理**

消息驱动处理是一种应用程序设计风格。

用此风格, 应用程序被分成几个分离的、不相连的功能块, 每个块都有明确定义的输入和输出参数。每个功能块是一个应用程序, 通过将值放在消息, 再将消息放入队列, 实现输入和输出参数与其它应用程序的交换。

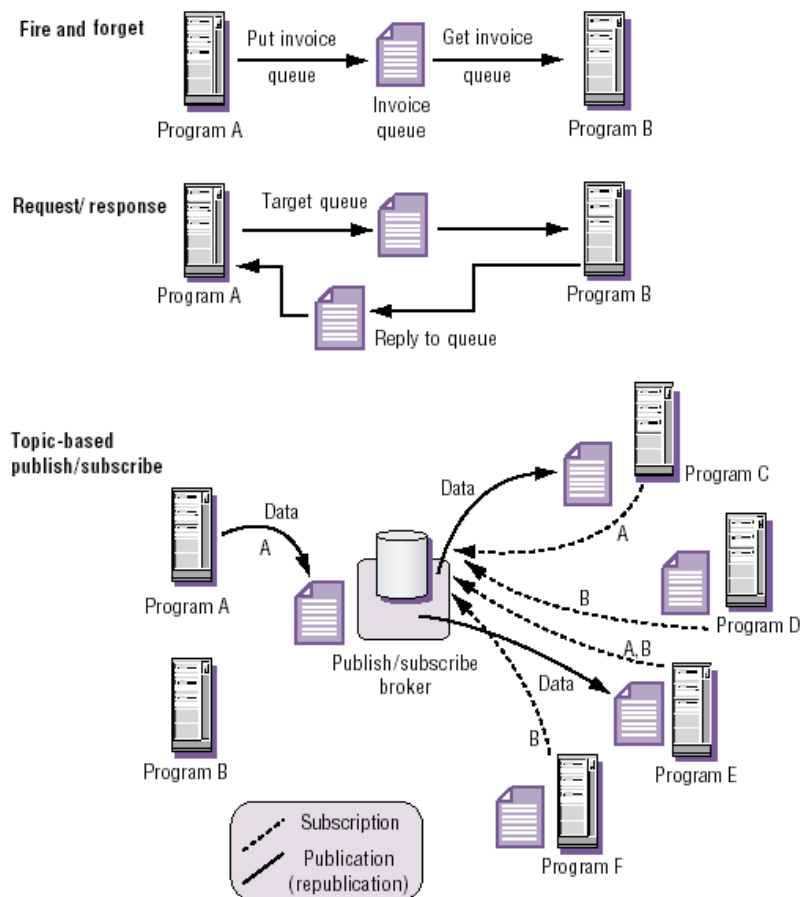
通过使用适当的 MQSeries 编程机制, 一个应用程序可因一个或多个消息到达队列而开始执行。如果需要, 程序能够在队列中的所有消息处理完后终止。

这种应用程序设计风格能比其它应用程序设计风格更快速地建立新的应用程序, 或修改现存的应用程序。

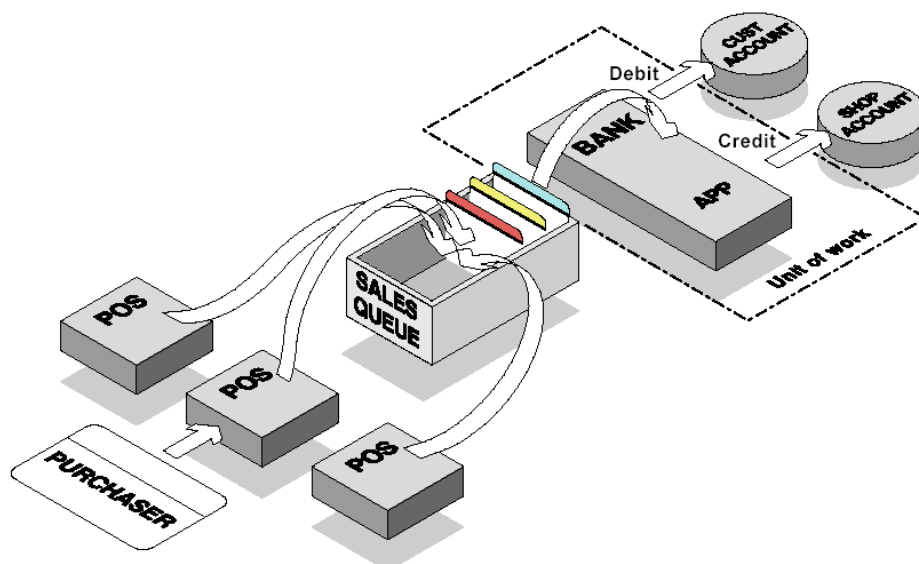
### **d. 数据完整性和资源保护**

MQSeries 应用程序能以非常高的一致性传送数据。

## **2. 传输示意图**



IBM 提供的示意图(pos 机和银行接口)



### 3. 消息和队列

#### a. 消息

消息两部分，*消息描述符*和 *应用程序数据*。应用程序数据的内容和结构由使用它们的应用程序来定义。消息描述符标识消息并包含了其它控制信息或属性，如创建消息的日期

和时间、消息类型及由发出消息的应用程序为消息分配的优先级

#### **b. 队列**

在物理概念上，一个队列就是一种类型的列表，用来存储消息直到它们被应用程序取走。

队列独立于使用它们的应用程序而存在。一个队列可以存在于：

如果队列是临时的，那么保存在主存储器中

如果必须保存该队列以便故障时恢复，那么保存在磁盘或类似的辅助存储器上

如果正在使用队列，并且必须保存该队列以便故障时恢复，那么同时保存在上述两种存储器上

每个队列属于一个队列管理器，它负责队列的维护。队列管理器将它检索到的消息放到适当的队列中。

队列可位于您的本地系统上，这种情况下它们被称为 本地队列；也可位于其它队列管理器上，这时则被称为 远程队列。

在 MQSeries 中，消息可被经适当授权的应用程序从队列中检索，且遵循下列检索算法：

先进先出(FIFO)。

消息优先级(在消息描述符中定义)。具有同一优先级的消息以 FIFO 的原则被检索。

一个程序请求特定的消息

#### **c. 消息的特点**

消息的大小

最大消息长度

消息分段

引用消息

#### **d. 队列的特点**

消息队列

事件队列

启动队列

传输队列

接受应答队列

死信队列

## **二、 WebSphere MQ server for WIN 安装**

### **1. 安装流程**

选择所有的文件安装在 d 盘目录下，以下所有的操作以此目录为依据。

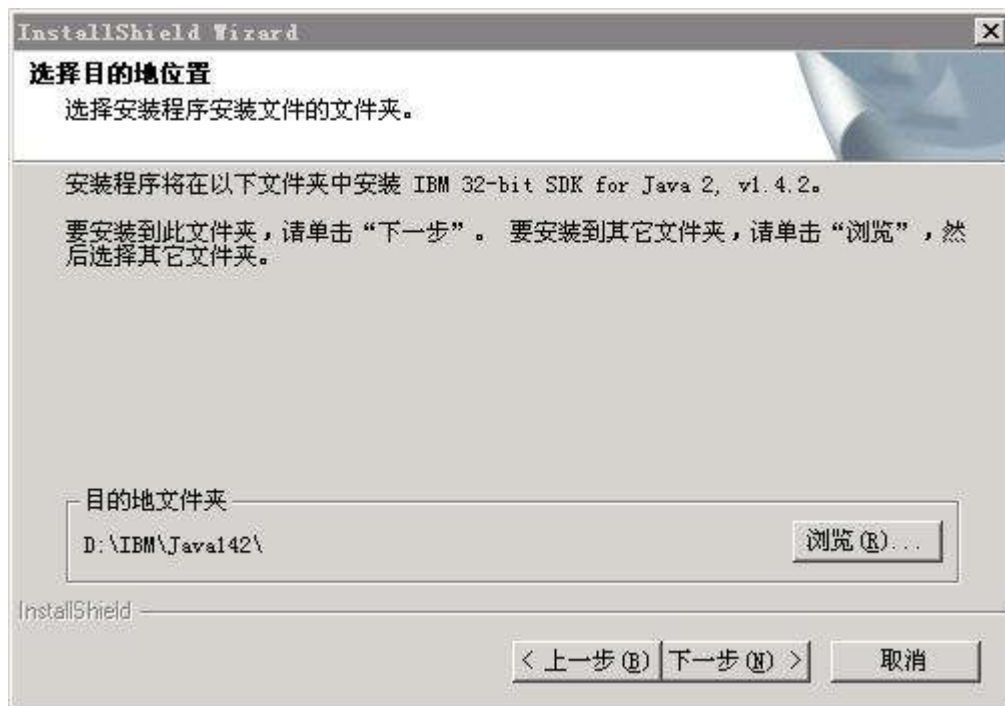
#### **一、 安装 IBM JDK**

此步骤可省略

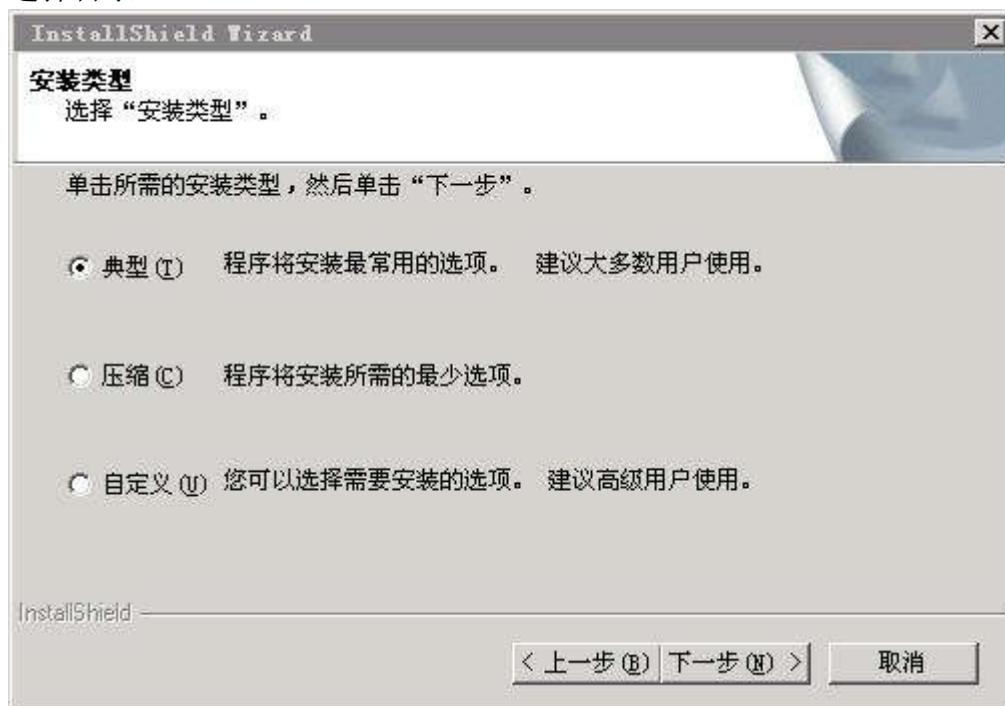
在光盘的 Prereqs\JDK 目录下，点击 ibm-java2-sdk-142.exe 进行安装



选择中文



选择目录



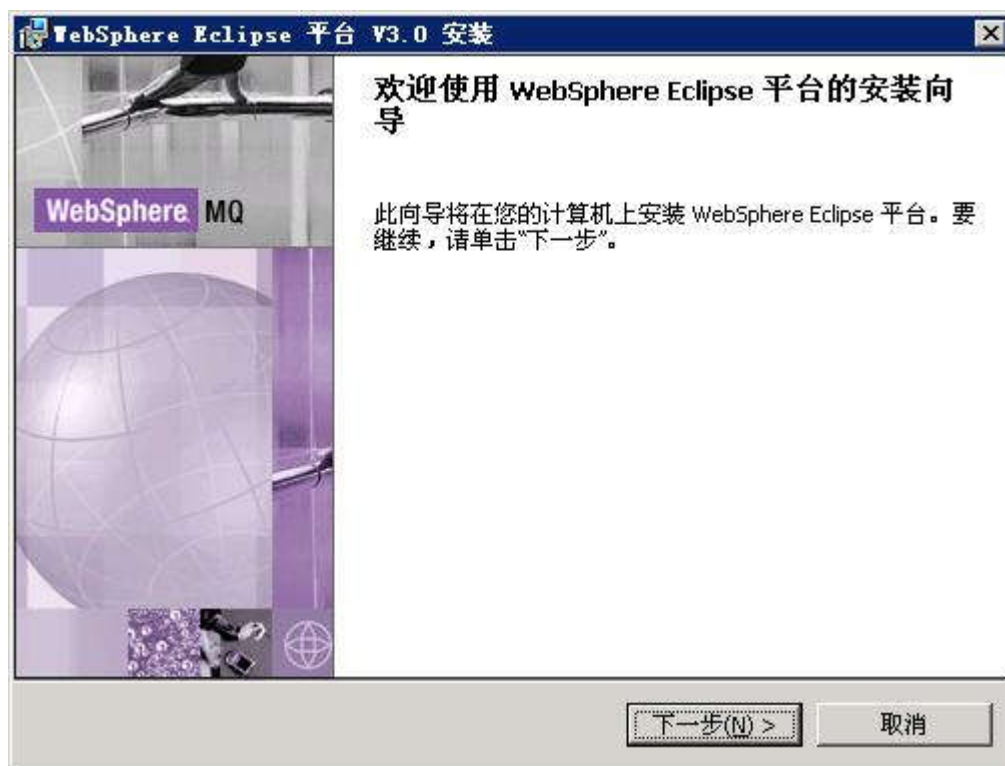
选择默认安装，等待结束

## 二、 安装 IES

IES 及为插肩式 ecilpse 用来做 MQ 的管理用，必须在安装 server 之前进行



选择语言



下一步



再下一步



等待结束。

### 三、安装服务

安装服务并没有创建 Active Directory，所以查看“软件需求”处没有问题以后就可以进行安装了



下一步

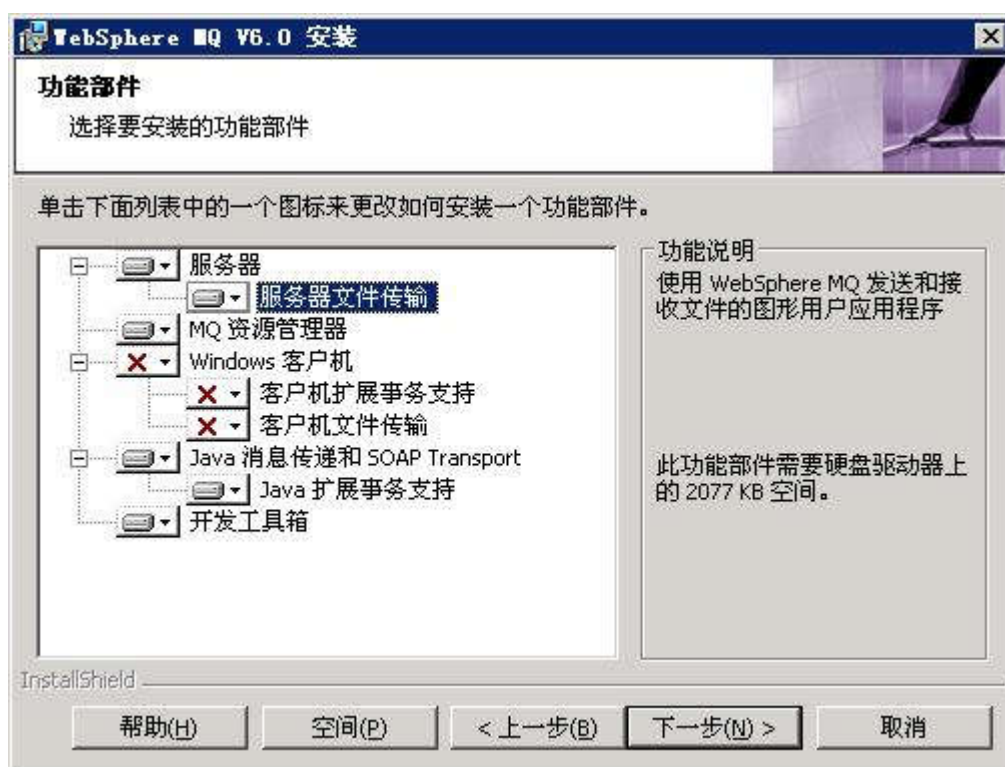


选择为 “定制”





## 设定目录



## 选择功能





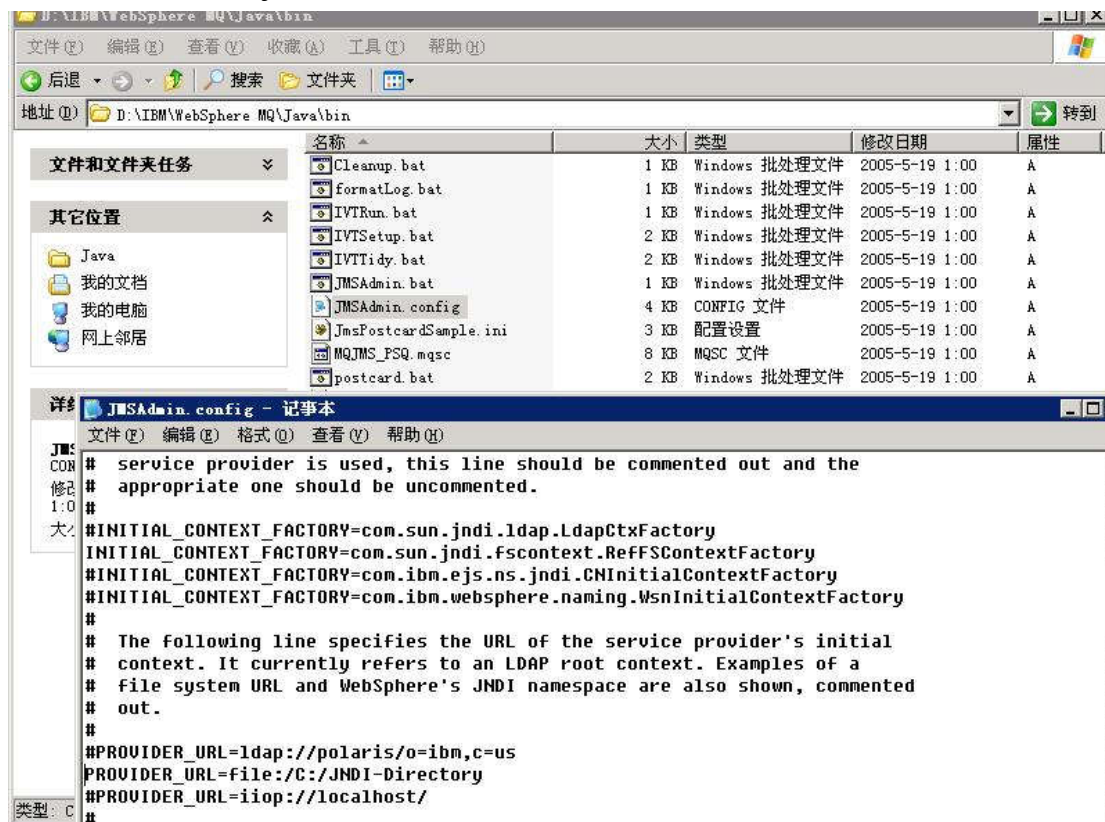
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>cd D:\IBM\WebSphere MQ\bin

C:\Documents and Settings\Administrator>d:
D:\IBM\WebSphere MQ\bin>runmqsc.exe QM_APPLE < "D:/IBM/WebSphere MQ/Java/bin/MQJMS_PSQ.mqsc" >log.txt

D:\IBM\WebSphere MQ\bin>
```

进行 JMS 以及 for java 的设置



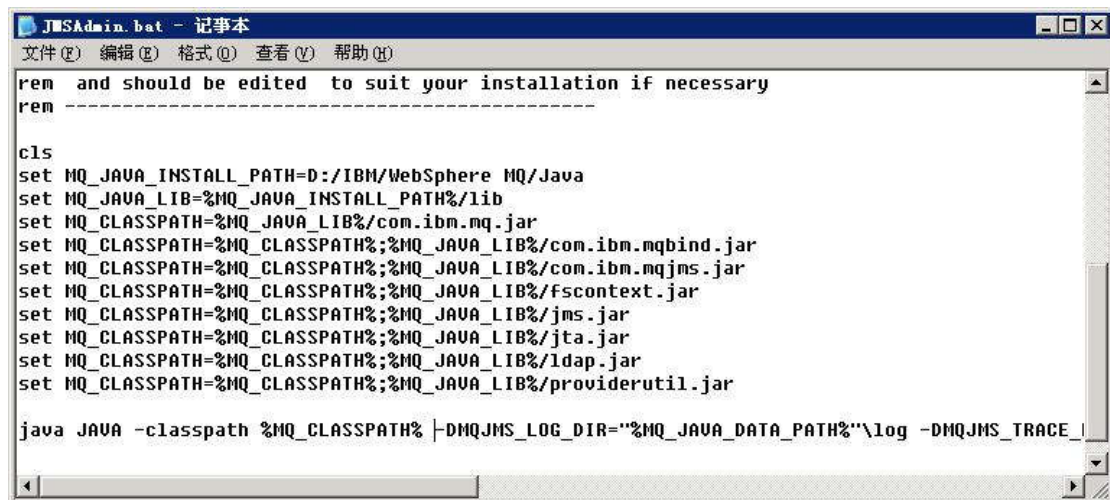
| 名称                    | 大小   | 类型            | 修改日期           | 属性 |
|-----------------------|------|---------------|----------------|----|
| Cleanup.bat           | 1 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |
| formatLog.bat         | 1 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |
| IVTRun.bat            | 1 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |
| IVTSetup.bat          | 2 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |
| IVTTidy.bat           | 2 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |
| JMSAdmin.config       | 4 KB | CONFIG 文件     | 2005-5-19 1:00 | A  |
| JmsPostcardSample.ini | 3 KB | 配置设置          | 2005-5-19 1:00 | A  |
| MQJMS_PSQ.mqsc        | 8 KB | MQSC 文件       | 2005-5-19 1:00 | A  |
| postcard.bat          | 2 KB | Windows 批处理文件 | 2005-5-19 1:00 | A  |

```
JMSAdmin.config - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

JMS
COR
修改
1:0
大小:

# service provider is used, this line should be commented out and the
# appropriate one should be uncommented.
#
#INITIAL_CONTEXT_FACTORY=com.sun.jndi ldap.LdapCtxFactory
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.ReffFSContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory
#
# The following line specifies the URL of the service provider's initial
# context. It currently refers to an LDAP root context. Examples of a
# file system URL and WebSphere's JNDI namespace are also shown, commented
# out.
#
#PROVIDER_URL=ldap://polaris/o=ibm,c=us
PROVIDER_URL=file:/C:/JNDI-Directory
#PROVIDER_URL=iiop://localhost/
#
```

修改 JMSAdmin.config 中 INITIAL\_CONTEXT\_FACTORY 以及 PROVIDER\_URL

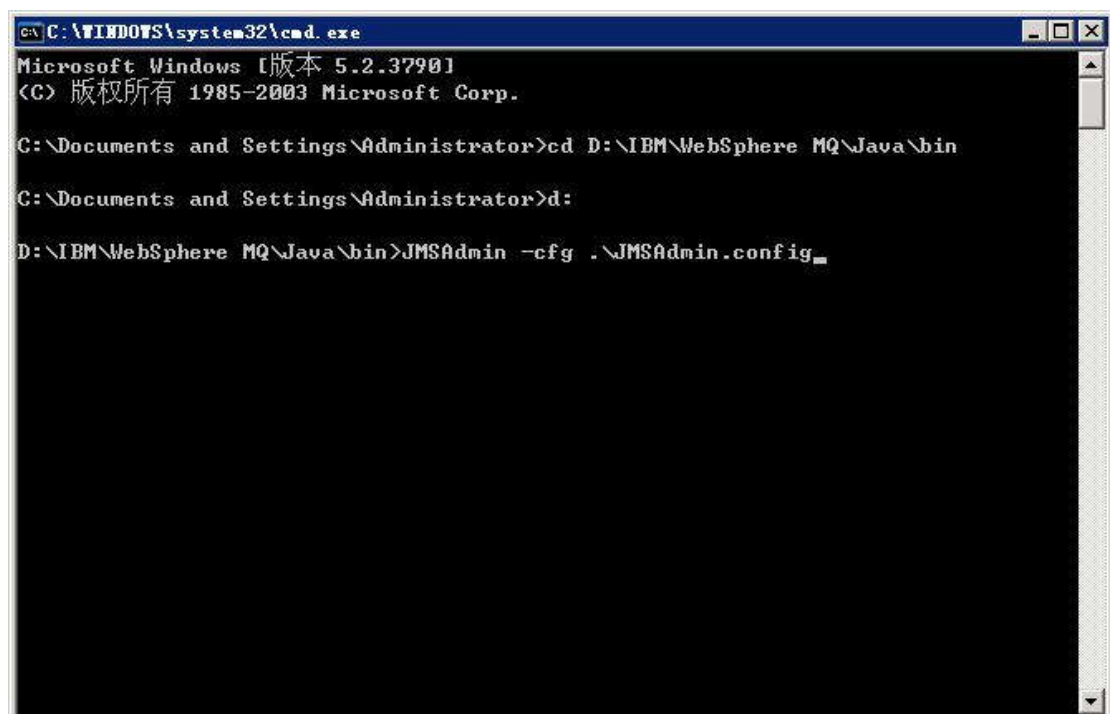


```
rem and should be edited to suit your installation if necessary
rem -----

cls
set MQ_JAVA_INSTALL_PATH=D:/IBM/WebSphere MQ/Java
set MQ_JAVA_LIB=%MQ_JAVA_INSTALL_PATH%/lib
set MQ_CLASSPATH=%MQ_JAVA_LIB%/com.ibm.mq.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/com.ibm.mqbind.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/com.ibm.mqjms.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/fscontext.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/jms.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/jta.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/ldap.jar
set MQ_CLASSPATH=%MQ_CLASSPATH%;%MQ_JAVA_LIB%/providerutil.jar

java JAVA -classpath %MQ_CLASSPATH% | -DMQJMS_LOG_DIR="%MQ_JAVA_DATA_PATH%\log -DMQJMS_TRACE'
```

修改 JMSAdmin.bat 配置好 lib 包，用来配置 jndi 用



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>cd D:\IBM\WebSphere MQ\Java\bin

C:\Documents and Settings\Administrator>d:

D:\IBM\WebSphere MQ\Java\bin>JMSAdmin -cfg .\JMSAdmin.config_
```

执行配置 jndi

```
C:\WINDOWS\system32\cmd.exe - JMSADMIN -cfg .\JMSAdmin.config
正在启动用于 Java(tm) 消息服务的 Websphere MQ 类管理

InitCtx> DEFINE XAQCF(bridge.mqQCFXA)

InitCtx> DEFINE QCF(bridge.mqQCF)

InitCtx> DEFINE Q(bridge.mqSendQueue) QUEUE(Q1)

InitCtx> DEFINE Q(bridge.mqReceiveQueue) QUEUE(Q1)

InitCtx> display ctx

内容是 InitCtx

      .bindings
a  bridge.mqReceiveQueue      com.ibm.mq.jms.MQQueue
a  bridge.mqSendQueue        com.ibm.mq.jms.MQQueue
a  bridge.mqQCF              com.ibm.mq.jms.MQQueueConnectionFactory
a  bridge.mqQCFXA            com.ibm.mq.jms.MQXAQueueConnectionFactory

5 对象
  0 上下文
  5 绑定, 4 管理

InitCtx>
```

以上为简单配置命令，详细输入命令请查看 命令文档，

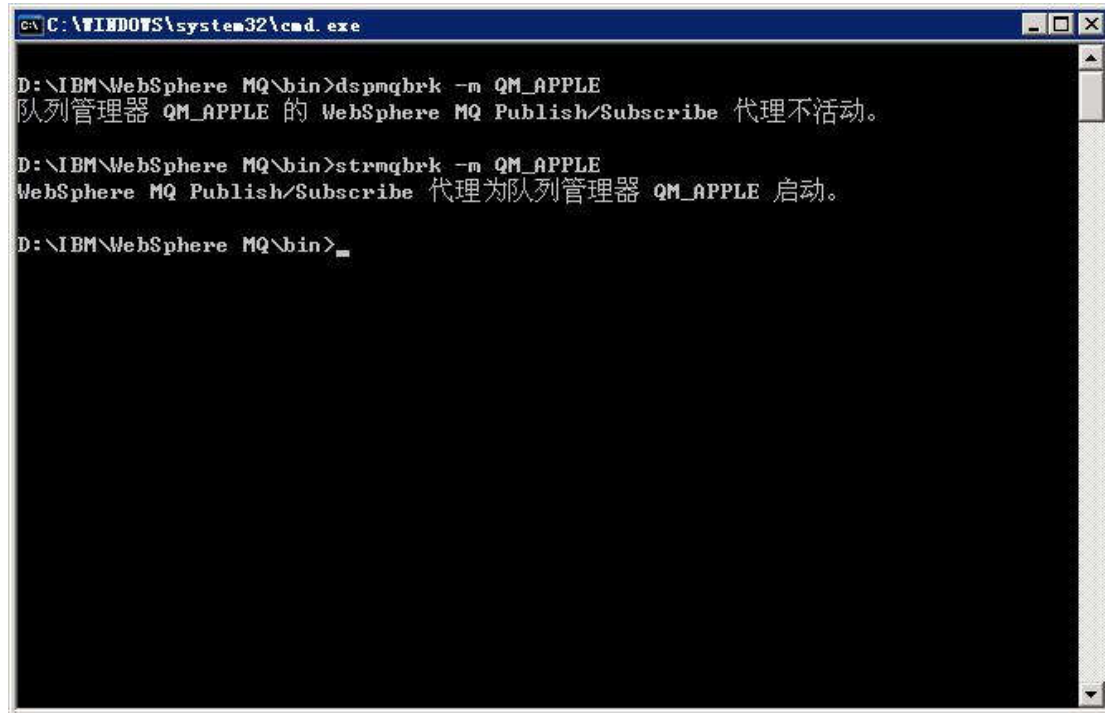
```
C:\WINDOWS\system32\cmd.exe - runmqsc QM_APPLE

C:\Documents and Settings\Administrator>runmqsc QM_APPLE
5724-H72 (C) Copyright IBM Corp. 1994, 2005. ALL RIGHTS RESERVED.
启动队列管理器 QM_APPLE 的 MQSC。

display QMGR CCSID
      1 : display QMGR CCSID
AMQ8408: 显示队列管理器的细节。
      QMNAME(QM_APPLE)                      CCSID(1381)
```

查看当前字符集，此字符集可在安装的时候生成，或安装重新设定





```
C:\WINDOWS\system32\cmd.exe

D:\IBM\WebSphere MQ\bin>dspmqbrk -m QM_APPLE
队列管理器 QM_APPLE 的 WebSphere MQ Publish/Subscribe 代理不活动。

D:\IBM\WebSphere MQ\bin>strmqbrk -m QM_APPLE
WebSphere MQ Publish/Subscribe 代理为队列管理器 QM_APPLE 启动。

D:\IBM\WebSphere MQ\bin>
```

启动 publish/subscribe 代理

## 2. 补充和关注点

1. 安装时候点 setup 出错 AMQ4739, 请用

<Installable\_Path>\setup.exe -a AMQSERVICESHIPLEVEL="2.1"

## 三、 WebSphere MQ for java 的简单例子

### 1..需要的 java 包

Com.ibm.mq.jar --MQ base

Com.ibm.mqjms.jar --MQ JMS

CL3Export.jar --publish/subscribe

CL3Nonexport.jar

dhbcore.jar

Rmm.jar

Connector.jar (version 1.0)

Fscontext.jar (version 1.2)

Jms.jar (version 1.1)

Jndi.jar (version 1.2.1)

Jta.jar (version 1.0.1)

Ldap.jar (version 1.2.2)

Providerutil.jar (version 1.2)

### 2.调用的两种方式

通过调用 MQI 的方式 TCP/IP 连接 作为 MQ 的 client

通过调用通用接口 JMS 的方式 连接 JNDI 到 MQ

简单的代码 (采用 IBM 提供的 java 例子)

### **JmsUtils.java**

```
package com.ibm.examples;

import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class JmsUtils {
    private static final String CF_CLASS_NAME =
        "com.ibm.mq.jms.context.WMQInitialContextFactory";
    private static final String WMQ_URL = "localhost:1414/SYSTEM.DEF.SVRCONN";

    protected Context getInitialContext() throws NamingException {
        Properties props = new Properties();
        props.put(Context.INITIAL_CONTEXT_FACTORY, CF_CLASS_NAME);
        props.put(Context.PROVIDER_URL, WMQ_URL);
        return new InitialContext(props);
    }
}
```

### **P2Utils.java**

```
package com.ibm.examples;

import javax.jms.JMSException;
import javax.jms.Queue;
import javax.jms.QueueConnection;
import javax.jms.QueueConnectionFactory;
import javax.naming.Context;
import javax.naming.NamingException;

public class P2Utils extends JmsUtils {

    private static final String QCF_NAME = "SampleQCF";
    private static final String QUEUE_NAME = "SampleQueue";

    public QueueConnection getConnection()
        throws NamingException, JMSException {
        Context context = getInitialContext();
```



```

        QueueConnectionFactory qcf =
            (QueueConnectionFactory) context.lookup(QCF_NAME);
        return qcf.createQueueConnection();
    }

    public Queue getQueue() throws NamingException {
        Context context = getInitialContext();
        return (Queue) context.lookup(QUEUE_NAME);
    }
}

```

### **PubSubUtils.java**

```

package com.ibm.examples;

import javax.jms.JMSEException;
import javax.jms.Topic;
import javax.jms.TopicConnection;
import javax.jms.TopicConnectionFactory;
import javax.naming.Context;
import javax.naming.NamingException;

public class PubSubUtils extends JmsUtils {

    private static final String TCF_NAME = "SampleTCF";
    private static final String TOPIC_NAME = "SampleTopic";

    public TopicConnection getConnection()
        throws NamingException, JMSEException {
        Context context = getInitialContext();
        TopicConnectionFactory qcf =
            (TopicConnectionFactory) context.lookup(TCF_NAME);
        return qcf.createTopicConnection();
    }

    public Topic getTopic() throws NamingException {
        Context context = getInitialContext();
        return (Topic) context.lookup(TOPIC_NAME);
    }
}

```

### **StandalonePublisher.java**

```

package com.ibm.examples;

```

```

import java.io.IOException;

import javax.jms.JMSEException;
import javax.jms.Message;
import javax.jms.Session;
import javax.jms.TopicConnection;
import javax.jms.TopicPublisher;
import javax.jms.TopicSession;
import javax.naming.NamingException;

public class StandalonePublisher {

    private PubSubUtils utils;
    private TopicConnection connection;
    private TopicSession session;
    private TopicPublisher publisher;

    public static void main(String[] args)
        throws NamingException, JMSEException, IOException {
        StandalonePublisher publisher = new StandalonePublisher();
        publisher.connect();
        String message = "ignored";
        while (message.length() > 0) {
            byte[] input = new byte[40];
            System.out.print("Enter a message: ");
            System.in.read(input);
            message = (new String(input, 0, input.length)).trim();
            if (message.length() > 0)
                publisher.sendMessage(message);
        }
        publisher.disconnect();
    }

    private StandalonePublisher() {
        utils = new PubSubUtils();
    }

    private void connect() throws NamingException, JMSEException {
        connection = utils.getConnection();
        session =
            connection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
        publisher = session.createPublisher(utils.getTopic());
        System.out.println("Publisher started.");
    }
}

```

```

    }

    private void sendMessage(String text) throws JMSEException {
        Message message = session.createTextMessage(text);
        publisher.publish(message);
        System.out.println(
            "Published message <"
                + text
                + "> with ID <"
                + message.getJMSMessageID()
                + ">");
    }

    private void disconnect() throws JMSEException {
        publisher.close();
        session.close();
        connection.close();
        System.out.println("Publisher stopped.");
    }
}

```

### **StandaloneReceiver.java**

```

package com.ibm.examples;

import java.io.IOException;

import javax.jms.JMSEException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.QueueConnection;
import javax.jms.QueueReceiver;
import javax.jms.QueueSession;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.naming.NamingException;

public class StandaloneReceiver implements MessageListener {

    private P2Utils utils;
    private QueueConnection connection;
    private QueueSession session;
    private QueueReceiver receiver;

```

```

public static void main(String[] args)
    throws NamingException, JMSEException, IOException {
    StandaloneReceiver receiver = new StandaloneReceiver();
    receiver.connect();
    System.in.read();
    receiver.disconnect();
}

private StandaloneReceiver() {
    utils = new P2PUtils();
}

private void connect() throws NamingException, JMSEException {
    connection = utils.getConnection();
    session =
        connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
    receiver = session.createReceiver(utils.getQueue());
    receiver.setMessageListener(this);
    connection.start();
    System.out.println("Receiver started.");
}

public void onMessage(Message message) {
    try {
        TextMessage tMessage = (TextMessage) message;
        String text;
        text = tMessage.getText();
        System.out.println(
            "Received message <"
                + text
                + "> with ID <"
                + message.getJMSMessageID()
                + ">");
    } catch (JMSEException e) {
        e.printStackTrace();
    }
}

private void disconnect() throws JMSEException {
    receiver.close();
    session.close();
    connection.stop();
    connection.close();
    System.out.println("Receiver stopped.");
}

```

```
    }  
}
```

### **StandaloneSender.java**

```
package com.ibm.examples;  
  
import java.io.IOException;  
  
import javax.jms.JMSException;  
import javax.jms.Message;  
import javax.jms.QueueConnection;  
import javax.jms.QueueSender;  
import javax.jms.QueueSession;  
import javax.jms.Session;  
import javax.naming.NamingException;  
  
public class StandaloneSender {  
  
    private P2PUtils utils;  
    private QueueConnection connection;  
    private QueueSession session;  
    private QueueSender sender;  
  
    public static void main(String[] args)  
        throws NamingException, JMSException, IOException {  
        StandaloneSender sender = new StandaloneSender();  
        sender.connect();  
        String message = "ignored";  
        while (message.length() > 0) {  
            byte[] input = new byte[40];  
            System.out.print("Enter a message: ");  
            System.in.read(input);  
            message = (new String(input, 0, input.length)).trim();  
            if (message.length() > 0)  
                sender.sendMessage(message);  
        }  
        sender.disconnect();  
    }  
  
    private StandaloneSender() {  
        utils = new P2PUtils();  
    }  
}
```

```

private void connect() throws NamingException, JMSEException {
    connection = utils.getConnection();
    session =
        connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
    sender = session.createSender(utils.getQueue());
    System.out.println("Sender started.");
}

private void sendMessage(String text) throws JMSEException {
    Message message = session.createTextMessage(text);
    sender.send(message);
    System.out.println(
        "Sent message <"
        + text
        + "> with ID <"
        + message.getJMSMessageID()
        + ">");
}

private void disconnect() throws JMSEException {
    sender.close();
    session.close();
    connection.close();
    System.out.println("Sender stopped.");
}
}

```

### **StandaloneSubscriber.java**

```

package com.ibm.examples;

import java.io.IOException;

import javax.jms.JMSEException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.jms.TopicConnection;
import javax.jms.TopicSession;
import javax.jms.TopicSubscriber;
import javax.naming.NamingException;

public class StandaloneSubscriber implements MessageListener {

```

```

private PubSubUtils utils;
private TopicConnection connection;
private TopicSession session;
private TopicSubscriber subscriber;

public static void main(String[] args)
    throws NamingException, JMSEException, IOException {
    StandaloneSubscriber subscriber = new StandaloneSubscriber();
    subscriber.connect();
    System.in.read();
    subscriber.disconnect();
}

private StandaloneSubscriber() {
    utils = new PubSubUtils();
}

private void connect() throws NamingException, JMSEException {
    connection = utils.getConnection();
    session =
        connection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
    subscriber = session.createSubscriber(utils.getTopic());
    subscriber.setMessageListener(this);
    connection.start();
    System.out.println("Subscriber started.");
}

public void onMessage(Message message) {
    try {
        TextMessage tMessage = (TextMessage) message;
        String text;
        text = tMessage.getText();
        System.out.println(
            "Received message <"
                + text
                + "> with ID <"
                + message.getJMSMessageID()
                + ">");
    } catch (JMSEException e) {
        e.printStackTrace();
    }
}

```

```
private void disconnect() throws JMSEException {  
    subscriber.close();  
    session.close();  
    connection.stop();  
    connection.close();  
    System.out.println("Subscriber stopped.");  
}  
}
```