使用 JMS 開發 IBM Lotus Domino 代理程式

級別：中等

IBM 應用和整合中介軟體架構設計師 William Griffith (wgriffith@us.ibm.com)

2008 年 1 月 29 日

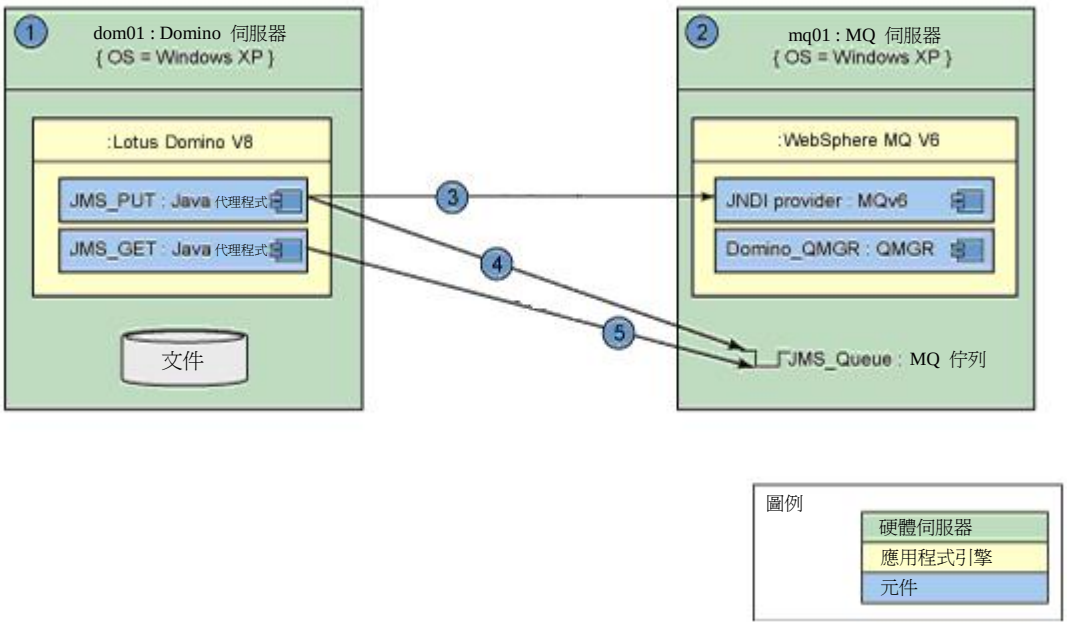
本文說明如何在 IBM Lotus Domino 中開發 Java 代理程式，以利用 Java 訊息系統 (Java Messaging System, JMS) 提供者（特別是 IBM WebSphere MQ）傳送及接收訊息。
許多企業現有的 IBM Lotus Domino 應用程式中都有其他應用程式所需的大量資訊和資料。本文說明如何在執行 IBM Lotus Domino Designer 的 Lotus Domino 伺服器上開發 Java 代理程式。這個代理程式使用 Java 訊息系統 (JMS) 1.1 API 以及 Java 命名與目錄介面 (Java Naming and Directory Interface, JNDI) API 來傳送訊息給 JMS 提供者，並從 JMS 提供者接收訊息。本文使用 IBM WebSphere MQ V6 作爲傳訊提供者，但是透過使用 JMS 1.1，這個程式碼可讓 Lotus Domino 應用程式連接任何支援 JMS 的企業服務匯流排 (ESB)。（這個程式碼是使用 Lotus Domino V8、WebSphere MQ V6 以及 JMS 1.1 API 進行測試的；但是這個程式碼應該可以和 Lotus Domino V5 及更新版本和 JMS 1.0 及更新版本一起搭配運作。）

本文是專爲 Lotus Domino 程式設計師所撰寫，設計師可利用 Lotus Domino 提供的 Java API 支援來和傳訊系統交互運作。本文假設您熟悉 Java 程式設計語言以及 Lotus Domino 程式設計模型。

建議架構

在我們探討解決方案之前，讓我們先全面瞭解一下我們所要達成的目標。圖 1 是一個統一塑模語言 (UML) 部署圖，可爲此範例應用程式的運作方式提供部分環境定義。如圖所示，此應用程式涉及兩個實際上各自獨立的伺服器：一部 Lotus Domino V8 伺服器和一部 WebSphere MQ V6 伺服器。您當然可以在同一部伺服器上執行該應用程式，但是以分散式網路情境實例來進行說明，可以提供較適當的解決方案。此外，圖 1 顯示，無須其他軟體，WebSphere MQ 伺服器也可以提供 JNDI 與 JMS 提供者支援。

圖 1. 架構部署圖



爲何使用此架構？

當然，從 Lotus Domino 取得資料，然後傳送至其他應用程式的方法有很多，所以爲什麼要使用傳訊，而且更明確一點地說，爲什麼要使用 JMS？傳訊可提供非同步通訊管道，讓 Lotus Domino 及所有 JMS 提供者以鬆散耦合的方式共用資料（也就是說，產生者或消費者無須知道彼此）。此外，透過使用 JMS API，這個 Java 程式碼不但可讓 Lotus Domino 和任何 JMS 訊息提供者交互運作，還可以連接至許多企業服務匯流排。

本文概觀

本文中的步驟和您可以建構此範例應用程式的順序類似。這些步驟和順序如下：

1. **配置 WebSphere MQ**。
   本文是與 Lotus Domino 與 JMS 提供者的連接有關，因此您必須安裝並配置一個 JMS 提供者。我們使用 WebSphere MQ V6 是因爲它支援 JMS 1.1，而且也是極受歡迎的訊息導向中介軟體 (Message Oriented Middleware, MOM)。因爲 JMS 需要一個 JNDI 提供者，所以我們也使用 WebSphere MQ 作爲 JNDI 提供者。這需要一個 supportPac，然後建立和 MQ 基礎實體連結的 JNDI 管理物件 (JNDI administered objects)。
2. **配置 Lotus Domino**。
   爲了說明 Lotus Domino 至 JMS 的概念，我們開發了一個簡單的應用程式，

其中包含表單、視圖及 Java 代理程式。在本節中，我們會說明 Lotus Notes 應用程式的基本觀念，並討論如何在 Lotus Domino Designer 中建構 Java 代理程式。最後，我們會討論 Java 程式碼及其運作方式。

3. **執行範例**。
   爲了說明該程式碼，我們會執行代理程式將訊息推送至 WebSphere MQ，接著再顯示訊息出現在 WebSphere MQ 中的情形。 然後我們會執行另一個代理程式，將訊息從 WebSphere MQ 提取回來再推送至 Lotus Domino，藉此說明我們可以支援傳訊的產生者端和消費者端。最後，我們會在 Lotus Notes 用戶端中，將擷取到的訊息開啓爲 Notes 文件，以顯示訊息內文。

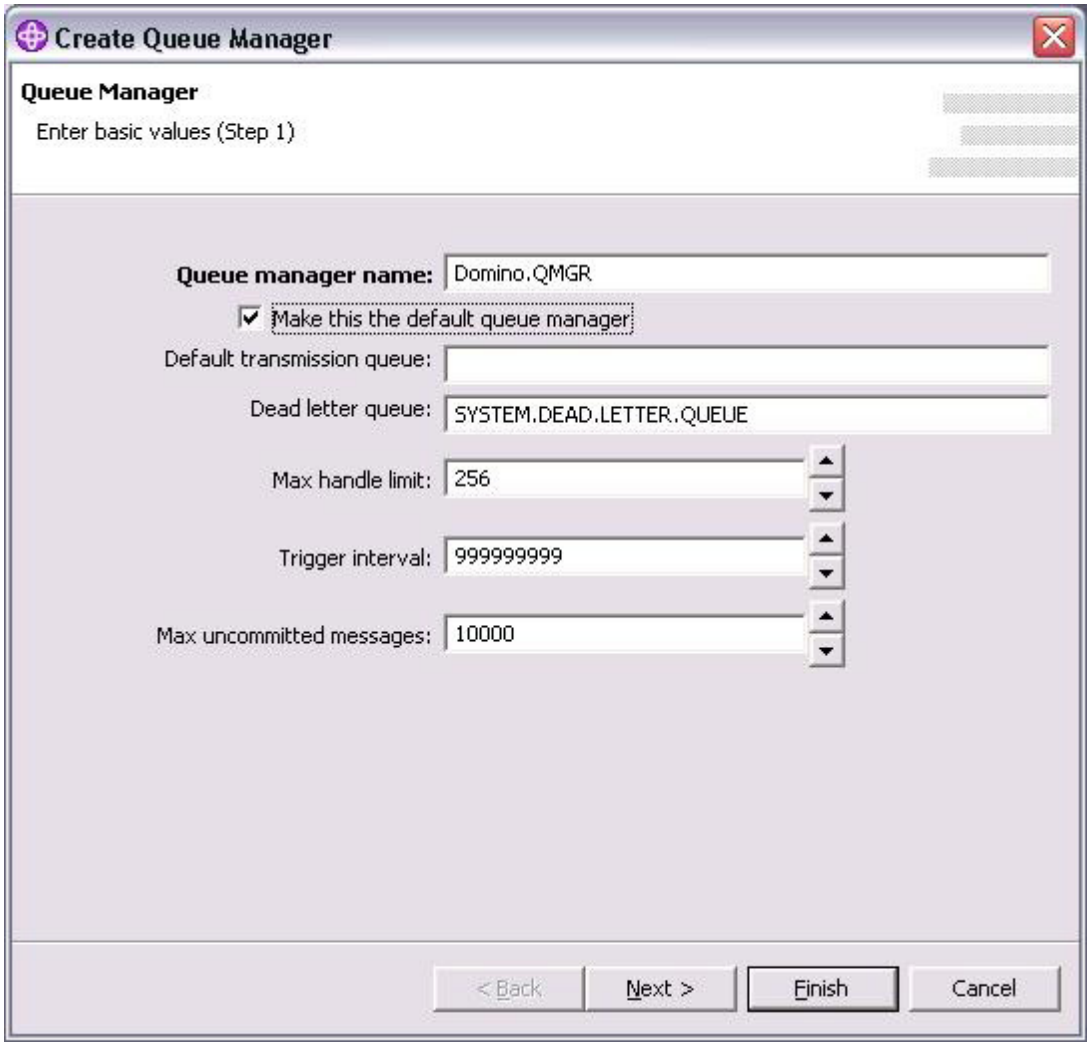
配置 WebSphere MQ

WebSphere MQ V6 支援 JMS 1.1。這代表 WebSphere MQ V6 可以像 JMS 1.1 提供者般運作。請記得，JMS 只是一個 API 規範；傳訊系統是藉由實作 JMS 規範來提供傳訊。

爲了讓 JMS 用戶端連接至 WebSphere MQ，您必須建立並配置一個佇列管理程式。佇列管理程式可以提供執行時期程序，進而提供 JMS 功能給 JMS 用戶端。WebSphere MQ 會提供 Eclipse 管理用戶端，以在此程序中提供協助。

1. 啓動 WebSphere MQ 探險家，方法是選擇開始 - 所有程式 - IBM Websphere MQ - WebSphere MQ 探險家。
2. 在 WebSphere MQ 探險家導覽器中選取 Queue Manager 資料夾，然後按一下滑鼠右鍵選取 New - Queue Manager。
3. 在精靈中，輸入 Domino.QMGR 作爲佇列管理程式的名稱。選取 "Make this the default queue manager" 選項。在 Dead letter queue 欄位中輸入 SYSTEM.DEAD.LETTER.QUEUE。保留其他欄位的預設值，然後按一下 Finish，如圖 2 所示。

圖 2. 佇列管理程式配置



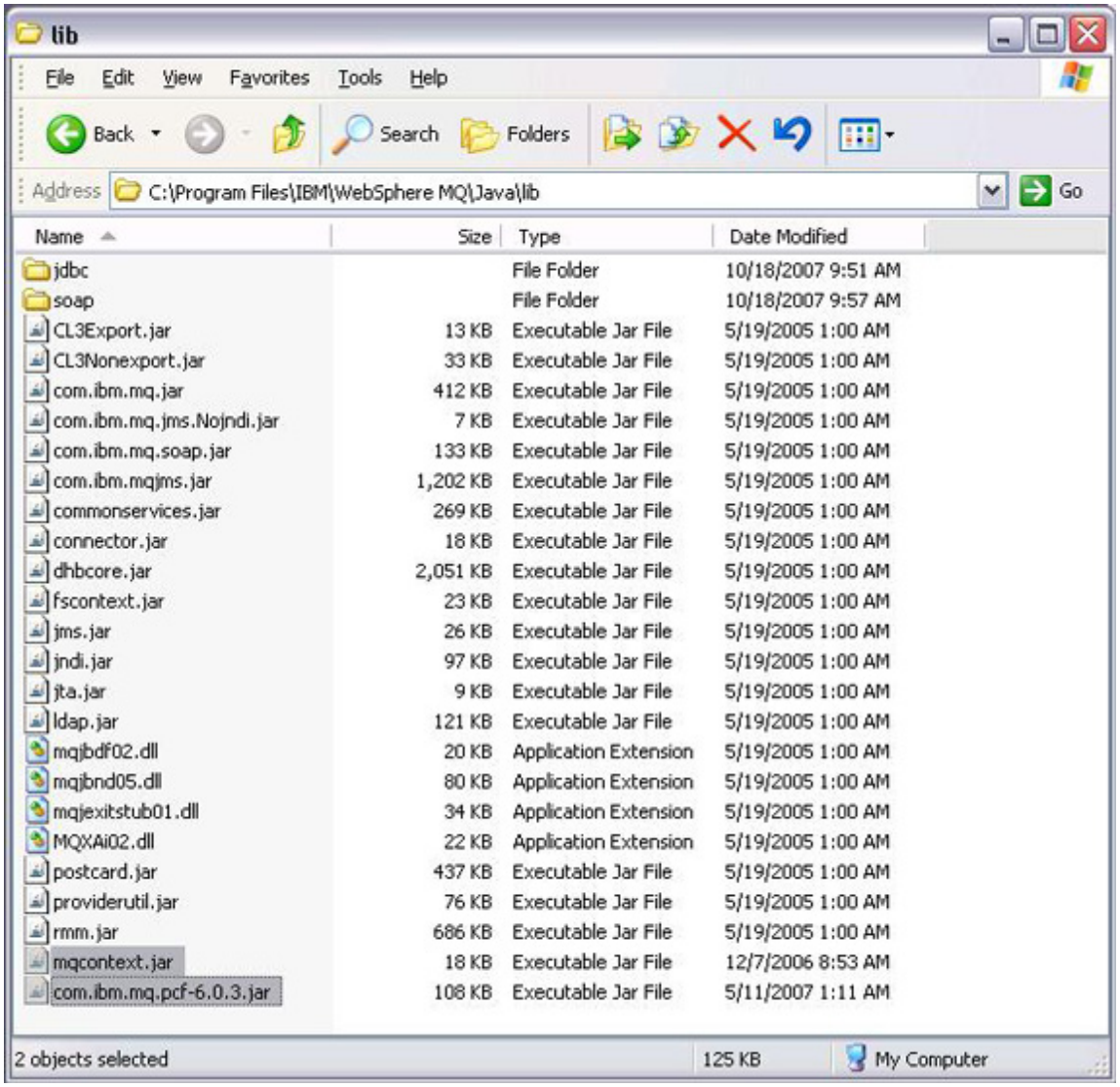
此時，您已在 Websphere MQ 中建立了一個可以儲存訊息的佇列管理程式，但是 JMS 用戶端也需要一個 JNDI 提供者。

配置 JNDI

JNDI 只是一個規範，因此需要一個提供者來提供實作。IBM WebSphere MQ V6 可以支援 supportPac ME01 來提供此實作，但要支援 supportPac ME01 則需要 supportPac MS0B（請參閱「資源」一節）。這些 supportPac 非常容易安裝：將它們解壓縮，然後將隨附的 JAR 檔複製到 <MQ_Install>\Java\lib 即可。圖 3 顯示已解壓縮並新增至 <MQ_Install>\Java\lib 的 JAR，即 mqcontext.jar 和 com.ibm.mq.pcf-6.0.3.jar。

透過使用 WebSphere MQ 做爲我們的 JNDI 提供者，就不需要使用個別的 JNDI 提供者來將 JMS 管理物件對映至實體的 Websphere MQ 佇列 – supportPac ME01 即可提供此功能。

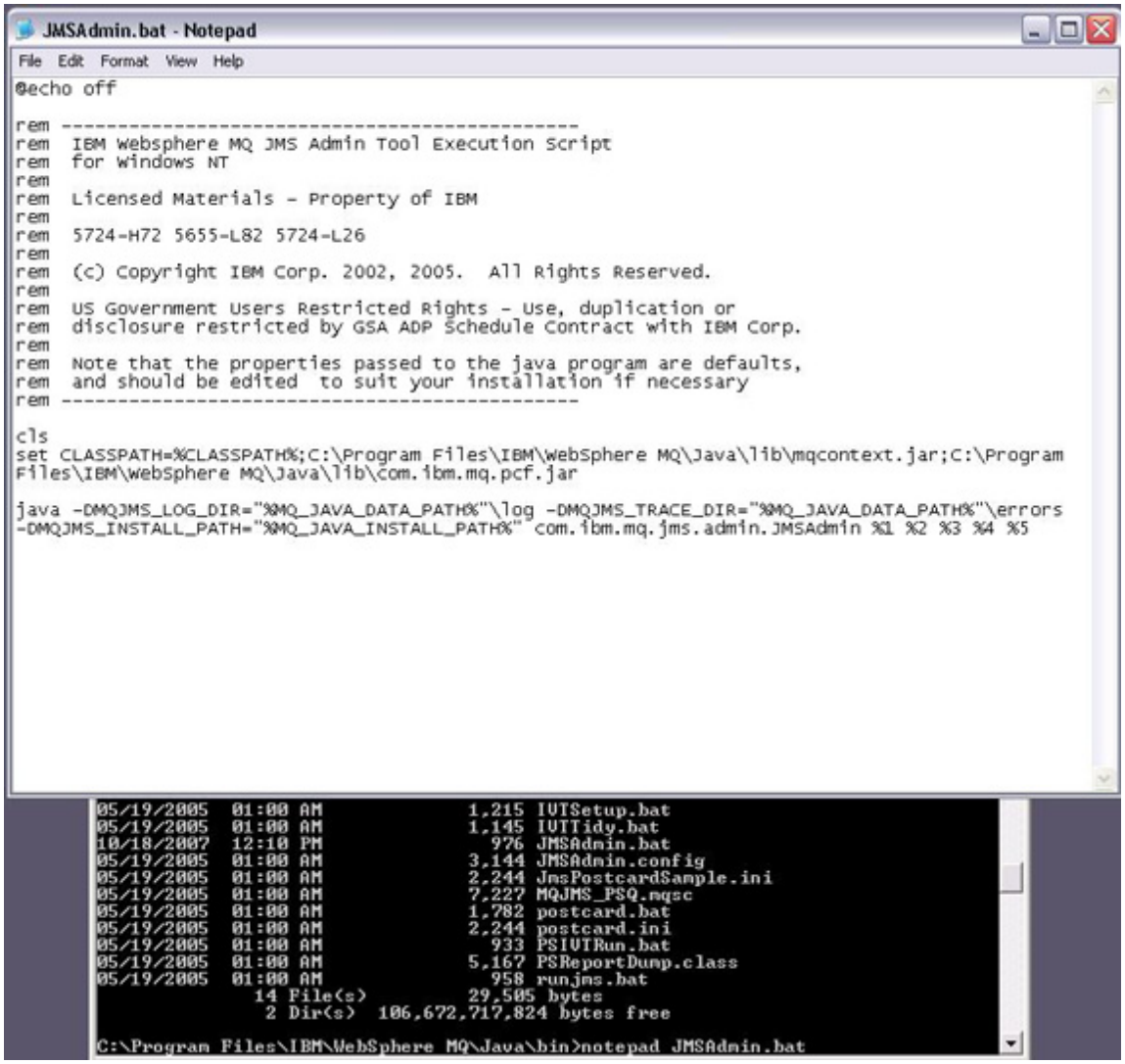
圖 3. JNDI supportPacs



JMSAdmin 工具是一個指令行互動式工具，隨附於 WebSphere MQ，可讓您在 JNDI 提供者（在此案例中爲 WebSphere MQ）中建立 JNDI 管理物件。爲了讓 JMSAdmin 工具能在 WebSphere MQ 中建立 JNDI 物件，您必須更新 CLASSPATH 以包含 supportPacs。使用文字編輯器開啓批次 Script (C:\Program Files\IBM\Websphere MQ\Java\bin\JMSAdmin.bat)，然後將 JAR 檔新增至 CLASSPATH 環境變數中，如圖 4 所示。

set CLASSPATH=%CLASSPATH%;C:\Program Files\IBM\Websphere MQ\Java\lib\mqcontext.jar;
C:\Program Files\IBM\Websphere MQ\Java\lib\com.ibm.mq.pcf.jar

圖 4. JMSAdmin.bat CLASSPATH 更新



使用 supportPacs 更新 JMSAdmin 工具之後，您便可以連接至您的 JNDI 提供者，以建立 JMS 管理物件。這是透過在 JMSAdmin 工具的配置檔 (JMSAdmin.config) 中設定連線屬性來完成。

您可以使用 WebSphere MQ 作爲 JNDI 提供者的 # JNDI 配置設定
INITIAL_CONTEXT_FACTORY=com.ibm.mq.jms.context.WMQInitialContextFactory
PROVIDER_URL=localhost:1414/SYSTEM.DEF.SVRCONN
因爲您是從 WebSphere MQ 伺服器執行 JMSAdmin 工具，所以請將 PROVIDER_URL 設定爲本端主機 (localhost)。如圖 1 中所描述，Lotus Domino Java 代理程式會遠端連接至 JNDI 提供者，因此會使用遠端伺服器的 DNS 名稱，而不是使用本端主機。

此時，您可以透過以下指令行啓動 JMSAdmin 工具：

C:\Program Files\IBM\Websphere MQ\Java\bin\JMSAdmin.bat

建立 JMS 管理物件

JMS 規範使用 ConnectionFactory 來建立 JMS 目的地（即佇列和主題）。因此，您必須建立的第一個 JNDI 管理物件就是 ConnectionFactory。請在 JMSAdmin 互動式工具中使用下列指令。（JMSAdmin 工具支援使用 Script 來代替互動式模式，但在本文中我們會使用互動式介面。）

```
def cf(JMS_ConnectionFactory) qmgr(Domino.QMGR) tran(client)
chan(SYSTEM.DEF.SVRCONN)
host(bpte-demo-8.austin.ibm.com) port(1414)
```

表 1 指令分解

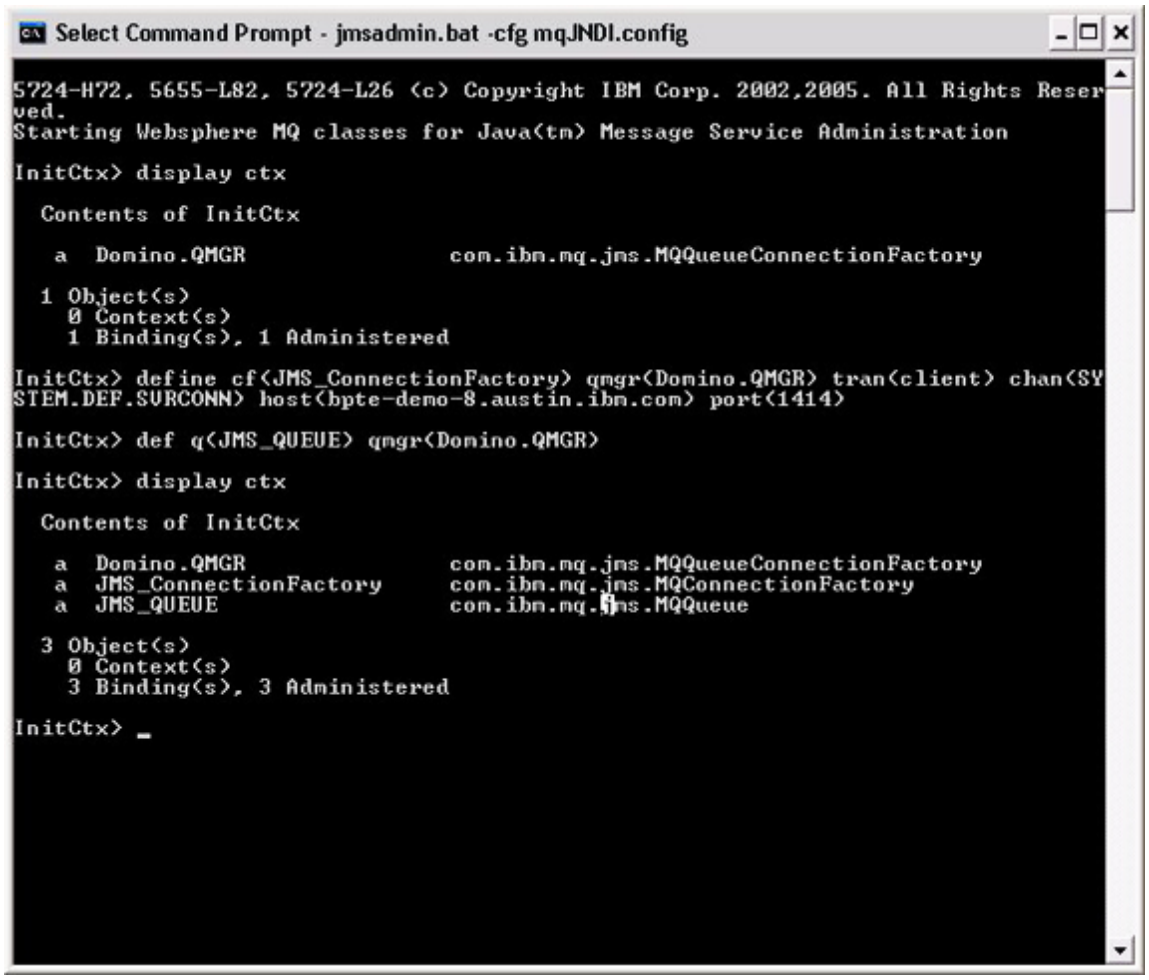
關鍵字	說明
Def	定義 – 用來定義新的 JNDI 管理物件。
Cf	ConnectionFactory – JMS 1.1 可以使用 ConnectionFactory 來建立佇列或主題。
qmgr	佇列管理程式 – 這會建立 ConnectionFactory 尋找其目的地佇列和主題的位置。
Tran	傳輸 – WebSphere MQ 可以使用用戶端或連結作爲傳輸。在本文中，我們會使用用戶端模式，以容許輕量型的遠端用戶端。
Chan	通道 – 用戶端必須有一個他們可以連接且接聽 WebSphere MQ 伺服器的通道，以存取佇列管理程式及其資源。
Host	主機 – 因爲您使用的是用戶端模式，所以請指定管理佇列管理程式的伺服器 DNS 名稱。
port	埠 – 因爲您使用的是用戶端模式，所以請指定接聽連線的 TCP/IP 埠。

然後建立保留訊息的佇列，如圖 5 所示。

```
def q(JMS_QUEUE) qmgr(Domino.QMGR)
```

這個指令會建立一個 WebSphere MQ 佇列，該佇列也會被儲存爲一個 JNDI 管理物件。請記住，JNDI 中的這個別名功能是可以變更您的基礎佇列或主題屬性，而不會影響用戶端程式碼的功能。事實上，您可以將該佇列移至一個完全不同的 JMS 提供者，而無須變更用戶端程式碼。

圖 5. JNDI 配置

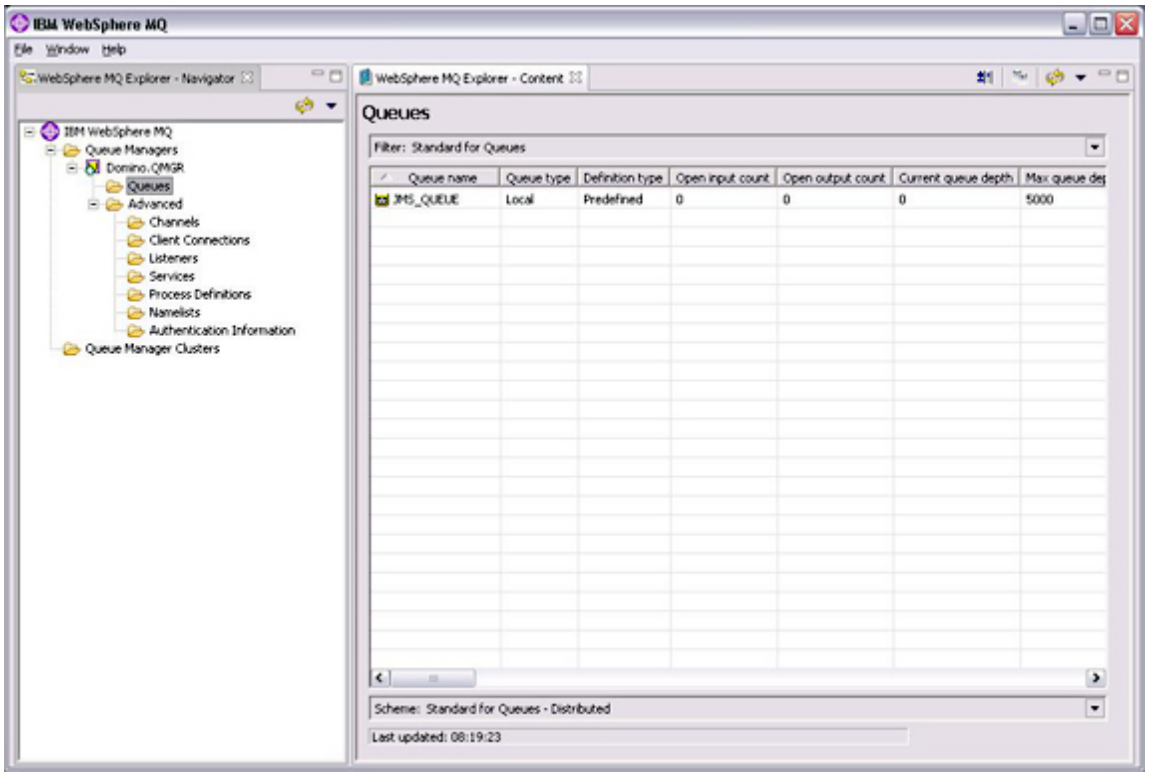


確認指令是否能如預期般運作通常是個好辦法。在 JMSAdmin 工具中，只要執行下列指令即可：

display ctx
這個指令會顯示您已新增至其中的 JNDI 環境定義。

此時，您的 JMS 提供者以及您的 JNDI 提供者都已配置完成。現在您可以前進至連接這些提供者的 JMS 用戶端程式碼，以尋找 JMS 目的地並傳送訊息給它們。同時也請注意，我們不需要使用 WebSphere MQ 探險家來建立佇列；佇列和 JNDI 實體先前已透過 JMSAdmin 工具同時建立。請參閱圖 6。

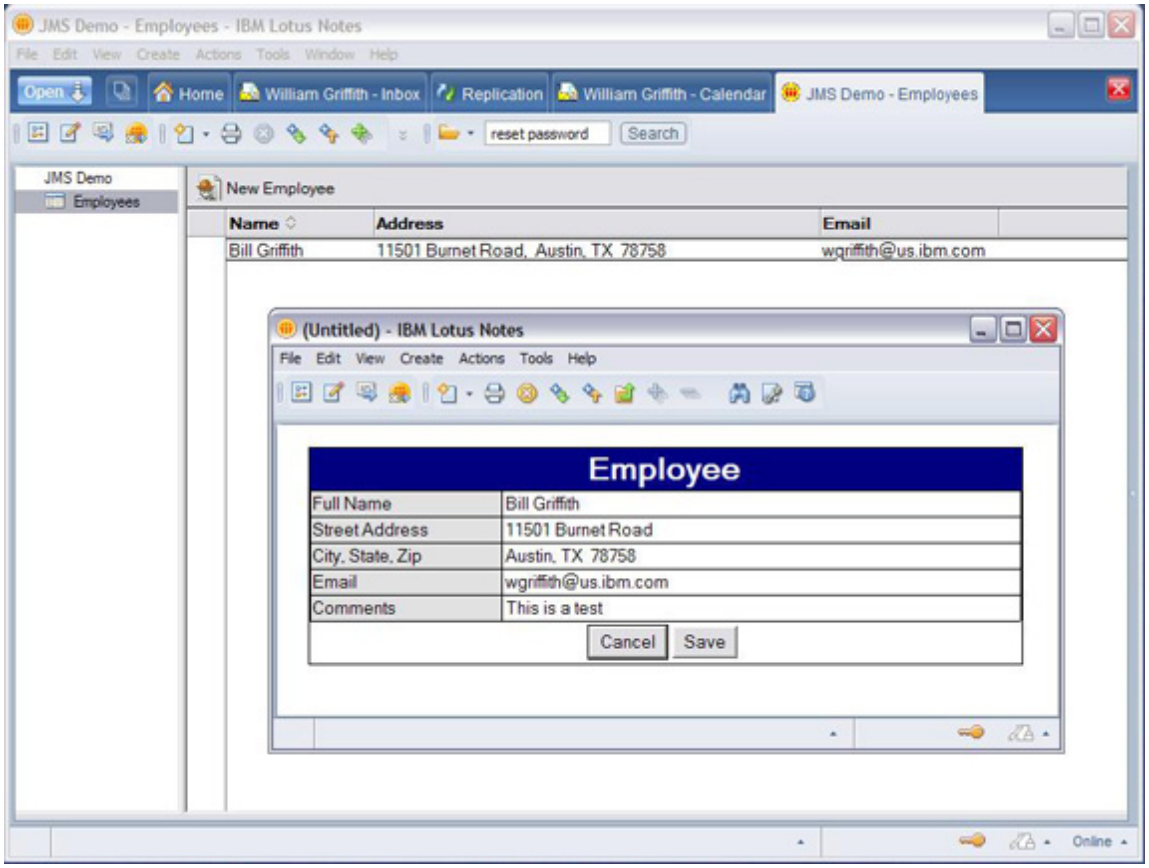
圖 6. WebSphere MQ 中的佇列



建立 Lotus Domino 程式碼

爲了說明這個概念，我們使用 Lotus Domino Designer V8 建立了一個簡單的 Lotus Domino V8 應用程式。這個應用程式包含一個簡單的聯絡資訊表單（請參閱圖 7），以及另一個用來保留擷取到的 JMS 訊息的表單。同樣地，我們也建立了兩個視圖，以分別顯示聯絡人和擷取到的訊息。請參閱本文「下載」一節中的程式碼 JMSDemo.zip 檔案。

圖 7. 範例 Lotus Domino 應用程式



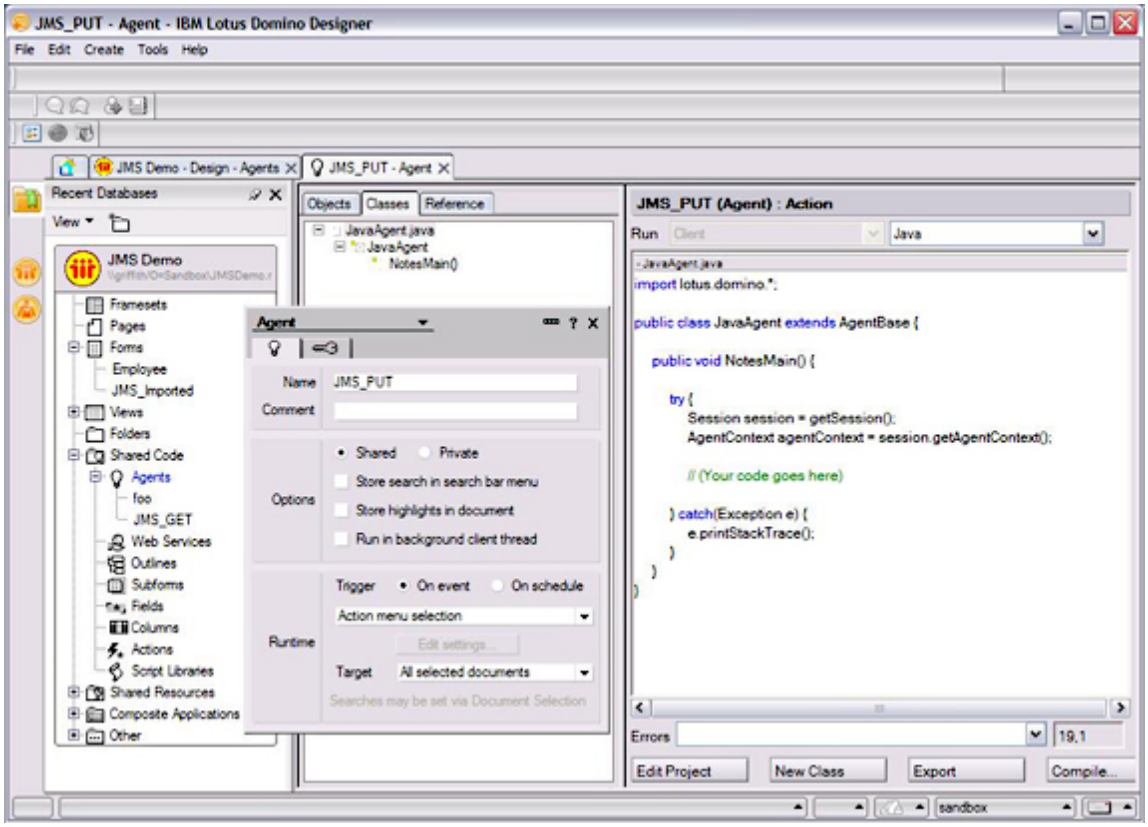
建立 Java 代理程式

範例應用程式中還包含了兩個 Java 代理程式：JMS_PUT 和 JMS_GET。這兩個代理程式都是由使用者所觸發的 Agent 功能表來執行。（您可能會想要不同的觸發策略；Lotus Domino 詳細記錄許多可能辦法。）JMS_PUT 會透過 Lotus Notes 用戶端，傳送訊息至 JMS 提供者，要求提供在 Employees 視圖中選取的所有文件。JMS_GET 代理程式也是從 Lotus Notes 用戶端執行，但它並不需要選取任何文件。這個代理程式實際上會尋找 JMS 提供者上的訊息。它會針對找到的每個訊息，使用該訊息來建立一個 Lotus Domino 文件，其中包含一個 JMS 訊息 ID 欄位，以及一個 JMS 訊息內文欄位。這個範例使用 Lotus Domino 提供的 DxlExporter 類別來將員工文件轉換成 XML 文件。這會用來作爲 JMS 訊息內文。

若要在 Lotus Domino Designer 中建立 Java 代理程式，只要擴充 JMS Demo 應用程式中的 Shared Code 資料夾即可。

1. 選取此資料夾中的 Agents 即可檢視代理程式以及可建立新代理程式的按鈕。
2. 按一下 New Agent 按鈕以建立 Java 代理程式（或開啓 JMSDemo.zip 檔案中的現有程式碼）。
3. 當您在 Lotus Domino Designer 中建立 Java 代理程式時，IDE 會爲您清除 Skeleton，如圖 8 中所描述。

圖 8. 已清除的 Java 代理程式



在介紹程式碼之前，我們必須先將 JMS 提供者實作類別及 API 新增至代理程式。請回想一下，JMS 只是一個規範，因此它會提供一組 JMS 提供者實作的介面。因爲我們是使用 WebSphere MQ 作爲提供者，因此我們會將 WebSphere MQ JAR 檔加入我們的代理程式中。如圖 9 所描述，在 Java 代理程式開啓的情況下，按一下 Edit Project 按鈕開啓對話框（圖 9），以將 JAR 檔新增至代理程式中。（本文爲了簡單起見，我們將 JAR 新增至代理程式中，但您可能會想要將這些 JAR 檔複製到您的 Lotus Domino 伺服器，並在該伺服器 Notes.ini 檔中的 JavaUserClasses 變數之下進行參照。）

圖 9. 新增 JAR 至 Java 代理程式

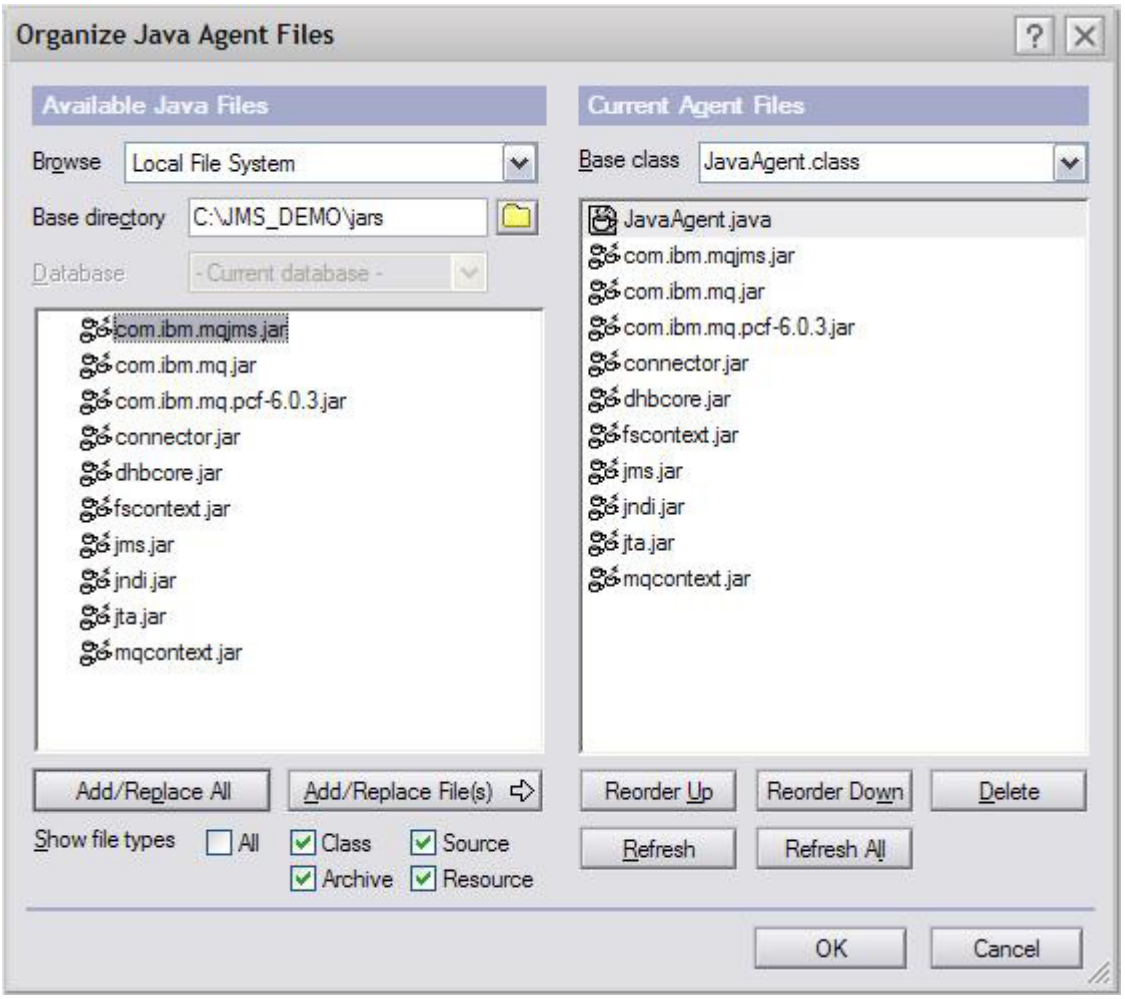


表 2 說明所需的每個 JAR 檔。

表 2 WebSphere MQ JMS 實作 JAR 檔

檔案	說明
com.ibm.mqjms.jar	提供 MQ JMS 實作類別
com.ibm.mq.jar	提供 MQ 實作類別
com.ibm.mq.pcf.jar	爲 PCF 提供 MQ 類別
connector.jar	提供 J2EE 連接器 API
dhbcore.jar	WebSphere MQ 所需
fscontext.jar	提供檔案系統 JNDI 實作
jms.jar	提供 JMS 介面類別
jndi.jar	提供 JNDI 介面類別
jta.jar	提供 JTA 實作
mqcontext.jar	提供 MQ JNDI 實作類別

這些 JAR 是從 C:\Program Files\IBM\Websphere MQ\Java\lib 目錄中的 WebSphere MQ 安裝目錄複製的。

程式碼

現在讓我們來瞭解程式碼，並檢視每個部分的功用。清單 1 顯示一般程式碼，該程式碼是訊息提供者和訊息消費者取得的相同程式碼。請注意程式碼的編號反映了清單 2 和清單 3 是接在清單 1 中的一般程式碼之後。

清單 1. 一般程式碼

```
1    lotus.domino.Session notesSession = getSession();
2    AgentContext agentContext = notesSession.getAgentContext();
3    Database db = agentContext.getCurrentDatabase();

4    Hashtable env = new Hashtable();
5    env.put(Context.INITIAL_CONTEXT_FACTORY,
                om.ibm.mq.jms.context.WMQInitialContextFactory   );
6    env.put(Context.PROVIDER_URL,
        pte-demo-8.austin.ibm.com:1414/SYSTEM.DEF.SVRCONN   );

7    javax.naming.InitialContext ctx = new InitialContext( env );
8    javax.jms.ConnectionFactory qcf = (ConnectionFactory)
     ctx.lookup("JMS_ConnectionFactory");
9    javax.jms.Connection jmsCon = qcf.createConnection();
10   jmsCon.start();
11   javax.jms.Session jmsSess =
       jmsCon.createSession(true,javax.jms.Session.AUTO_ACKNOWLEDGE);

12   javax.jms.Destination jmsDest = (Destination) ctx.lookup("JMS_QUEUE");
```

第 1、2 及 3 行只是要取得 Lotus Domino 代理程式和代理程式執行所在的資料庫的環境定義。第 4、5 及 6 行會設定 JNDI 的屬性，以容許此代理程式程式碼連接 JNDI 提供者，即遠端執行的 WebSphere MQ V6。實例化 JNDI InitialContext 之後，您就可以使用它來尋找 JMS ConnectionFactory，如第 8 行所示。這個值必須和使用 JMSAdmin 工具建立的 JMS 管理物件名稱相符。然後啓動連線並建立 JMS 階段作業。現在您便可以尋找您的 JMS 目的地，並開始存放訊息或使用訊息。第 12 行顯示透過尋找 JNDI 名稱 JMS_QUEUE 以找出 JMS 目的地。（這是由上述的 JMSAdmin 工具所建立的，如圖 6 所示。）

清單 2 顯示 JMS 產生者程式碼，說明如何建立 JMS 訊息，並將其遠端傳送給 JMS 提供者。因爲此代理程式必須依據在視圖中所選取的文件執行，第 16 行會取得這些文件的組合。然後我們將組合中的每個 Lotus Domino 文件逐一轉換成可在 JMS 中寫成純文字的 XML 文件（第 19 行）。

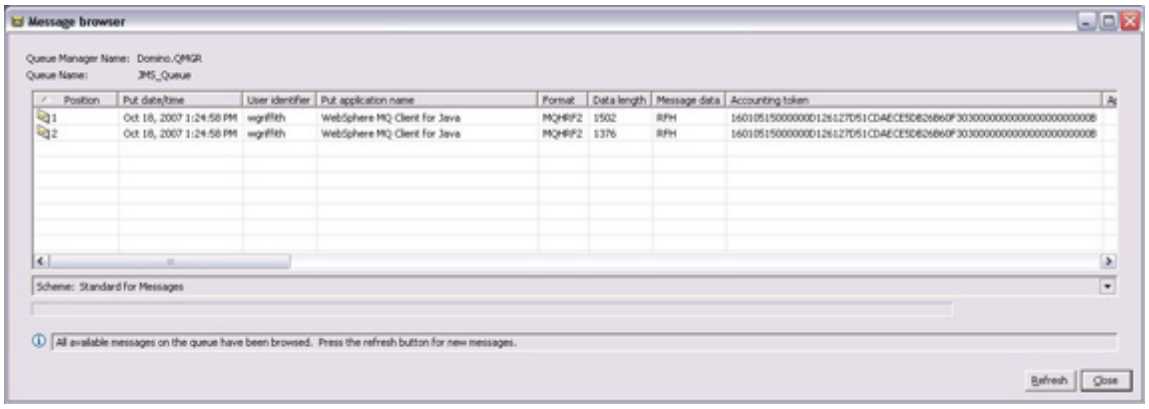
清單 **2. 訊息產生者**

```
13      MessageProducer jmsProducer = jmsSess.createProducer(jmsDest);
14      TextMessage jmsMsg = jmsSess.createTextMessage();

       // convert Domino document to an XML fragment
15      DxlExporter xmlGen = notesSession.createDxlExporter();

       // get collection of selected documents from agent
16      DocumentCollection dc =    agentContext.getUnprocessedDocuments();
17      Document doc = dc.getFirstDocument();

18      while (doc != null) {
           // write document contents to JMS destination
19          jmsMsg.setText(xmlGen.exportDxl(doc));
20          jmsProducer.send(jmsMsg);
21          jmsSess.commit();
22          doc = dc.getNextDocument();
    }
```

執行代理程式之後，您可以使用 WebSphere MQ 探險家來確認訊息是否已到達 WebSphere MQ。在 WebSphere MQ 探險家中，選取 Queues 資料夾，以檢視這個佇列管理程式中的所有佇列，然後用滑鼠右鍵按一下 JMS_Queue，並選擇 Browse Messages，以檢視該佇列中的訊息。

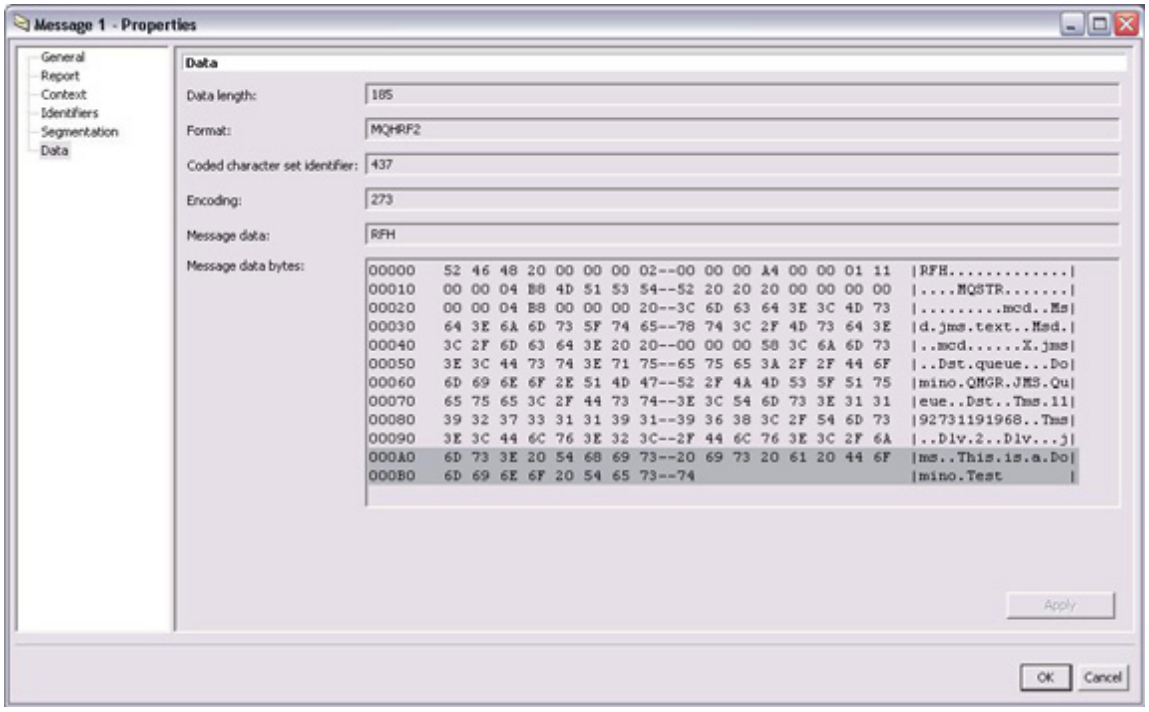
這會顯示在我們的 JMS_PUT 代理程式執行期間所選取的每個文件的訊息（請參閱圖 10）。

圖 10. 佇列中的訊息



按兩下其中一個訊息會顯示 Properties 對話框，其中會顯示該訊息，而顯示方式就如同在 WebSphere MQ 中一樣（請參閱圖 11）。

圖 11. WebSphere MQ 中的訊息



此時，您已成功地在一個 WebSphere MQ 佇列中，將一個完整的 Lotus Domino 文件儲存爲 XML。而且您的程式碼中沒有僅適用於 WebSphere MQ 的類別，只有同樣能夠輕鬆地將此訊息傳送至 WebSphere 企業服務匯流排的 JMS 程式碼。

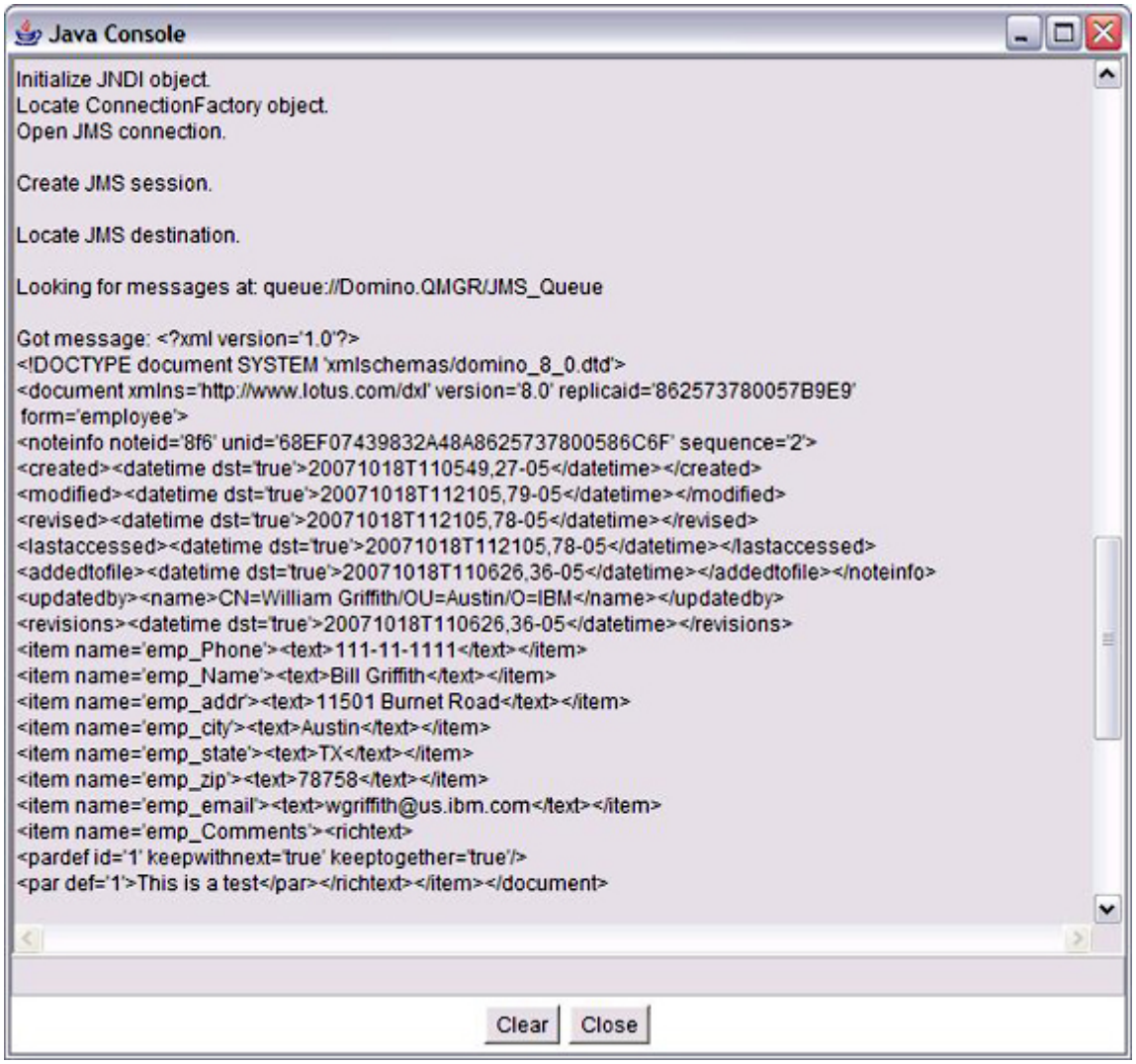
爲了說明您也可以透過 JMS 從 WebSphere MQ 取得訊息，我們建立了另一個代理程式來使用訊息。就和產生者程式碼一樣，這個程式碼並不知道它使用的是哪個 JMS 提供者 – 它只知道如何連結 JMS。清單 3 顯示消費者程式碼。

清單 **3. 訊息消費者**

```
13      MessageConsumer jmsConsumer = jmsSess.createConsumer(jmsDest);
14      Message jmsIncoming = null;

15      do {
          // The consumer will wait 1 second (1000 milliseconds) between polls
16          jmsIncoming = jmsConsumer.receive(1000);

17          if( jmsIncoming instanceof TextMessage ) {
18        System.out.println( "\n" + "Got message: "+((TextMessage) jmsIncoming).getText());
19              Document doc = db.createDocument();
20              doc.replaceItemValue("Form", "JMS_Imported");
21              doc.replaceItemValue("JMS_MsgID", jmsIncoming.getJMSMessageID());
22            doc.replaceItemValue("JMS_Message", ((TextMessage) jmsIncoming).getText());
23            doc.save(true, true);
24        }
25        jmsSess.commit();
26    } while ( jmsIncoming != null );
```

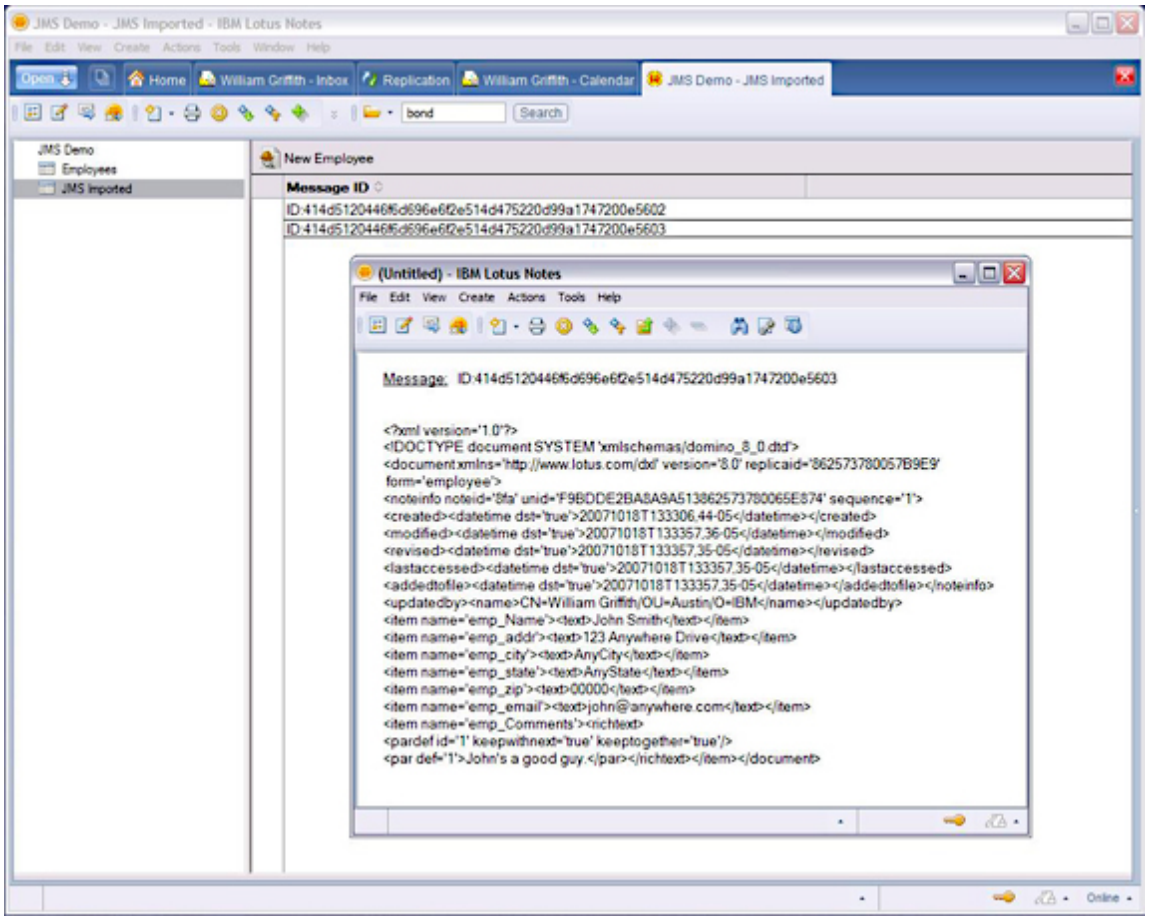
消費者程式碼最大的不同點是，消費者必須等待並接聽訊息，如第 16 行所示。 收到訊息後，我們會建立一個新的 Lotus Domino 文件，以將 JMS 訊息儲存至 Lotus Domino 資料庫（第 19 到 23 行）。爲了達到除錯的目的，我們會將訊息記錄到 Java 主控台中，該主控台可以透過在 Lotus Domino Designer 中選擇 Tools - Show Java Debug Console 來檢視，如圖 12 所示。

圖 12. 記錄到 Java 主控台的訊息



此外，開啓 JMS_Imported 視圖可以顯示執行 JMS 消費者程式碼所建立的文件。開啓該文件會顯示擷取到的 XML 文字（請參閱圖 13）。

圖 13. 儲存爲 Domino 文件的訊息



總結

在本文中，我們打算說明如何從 Lotus Domino 傳送訊息給 JMS 提供者，並從 Lotus Domino 接收其訊息。這提供了我們一個簡單實惠的方式將 Lotus Domino 連接至匯流排（即企業服務匯流排）。在本文中，我們是使用 WebSphere MQ 作爲傳訊提供者來進行示範，但是該程式碼可以和任何 JMS 提供者搭配運作。

致謝

特別感謝 Bob Balaban 和 Bobby Woolf 審閱本文。

下載

名稱	大小	下載方式
JMSDemo.zip	4.85MB	HTTP

下載方法的相關資訊

關於作者

Bill Griffith 是 IBM 軟體事業處的應用和整合中介軟體架構設計師。他已爲客戶開發了數十個 J2EE 和 Lotus Domino 應用程式；目前主要著重於開發服務導向架構的啓用與流程，並提供建議。