

IBM WebSphere 開發人員技術日誌：在 WebSphere MQ V6.0 上執行獨立的 Java 應用程式

Rob Wyatt (t.rob.wyatt@us.ibm.com)，T. IBM IT 專家

Bobby Woolf，IBM ISSW WebSphere J2EE 顧問

Kulvir Singh Bhogal (kbhogal@us.ibm.com)，IBM 資深 IT 專家

原始文章刊載於：

http://www-128.ibm.com/developerworks/websphere/techjournal/0610_woolf/0610_woolf.html

瞭解如何開發使用 IBM® WebSphere® MQ V6.0 收發訊息的 J2SE 應用程式。開發出來的應用程式會使用 J2EE™ 的標準 JMS 和 JNDI API，但無須部署在 J2EE 應用程式伺服器上。（摘自 IBM WebSphere 開發人員技術日誌）

簡介

「開發 WebSphere MQ 的獨立 Java 應用程式」(Developing a standalone Java application for WebSphere MQ) 一文介紹了如何使用 IBM WebSphere MQ V5.3，做為 Java 2 Standard Edition (J2SE) 應用程式（也就是不在 Java 2 Enterprise Edition (J2EE) 應用程式伺服器上執行的應用程式）的 Java™ 傳訊服務 (JMS) 和 Java 命名與目錄介面 (JNDI) 提供者。本文將探討相同的主题，但此次以 WebSphere MQ V6.0 為主，該產品現加入了建立 JMS 應用程式所需的 JMS 類別，而且也能直接當做實作 JMS 主題的發佈/訂閱分配管理系統。這樣的平台不但更容易開發應用程式，也能更快部署，管理起來也更為輕鬆。

本文隨附的範例程式碼和配置均已針對這些變更大幅調整。此外，本文也另闢數個章節，針對 JMS 1.1 中的統一網域類別、例外處理、傳輸連結以及其他最佳實務進行說明。熟悉第一篇文章的讀者也將能從本文獲得更多實用的資訊。

為什麼要使用獨立 JMS 應用程式？

可能有人會想問，為什麼要撰寫在 J2EE 環境以外運作的 JMS 傳訊程式？的確，J2EE 與傳訊的關係極為密切，但在 J2EE 出現之前，傳訊就已經存在，當初使用傳訊的許多工作現在還是得繼續，這些工作對 J2EE 環境來說不是大材小用，不然就是不適合。這些工作包括批次作業、橋接、轉換、公用程式、系統管理、設備測試、監視以及實際事件通知等等。不過，在 J2EE 環境以外撰寫程式碼，並不代表就不能使用 JMS。由於 Java 是首選的程式語言，而 JMS 又是 Java 的標準傳訊 API，所以真正的問題應該是：何不使用 JMS 來滿足這些傳訊需求呢？

JMS 應用程式的各個階層

執行中的 JMS 應用程式包含三層不同的程式碼，如圖 1 所示。

1. JMS 應用程式 - 獨立於提供者之外；使用 JNDI 透過 JMS 存取提供者。
2. JMS 實作 - 提供者專屬的 JMS 物件。
3. 傳訊提供者 - 這是傳訊系統本身，在此即為 WebSphere MQ。

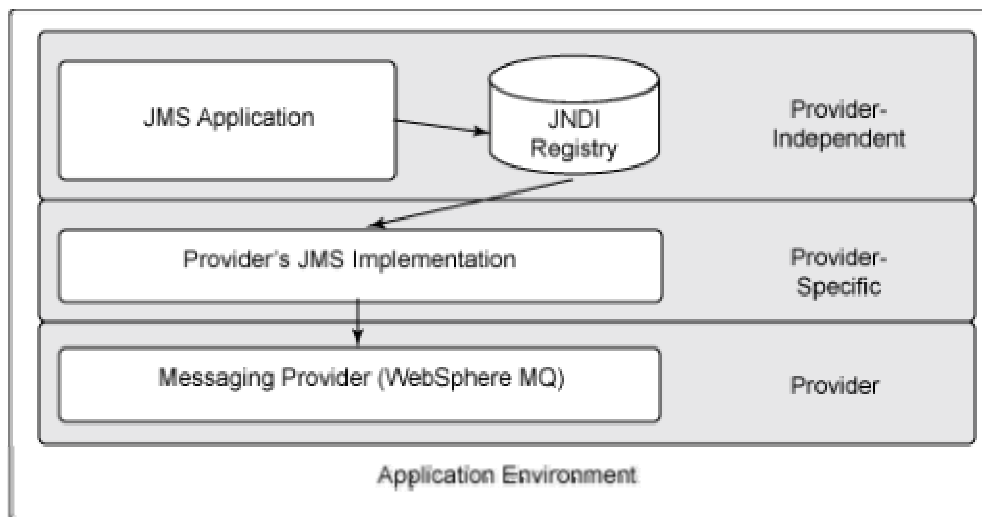


圖 1. JMS 應用程式的階層

如圖 1 所示，應用程式碼並非直接存取傳訊提供者，而是由提供者專屬的那一層程式碼存取，透過 JNDI 存取個別物件。

由於 JMS 隱藏了 JMS 實作和提供者層的大部分細節，因此 Java 開發人員可以編寫與特定提供者幾近完全無關的 JMS 應用程式。不過，負責管理 JMS 提供者的人員還是需要對 JMS 實作層有所瞭解，因為設定應用程式與提供者搭配運作的配置就在此進行。JMS 規格雖描述了程式設計 API，但實作的細節則交由廠商自行負責。因此，儘管 Java 開發人員看不到標準 JMS API 的實作細節，但 JMS 提供者管理員還是必須知道配置細節，而這些配置細節會因各廠商的產品而異。

由於有必要瞭解 JMS 應用程式、JMS 實作以及傳訊提供者彼此之間如何互動，因此下文將就每一層進行說明，首先從提供者開始。

傳訊提供者

傳訊提供者會透過網路連線，在程序之間可靠地傳送訊息。傳訊提供者不一定要支援 JMS。例如，WebSphere MQ 早在 JMS 規格出現之前即已存在（前身為 IBM MQSeries），

即使到今天，許多非 Java 應用程式都還是使用 WebSphere MQ，而從未用過 JMS。不過，本文假設傳訊提供者是 WebSphere MQ，而應用程式將透過該產品選配的 JMS 介面存取提供者。

WebSphere MQ 是以一或多個佇列管理程式的形式執行。每個佇列管理程式都是一組程序，這些程序會定義一組佇列、組織這些佇列中的訊息、將這些訊息儲存在記憶體和磁碟上，然後將訊息送到傳訊用戶端和其他佇列管理程式。WebSphere MQ 使用佇列實現點對點傳訊。WebSphere MQ 中有一個名為「發佈/訂閱分配管理系統」的程序提供發佈/訂閱功能，使用內建的 WebSphere MQ 佇列做為資料儲存庫。WebSphere MQ 使用這些功能建置 JMS 佇列和主題。

JMS 沒有物件能夠對應於 WebSphere MQ 佇列管理程式，也沒有 JMS API 可透過程式設計的方式執行系統管理和配置作業，例如啟動或停止佇列管理程式、定義或刪除 WebSphere MQ 物件，或執行任何其他與 WebSphere MQ 相關的管理作業。因此，只有具備基本的 WebSphere MQ 管理知識，才能確保佇列管理程式正常執行，而且也用了 JMS 應用程式所需的資源配置。這一般是由 WebSphere MQ 管理者負責的工作。

如果您對發佈/訂閱傳訊網域有興趣，請務必注意，在 WebSphere MQ 中，當佇列管理程式啟動時，發佈/訂閱分配管理系統並不會跟著自動啟動。在啟動佇列管理程式之後，還必須啟動分配管理系統，這樣應用程式才能執行發佈/訂閱傳訊。從 WebSphere MQ V6 開始，分配管理系統可以配置由佇列管理程式控制啟動和停止。

用戶端模式 vs. 連結模式

在 WebSphere MQ 中，應用程式可以利用兩種方法與 WebSphere MQ 佇列管理程式連線。這兩種方法分別為連結模式 (bindings mode) 和用戶端模式 (client mode)。瞭解兩種模式的差異、好處及壞處是很重要的。

- 連結模式是 WebSphere MQ 的原生連線模式。在連結模式中，JMS 應用程式必須與佇列管理程式在同一部主機上執行，而且使用跨處理程序 (IPC) 協定通訊。
- 在用戶端模式中，JMS 應用程式則透過 TCP/IP 與佇列管理程式的代理程式建立網路階段作業。然後，代理程式再使用連結模式與佇列管理程式通訊，代表應用程式執行 API 呼叫。

連結模式和用戶端模式的傳訊用戶端 API 是一樣的。在 JMS 中，這些模式的連線參數全都儲存在連線 Factory 物件中，應用程式是看不到這些參數的。不過，在操作時還是有些重要事項需要注意：

- 所有用戶端模式 API 呼叫都是以三個步驟執行：
 1. 將 API 呼叫從用戶端應用程式傳送到代理程式。
 2. 代理程式對佇列管理程式執行呼叫。
 3. 將呼叫結果從代理程式傳回用戶端應用程式。

如果網路連線在這期間中斷，用戶端應用程式並無法判斷連線是在第一個步驟（即 API 呼叫傳送到代理程式之前）中斷，或在第三個步驟（準備送回結果）中斷。這樣一來，如果用戶端模式的連線中斷，就無法確定最後一個 API 呼叫的結果為何。因此，如果要使用（或可能使用）用戶端模式，建議使用交易階段作業並撰寫明確的 `commit()` 和 `rollback()` 方法呼叫。使用這種方法時，應用程式雖然在連線中斷之後可能會收到重複的訊息，但確定不會遺失任何訊息。

- 使用基本 WebSphere MQ 用戶端時，用戶端模式無法使用 XA 交易。不過，IBM 提供名為「延伸交易用戶端」(Extended Transactional Client) 的進階版本，這種用戶端提供用戶端模式連線所需的 XA 功能，但這是要另外授權和購買的產品，不應與免費的 WebSphere MQ 基本用戶端混為一談。
- 由於連結模式的連線使用共用記憶體，因此不會因網路連線傳輸造成負荷。用戶端模式的效能則會視網路負載的情況而定，一直到連線中斷為止。此外，防火牆和路由器也可能逾時，使得用戶端階段作業中止。

綜合上述原因，最好還是盡可能使用連結模式。不過另一方面來說，在連結模式下，佇列管理程式必須與應用程式安裝在同一部主機上（但這不一定都可行），用戶端模式則可讓應用程式存取不同主機上的佇列管理程式。因此，連結模式雖然一般較佳，但用戶端模式也有其好處。最佳做法是：J2SE JMS 應用程式所在的主機上應盡可能安裝佇列管理程式，而應用程式則透過連結模式連線。範例程式碼包含了用戶端和連結模式所需的連線 Factory，示範如何實作兩者。

要使用用戶端模式連線，還必須對 WebSphere MQ 做些配置。首先，必須啟動接聽器程序。WebSphere MQ 提供名為 `runmqslr` 的執行檔，預設會在埠 1414 上接聽。用戶端模式也需要在佇列管理程式上定義 `SVRCONN` 通道。為了能夠連線，連線 Factory 必須包含通道名稱、接聽器埠，以及佇列管理程式所在電腦的主機名稱或 IP 位址。執行範例程式碼所需的接聽器和通道配置將在下節中說明。

另外也請注意，要提供 WebSphere MQ 用戶端功能，最佳方式是安裝最新的 WebSphere MQ 軟體套件（伺服器或用戶端版本皆可）。很多人會直接從安裝檔取得 Java JAR 檔案，然後再將這些檔案改放在用戶端電腦上。這種方法雖然也可以讓應用程式執行，但就無法安裝完整套件中許多有用的公用程式和診斷工具。特別是，完整安裝 WebSphere MQ 用戶端不但容易維護和套用修補程式，也可以查看安裝的版本為何。為了能在本端建立佇列管理程式，本文假設安裝 WebSphere MQ 伺服器，而且也選取安裝用戶端支援的選項（如需下載

WebSphere MQ 試用版以用於此範例，請參閱「資源」一節)。

JMS 實作

每個提供者都會以其能運作的方式實作 JMS API。這種提供者專屬的實作方式可讓 JMS 程式中非屬特定提供者的程式碼附加到特定的 JMS 提供者。由於每個提供者的實作方式都不同，因此負責管理 JMS 物件或傳訊提供者的人員必須瞭解如何配置實作層。本文所使用的提供者為 WebSphere MQ，因此會提供 WebSphere MQ 專屬的指示說明。

JMS 管理物件

管理物件是提供者專屬的根物件，用來實作 JMS 介面，而且可直接由應用程式存取。JMS 應用程式使用 Java 命名與目錄介面 (JNDI)，根據名稱從登錄擷取管理物件。登錄項目包含 JMS 應用程式使用的公用屬性，例如物件類型（像是佇列或主題），也包含提供者的 JMS 實作類別所使用的私用屬性。雖然登錄中有許多資源類型，但主要與 JMS 相關的管理物件類型有三種：連線 Factory、佇列以及主題。JMS 應用程式會透過這些管理物件與 WebSphere MQ 佇列管理程式通訊。

連線 Factory 可讓 JMS 程式與佇列管理程式建立連線來收發訊息。然後再使用佇列或主題物件產生或消費訊息。JMS 佇列相當於 WebSphere MQ 的佇列實作，不過 WebSphere MQ 沒有物件能直接對應於 JMS 主題。WebSphere MQ 使用發佈/訂閱分配管理系統實作主題，該系統會使用分配管理控制佇列和發佈訊息的串流佇列。這些執行時期的細節是 WebSphere MQ 的 JMS 實作的一部分，從 JMS 應用程式是完全看不到的。從應用程式的角度來看，WebSphere MQ 中的佇列和主題實作都是透過 JNDI 以 JMS 管理物件的形式存取，作用就跟 JMS 規格所述完全相同。

這些提供者專屬的物件還有另一個重要功能，就是設定和強制執行系統管理政策或商業要求。例如，訊息的優先順序可在應用程式碼中指定，但也可以透過管理物件中的設定予以置換。佇列、主題和連線 Factory 還有許多其他提供者專屬物件的屬性，值得花時間好好研究。

乍看之下，這些提供者專屬物件好像是要另外學習和管理的東西，但其實只要執行簡單的程式碼即可。不過，這些提供者專屬物件之所以強大，是因為可以讓程式碼與提供者分開獨立，使其餘程式碼得以不受環境限制，而且可供移植。例如，在不同環境（開發、測試、實際執行等等）或在應用程式的不同版本中，佇列常會有不同的名稱。由於 WebSphere MQ 名稱是封裝在 JMS 管理物件內，因此應用程式碼不會受到這些變化的影響。事實上，大部分的應用程式碼都不需要知道物件代表的是佇列或主題，甚至也不需知道使用的是哪一家廠商的提供者。提供者專屬物件能讓所有這些實作細節獨立於程式碼之外。

管理物件登錄

JMS 管理物件必須登錄在實作 JNDI API 的目錄中。JNDI 可讓 Java 元件透過唯一名稱存取資源，無論資源儲存在何處、如何實作，或是容器和其 JNDI 提供者實際如何提供資源的存取權。資源可能是程式需要全域存取的任何物件，例如 Java 資料庫連線功能 (JDBC) 資料來源、Enterprise JavaBeans (EJB) 起始點、J2EE Connector Architecture 連接器等等。

WebSphere MQ 對三種不同的目錄服務提供 JNDI 支援：J2EE 環境（如 WebSphere Application Server）、LDAP 以及檔案系統實作。本文將使用檔案系統服務，因為這除了 WebSphere MQ 安裝檔之外，並不需要另外的軟體或硬體元件。JMS 應用程式需要至少一個連線 Factory 和至少一個佇列或主題。接下來，我們將討論 WebSphere MQ 如何將這些物件設為可透過 JNDI 存取的登錄項目。

連線 Factory 提供應用程式連線到 WebSphere MQ 佇列管理程式需要的所有資訊。應用程式並不需要知道它是透過用戶端或連結模式連線，也不需要知道佇列管理程式的名稱。應用程式碼只要使用 JNDI 查閱連線 Factory 的名稱，存取佇列管理程式和建立連線的低階細節就會由 WebSphere MQ 的物件處理。

連線 Factory 實作包含二十多個 WebSphere MQ 屬性。預設設定會使用最保守的選項，這對執行簡單的程式就足夠了，例如本文隨附的範例。不過，要有效管理 JMS 物件或 WebSphere MQ，就必須瞭解兩者之間是如何互動的。

從 JMS 1.1 開始，JMS 應用程式就無須區分佇列和主題。在應用程式碼中，這兩者都是產生或消費訊息的目的地。不過，佇列和主題就傳輸層面來看有著本質上的不同，因此，任何目的地的登錄項目都必須明確定義為佇列或主題。當 JMS 應用程式透過 JNDI 查閱目的地時，會根據名稱找出正確的物件，因此傳回的結果可能是佇列或主題的實例。大多數情況下，應用程式可以只使用目的地物件，而不管物件代表的是佇列或主題，然後視需要產生或消費訊息即可。如有必要，可以使用操作員實例來判斷目的地的子類型。

如同連線 Factory，佇列和主題登錄項目也都有多個 WebSphere MQ 屬性。同樣地，這些屬性的預設值對本文隨附的簡單範例應用程式即足夠，但管理者最好能先熟悉所有屬性以及作用，然後再部署實際執行的應用程式。

JMS 應用程式

JMS 應用程式堆疊的最上層是 JMS 應用程式本身。應用程式使用 JNDI 查閱所需的各種 JMS 管理物件，然後使用這些物件附加到 WebSphere MQ 傳訊提供者。應用程式只需要知道資源的 JNDI 名稱。這些名稱必須完全相符，而且區分大小寫。

從 JMS 的角度來看，如前所述，最重要的變更之一就是將點對點傳訊與發佈訂閱合併到同一網域中。雖然仍可以使用網域專屬的類別，例如 `Queue`、`Topic`、`QueueConnectionFactory` 和 `TopicConnectionFactory`，但這些類別已被中性類別取代，例如 `Destination` 和 `ConnectionFactory`，這些類別適合同時用於佇列和主題。

將佇列與主題網域統一起來是大工程，值得更深入探討。在舊規格之下，應用程式碼必須區分佇列或主題，因為類別是有分網域的。因此，若要將佇列變更為主題，就必須重新編碼、重新編譯，然後再重新部署應用程式。這等於是在設計期間就必須知道目的地是佇列或主題。在 JMS 1.1 之下，則通常只需要到執行時期再知道目的地為哪種類型的物件。這表示可以將程式編寫為只要知道高階訊息流程即可。

本文隨附的範例程式碼示範了如何使用這些統一的網域類別。此範例在佇列管理程式上使用了佇列和主題，然後利用 JNDI 連結在兩者之間切換。請注意，如果在程式執行時變更 JNDI 連結，程式不會動態切換佇列和主題。這項變更只會在 JNDI 查閱物件時生校，這通常只會執行一次，也就是在程式初始化的時候。

範例應用程式

現在我們已經知道 JMS 應用程式如何附加到 WebSphere MQ，接下來就來看看範例 JMS 程式，以及如何配置 WebSphere MQ 來執行該程式。若要這麼做，需執行幾個步驟：

1. 配置 WebSphere MQ。
2. 取得範例程式碼。
3. 配置管理物件。
4. 執行範例程式碼。

1. 配置 WebSphere MQ

若要配置 WebSphere MQ 執行範例程式碼，必須執行下列步驟：

A. 建立佇列管理程式。

若要執行範例程式碼，就需要有佇列管理程式。當建立佇列管理程式時，會有許多不同的選項可供選擇，不過我們需要的是配置寄不出信件佇列的選項，以及設為預設佇列管理程式的選項：

- 佇列管理程式和 Java 類別會使用寄不出信件佇列，在其中存放無法送達指定目的地的訊息。每一個佇列管理程式最好有一個寄不出信件佇列。
- 當佇列管理程式設為預設佇列管理程式時，應用程式就可以在不知道名稱的情況下存取它。這可讓範例程式碼更容易撰寫，而且應用程式不該依賴此設定。

定義新佇列管理程式的方法有好幾種，最快的就是從 GUI，也就是「WebSphere MQ 探險家」：

- I. 從 Windows® 的「開始」功能表選取「所有程式」=> IBM WebSphere MQ => WebSphere MQ Explorer (WebSphere MQ 探險家)。一旦 WebSphere MQ 探險家執行，用滑鼠右鍵按一下 Queue Managers (佇列管理程式) 資料夾。然後選取 New (新增) => Queue Manager (佇列管理程式) 啟動 New Queue Manager (新增佇列管理程式) 精靈。(圖 2)

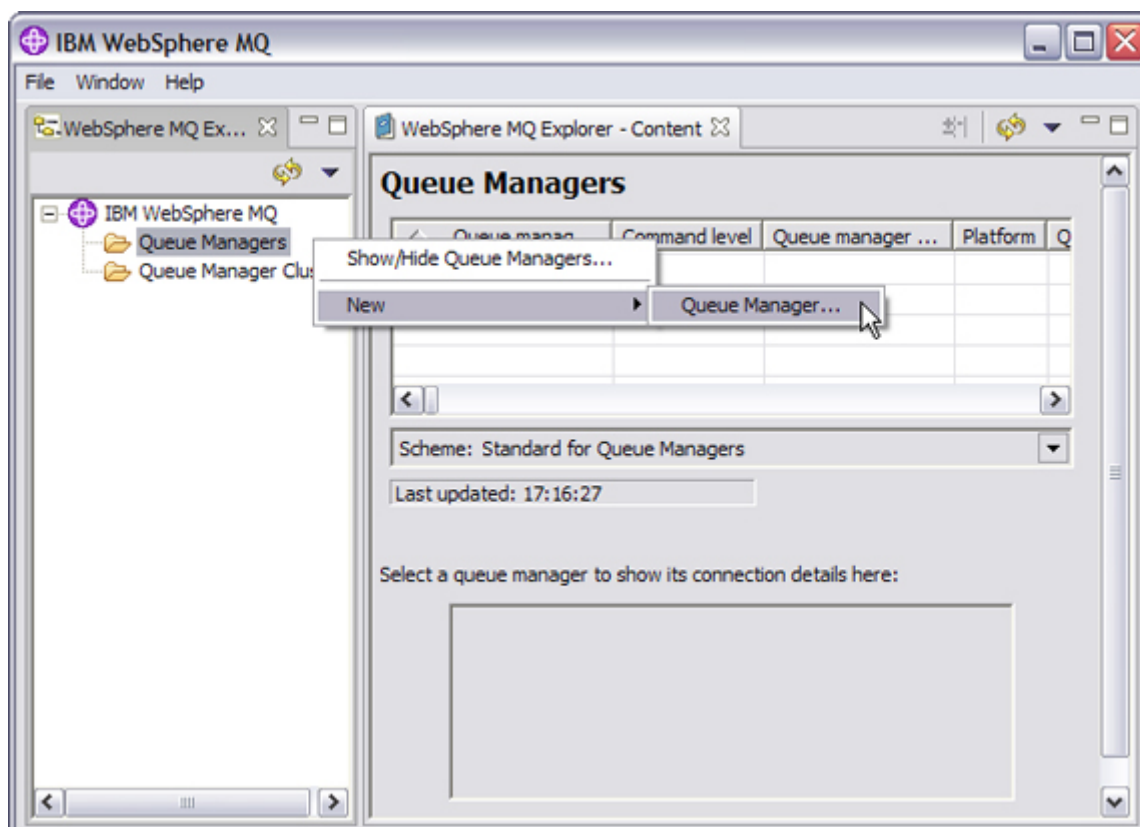


圖 2. 建立新的佇列管理程式

- II. 當精靈開啓時，輸入佇列管理程式名稱 JMSDEMO，然後勾選 Make this the default queue manager (設為預設佇列管理程式) 方塊。接下來，輸入預設寄不出信件 (Dead Letter Queue) 的名稱：SYSTEM.DEAD.LETTER.QUEUE。請確定全部輸入大寫字母。(圖 3)

Create Queue Manager

Queue Manager
Enter basic values (Step 1)

Queue manager name: JMSDEMO

Make this the default queue manager

Default transmission queue: _____

Dead letter queue: SYSTEM.DEAD.LETTER.QUEUE

Max handle limit: 256

Trigger interval: 999999999

Max uncommitted messages: 10000

< Back Next > Finish Cancel

圖 3. 定義佇列管理程式

III. 這時按 **Finish** (完成) 按鈕，以預設設定建立佇列管理程式 (圖 4)。這些預設設定包括：

- ◆ 佇列管理程式會立即啟動。
- ◆ 佇列管理程式會設定在電腦重新開機時啟動。
- ◆ 會建立新的接聽器並在埠 1414 上啟動。
- ◆ 新的佇列管理程式現在應出現在 **WebSphere MQ 探險家** 視窗中。

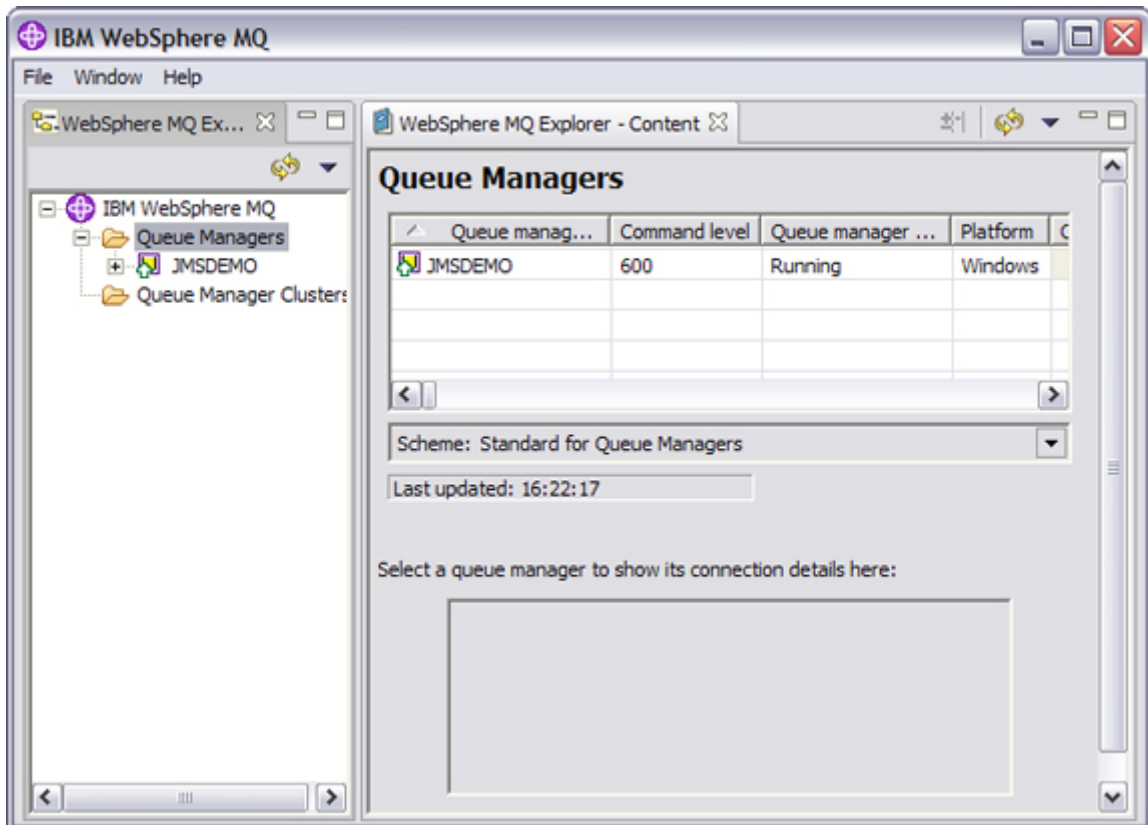


圖 4. 佇列管理程式清單

B. 建立佇列

從 WebSphere MQ 探險家定義佇列也相當簡單（圖 5）：

- I. 按一下 JMSDEMO 佇列管理程式旁邊的加號，展開資料夾樹狀結構，然後按一下 Queues（佇列）資料夾。系統佇列預設會篩選掉，而由於佇列管理程式是剛建立的，因此右邊窗格應該不會有任何佇列出現。
- II. 接下來，用滑鼠右鍵按一下 Queues（佇列）資料夾，然後從快顯功能表選取 New（新增）=> Local Queue（本端佇列），開啓 Local Queue（本端佇列）精靈。

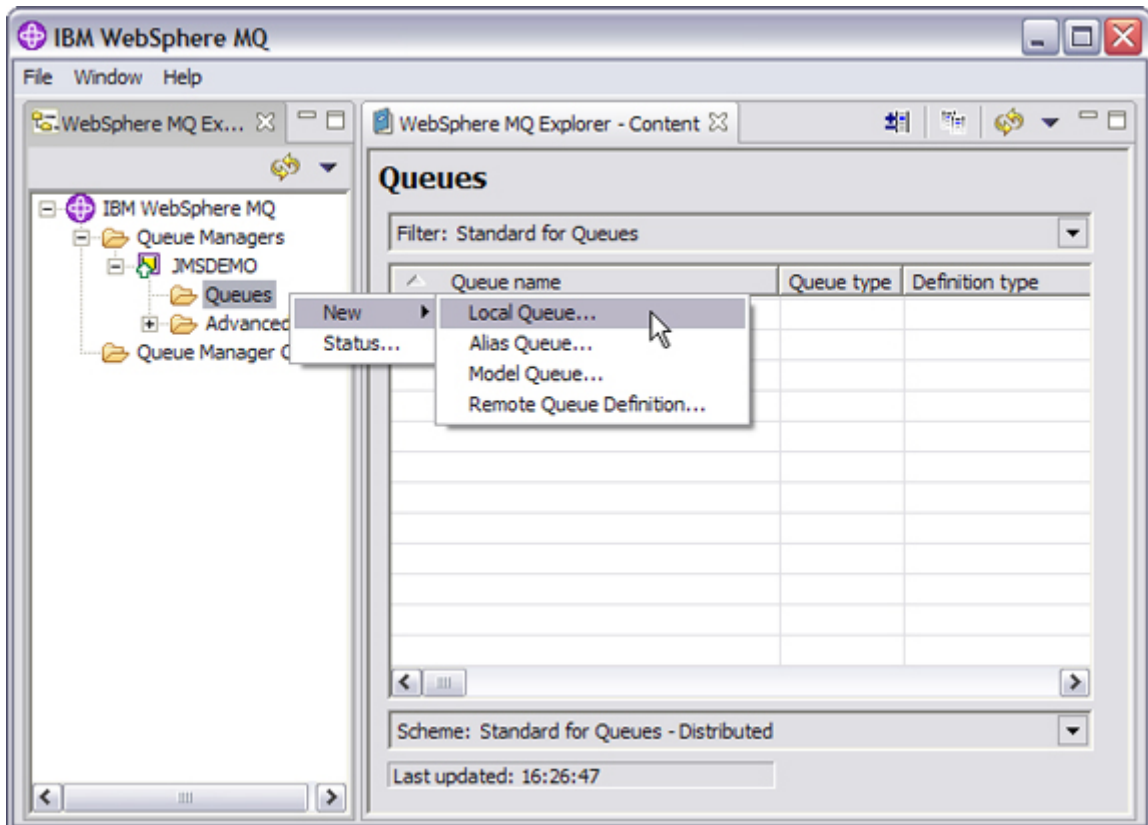


圖 5. 建立新佇列

- III. 在精靈中，輸入 JMSDEMO.QL 做為佇列的名稱，然後按一下 Finish（完成）按鈕。由於稍後建立的 JNDI 目錄項目會搜尋這個佇列名稱，因此請務必按照圖中所示輸入。（圖 6）

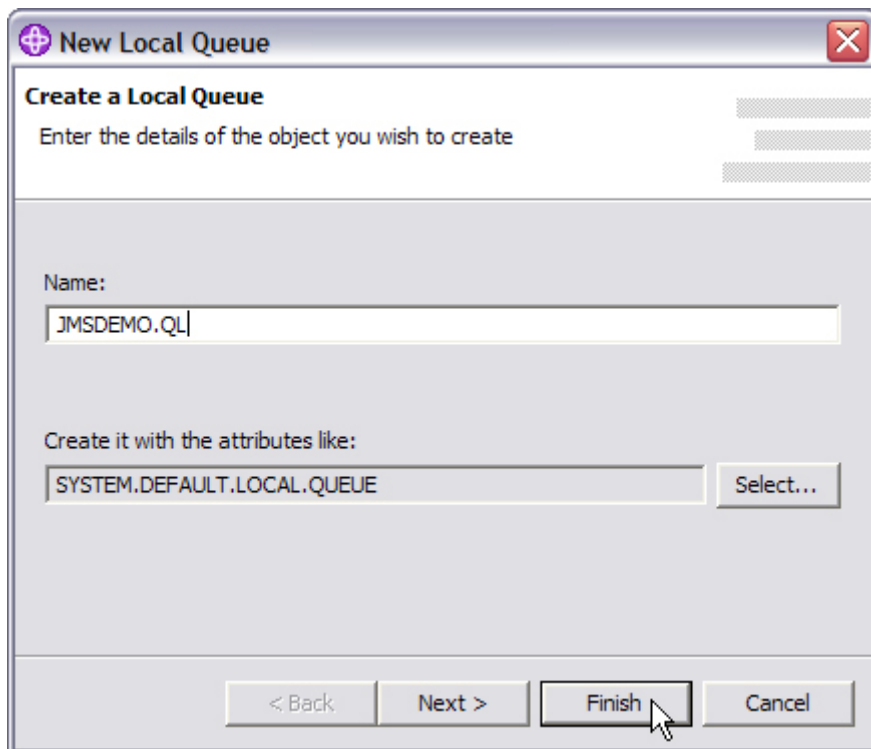


圖 6. 定義新佇列

IV. 當精靈完成時，新佇列應出現在 Queues（佇列）視窗中。（圖 7）

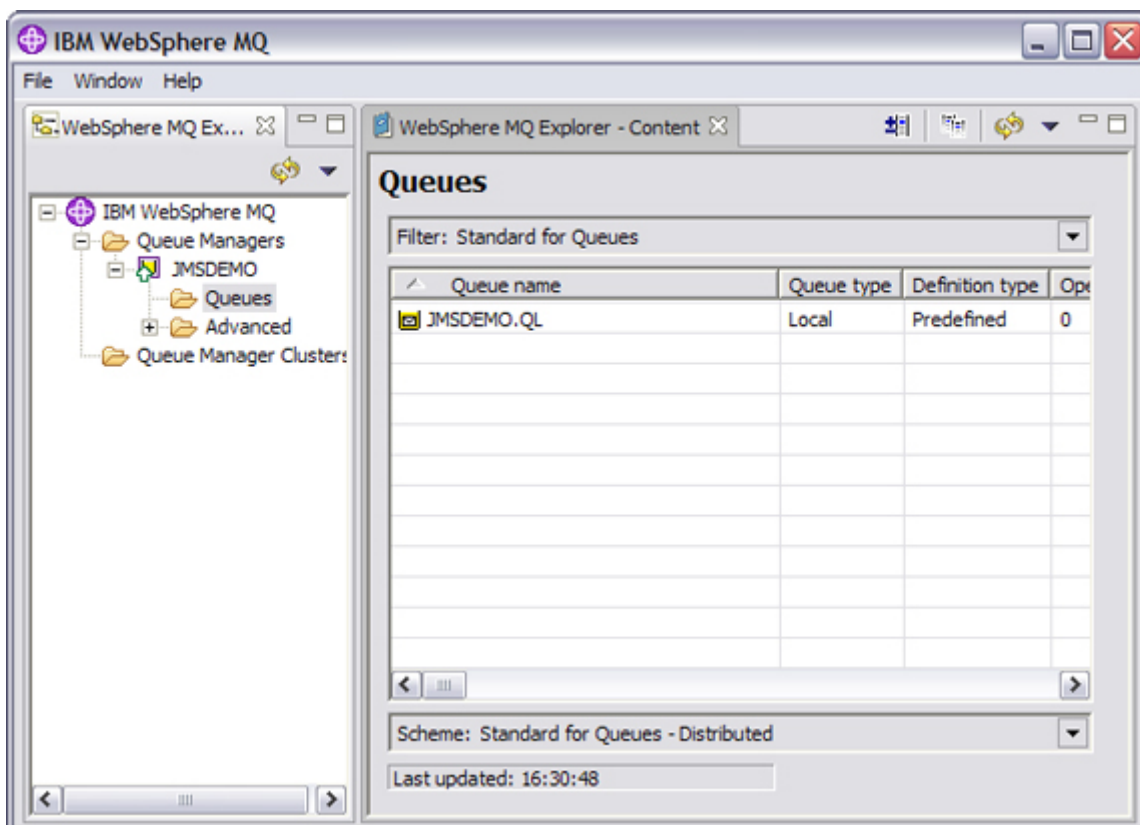
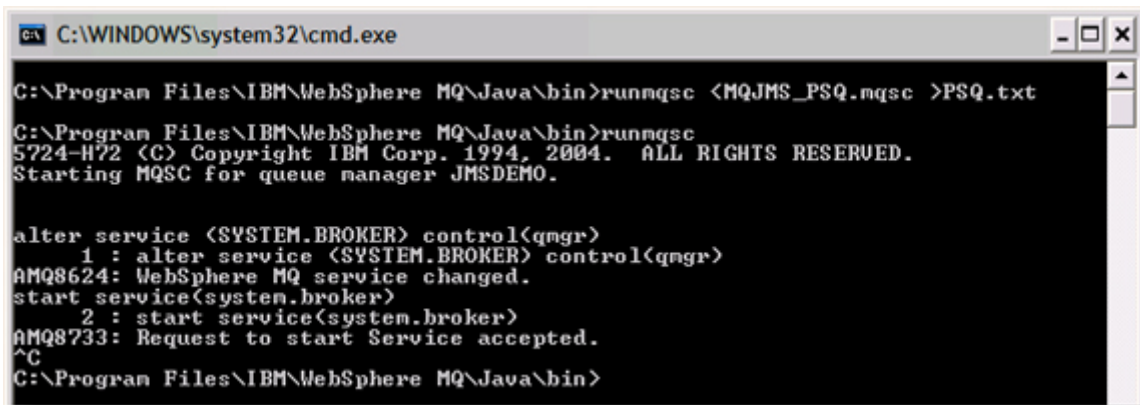


圖 7. 佇列清單

C. 設定發佈/訂閱

到目前為止，我們已經定義並啟動一個佇列管理程式，並建立一個佇列供點對點傳訊。設定發佈/訂閱還需要幾個步驟。分配管理系統會將內部狀態儲存在佇列中，因此必須先建立系統才能予以啟動。IBM 將所有定義都放在指令碼中，因此只要執行 `runmqsc` 指令直譯器中的指令碼即可。一旦完成之後，接下來就要設定由佇列管理程式控制發佈/訂閱分配管理系統啟動和停止。

- I. 開啓指令視窗，然後移到安裝 WebSphere MQ 的目錄。從這個目錄往下切至 `Java/bin` 目錄，這裡會有 `MQJMS_PSQ.mqsc` 指令碼。執行 `runmqsc`，並將這個指令碼重新導向成爲其輸入（如圖 8 所示）。將輸出重新導向至某個檔案以供稍後檢視。



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

C:\Program Files\IBM\WebSphere MQ\Java\bin>runmqsc <MQJMS_PSQ.mqsc >PSQ.txt
C:\Program Files\IBM\WebSphere MQ\Java\bin>runmqsc
5724-H72 (C) Copyright IBM Corp. 1994, 2004. ALL RIGHTS RESERVED.
Starting MQSC for queue manager JMSDEMO.

alter service (SYSTEM.BROKER) control(qmgr)
  1 : alter service (SYSTEM.BROKER) control(qmgr)
AMQ8624: WebSphere MQ service changed.
start service(system.broker)
  2 : start service(system.broker)
AMQ8733: Request to start Service accepted.
^C
C:\Program Files\IBM\WebSphere MQ\Java\bin>
```

圖 8. 重新導向 runmqsc 指令碼

- II. 接下來就需要將分配管理系統配置為服務。服務是 WebSphere MQ 第 6 版的新功能，這裡要使用的是名為 SYSTEM.BROKER 的服務。執行的步驟很簡單，就是把服務設在佇列管理程式的控制之下，然後啟動服務。

一開始先啟動 WebSphere MQ 指令直譯器 runmqsc。使用 alter service(SYSTEM.BROKER) control(qmgr) 指令，將 SYSTEM.BROKER 服務改由佇列管理程式控制。由於佇列管理程式已在執行中，因此第一次必須手動啟動分配管理系統。輸入 start service(system.broker) 指令（請注意，除非另外註明，否則 runmqsc 參數不區分大小寫）。按 CTRL-C 結束 runmqsc。

我們會使用名為 JMSDemo 的主題，不過由於使用預設設定，所以無須預先定義主題。這就完成了執行 JMSDemo 應用程式所需的配置。

2. 取得範例程式碼

範例應用程式是以壓縮的格式提供。請使用一般慣用的解壓縮程式，將檔案解壓縮到本端磁碟。請選取保留目錄的選項，讓檔案解壓縮到本端磁碟上名為 JMSDEMO 的新目錄中。該目錄下會有執行程式需要的所有檔案。請花些時間熟悉這些檔案：

- JMSAdmin.config - 指示 JMSAdmin 工具使用 JNDI 登錄服務所需的檔案系統。
- JMSAdmin.bat - 執行 WebSphere MQ JMS 管理工具。
- JMSDemoPub.bat - 將範例程式當做發佈者執行。
- JMSDemoReceive.bat - 執行範例程式來消費佇列中的訊息。
- JMSDemoSend.bat - 執行範例程式來產生佇列中的訊息。
- JMSDemoSub.bat - 將範例程式當做訂閱者執行。
- jmsdemoJNDIbindings.scf - 包含管理物件定義並將連線 Factory 設為使用連結模式。
- jmsdemoJNDIClient.scf - 包含管理物件定義並將連線 Factory 設為使用用戶端模式。

3. 配置管理物件

範例應用程式中的每個連線 **Factory** 或目的地都必須用唯一名稱登錄在 **JNDI** 目錄中。範例應用程式會使用該名稱查閱 **JMS** 管理物件。提供者專屬的物件使用基本的 **WebSphere MQ** 物件名稱和屬性配置。範例定義包含在下載檔所附的 **.scp** 檔中。

請務必瞭解管理物件的屬性，因為這些屬性會控制 **JMS** 應用程式與 **WebSphere MQ** 之間的互動。各物件屬性的設定對 **JMS** 應用程式的運作和效能都會有極大的影響。但此配置不是 **JMS** 應用程式的一部分；配置的細節會視 **WebSphere MQ** 而定。

若要配置所需的管理物件：

I. 配置連線 **Factory**

連線 **Factory** 需要宣告佇列管理程式名稱參數，但此參數不一定要包含值。如果要在參數填入值，該值必須與佇列管理程式名稱完全相同，否則連線就會失敗。如果此值空白，那麼若本端佇列管理程式設為預設佇列管理程式，使用連結模式的連線會成功，而在用戶端模式下的任何佇列管理程式也會連線成功。此範例將值保留為空白。

另外也需要傳輸參數，指示 **JMS** 使用用戶端或連結模式連線。當使用用戶端模式時，必須指定 **host**、**chan** 和 **port** 參數：

- **host** - 佇列管理程式所在主機的名稱或 **IP** 位址。
- **chan** - 包含用來連線到佇列管理程式的 **SVRCONN** 通道名稱。
- **port** - 佇列管理程式接聽的埠號。

連線 **Factory** 的部分參數雖然不強制要求，但重要性值得注意：

- **FAILIFQUIESCE** - 可讓佇列管理程式中斷任何 **API** 呼叫以利關機。此參數預設為啟用；請注意，停用此參數可能會對佇列管理程式有不利的影響，尤其當使用用戶端模式連線時。
- **SYNCPOINTALLGETS** - 有助於確保應用程式未遺失任何訊息。這在用戶端模式下執行時尤為重要，因為訊息從佇列離開到透過網路送達應用程式之間，會有時間上的延遲。如果網路在這段時間中斷，除非設有同步點，否則訊息就無法回復。在連線 **Factory** 中設定 **SYNCPOINTALLGETS** 之後，可確保回復未認可的訊息。認可是自動執行的動作，因此這不會對程式碼有所影響（如需同步點操作的完整說明，請參閱「**WebSphere MQ 應用程式設計手冊**」(**WebSphere MQ Application Programming Guide**) 和「**WebSphere MQ 使用 Java**」(**WebSphere MQ Using Java**) 手冊)。

範例程式提供了兩個連線 Factory 目錄定義，其中一個設為連結模式，另一設為用戶端模式，如下所示：

連結模式連線 Factory 定義（取自於 `jmsdemoJNDIbindings.scf` 檔案）：

```
#-----  
# Connection Factory for Bindings mode  
# Delete the Connection Factory if it exists  
DELETE CF(JMSDEMOCF)  
  
# Define the Connection Factory  
DEFINE CF(JMSDEMOCF) +  
    SYNCPOINTALLGETS(YES) +  
    TRAN(bind) +  
    QMGR( )  
  
# Display the resulting definition  
DISPLAY CF(JMSDEMOCF)
```

用戶端模式連線 Factory 定義（取自於 `jmsdemoJNDIClient.scf` 檔案）：

```
#-----  
# Connection Factory for Client mode  
# Delete the Connection Factory if it exists  
DELETE CF(JMSDEMOCF)  
  
# Define the Connection Factory  
DEFINE CF(JMSDEMOCF) +  
    SYNCPOINTALLGETS(YES) +  
    TRAN(client) +  
    HOST(local host) CHAN(SYSTEM.DEF.SVRCONN) PORT(1414) +  
    QMGR( )  
  
# Display the resulting definition  
DISPLAY CF(JMSDEMOCF)
```

每個佇列管理程式預設都會有 `SYSTEM.DEF.SVRCONN` 通道。為方便起見，本文使用該通道，但請注意，一般最好不要使用 `SYSTEM.*` 物件。如果您要在其他人管理的佇列管理程式上執行範例，WebSphere MQ 管理者可能需要為您另外定義 `SVRCONN` 通道。在此情況下，請確定在 `jmsdemoJNDIClient.scf` 檔案中輸入正確的通道名稱。

II. 配置目的地

其餘的定義對 `jmsdemoJNDIClient.scf` 檔案和 `jmsdemoJNDIbindings.scf` 檔案都相同。

前面提到，每個 JMS 管理物件都必須登錄在可透過 JNDI 存取的目錄中。為了將目錄名稱對映到 WebSphere MQ 已知的佇列或主題名稱，必須設定佇列或主題屬性。此屬性沒有預設值，但區分大小寫。

FAILIFQUIESCE 屬性同時適用於佇列和主題物件。不過，這不是從連線 Factory 繼承的，而是預設為 "YES" 的正確值，因此不會包含在範例定義中。除了 WebSphere MQ 版本的佇列名稱，所有其他屬性都可以使用預設值。

佇列定義（在 jmsdemoJNDIClient.scf 和 jmsdemoJNDIbindings 檔案中）：

```
#-----  
# Queue Object  
# Delete the Queue if it exists  
DELETE Q(JMSDEMOQueue)  
  
# Define the Queue object  
DEFINE Q(JMSDEMOQueue) QUEUE(JMSDEMO.QL)  
  
# Display the resulting Queue object definition  
DISPLAY Q(JMSDEMOQueue)
```

除了與佇列相關的屬性之外，WebSphere MQ 主題定義還包括數個屬性，可用來與發佈/訂閱分配管理系統通訊。這些包括不同的服務類別、訂閱期限，以及發佈或讀取訊息的佇列。如果您在自己建立的佇列管理程式上執行範例，使用預設值即可。如果您在其他人管理的佇列管理程式上執行，則要由管理者提供其中部分屬性的值，特別是 BROKERPUBQ 和 BROKERSUBQ。範例定義全使用預設值。

主題定義（在 jmsdemoJNDIClient.scf 和 jmsdemoJNDIbindings 檔案中）：

```
#-----  
# Topic Object  
# Delete the Topic if it exists  
DELETE T(JMSDEMOTopic)  
  
# Define the Topic  
DEFINE T(JMSDEMOTopic) TOPIC(JMSDEMOtopic)  
  
# Display the resulting Topic definition  
DISPLAY T(JMSDEMOTopic)
```

III. 連結管理物件

前面步驟建立的管理物件必須登錄在目錄中，才能供 JMS 應用程式使用。登錄項目就等於是將 JNDI 名稱與提供者的資源「連結」起來。WebSphere MQ 提供管理工具，可在任何 JNDI 存取的目錄服務中建立和處理管理物件。範例程式碼即包含呼叫 WebSphere MQ JMS 管理工具的指令碼，以及將要使用的管理物件的定義指令碼。

- i. 開啟指令提示視窗，然後切換到 JMSDEMO 目錄。
- ii. 使用連結定義執行 JMSAdmin 指令碼，如圖 9 所示。


```
C:\WINDOWS\system32\cmd.exe
C:\JMSDEMO>JMSAdmin.bat <jmsdemoJNDIbindings.scp
```

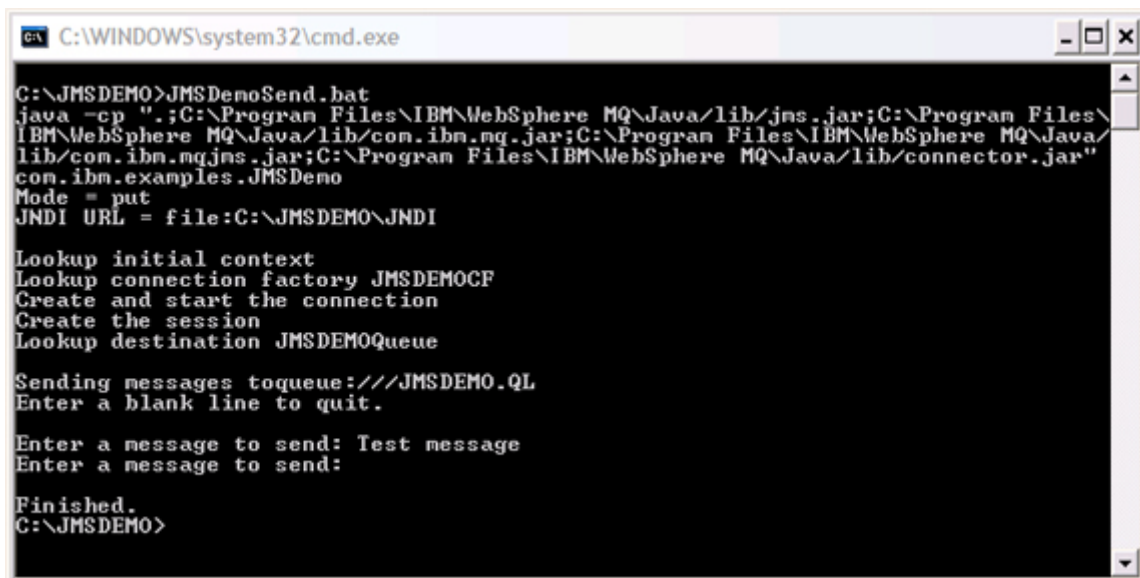
圖 9. 執行 JMSAdmin

這時您應該繼續下一個步驟並執程式碼。等到使用連結模式成功執行範例之後，您可以回到此步驟，再使用用戶端模式定義執行 JMSAdmin 指令碼。這可將範例程式碼設成透過 TCP/IP（而非透過 IPC）連線到佇列管理程式。您可以視需要在連結模式和用戶端模式之間切換，而不會影響範例應用程式。當使用用戶端模式時，SVRCONN 通道會出現在 WebSphere MQ 探險家中，並顯示狀態為 ACTIVE（作用中）。

4. 執行範例程式碼

現在已經配置好 WebSphere MQ 並在 JNDI 目錄中定義了管理物件，接下來就可以執行本文隨附的範例。若要這麼做，請遵循下列步驟：

- I. 執行點對點範例
 - i. 開啓三個指令提示視窗，每一個視窗都切換到 JMSDEMO 目錄。
 - ii. 若要執行點對點範例，請在一個視窗執行 JMSDemoSend.bat（圖 10），然後在另一個視窗執行 JMSDEMOResceive.bat（圖 11）。程式會顯示一些診斷資訊，然後等待輸入。
 - iii. 在 JMSDemoSend.bat 視窗中輸入任何想要的訊息，此訊息應會出現在 JMSDEMOResceive.bat 視窗中。輸入空白行結束。



```
C:\WINDOWS\system32\cmd.exe
C:\JMSDEMO>JMSDemoSend.bat
java -cp ".;C:\Program Files\IBM\WebSphere MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar" com.ibm.examples.JMSDemo
Mode = put
JNDI URL = file:C:\JMSDEMO\JNDI
Lookup initial context
Lookup connection factory JMSDEMOCF
Create and start the connection
Create the session
Lookup destination JMSDEMOQueue
Sending messages to queue:///JMSDEMO.QL
Enter a blank line to quit.
Enter a message to send: Test message
Enter a message to send:
Finished.
C:\JMSDEMO>
```

圖 10. JMSDemoSend.bat 視窗

```
C:\WINDOWS\system32\cmd.exe
C:\JMSDEMO>JMSDemoReceive.bat
java -cp ".;C:\Program Files\IBM\WebSphere MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar"
com.ibm.examples.JMSDemo
Mode = get
JNDI URL = file:C:\JMSDEMO\JNDI

Lookup initial context
Lookup connection factory JMSDEMOCF
Create and start the connection
Create the session
Lookup destination JMSDEMOQueue

Getting messages from queue:///JMSDEMO.QL

Got message: Test message

Finished.C:\JMSDEMO>
```

圖 11. JMSDemoReceive.bat 視窗

請注意，在第一個視窗中（圖 10），我們送出了內容為 "Test message" 的訊息，然後在第二個視窗中（圖 11），我們收到了內容為 "Test message" 的訊息。

II. 執行發佈/訂閱範例

若要執行發佈/訂閱範例：

- iv. 輸入 `strmqbrk` 指令，確定發佈/訂閱分配管理系統正在執行。
- v. 在一個視窗中執行 `JMSDemoPub.bat`（圖 12），然後在另兩個視窗中執行 `JMSDemoSub.bat`（圖 13）。
- vi. 在 `JMSDemoPub.bat` 視窗中輸入任何想要的訊息，此訊息會出現在兩個 `JMSDemoSub.bat` 視窗中。輸入空白行結束。

```
C:\WINDOWS\system32\cmd.exe
C:\JMSDEMO>JMSDemoPub.bat
java -cp ".;C:\Program Files\IBM\WebSphere MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar"
com.ibm.examples.JMSDemo
Mode = put
JNDI URL = file:C:\JMSDEMO\JNDI

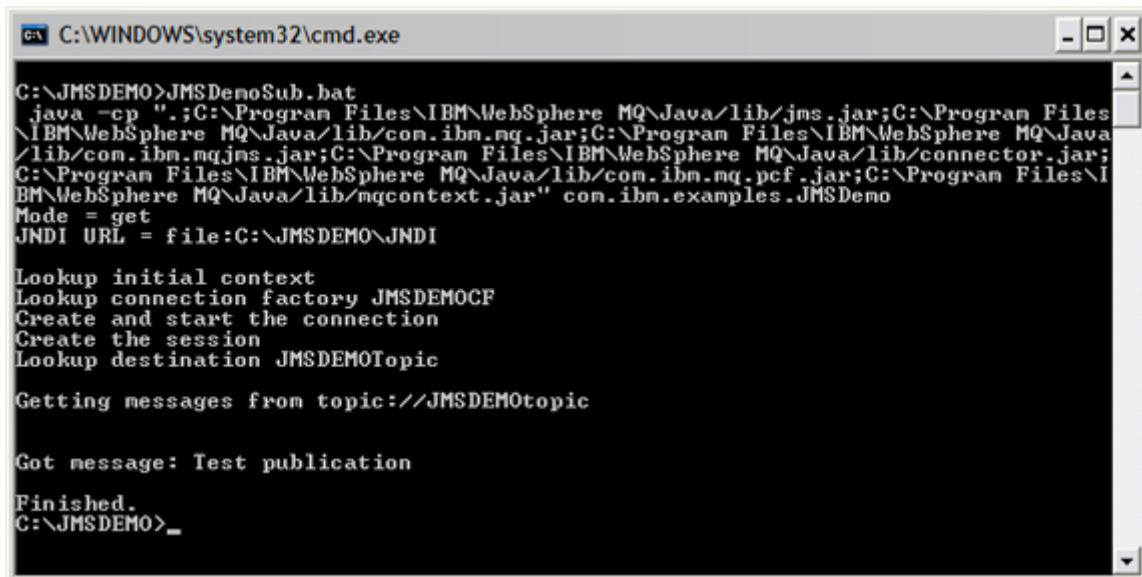
Lookup initial context
Lookup connection factory JMSDEMOCF
Create and start the connection
Create the session
Lookup destination JMSDEMOTopic

Sending messages totopic:///JMSDEMOtopic
Enter a blank line to quit.

Enter a message to send: Test publication
Enter a message to send:

Finished.
C:\JMSDEMO>_
```

圖 12. JMSDemoPub.bat 視窗



```
C:\WINDOWS\system32\cmd.exe
C:\JMSDEMO>JMSDemoSub.bat
java -cp ".;C:\Program Files\IBM\WebSphere MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\con.ibm.mq.pcf.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\mqcontext.jar" con.ibm.examples.JMSDemo
Mode = get
JNDI URL = file:C:\JMSDEMO\JNDI

Lookup initial context
Lookup connection factory JMSDEMOCF
Create and start the connection
Create the session
Lookup destination JMSDEMOTopic

Getting messages from topic://JMSDEMOTopic

Got message: Test publication

Finished.
C:\JMSDEMO>_
```

圖 13. 兩個 JMSDemoSub.bat 視窗應完全相同

請注意，在第一個視窗中（圖 12），我們送出了內容為 "Test publication" 的訊息，然後在第二個和第三個視窗中（圖 13），都收到了內容為 "Test publication" 的訊息。

結論

本文說明了 J2SE 應用程式如何使用 JMS 和 JNDI 將 WebSphere MQ 當做訊息提供者存取，而無須部署在 J2EE 應用程式伺服器（如 WebSphere Application Server）上。

這些內容包括：

- JMS 應用程式可分為三層：
 - 獨立於提供者之外的 JMS 應用程式。
 - 提供者專屬的 JMS 實作。
 - JMS 提供者。
- 應用程式如何使用兩種模式（連結和用戶端模式）附加到 WebSphere MQ。
- 應用程式如何使用 JNDI 存取 JMS 實作層中的 JMS 管理物件。
- 如何配置 WebSphere MQ。
- 如何管理 WebSphere MQ 資源，使這些資源能夠當做 JMS 管理物件並使用 JNDI 存取。
- 您現在就可以部署新的和既有 Java 應用程式來使用 WebSphere MQ 了。

特此鳴謝

作者群感謝 Graham Oakes 和 Emir Garza 協助撰寫本文。




下載

說明	名稱	檔案大小	下載方法
程式碼範例	JMSDEMO.zip	10 KB	HTTP

[下載方法相關資訊](#)

取得 [Adobe® Reader®](#)

作者簡介

	<p>T. Rob Wyatt 是 IBM WebSphere 軟體服務的 IT 專家，負責協助客戶在 WebSphere MQ 方面的管理、架構和安全作業。他先前針對 WebSphere MQ 管理所撰寫的文章已發表在 Xephon MQ Update。他曾對 NY/NJ WebSphere MQ 使用者群發表簡報，而且擔任過以前大西洋中部 MQSeries 使用者群的副總裁。T. Rob 活躍於 MQSeries Listserv 社群，而且也參與了幾個 SupportPac 的開發工作。</p>
	<p>Bobby Woolf 是 IBM WebSphere 軟體服務顧問，負責協助客戶使用 WebSphere 產品來追求成功目標。他是 Enterprise Integration Patterns 和 The Design Patterns Smalltalk Companion 兩本書的共同作者。Bobby 負責協助客戶使用 IBM WebSphere Integration Developer 開發具服務導向架構的應用程式，以在 IBM WebSphere Process Server 上使用。Bobby 也經常在各大研討會發表演說。Bobby 在 developerWorks 的部落格有更多的文章可供閱讀。</p>
	<p>Kulvir Singh Bhogal 是 IBM WebSphere 軟體服務顧問，負責為全美各地的客戶設計和導入 J2EE 解決方案。</p>