

IBM WebSphere 開發人員技術日誌：使用 WebSphere ESB 建置企業服務匯流排，第二部分

JMS 傳訊示例

等級：中級

IBM 資深顧問 IT 專家，Rachel Reinitz (rreinitz@us.ibm.com)

IBM 資深技術人員，Andre Tost (atost@us.ibm.com)

2006 年 12 月 06 日

第一部分主要介紹 IBM® WebSphere® Enterprise Service Bus (ESB，企業服務匯流排) 用於建置 ESB 的重要功能，並以一個貫穿全系列的商業情境為例，說明 WebSphere Application Server 與 WebSphere ESB 的 SIBus 功能之間的關係。第二部分示範如何以 JMS 傳送訊息的方式，將 J2EE™ 用戶端應用程式連接至 ESB，並由 ESB 記錄該訊息，然後利用 MDB (message-driven bean) 將訊息遞送給服務提供程式。因此，本文將說明 ESB 如何支援 JMS 服務要求程式及 JMS 服務提供程式。

摘自 IBM WebSphere 開發人員技術日誌。

前言

我們從第一篇 WebSphere Application Server ESB 系列文章開始，便以一個示例說明「系統整合匯流排」(System Integration Bus, SIBus) 做為應用程式伺服器的預設 JMS 提供者，文中說明以傳送 JMS 純文字訊息，可將 J2EE 應用程式用戶端連接至應用程式伺服器上所執行的 JMS 服務提供程式，並實作為 MDB。而在本文中，我們使用相同的示例，但並非從用戶端傳送訊息至 SIBus 佇列，再由 MDB 服務從佇列取得該訊息，而是透過 WebSphere ESB 中所執行的中介軟體來傳訊。

為使示例符合真實的商業環境，我們仍然以第一部分中的運輸公司為例，假設包裹順利遞送給客戶時，都必須傳送訊息到主系統，以確認遞送狀態。確認訊息是以非同步方式傳送，也就是不需要回應，訊息只會佇列在主系統中等待處理。

強化的架構

ESB 可提供虛擬服務介面，以建置分立的系統；虛擬服務介面是指客戶不會直

接存取實際的服務提供程式，而是與 ESB 交換訊息，再由 ESB 與實際服務提供程式進行訊息的遞送和接收。這種做法不只適用於 Web 服務（即透過 SOAP/HTTP 連結所提供的服務），也適用於其他任何服務。在我們的示例中，是以 J2EE 應用程式接收來自 JMS 佇列的純文字訊息，在 ESB 中，我們將這個應用程式視為服務提供程式。同樣的，我們也提供服務介面給 JMS 用戶端應用程式。

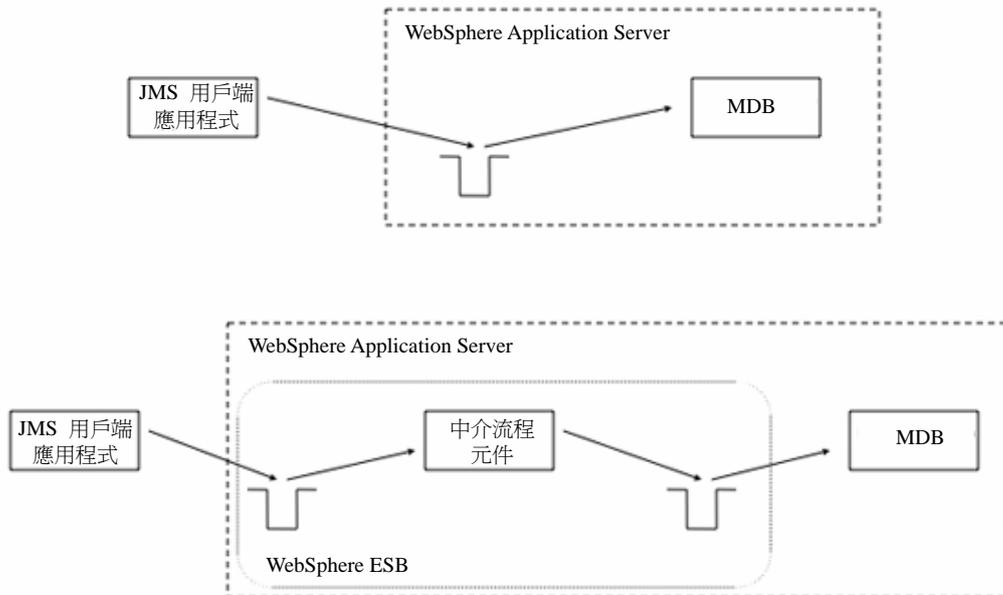
這裏我們並不詳述示例中所使用的服務要求程式及提供者程式碼（程式碼其實類似其他典型的 JMS 程式碼，請參閱[參考資料](#)取得 JMS 指導教學），只提出一些基本概念：

- J2EE 用戶端應用程式只會傳送內含遞送包裹數目的訊息。
- 由於傳送者程式碼是在 J2EE 用戶端應用程式中執行，所以使用的 JMS 資源名稱不會寫入程式碼，而是以 `java:comp/env` 名稱空間來代替。兩個名稱（一個用於連線工廠，一個用於實際佇列）的連結，是利用用戶端應用程式開發描述子中的資源參照。
- 撰寫用戶端應用程式，原本是為了傳送純文字訊息，但現在更新為可傳送 XML 訊息。
- 對 MDB 而言，所有設定更加簡化：每個 MDB 都有一個 `onMessage()` 的方法，只要訊息一抵達 MDB，所接聽的佇列便會開啓，並將訊息列印至 `System.out`。

這個做法有一點值得注意，亦即經由 WebSphere ESB 遞送訊息，可將用戶端的 JMS 配置和 MDB 配置區分開，兩者都與 WebSphere ESB 連結。如果 MDB 部署在另一部伺服器上，用戶端應用程式就不需要變動。由於訊息會經過 WebSphere ESB，所以可在 ESB 內進行中介處理。

圖 1 的設定是透過 SIBus 佇列而直接連接 JMS 要求程式和 MDB（如前一系列「第三部分」所述），並與本文中的設定相比較。

圖 1：情境示例的設定



請注意，在更新的架構中，我們使用兩個佇列：一個在用戶端和 WebSphere ESB 中介軟體之間，另一個在中介軟體和實際接收者（也就是服務提供程式）之間。圖 1 也顯示 WebSphere ESB 機制如何在 WebSphere Application Server 內持續執行；WebSphere Application Server 可提供 JMS 的執行時期，並裝載了 MDB。為了簡化起見，我們在同一部伺服器上執行 MDB 與 WebSphere ESB，但在正式作業環境中，MDB 通常是在獨立的伺服器上執行，所以需要另外設定配置。

WebSphere ESB 的中介流程元件只是「服務元件架構」(Service Component Architecture, SCA) 中所定義的另一種服務元件，SCA 需要使用服務介面定義，以描述匯流排(即用戶端應用程式將要呼叫的介面)上的服務端點。因為 SCA 使用正式的 (WSDL) 描述，所以可將 JMS 訊息的交換視為一種服務呼叫作業。（有關 SCA 的介紹，請參閱「參考資料」。）

建立與配置 ESB：簡介

在開始之前，我們先簡介本文中所執行的步驟。（如果要盡量減少必要的步驟，請見本文隨附的下載文件，其中包含可用於服務要求程式和服務提供程式的 EAR，以及一個專案交換檔案及完整的 WebSphere ESB 中介軟體。如果選擇使用專案交換檔案，就需要配置用戶端和 MDB 應用程式的部署描述子，以及伺服器本身。）

1. 建立 WebSphere ESB 伺服器。

2. 建立服務介面。使用 **WebSphere Integration Developer** 來建立服務介面的 **WSDL** 描述，以供服務要求程式傳送訊息至匯流排，並使用 **MDB** 做為服務提供程式。
3. 建立中介軟體。建立中介模組，並建置中介流程元件。中介軟體的匯出和匯入，都是利用上述步驟 2 所建立的服務介面來進行配置。匯出和匯入之間會建立 **JMS** 連結，中介軟體將會記錄經過的訊息。
4. 設定服務要求程式。修改用於傳送簡單文字字串的原始測試用戶端，以傳送 **XML** 訊息。設定用戶端應用程式和 **WebSphere Application Server** 的 **JMS** 配置，其中 **WebSphere ESB** 在 **WebSphere Application Server** 上執行，也與用戶端相連。**WebSphere ESB** 所產生的 **SIBus** 佇列將做為配置的一部分。
5. 設定服務提供程式。使用 **WebSphere ESB** 所產生的 **SIBus** 佇列來修改 **MDB EAR** 配置。
6. 執行端對端測試。

版本

請使用最新版的 **WebSphere Integration Developer** 和 **WebSphere Enterprise Service Bus**。本文使用的是 **WebSphere Integration Developer 6.0.1.2** 版和 **WebSphere ESB 6.0.1.2** 版。2006 年 12 月底推出具備重要功能和效能加強的修正套件，可將這兩項產品升級到 6.0.2 版，因此在推出修正套件後請使用 6.0.2 版。

A. 建立 **WebSphere ESB** 伺服器。

在 **WebSphere Integration Developer** 中建立一個用於測試的伺服器。

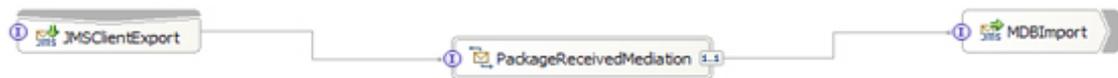
1. 在 **WebSphere Integration Developer** 的 **Servers** 畫面上選取 **New => Server**。
2. 選取 **WebSphere ESB Server v6.0**，接受預設值，然後選取 **Finish**。

如果已經建立了測試伺服器，可以重複使用，但請注意，其中可能有預先存在的機制會導致衝突。

B. 建立服務介面。

服務元件會利用匯入和匯出，與外部合作夥伴進行交互運作。在我們的案例中，匯出會與 JMS 用戶端應用程式交互運作，匯入則與 MDB 交互運作。

圖 2：SCA 的中介組件



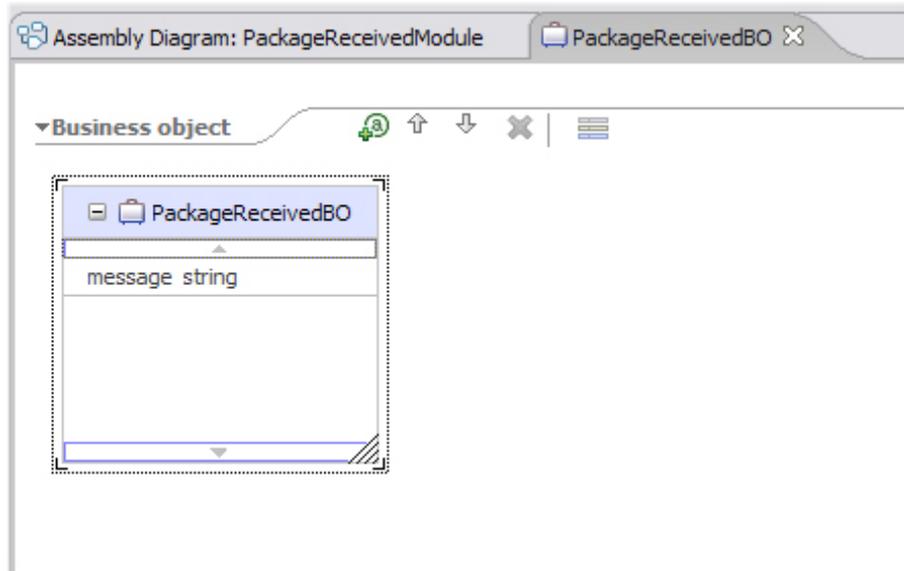
匯出和匯入都需要使用介面定義，以明確描述交換資料的格式。匯出和匯入也都需要連結，以說明要用於傳送資料的基礎通訊協定詳細資訊。在本例中，匯入和匯出介面是相同的。

我們先從介面開始，在原先的例子中，只能傳送一個字串（即「已收到包裹」通知），而在更新版本中，則定義 XML 綱目（其中包含訊息結構定義），讓所傳送的字串正式化，這樣中介軟體和最終接收程式能夠更容易處理訊息。我們使用 WebSphere Integration Developer 中的商業物件編輯器和介面編輯器，來建立介面。商業物件編輯器是用於說明訊息內容，介面編輯器則用於說明如何將訊息封裝到作業封包中。這些圖形化工具可產生用以描述介面的綱目和 WSDL，並提供給服務要求及提供程式。

1. 在 Business Integration 視圖中開啓 WebSphere Integration Developer。
2. 建立新的中介模組，命名為 PackageReceivedModule。中介模組是中介元件與其內建邏輯的儲存器，並與封面下的可部署 EAR 專案相對映。
3. 在樹型導覽表中選取 **Data Types**（資料類型）節點，以開啓商業物件編輯器。用滑鼠右鍵按一下該節點，即可建立新的商業物件。
4. 在新增商業物件精靈中，將商業物件命名為 PackageReceivedBO，並保留其他欄位中的預設值。按一下**完成**。
5. 開啓商業物件編輯器後，在商業物件中加入 message（訊息）這個屬性，然後將類型設定為 **string**（字串）。最終定義如圖 3 所示。

以上的說明非常簡略，並假設您之前已使用過這個工具，熟悉基本用法。有關如何在 WebSphere Integration Developer 中建立機制，相關文章及指導教學請見參考資料。

圖 3：商業物件定義



訊息經過 WebSphere ESB 時，訊息屬性會儲存訊息的有效負載。另外，商業物件定義會儲存在 .xsd 檔案中做為 XML 綱目，您可以在 XML 綱目編輯器中修改內容。不過本文的例子中並不需要進行修改。

6. 儲存所有變更。
7. 現在您已經準備好建立真正的服務介面。用滑鼠右鍵按一下樹型導覽表中的 **Interfaces**（介面）節點，然後選取 **Create a new Interface**（建立新介面）。
8. 將新介面命名為 **PackageReceivedIF**，並保留其他預設值。按一下**完成**，介面編輯器隨即開啓。
9. 在介面編輯器中，選取**新增單向作業**，並命名為 **package_received**。
10. **新增**輸入參數，然後將商業物件類型 **PackageReceivedBO** 命名為 **packageReceivedBO**（選取您在上述步驟中所建立的類型）。介面現在應該如圖 4 所示。

圖 4：介面定義



11. 儲存所有變更。

若要檢視所產生的 WSDL，請選取 **PackageReceivedIF**，然後選取 **Open with => XML Source Page Editor**。同樣的，您也可以開啓 **PackageReceivedBO** 綱目。

現在您已經準備好建置真正的中介元件。

C. 建立中介軟體。

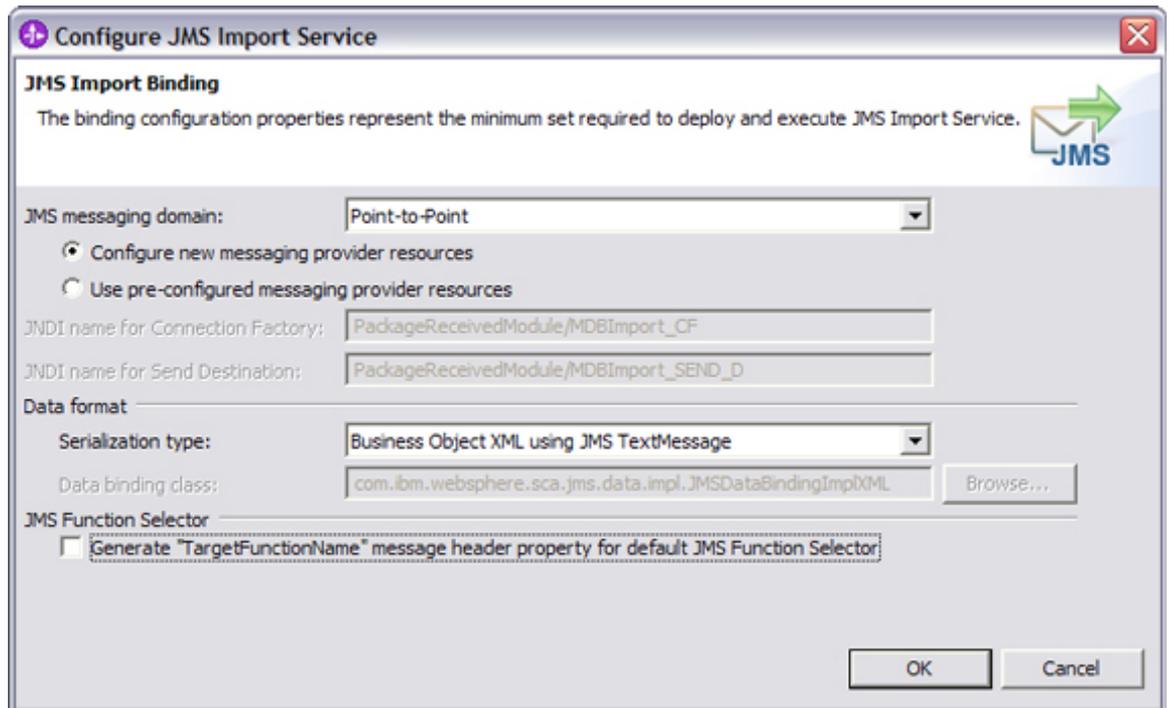
1. 在 **WebSphere Integration Developer** 中，按兩下

PackageReceivedModule 組件圖示 ，以開啓組件編輯器。隨即建立名為 **Mediation1** 的預設中介流程元件。將這個元件重新命名為 **PackageReceivedMediation**。

2. 接下來，必須建立一個匯入項，以連接中介軟體和實際服務提供程式(在我們的案例中，即指 **MDB**)。從選用區將匯入項拖放到中介流程元件的右側。將匯入項重新命名為 **MDBImport**。用滑鼠右鍵按一下匯入項，然後選取 **Add Interface** (新增介面)。
3. 在下一個對話框中，選擇您之前建立的 **PackageReceivedIF** 介面。簡單的說，就是指派了建立的介面以傳送訊息至 **MDB**，使 **ESB** 送出的訊息都要與這個介面相符。
4. 使用 **wire tool** 將匯入項連接至中介流程元件。(出現任何蹦現視窗詢問是否要在中介流程元件上產生參照時，請按一下 **OK**。)
5. 因為服務提供程式將會實作為 **MDB**，所以我們能夠透過 **JMS** 與其通訊，也因此，必須在匯入項上定義 **JMS** 連結。用滑鼠右鍵按一下匯入項，然後選取 **Generate Binding... => JMS Binding** 功能表選項。在出現的對話框中，可以定義一組重要的值，如圖 5 所示。

本節說明如何重新建立必要的中介模組。然而，如果您想略過這個步驟，可以匯入本文下載檔中隨附的備妥模組，其中包含了 WebSphere Integration Developer 專案交換檔，檔名為 PackageReceivedModule.zip，此檔案內含完整的解決方案。

圖 5：配置 JMS 匯入服務



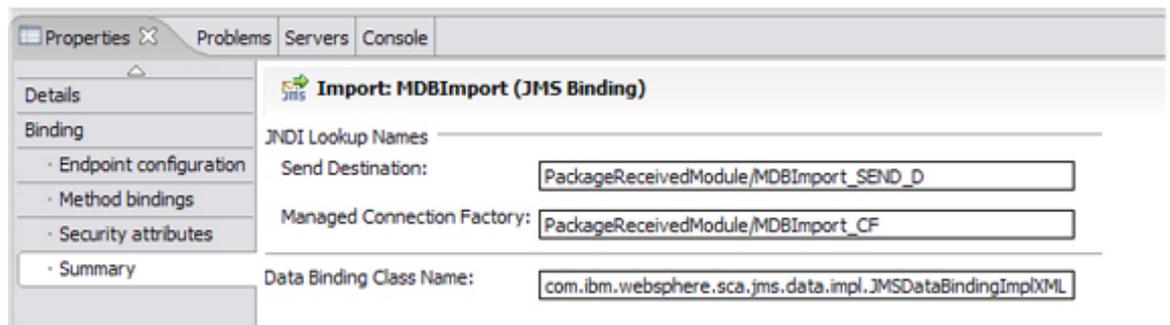
- **JMS messaging domain (JMS 傳訊網域)**：保留預設值 **Point-to-Point (點對點)**，因為我們會使用特定佇列來傳送訊息，而不是使用發布與訂閱 (pub/sub) 模型。
- **Configure new messaging provider resources (配置新的傳訊提供程式資源)**：選取此選項，表示所有相關的 **JMS** 資源，包括佇列與個別的 **SIBus** 機制，都會在您部署中介模組到伺服器時自動產生。
- **Serialization type (序列化類型)**：經過中介軟體的訊息透過匯出項接收時，會轉換成商業物件；然後透過匯入項傳出時，會轉換成適當的輸出格式。以 **JMS** 為例，您必須告訴系統要使用哪種類別來進行這項轉換。**WebSphere ESB** 提供幾組預先建置的類別，專門用於將格式化 **XML** 訊息與商業物件格式互相轉換。

在我們的案例中，是使用一個類別來取得 **JMS TextMessage**，並轉換成商業物件。該文字訊息必須遵循特定的格式，如此轉換器類別才能判斷所要使用的正確商業物件。後文將繼續討論此主題。

（**WebSphere ESB 6.0.2** 版支援不含 **XML** 的 **JMS** 訊息，使訊息得以對映至商業物件，同時又能夠擁有任意內容。如需為專用的 **JMS** 內容建立序列化邏輯，詳見「[使用 JMS 和 WebSphere ESB 以建置強大可靠的 SOA](#)」。）

- **JMS Function Selector (JMS 函數選取器)**：還記得我們說明如何將接收 **JMS** 訊息的應用程式視為一種服務嗎？這表示該應用程式能夠支援單或多項作業。以 **JMS** 型服務為例，多個作業會重覆使用同一個佇列，因此，您必須告訴系統哪種訊息類型該與哪種服務作業相對映。**WebSphere ESB** 利用名為 **TargetFunctionName** 的 **JMS** 標題欄位來達成這個目的，該標題欄位中包含了傳出訊息所呼叫的作業名稱。接收應用程式（在我們的案例中，即指 **PackageReceivedMDB**）現在只要檢測這標題，就能分辨出各種作業。然而，在我們的情境示例中，不需要使用這個標題，因為 **MDB** 只能支援一項作業。因此，請取消勾選這個選項。
6. 如上所述，設定所有數值之後，請按一下 **OK** 以建立連結。
 7. 選取組件編輯器中的匯入項，然後前往工具底端的 **Properties**（內容）標籤。在 **Properties**（內容）檢視中，選取 **Summary**（摘要）標籤。現在檢視畫面應該如圖 6 所示。

圖 6：匯入內容

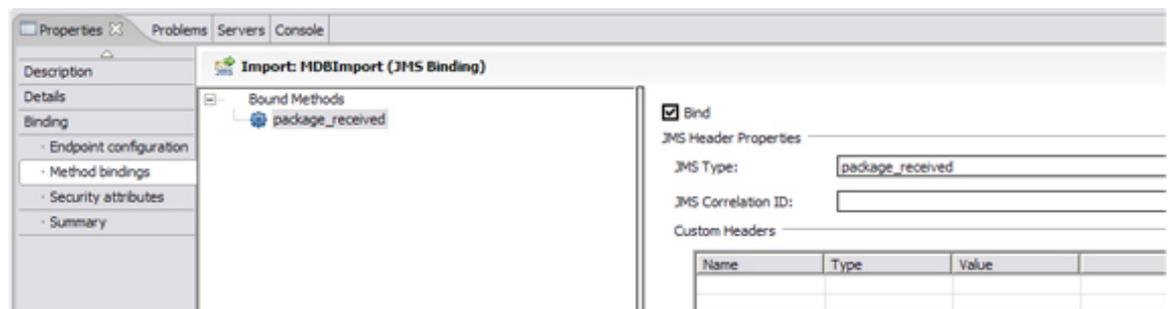


請注意，**send**（傳送）佇列的名稱是設定為

PackageReceivedModule/MDBImport_SEND_D。這個名稱是在建立連結時自動產生的，部署時還會產生個別機制。後面配置 MDB 時，將需要使用這個佇列名稱。

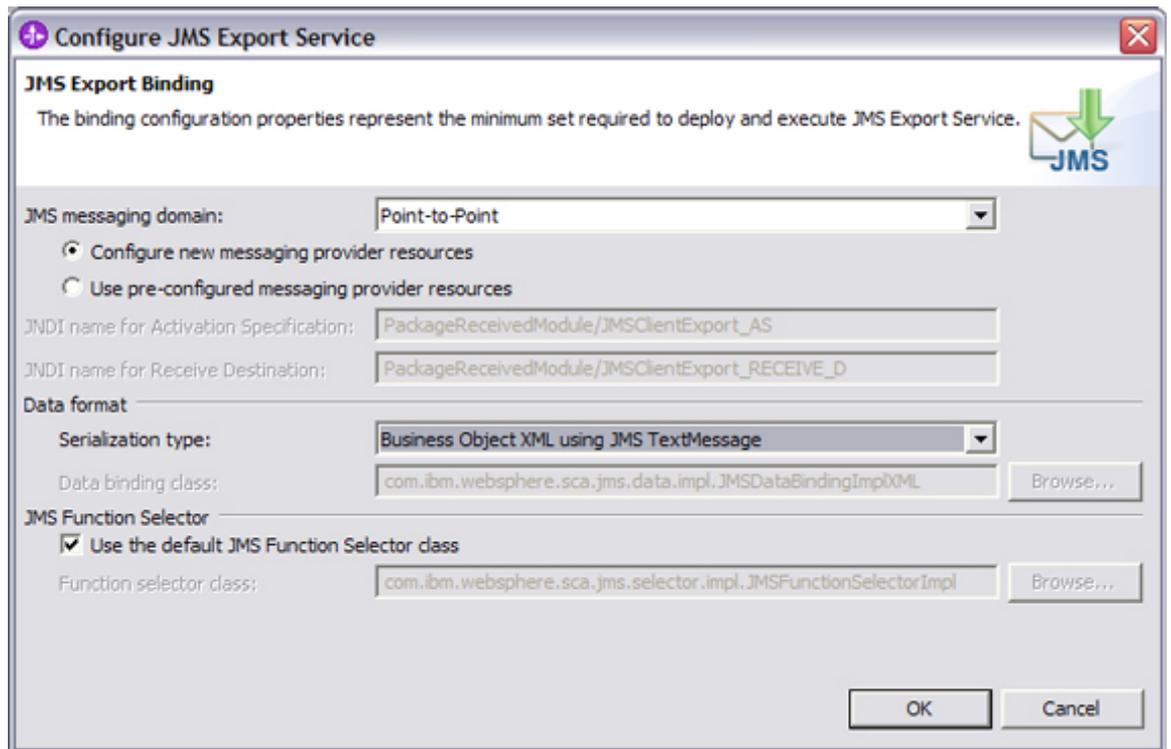
8. 還有一件事，傳出訊息必須定義適當的 JMSType。MDB 只能處理 JMSType 標題欄位設定為 package_received 的訊息（該名稱定義在 MDB 的部署描述子中）。在 Properties 檢視的 **Method bindings**（方法連結）標籤中，定義 JMSType，如圖 7 所示。

圖 7：方法連結



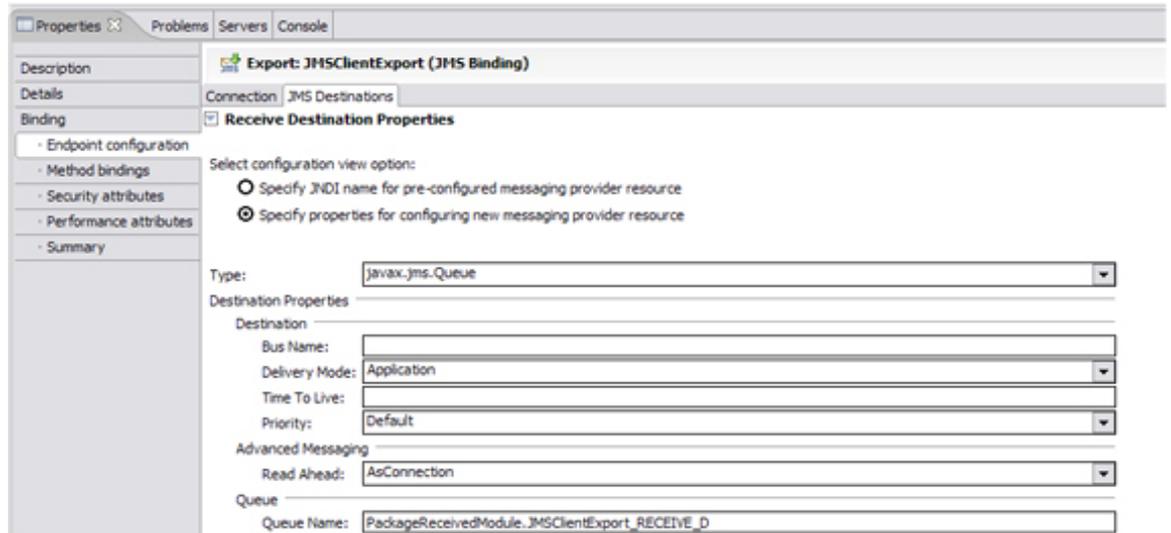
9. 返回組件編輯器，將匯出項拖放至中介元件的左側，並命名為 JMSClientExport。這個匯出項會向 JMS 測試用戶端顯示中介軟體。
10. 將 PackageReceivedIF 介面新增至匯出項，然後連接到中介流程元件。這個動作也會在中介流程元件中建立一個介面。請使用 wire tool 將這個匯出項連接到 PackageRecievedMediation。
11. 與匯入項類似，我們需要連結 JMS 與匯出項，因為用戶端是透過連結與中介軟體通訊的。用滑鼠右鍵按一下匯出項，然後選取 **Generate Binding... => JMS Binding** 功能表選項。出現的對話框幾乎與匯入項的完全相同，您在這裡需要使用相同的定義，只有一個地方不同，如圖 8 所示。

圖 8：匯出項的 JMS 連結



12. 在 **Serialization type** 一欄，請務必選取 **Business Object XML using JMS TextMessage**（使用 **JMS TextMessage** 的商業物件 XML）。匯出項將會使用預設的 **JMS** 函數選取器類別。如上所述，傳入的 **JMS** 訊息必須對映到服務介面上的特定作業，且預設選取器類別要求將 **TargetFunctionSelector JMS** 標題欄位設定為所呼叫作業的名稱。更新下列用戶端應用程式，以設定這個 **JMS** 標題欄位。
13. 按一下 **OK** 即可產生連結，並查看其 **Properties** 檢視。
14. 選取 **Binding - Endpoint configuration**（連結 - 端點配置）標籤內的 **JMS Destinations**（**JMS** 目的地）標籤。展開 **Receive Destination Properties**（接收目的地內容）。檢視畫面應該如圖 9 所示。

圖 9：接收目的地內容



請注意，中介軟體用來接收訊息的佇列名稱設定為 "PackageReceivedModule.JMSClientExport_RECEIVE_D"。我們將使用此佇列名稱來配置用戶端應用程式。

15. 儲存所有變更之後，即可進行最後一個步驟，就是建立流程元件的實作。用滑鼠右鍵按一下組件編輯器中的 **PackageReceivedMediation**，然後選取 **Generate Implementation**（產生實作），中介流程編輯器隨即開啓。
16. 本例中僅使用非常簡單的中介邏輯：記錄經過中介軟體的每則訊息，讓您能夠看到經過 ESB 的所有訊息。請將左側 **PackageReceivedIF**（亦即中介流程元件連接至匯出項的介面）上的 **package_received** 作業，連接至 **PackageReceivedIFPartner**（亦即中介流程元件連接至匯入項的參照）上的 **package_received** 作業。在編輯器的下半部，將 **Message Logger**（訊息記錄器）的中介基本節點拖放到畫面上，並設定訊息流程路線，使每則訊息都能經過記錄器，如圖 10 所示。

圖 10：設定路線使每則訊息都能經過記錄器



17. 儲存中介流程編輯器和組件編輯器中的所有變更，中介模組已經完成，可以部署到伺服器了。開始部署之前，還需要調整要求程式碼及提供程式的配置。

D. 設定服務要求程式。

要完成作業，需要執行下列一般步驟：

- 將 J2EE 測試用戶端應用程式匯入至 WebSphere Integration Developer。
- 將部署描述子中的佇列名稱配置為 WebSphere ESB 中介模組匯出項所產生的佇列名稱。
- 根據 WebSphere ESB 需求，更新應用程式碼，以便傳送 XML 訊息及設定 JMSHeader TargetFunctionName。
- 在 WebSphere ESB 伺服器中建立 JMS 連線工廠，以供使用者取得連線。

以下是詳細執行資料：

1. 匯入位於 PackageReceivedClient.ear 檔案中的測試用戶端應用程式。這是我們在上篇文章中所用的 EAR 檔的更新版本，更新說明如下：
 - a. 選取 **Import => EAR** 時，如果尚未啟用「基礎 J2EE 支援」，將會要求啟用。如果出現要求訊息，請回答 OK，然後切換至 J2EE 視圖。
 - b. 在匯入期間，必須將 EAR 專案命名為 **PackageReceivedClientEAR**，並選取 **WebSphere ESB Server 6.0 版** 做為目標伺服器。名為 **PackageReceivedClient** 的新應用程式用戶端專案中有一個 **Main.java** 檔，其中包含用於測試用戶端的程式

碼。

2. 接下來，請變更佇列名稱，讓佇列傳送訊息至 WebSphere ESB 匯出項，而非傳送至 MDB 所接聽的佇列。這個名稱會參照至用戶端專案的部署描述子，因此按兩下即可開啓這部署描述子。部署描述子中包含名為 `.jms/PackageReceivedQueue` 的參照，請將其 **WebSphere Bindings => JNDI 名稱變更爲** `PackageReceivedModule/JMSClientExport_RECEIVE_D`，這是 WebSphere ESB 為匯出項所產生的佇列名稱。
3. 在原本的測試用戶端中，我們傳送給佇列的是純文字字串，而在更新的情境中，則是傳送 XML 格式的字串，以代表中介模組中定義的商業物件。如此可讓預設的序列化程式將 XML 訊息轉換成商業物件，便於在中介軟體中操作。根據您之前建立的介面和商業物件定義綱目，XML 字串應如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<ns: package_received
xml ns: ns="http://PackageReceivedModule/PackageReceivedIF">
  <packageReceivedB0>
    <message>Package Received - 24595023</message>
  </packageReceivedB0>
</ns: package_received>
```

因此，更新的 `Main.java` 檔將會傳送這則新訊息：

```
// private final static String messageText = "Package Received - 24595023";

private final static String messageText = "<?xml version=\"1.0\"
encoding=\"UTF-8\"?> " +
  "<ns: package_received
xml ns: ns=\"http://PackageReceivedModule/PackageReceivedIF\"> " +
  "  <packageReceivedB0>" +
  "  <message> Package Received - 24595023</message> " +
  " </packageReceivedB0> " +
  " </ns: package_received> ";
```

此外也請記住，在 WebSphere ESB 中使用的是預設 JMS 函數選取器，並會在服務介面上自行挑選適當作業以進行呼叫。您必須新增 JMS

標題欄位 (TargetFunctionName) ，以指明所要呼叫的作業 (package_received) :

```
...
// Create MessageProducer and TextMessage
MessageProducer queueSender = session.createProducer(q);
TextMessage outMessage = session.createTextMessage();
outMessage.setText(messageText);

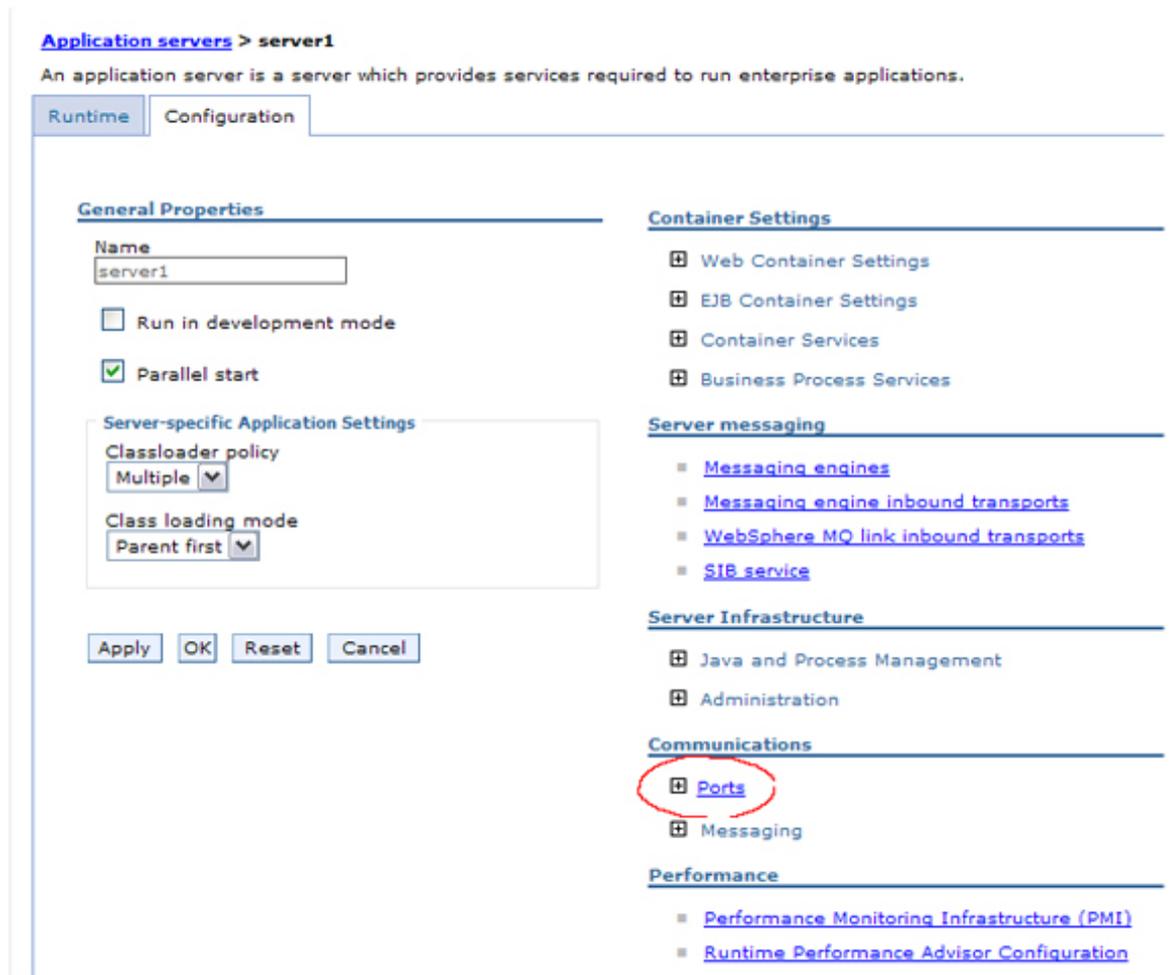
// set target operation name for WESB default JMS function selector
outMessage.setStringProperty("TargetFunctionName",
                             "package_received");

// Set type and destination and send
outMessage.setJMSType("package_received");
outMessage.setJMSDestination(q);
...
```

由於已經對程式碼做了上述的變更，所以此時不需要採取任何動作。

4. 您也需要執行部分 JMS 配置，讓 J2EE 用戶端應用程式（服務要求程式）能夠使用其 JMS 資源與 WebSphere ESB 伺服器連線。前往 Servers（伺服器）檢視畫面，開啓測試伺服器的管理主控台（如果尚未啓動伺服器，請啓動）。用滑鼠右鍵按一下 **WebSphere ESB v6.0** 伺服器，然後選取 **Run administrative console**（執行管理主控台）。
5. J2EE 用戶端應用程式會使用 JMS 連線工廠來連線至 JMS 佇列。在伺服器上建立這個連線工廠，以供用戶端應用程式從連線工廠取得連線。用戶端隨即開啓連線，與伺服器上的 JMS 佇列相連。請注意，用戶端使用特定連接埠與伺服器上的傳訊引擎進行通訊，預設連接埠是 7276。然而，工具內所執行的測試伺服器很可能使用不同的連接埠，因此您必須使用正確的埠號來配置連線工廠。若要找出伺服器所使用的埠號，請前往管理主控台，然後按一下 **Servers => Application Servers...** 節點。選取名為 **server 1** 的伺服器。現在畫面上會顯示該伺服器的相關資訊，包括名為 Ports 的鏈結，如圖 11 所示。

圖 11：尋找伺服器所用的連接埠



選取 **Ports** 鏈結以擷取伺服器正在使用的連接埠清單。JMS 通訊所用的埠號名稱是 `SIB_ENDPOINT_ADDRESS`，在圖 12 中，埠號是 7278。

圖 12：伺服器所用的連接埠

Application servers

Application servers > server1 > Ports

Configure important TCP/IP ports which this server uses for connections.

Preferences

New Delete

| Select | Port Name | Host | Port | Transport Details |
|--------------------------|---------------------------------------|-----------------------|------|--|
| <input type="checkbox"/> | BOOTSTRAP_ADDRESS | atost.rchland.ibm.com | 2811 | No associated transports |
| <input type="checkbox"/> | CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS | atost.rchland.ibm.com | 9409 | No associated transports |
| <input type="checkbox"/> | CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS | atost.rchland.ibm.com | 9408 | No associated transports |
| <input type="checkbox"/> | DCS_UNICAST_ADDRESS | * | 9355 | View associated transports |
| <input type="checkbox"/> | ORB_LISTENER_ADDRESS | atost.rchland.ibm.com | 9102 | No associated transports |
| <input type="checkbox"/> | SAS_SSL_SERVERAUTH_LISTENER_ADDRESS | atost.rchland.ibm.com | 9407 | No associated transports |
| <input type="checkbox"/> | SIB_ENDPOINT_ADDRESS | * | 7278 | View associated transports |
| <input type="checkbox"/> | SIB_ENDPOINT_SECURE_ADDRESS | * | 7288 | View associated transports |
| <input type="checkbox"/> | SIB_MQ_ENDPOINT_ADDRESS | * | 5560 | View associated transports |
| <input type="checkbox"/> | SIB_MQ_ENDPOINT_SECURE_ADDRESS | * | 5580 | View associated transports |
| <input type="checkbox"/> | SOAP_CONNECTOR_ADDRESS | atost.rchland.ibm.com | 8882 | No associated transports |
| <input type="checkbox"/> | WC_adminhost | * | 9062 | View associated transports |
| <input type="checkbox"/> | WC_adminhost_secure | * | 9045 | View associated transports |
| <input type="checkbox"/> | WC_defaulthost | * | 9082 | View associated transports |
| <input type="checkbox"/> | WC_defaulthost_secure | * | 9445 | View associated transports |

Total 15

請記下此埠號。

6. 現在，請在管理主控台中導覽至 **Resources => JMS Providers => Default Messaging**。按一下 **JMS** 連線工廠的鏈結，再按一下 **New**（新建）。輸入下列數值：

- Name（名稱）：TheConnectionFactory
- JNDI name（JNDI 名稱）：jms/TheConnectionFactory
- Bus name（匯流排名稱）：SCA.APPLICATION.esbCell.bus
- Provider endpoints（提供程式端點）：
localhost:7278:BootstrapBasicMessaging

Provider endpoints（提供程式端點）定義中應該包含傳訊引擎的埠號（即之前所查詢的資訊）；在我們的示例中是 7278。（特定 Cell 的名稱會因個別配置而有所不同，所以不同安裝之間的預設 SCA 應用程式匯流排名稱也可能不同。）

有關連線至非預設 JMS 提供程式連接埠，詳見 [WebSphere Application Server 資訊中心](#)。

其他各欄位保留預設值，然後按一下 **OK**。

7. 儲存所有變更，然後關閉管理主控台。

E. 設定服務提供程式

如前所述，用來擷取 JMS 訊息的服務提供程式是以 MDB 所實作的。我們使用前一系列文章中相同的應用程式，適當的企業保存檔 (EAR) 已包含在本文隨附的下載文件中。

要完成作業，需要執行下列一般步驟：

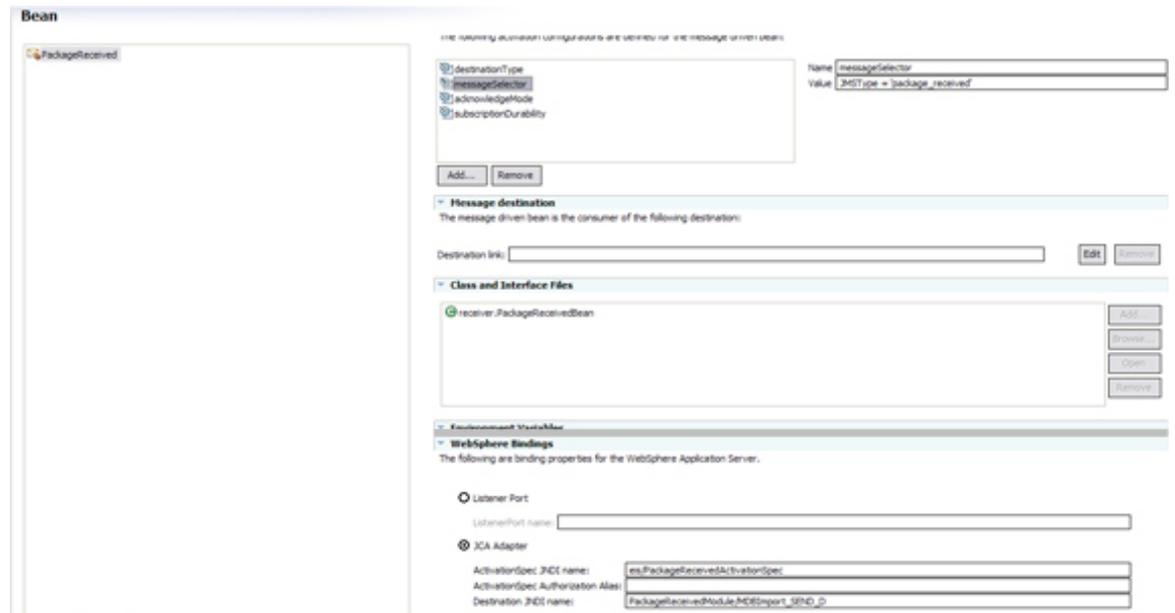
- 將具有 MDB 的 EAR 匯入至 WebSphere Integration Developer。
- 配置部署描述子，以接聽 WebSphere ESB 中介模組匯入項所產生的佇列。
- 在 WebSphere ESB 伺服器上為 MDB 建立 ActivationSpec (因為是執行 MDB EAR 的同一個伺服器)。

以下是詳細執行資料：

1. 在 WebSphere Integration Developer 中，切換至 J2EE 視圖。在匯入 PackageReceived.ear 檔案之前，請先在 WebSphere Integration Developer 中啟動 WebSphere ESB 測試伺服器 (若尚未啟動的話)。
 - a. 匯入 EAR 檔時，必須將 EAR 專案命名為 PackageReceivedEAR。
 - b. 選取 **WebSphere ESB Server v6.0** 做為專案的目標伺服器。其他各欄位保留預設值。
 - c. 完成匯入後，應該會看到名為 PackageReceived 的新 EJB 專案，其中包含一個名為 PackageReceived 的 MDB。
2. 變更 MDB 的部署描述子，並配置從 WebSphere ESB 放置訊息的佇列處取得訊息。按兩下部署描述子，即可在編輯器中開啓。
3. 在 PackageReceived 的 **Bean** 標籤中，移除 Message destination (訊息目的地) 輸入項。您也必須變更佇列的 **WebSphere Bindings => Destination JNDI** 名稱，以便從

PackageReceivedModule/MDImport_SEND_D（亦即 WebSphere ESB 為中介模組匯入項所產生）擷取訊息。（您會看到 messageSelector 的值已設定為 JMSType，與中介軟體的匯入連結中的設定一樣。）現在部署描述子應該如圖 13 所示。

圖 13：已更新的 MDB 部署描述子



如果在匯入之後看到其他編譯錯誤，請選取 **Project => Clean...** 以重新建置專案，解決那些錯誤。

4. 您也需要在執行 MDB 的伺服器上採用部分 JMS 配置，這樣就能簡化 WebSphere ESB 伺服器。請前往 **Servers** 檢視畫面，開啓測試伺服器的管理主控台。用滑鼠右鍵按一下 **WebSphere ESB 6.0** 版伺服器，然後選取 **Run administrative console**（假設伺服器已經啓動；若未啓動，請現在啓動）。
5. 服務提供程式會使用 MDB 從 ESB 處擷取訊息。這個 MDB 配置為內含佇列定義的啓動規格，這些都是標準的 J2EE 作業，因此在部署應用程式之前，必須先配置啓動規格。在管理主控台中，展開 **Resources => JMS Providers => Default messaging**。
 - d. 在畫面右下角，按一下名為 **JMS activation specification**（JMS 啓動規格）的鏈結。
 - e. 在下一個畫面中，按一下 **New**，然後輸入下列新啓動規格：
 - Name（名稱）：PackageReceivedActivationSpec

- JNDI name (JNDI 名稱): eis/PackageReceivedActivationSpec
- Destination JNDI name (目的地 JNDI 名稱):
PackageReceivedModule/MDBImport_SEND_D
- Bus name (匯流排名稱): SCA.APPLICATION.esbCell.bus

(同樣的，實際的匯流排名稱可能因個別配置而有所不同。)

圖 14：啟動規格

[Default messaging provider](#) > [JMS activation specification](#) > **New**

A JMS activation specification is associated with one or more message-driven beans and provides

Configuration

General Properties

| Administration | | Related Items |
|----------------|--|---------------|
| * Scope | <input type="text" value="cells:esbCell:nodes:esbNode"/> | |
| * Name | <input type="text" value="PackageReceivedActivationSpec"/> | |
| * JNDI name | <input type="text" value="eis/PackageReceivedActivatic"/> | |

| Destination | |
|--------------------------------|---|
| * Destination type | <input type="text" value="Queue"/> |
| * Destination JNDI name | <input type="text" value="PackageReceivedModule/MDBImport_SEND_D"/> |
| Message selector | <input type="text"/> |
| Bus name | <input type="text" value="SCA.APPLICATION.esbCell.Bus"/> |
| Acknowledge mode | <input type="text" value="Auto-acknowledge"/> |
| Target inbound transport chain | <input type="text"/> |

6. 保留其他所有預設值，然後按一下 **OK**。儲存所有變更。

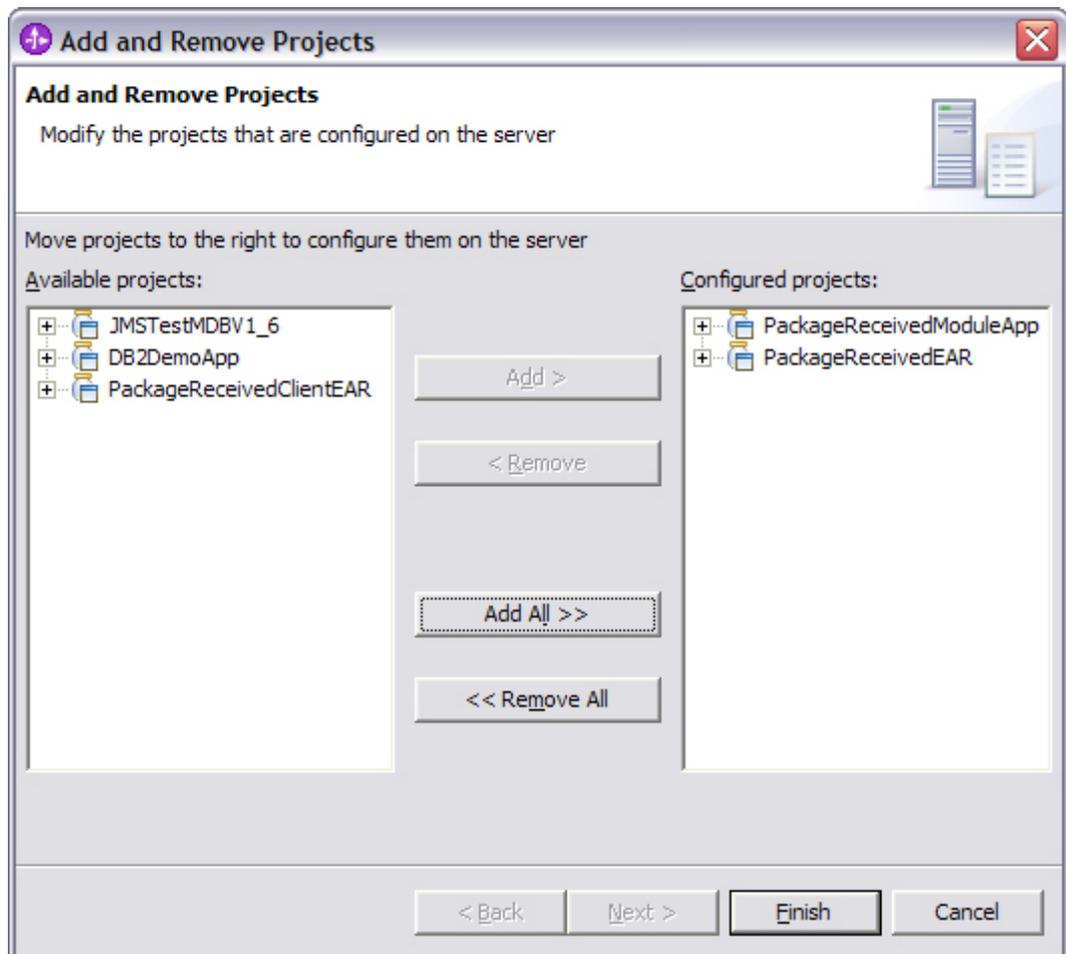
以上就是服務提供程式所需要的各項設定。

F. 執行端對端測試。

現在終於準備好，可將完整的示例部署到測試伺服器，並開始執行。

1. 在 WebSphere Integration Developer 中，請從上方功能表選取 **Project => Clean... => Clean all projects => OK**。這個動作可以讓專案所產生的程式碼保持一致。在 Servers（伺服器）檢視畫面中，用滑鼠右鍵按一下 **WebSphere ESB Server v6.0**，然後選取 **Add and remove projects...**（新增和移除專案）。
2. 在出現的對話框中，選取 **PackageReceivedModuleApp**，然後按一下 **Add >**。
3. 接著，將 **PackageReceivedEAR** 新增至伺服器，然後按一下 **Finish**。以這個順序新增，可以讓 **PackageReceivedModuleApp** 最先載入，因為它會設定 **MDB EAR** 所用的目的地。如果載入專案的順序錯誤，在載入 **ActivationSpec** 時可能出現「找不到目的地」的錯誤訊息。

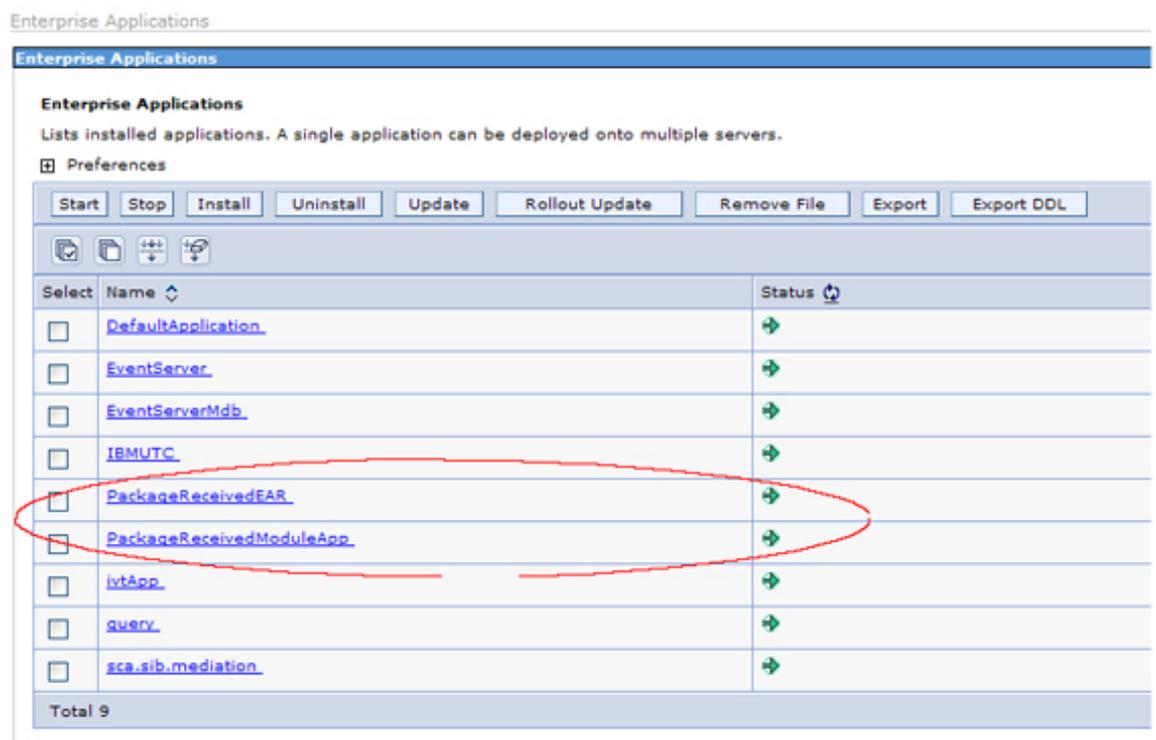
圖 15：將可用的專案加入至已配置的專案中



請注意，用戶端應用程式並不會部署到伺服器，因為它是以用戶端來執行，所以並不需要部署。

4. 完成發佈步驟之後，請使用工具中的 **Restart**（重新啟動）或 **Stop and Start**（停止並啟動）按鈕，以停止伺服器，然後重新啟動。重新啟動後，兩個應用程式應該都能順利啟動。要確認應用程式是否順利啟動，一個方法是再次啟動管理主控台，然後選取 **Applications => Enterprise Applications**。此時應該會顯示一份應用程式清單，內含您剛才安裝的兩個應用程式，且所有應用程式應該都已啟動（圖 16）。

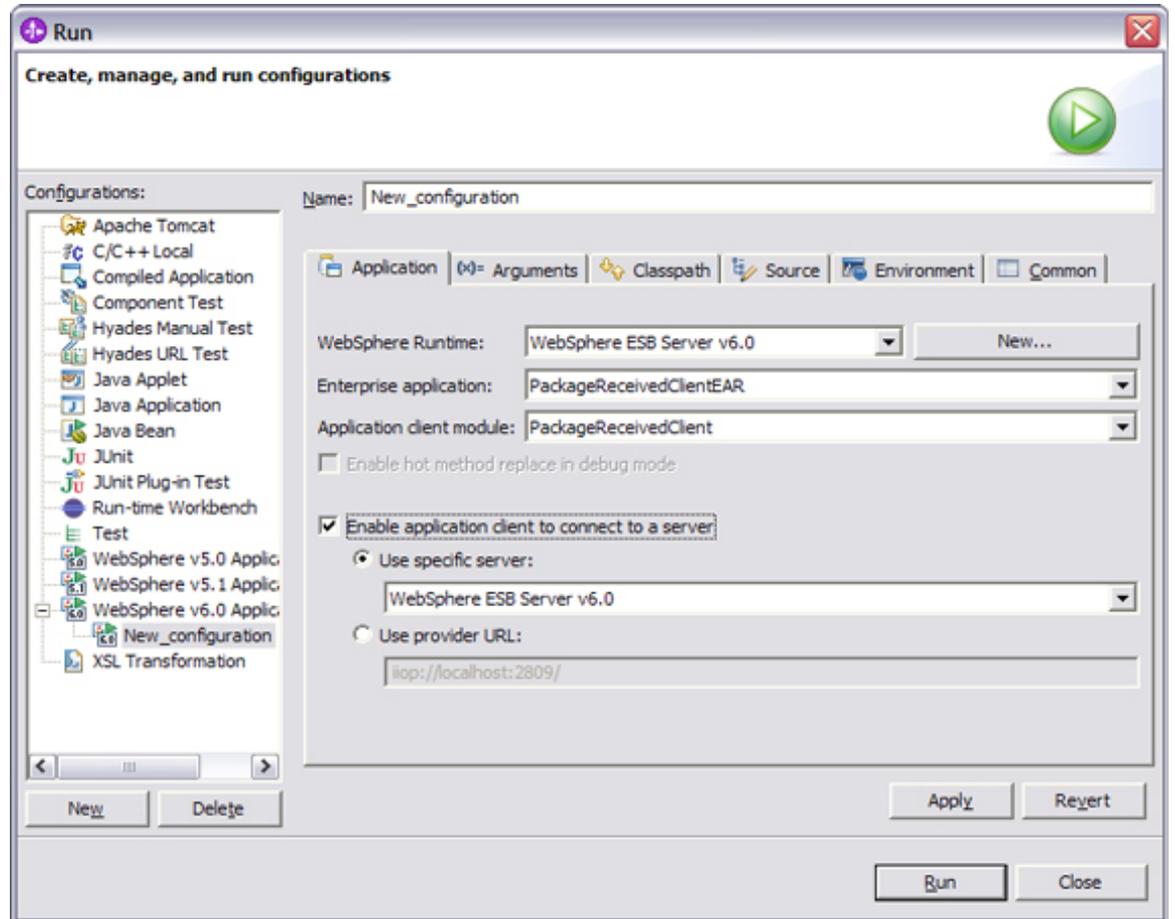
圖 16：應用程式狀態



5. 若要執行測試用戶端應用程式，請從 WebSphere Integration Developer 主功能表選取 **Run => Run...**。
6. 在出現的對話框中，選取 **WebSphere v6.0 Application Client** 配置，然後按一下 **New**。
7. 在新的配置中，請選取 **WebSphere ESB Server v6.0** 做為 WebSphere 執行時期，並輸入 **PackageReceivedClientEAR** 做為企業應用程式，同時使用 **PackageReceivedClient** 做為應用程式用戶端模組。此外，請勾選 **Enable application client to connect to a server**（讓應用程式用戶端連線至伺服器）勾選框，然後選取 **Use specific server with the**

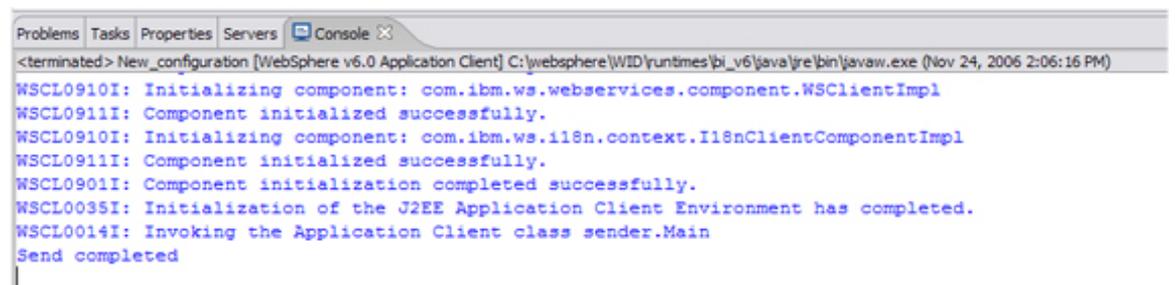
WebSphere ESB Server v6.0（使用特定伺服器來搭配 WebSphere ESB Server 6.0 版）選項，如圖 17 所示。

圖 17：選取配置



8. 現在可以按一下 **Run** 以啓動用戶端。完全開啓用戶端之後，主控台將會顯示下列資訊：

圖 18：主控台中的應用程式狀態訊息

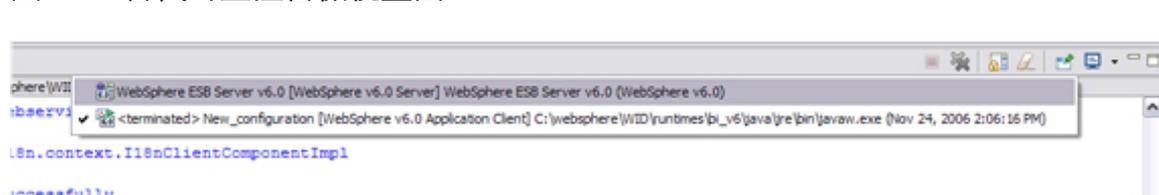


這些主控台訊息說明訊息的狀態：訊息已遞送至第一個佇列（並在此遞

送共至中介流程元件)，加以記錄，然後再轉送至第二個佇列（也就是遞送至 MDB）。

9. 您可以切換不同程序的主控制台檢視畫面，以顯示應用程式用戶端及伺服器程序的輸出狀態：

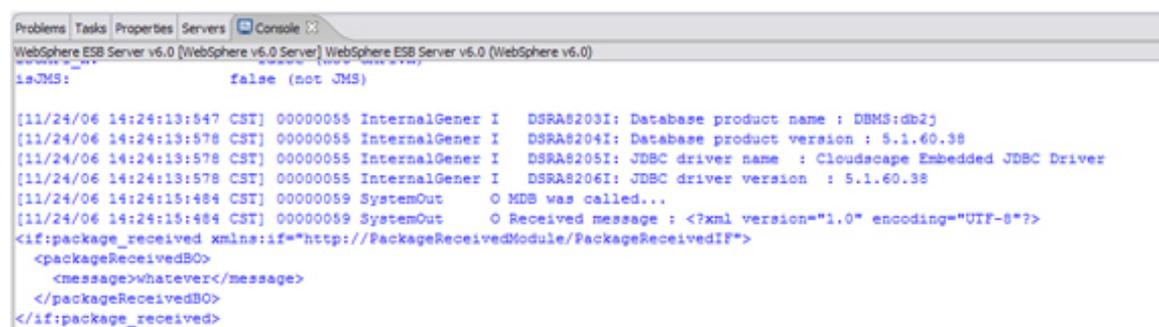
圖 19：替代的主控制台檢視畫面



將主控台的輸出切換至測試伺服器的程序。

10. 因為兩個中介流程元件和 MDB 是在同一部伺服器上執行，所以輸出會同時顯示在主控台中。

圖 20：中介流程元件與 MDB 的主控制台訊息



資料庫相關的列印輸出顯示，訊息日誌程式基本節點已執行，而 System.out 訊息是來自 MDB。

總結

本文說明如何使用傳送者應用程式和接收者應用程式，將簡單的點對點 JMS 情境示例，改爲可在其間建立 ESB 中介軟體。ESB 能夠將不同的功能分開設定，因爲像記錄這類工作，可在 ESB 中介軟體內處理，無需由提供程式或消費者程式碼 (consumer code) 本身來進行，因此能夠建立起 SOA 的兩大原則。

在我們的情境示例中，使用 WebSphere ESB 及其支援 JMS 通訊協定連結的功能，來建立中介軟體。雖然我們同時使用 JMS 做爲 ESB 兩端的通訊協定，但

在下篇文章中我們會說明進入和傳出中介元件時，如何使用 **ESB** 自行切換使用不同的通訊協定。

下篇文章將著重說明更符合 **Web** 服務導向的情境示例，而後再繼續探討如何整合 **JMS/MQ** 和 **Web** 服務這兩個範圍。請不要錯過！

此外，在本系列的第三部分，我們將開始運用全新 6.0.2 版的 **WebSphere ESB** 和 **WebSphere Integration Developer** 工具！

本系列其他部分：

- [第一部分：使用 WebSphere ESB，或 WebSphere ESB 和 SIBus 比較簡介](#)
- [第三部分：新增 Web 服務和提升內容](#)

下載

| 說明 | 名稱 | 大小 | 下載方法 |
|----------------|------------------------------|-------|-------------|
| 應用程式舉例 | PackageReceivedModule.zip | 20 KB | HTTP |
| 應用程式舉例 EAR 檔 1 | PackageReceivedClientEAR.ear | 4 KB | HTTP |
| 應用程式舉例 EAR 檔 1 | PackageReceived.ear | 4 KB | HTTP |

→ [下載方法的相關資訊](#)

資源

- [Building an ESB with WebSphere Application Server V6 Part 3: A simple JMS messaging example](#)
- [An introduction to the IBM Enterprise Service Bus](#)
- [JMS Tutorial](#)
- [WebSphere Enterprise Service Bus product page](#)
- [Getting started with WebSphere Enterprise Service Bus and WebSphere Integration Developer](#)
- [Developing custom mediations for WebSphere Enterprise Service Bus](#)
- [Building a powerful, reliable SOA with JMS and WebSphere ESB](#)

- [Dynamic routing at runtime in WebSphere Enterprise Service Bus](#)
- [Tutorial: Invoking a Web service with a JMS client](#)
- [Service Component Architecture](#)
- [Redbook: Enabling SOA Using WebSphere Messaging](#)

關於作者

Rachel Reinitz 是 IBM Software Services for WebSphere 的資深顧問 IT 專家，專長為 Web 服務。Rachel 為客戶和 ISV 提供的諮詢，主要是有關服務導向架構及 Web 服務如何能達成商業與技術上的目標。她開發了「IBM 進階 Web 服務訓練」課程，也經常出席會議發表演說。Rachel 也是「IBM 知識大學 (IBM Academy)」的成員，具有豐富的 eXtreme 程式設計課程教學經驗，並有四年的 XP 實務經歷。她現在居住於美國加州灣區，興趣是登山、參加社交活動和國外旅行。

Andre Tost 於 IBM 「軟體事業處」的「整合解決方案」部門，擔任資深技術人員，協助 IBM 的顧客建置「服務導向架構」(SOA)，他的專長是 Web 服務技術。在此之前，他在 IBM 軟體開發中有長達十年的時間服務於合作夥伴啓用、開發和架構等各種相關業務，現於「WebSphere 業務開發處」服務。他來自德國，目前居住和就職於美國明尼蘇達州羅契斯特市。他休閒時喜歡與家人相處，有時間便踢足球、觀賞足球比賽。