IBM WebSphere Commerce

程式設計手冊與指導教學

5.5 版

IBM WebSphere Commerce

程式設計手冊與指導教學

5.5 版

注意事項：

在使用本資訊及其支援的產品之前，請記得先閱讀「注意事項」一節中的資訊。

第一版（2003 年 6 月）

本修訂版適用於 IBM WebSphere Commerce Business Edition 5.5 版、 IBM WebSphere Commerce Professional Edition 5.5 版，以及所有後續版次及修訂，直到新版中另有指示。請確定您使用的是該產品層次的正確版本。

請向 IBM 業務代表或向當地的 IBM 分公司訂購出版品。下列位址並不供應。

IBM 歡迎您提供意見。您可以使用線上 IBM WebSphere Commerce 文件讀者意見表來傳送您的意見，您可以從下列 URL 取得這份表格：

http://www..ibm.com/software/webservers/commerce/rcf.html

當您傳送資訊給 IBM 時，即授與 IBM 非獨占的資訊使用或公佈權利，IBM 不需對您負任何責任。

開始之前

本書更新日期爲 2003 年 5 月 30 日。

*WebSphere Commerce 程式設計手冊與指導教學*提供關於 WebSphere® Commerce 架構與程式設計模型的資訊。尤其是本書提供下列主題的詳細資訊：

- 元件的互動
- 設計型樣
- 持續性物件模型
- 存取控制
- 錯誤的處理與訊息
- 指令的實作
- 開發工具
- 自訂程式碼的部署

此外，本書亦包含下列的指導教學：

- 建立新的商業邏輯
- 修改現有的控制程式指令
- 延伸物件模型以及修改現有的作業指令
- 延伸現有的 WebSphere Commerce Entity Bean

本書的更新

本書的副本及任何更新版本，都可以在 WebSphere Commerce 網站的 Technical Library 區段中找到（PDF 檔案格式）：

`http://www.ibm.com/software/commerce/library/`

有關其他的支援資訊，請造訪 WebSphere Commerce 支援網站：

`http://www.ibm.com/software/commerce/support/`

本書的更新版本也可以在 WebSphere Developer Domain 的 WebSphere Commerce Zone 中找到，網址爲：

`http://www.ibm.com/websphere/developer/zones/commerce`

本書中的使用慣例

本書使用下列標明慣例：

粗體字型表示指令或圖形式使用者介面（GUI）控制項，如欄位名稱、按鈕或功能表選項。

等寬字型，表示輸入必須完全相同的文字範例，以及目錄路徑。

斜體字型，用來強調並代表可換成您所要之值的變數。

此圖示表示「要訣」-- 此額外資訊有助您完成作業。

Windows 表示 WebSphere Commerce for Windows® 2000 特定的資訊。

AIX 表示 WebSphere Commerce for AIX® 專用的資訊。

Solaris 表示 WebSphere Commerce for Solaris Operating Environment 軟體特定的資訊。

400 表示 WebSphere Commerce for the IBM® @server iSeries™ 400®（舊稱 AS/400®）專用的資訊

DB2 表示 DB2 Universal Database™ 特定的資訊

Oracle 表示 Oracle 特定的資訊

基本知識要求

本書適合需瞭解如何自訂 WebSphere Commerce 應用程式的商店程式開發人員閱讀。負責執行程式設計延伸的商店程式開發人員應熟悉下列領域：

- Java™
- EnterpriseJavaBeans 元件架構
- JavaServer Pages 技術
- HTML
- 資料庫技術
- WebSphere Studio Application Developer

路徑變數

本手冊使用下列變數來代表目錄路徑：

WC_installdir

這是 WebSphere Commerce 的安裝目錄。以下是 WebSphere Commerce 在各種作業系統中的預設安裝目錄：

- Windows C:\Program Files\WebSphere\CommerceServer55
- AIX /usr/WebSphere/CommerceServer55
- Solaris /opt/WebSphere/CommerceServer55
- Linux /opt/WebSphere/CommerceServer55
- 400 /QIBM/ProdData/CommerceServer55

WC_userdir

WebSphere Commerce 使用的所有資料（可供使用者修改或必須由使用者配置）的目錄。

- 400 /QIBM/UserData/CommerceServer55

WAS_installdir

這是 WebSphere Application Server 的安裝目錄。以下是 WebSphere Application Server 在各種作業系統中的預設安裝目錄：

- Windows C:\Program Files\WebSphere\AppServer
- AIX /usr/WebSphere/AppServer
- Solaris /opt/WebSphere/AppServer
- Linux /opt/WebSphere/AppServer
- 400 /QIBM/ProdData/WebAs5/Base

WAS_userdir

WebSphere Application Server 使用的所有資料（可供使用者修改或必須由使用者配置）的目錄。

- 400 QIBM/UserData/WebAS5/Base/*WAS_instancename*
  和 *WAS_instance_name* 代表與您的 WebSphere Commerce 實例相關的 WebSphere Application Server 名稱。

WCStudio_installdir

WebSphere Commerce Studio 的安裝目錄。以下是預設安裝目錄：C:\WebSphere\CommerceStudio55

其他相關資訊的位置

有關 WebSphere Commerce 的其他資訊，請造訪下列網站：

http://www.ibm.com/software/commerce/library/

目錄

第 1 篇 概念與結構

第 1 章 概觀

WebSphere Commerce 的軟體元件

在檢查 WebSphere Commerce Server 如何運作之前，察看與 WebSphere Commerce 相關的軟體元件概要非常有用。下圖顯示這些軟體產品的概要檢視：

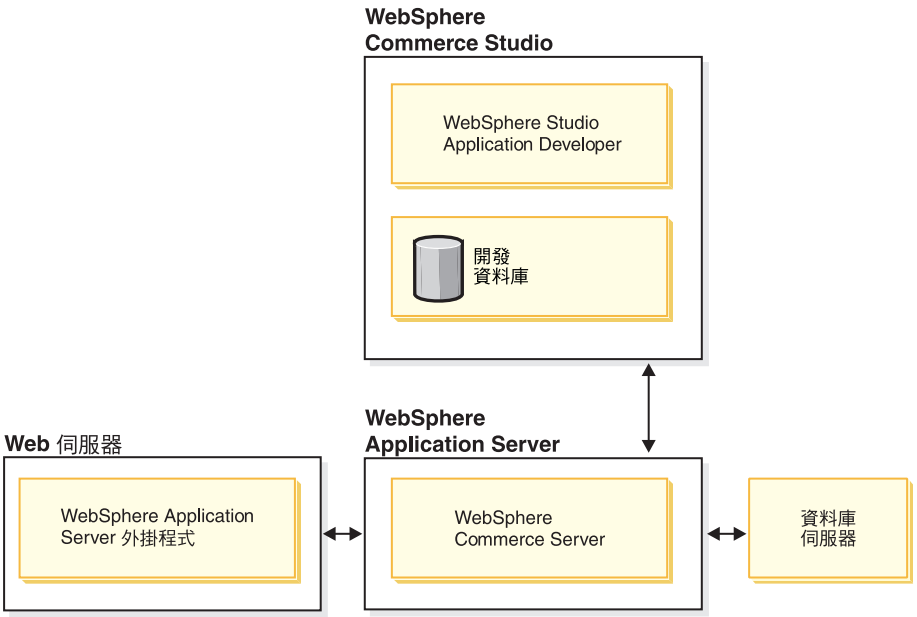

圖 1.

Web 伺服器是向您電子商務應用程式提出要求之傳入 HTTP 要求的第一個聯絡點。爲了成爲與 WebSphere Application Server 之間的有效介面，它使用 WebSphere Application Server 外掛程式。

WebSphere Commerce Server 是在 WebSphere Application Server 中執行，可讓它善用應用程式伺服器的許多特性。資料庫伺服器中保留您應用程式大部份的資料，包括產品與購物者資料。一般而言，是藉由修改或延伸 WebSphere Commerce Server 的程式碼來延伸您的應用程式。另外，您可能需要將 WebSphere Commerce 資料庫綱目領域外的資料儲存在資料庫中。

程式開發人員使用 WebSphere Studio Application Developer 來執行下列作業：

- 建立及自訂商店前端資產，例如 JSP 範本和 HTML 頁面
- 使用 Java 來建立新的商業邏輯
- 使用 Java 來修改現有的商業邏輯
- 測試程式碼以及商店前端資產

WebSphere Commerce Studio 使用開發資料庫。程式開發人員可以使用偏好的資料庫工具（包括 WebSphere Studio Application Developer）來對資料庫進行修改。

WebSphere Commerce 應用程式結構

現在，您已經知道各種軟體元件與 WebSphere Commerce 是如何相互配合，這對瞭解應用程式的架構來說是很重要的。這將有助您瞭解屬於基礎階層的部份以及可修改的部份。下圖顯示組成應用程式結構的各階層：

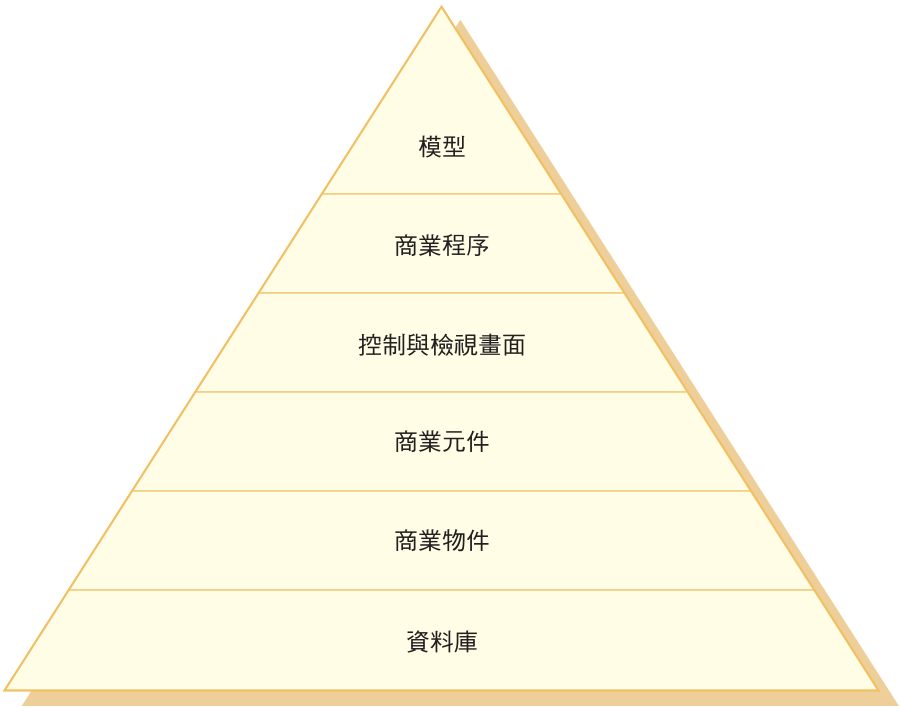

圖 2.

應用程式結構中的每一個階層的說明如下：

資料庫 WebSphere Commerce 利用專爲電子商務應用程式及其資料需求而設計的資料庫綱目。以下是此綱目中的表格範例：

- USERS

- ORDERS
- INVENTORY

商業物件

商業物件代表商業領域中的實體，另概括了擷取或解譯資料庫中資訊所需的邏輯（以資料爲主）。這些實體皆符合 Enterprise JavaBeans 規格。

這些 Entity Bean 是作爲商業元件與資料庫之間的介面。此外，Entity Bean 比資料庫表格中各直欄間的複雜關係更易於瞭解。

商業元件

商業元件爲商業邏輯的單元。主要是執行粗略的程序性商業邏輯。這個邏輯是利用 WebSphere Commerce 的控制指令和作業指令模型來實作。舉例來說，像 OrderProcess 控制程式指令即屬於此種元件類型。此特定指令概括了處理典型訂單所需的所有商業邏輯。電子商務應用程式會呼叫 OrderProcess 指令，而此指令會依序呼叫幾個作業指令以執行個別的工作單元。例如，各作業指令會確定有足夠的庫存量符合訂單的要求，會處理付款，會更新訂單的狀態，並在程序完成時依適當量降低庫存量。

控制與檢視畫面

Web 控制程式會決定適當的控制程式指令實作以及要使用的檢視畫面。實作方式可視商店而定。

檢視畫面中顯示指令與使用者動作的結果。這些檢視畫面是利用 JSP 範本來實作。檢視畫面的範例包括 ProductDisplayView（傳回產品頁面，以顯示購物者所選取的產品的相關資訊）以及 OrderCancelView。

商業程序

商業元件與檢視畫面的組合，而構成工作流程與網站流程，亦稱爲商業程序。商業程序的範例有：

建立電子郵件活動

這個商業程序包含在建立電子郵件活動的程序時牽涉到的所有步驟的相關商業元件與檢視畫面。

準備線上型錄

這個商業程序包含與建立線上型錄相關的商業元件與子程序。其中包括設計型錄、載入型錄資料、建立產品推銷連結，以及設定計價資訊。

模型 整體而言，圖中的低階層構成了電子商務商業模型。電子商務商業模型的其中一個範例就是「流行館」範例商店中顯示的消費者市場模型。另一個範例是「工具屋」範例商店中所顯示的「企業消費型」模型。

WebSphere Commerce 執行期間結構

上節所介紹的應用程式結構是從商業應用程式的觀點來說明 WebSphere Commerce 應用程式中的各個階層。本節則說明執行期間結構是如何實作的。

以下是 WebSphere Commerce 執行期間結構的主要元件：

- Servlet 引擎
- 通訊協定接收程式
- 配接器管理程式
- 配接器
- Web 控制程式
- 指令
- Entity Bean
- 資料 Bean
- 資料 Bean 管理程式
- 顯示頁面
- XML 檔

下圖顯示 WebSphere Commerce 元件之間的互動。我們會在後面章節中詳述各個元件。

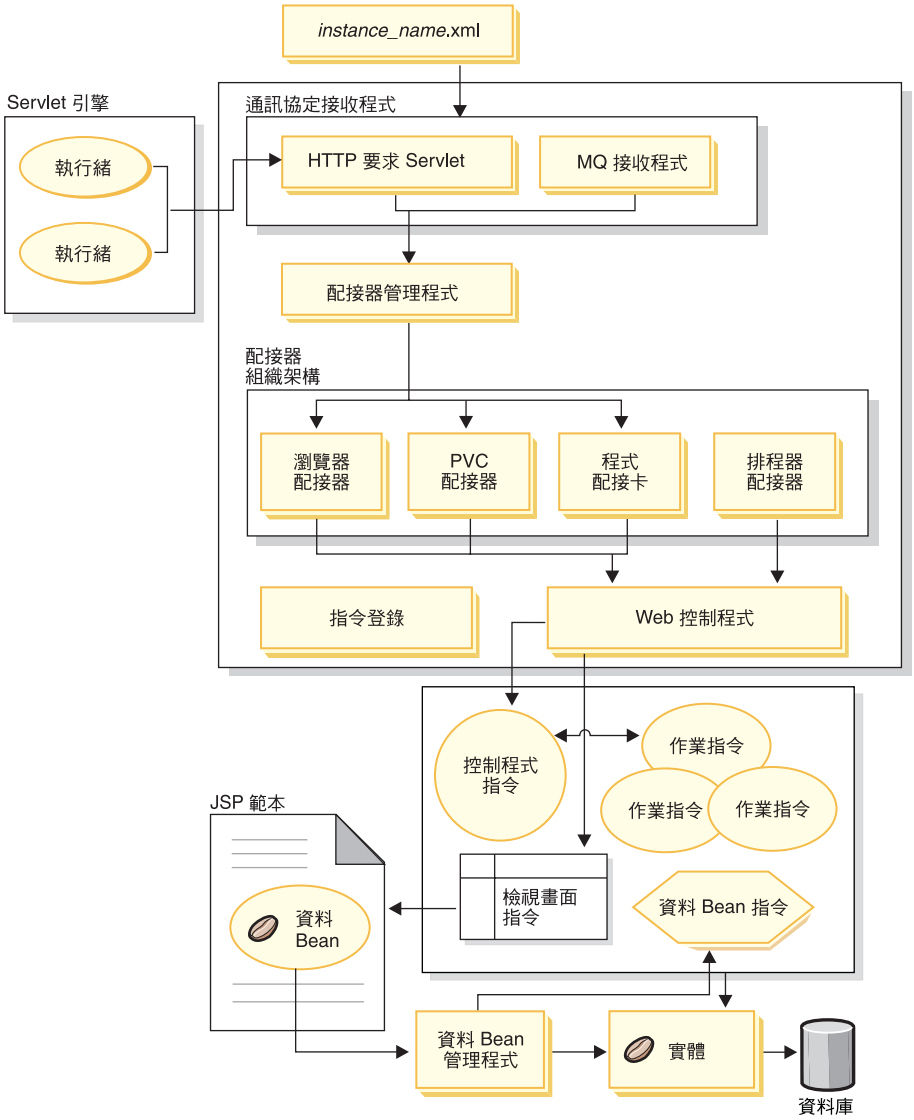

圖 3.

Servlet 引擎

Servlet 引擎爲 WebSphere Application Server 執行期間環境的一部份，主要扮演入埠 URL 要求的要求分派程式。Servlet 引擎負責管理處理要求的執行緒儲存池。每一個入埠要求會分別放在個別的執行緒中執行。

通訊協定接收程式

您可從各種裝置來呼叫 WebSphere Commerce 指令。舉例來說，可呼叫指令的裝置有：

- 典型的網際網路瀏覽器
- 使用網際網路瀏覽器的行動電話
- 使用 MQSeries® 傳送 XML 訊息的企業消費型商務應用程式
- 採用「XML 透過 HTTP」來傳送要求的採購系統
- 執行背景工作的 WebSphere Commerce 排程器

裝置可使用各種不同的通訊協定。通訊協定接收程式爲一種執行期間元件，它會接收傳輸端傳來的入埠要求，並根據所用的通訊協定將這些要求發送到適當的配接器。通訊協定接收程式包括：

- 要求 Servlet
- MQSeries 接收程式

當「要求 Servlet」收到 Servlet 引擎傳來的 URL 要求時，會將要求傳遞給配接器管理程式。接著配接器管理程式會查詢配接器類型，以判斷哪種配接器可處理該要求。一旦決定特定的配接器後，即會將要求傳給該配接器。

在起始設定要求 Servlet 時，它會讀取 *instance_name*.xml 配置檔（其中 *instance_name* 是 WebSphere Commerce 實例的名稱）。XML 檔其中一個配置區塊會定義所有的配接器。「要求 Servlet」的 init() 方法會起始設定所有已定義的配接器。

MQSeries 接收程式會接收遠端程式所傳來的 XML 型 MQSeries 訊息，並將要求分派給非 HTTP 配接器的管理程式。

工作排程器不需要通訊協定接收程式。

配接器管理程式

配接器管理程式會先判斷哪個配接器能夠處理要求，再將要求轉遞給該配接器。

配接器

WebSphere Commerce 配接器爲裝置特有的元件，會在將要求傳給 Web 控制程式前先執行各項處理功能。舉例來說，配接器所執行的處理作業有：

- 指示 Web 控制程式採用該裝置類型特有的方法來處理要求。例如，普遍運算（PVC）裝置配接器可指示 Web 控制程式忽略原始要求中的 HTTPS 檢查。
- 將入埠要求的訊息格式轉換成 WebSphere Commerce 指令可以剖析的一組內容。

- 讓裝置特有的階段作業持續。

下圖顯示 WebSphere Commerce 配接器組織架構的實作類別階層。

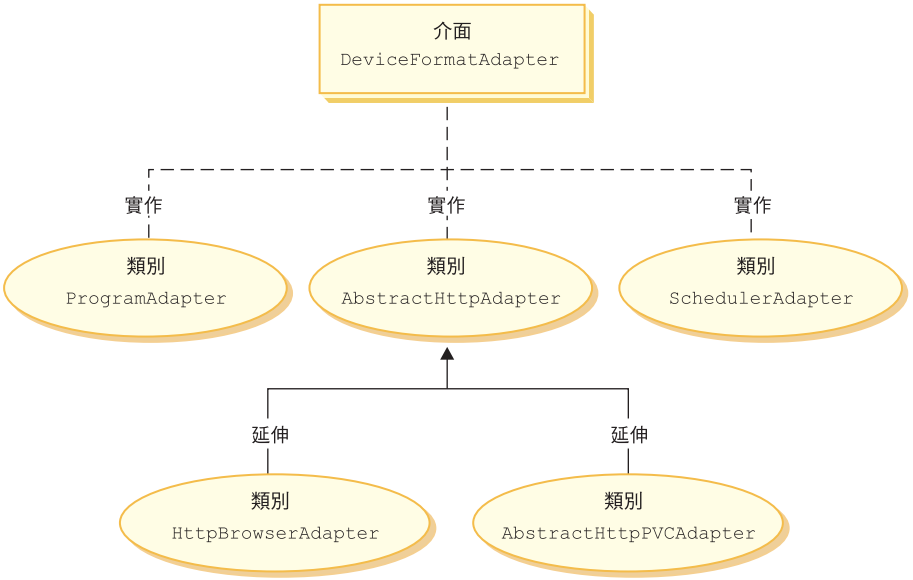

圖 4.

如上圖所示，所有配接器皆實作 DeviceFormatAdapter 介面。以下是 WebSphere Commerce 執行期間環境所用的配接器：

程式配接器

程式配接器可支援會呼叫 WebSphere Commerce 指令的遠端程式。程式配接器會接收要求，並使用訊息對映器將要求轉換成 CommandProperty 物件。當轉換好後，程式配接器會使用 CommandProperty 物件並執行要求。

排程器配接器

排程器配接器可支援當成背景工作執行的 WebSphere Commerce 指令。

HTTP 瀏覽器配接器

HTTP 瀏覽器配接器可支援 HTTP 瀏覽器所接收的呼叫 WebSphere Commerce 指令的要求。

HTTP PvC 配接器

這是一種抽象配接器類別，可用來開發特定的 PvC 裝置配接器。 例如，如果您需要開發一種用於特定行動電話的配接器，則將從此種配接器延伸而來。

必要時您可以採用下列兩種方法來延伸配接器組織架構：

- 建立特定 PvC 裝置的配接器（例如建立 HttpIModePVCAdapterImpl 類別，以支援 i-mode 裝置）。此種類型的配接器必須是 AbstractHttpAdapterImpl 類別的延伸。
- 建立連接新通訊協定接收程式的新配接器。這種新配接器必須實作 DeviceFormatAdapter 介面。

Web 控制程式

WebSphere Commerce Web 控制程式是一種應用程式儲存器，其遵循類似 EJB 儲存器的設計型樣。此儲存器可簡化指令的職務，因爲它可提供下列各項服務：階段作業管理（根據配接器所建立的階段作業持續性而定）、交易控制、存取控制以及鑑別。

Web 控制程式亦負責實施商務應用程式的程式設計模型。舉例來說，程式設計模型定義了應用程式應撰寫的指令類型。每一種指令類型各負責處理一種特定目的。商業邏輯必須用控制程式指令來實作，而檢視邏輯必須用檢視指令來實作。Web 控制程式會期待控制程式指令傳回一個檢視畫面名稱。如果未傳回檢視畫面名稱，則會擲出異常狀況。

就 HTTP 要求而言，Web 控制程式會執行下列作業：

- 使用 javax.transaction 套件中的 UserTransaction 介面開始進行交易。
- 從配接器取得階段作業資料。
- 判斷使用者在呼叫指令之前是否必須先登入。若有必要，則會將使用者的瀏覽器重新導向到登入 URL。
- 檢查該 URL 是否需要安全 HTTPS。如果需要但目前的要求並未使用 HTTPS，則會將 Web 瀏覽器重新導向到一個 HTTPS URL。
- 呼叫控制程式指令，並將指令環境定義與輸入內容物件傳給該控制程式指令。
- 如果發生交易回復異常狀況，而控制程式指令可以重試，則會重試該控制程式指令。
- 如果有檢視畫面指令要傳回給用戶端，則控制程式指令通常會傳回一個檢視畫面名稱。 Web 控制程式會呼叫該對應檢視畫面的檢視畫面指令。形成回應檢視畫面的方法有許多。像是重新導向至不同的 URL，轉遞到一個 JSP 範本，或在回應物件中撰寫一份 HTML 文件。
- 儲存階段作業資料。
- 確定階段作業資料。
- 當成功時確定目前的交易。
- 當失敗時，回復目前的交易（視情況而定）。

指令

WebSphere Commerce 指令爲一些 Bean，內含與處理特定要求有關的程式設計邏輯。

WebSphere Commerce 指令有四種主要類型：

控制程式指令

控制程式指令概括了與特定商業程序有關的邏輯。控制程式指令的範例包括用於訂單處理的 *OrderProcessCmd* 指令，以及可容許使用者登入的 *LogonCmd*。一般而言，控制程式指令含有一些控制陳述式（例如：if、then、else），並會呼叫作業指令以執行商業程序中的個別作業。一旦完成，控制程式指令會傳回一個檢視畫面名稱。接著 Web 控制程式會決定該檢視畫面指令的適當實作類別，並執行該檢視畫面指令。

作業指令

作業指令是實作應用程式邏輯中的特定單元。通常，控制程式指令會結合一組作業指令，共同實作某個 URL 要求的應用程式邏輯。作業指令是在和控制程式指令相同的儲存器中執行。

資料 Bean 指令

資料 Bean 指令是在將某個資料 Bean 實例化時，由資料 Bean 管理程式所呼叫。資料 Bean 指令的主要功能是在資料 Bean 中移入資料。

檢視畫面指令

檢視畫面指令會撰寫一個檢視畫面，作爲用戶端要求的回應。檢視畫面指令的類型有下列三種：

重新導向檢視畫面指令

此種檢視畫面指令會使用重新導向通訊協定（如：URL 重新導向）來傳送檢視畫面。控制程式指令如果要使用重新導向通訊協定來傳回檢視畫面，應使用此種檢視類型來傳回檢視畫面指令。當使用重新導向通訊協定時，將會變更瀏覽器中的 URL 堆疊。一旦輸入重新載入鍵值時，將會執行重新導向的 URL，而非執行原來的 URL。

直接導向檢視畫面指令

此檢視畫面指令會將回應檢視畫面直接傳給用戶端。

轉遞檢視畫面指令

此檢視畫面指令會將檢視要求轉遞給另一個 Web 元件，像是 JSP 範本。

呼叫檢視畫面指令的方法有下列三種：

- 控制程式指令在順利完成要求時，指定一個檢視畫面指令名稱。

- 用戶端直接要求檢視畫面。
- 指令偵測到錯誤，而必須執行錯誤作業以處理該錯誤。指令擲出設有檢視畫面指令名稱的異常狀況。當異常狀況傳播給 Web 控制程式時，它會執行錯誤檢視畫面指令並將回應傳回給用戶端。

WebSphere Commerce Entity Bean

Entity Bean 爲 WebSphere Commerce 所提供的一些持續性交易的商務物件。如果您熟悉商務領域， Entity Bean 直覺代表 WebSphere Commerce 資料。亦即，您不用全盤瞭解資料庫綱目，便可以從 Entity Bean（密切仿造商務領域中的概念與物件）中存取資料。您可以延伸現有的 Entity Bean。此外，您可以根據應用程式特有的商業需求，來完整部署新 Entity Bean。

Entity Bean 是根據 Enterprise JavaBeans (EJB) 元件模型來實作。

有關 Entity Bean 的詳細資訊，請參閱第 45 頁的『WebSphere Commerce Entity Bean 的實作』。

資料 Bean

資料 Bean 是一種 Java Bean，主要供 Web 設計人員使用。它們絕大部份可提供 WebSphere Commerce 實體的存取權。Web 設計人員可將這些 Bean 置於 JSP 範本中，以便在顯示頁面時將動態資訊移入頁面中。 Web 設計人員只需瞭解 Bean 可提供哪些資料，以及 Bean 需將哪些資料當成輸入。和將顯示與商業邏輯區隔開的主題一致， Web 設計人員不需瞭解 Bean 如何運作。

資料 Bean 管理程式

插入到 JSP 範本中的 WebSphere Commerce 資料 Bean 可容許在頁面中列入動態內容。資料 Bean 管理程式會啓用資料 Bean，以便在將下面這一行的程式碼插入到頁面時，將資料 Bean 的值移入：

```
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

其中 *data_bean* 是要啓用的資料 Bean，而 *request* 是 HTTPServletRequest 物件。

JavaServer Pages 範本

JSP 範本爲一種特殊的 Servlet，通常用於顯示上。一旦完成 URL 要求，Web 控制程式會呼叫一個檢視畫面指令以呼叫 JSP 範本。用戶端也可以直接從瀏覽器呼叫 JSP 範本，而不必透過相關聯的指令。在此情況下， JSP 範本的 URL 必須在其路徑中含有「要求 Servlet」，如此才能在單一交易中啓動 JSP 範本所需的所有資料 Bean。「要求 Servlet」可將 URL 要求轉遞給 JSP 範本，並在單一交易中執行 JSP 範本。

資料 Bean 管理程式會拒絕任何其路徑中不含「要求 Servlet」的 JSP 範本 URL。有關保護 JSP 範本以及其他資源的相關資訊，請參閱第 85 頁的第 4 章, 『存取控制』。

instance_name.xml 配置檔

instance_name.xml 配置檔（其中 *instance_name* 是 WebSphere Commerce 實例的名稱）會設定 WebSphere Commerce 實例的配置資訊。在起始設定「要求 Servlet」時會讀取此配置檔。

要求的摘要說明

本節摘要說明在產生要求的回應方面各元件間的互動流程。

圖解之後，有每個步驟的說明。

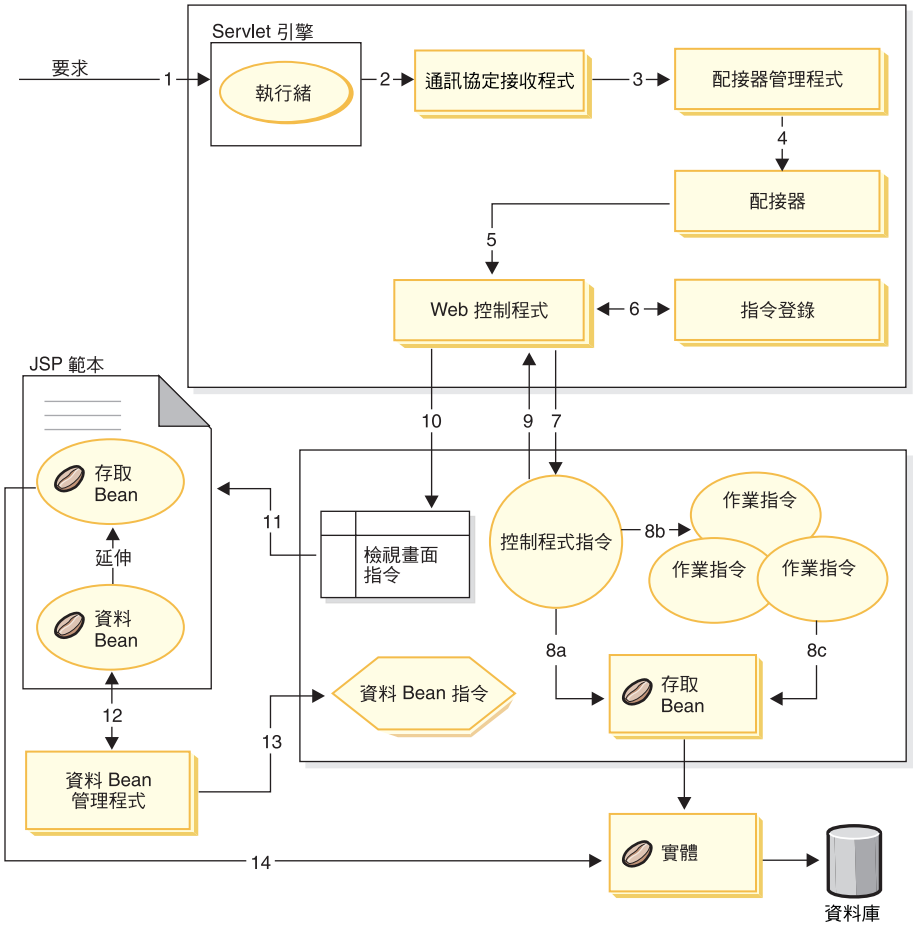

圖 5.

下列資訊與前圖相呼應。

1. WebSphere Application Server 外掛程式將要求引導到 Servlet 引擎。
2. 在要求本身的執行緒中執行要求。Servlet 引擎將要求分派給通訊協定接收程式。通訊協定接收程式可爲 HTTP 要求 Servlet 或 MQ 接收程式。
3. 通訊協定接收程式將要求傳給配接器管理程式。
4. 配接器管理程式會先判斷哪個配接器能夠處理要求，再將要求轉遞給適當的配接器。舉例來說，如果該要求由網際網路瀏覽器發出，則配接器管理程式會將要求轉遞給 HTTP 瀏覽器配接器。
5. 配接器將要求傳給 Web 控制程式。
6. Web 控制程式查詢指令登錄，以判斷要呼叫哪個指令。

7. 假設該要求需用到控制程式指令，則 Web 控制程式會呼叫適當的控制程式指令。
8. 一旦控制程式指令開始執行，會有多個可能的路徑：
   a. 控制程式指令可使用存取 Bean 與對應的 Entity Bean 來存取資料庫。
   b. 控制程式指令可呼叫一或多個作業指令。接著，作業指令可使用存取 Bean 與其對應的 Entity Bean 以存取資料庫（請見 8(c)）。
9. 一旦完成，控制程式指令會傳回一個檢視畫面名稱給 Web 控制程式。
10. Web 控制程式查看 VIEWREG 表格中的檢視畫面名稱。它會呼叫針對要求端之裝置類型所登錄的檢視畫面指令實作方式。
11. 檢視畫面指令將要求轉遞給 JSP 範本。
12. 在 JSP 範本中，需使用資料 Bean 以擷取資料庫中的動態資訊。資料 Bean 管理程式會啓動資料 Bean。
13. 必要時，資料 Bean 管理程式會呼叫資料 Bean 指令。
14. 從資料 Bean 延伸而來的存取 Bean 會使用其對應的 Entity Bean 來存取資料庫。

第 2 篇 程式設計模型

第 2 章 設計型樣

用來開發 WebSphere Commerce 組織架構的設計型樣與機制相當多樣。WebSphere Commerce 中會提供各 WebSphere Commerce 應用程式所應遵循的高階設計型樣。本章所討論的設計型樣如下：

- 「模型-檢視畫面-控制程式」設計型樣
- 「指令」設計型樣
- 「顯示」設計型樣

「模型-檢視畫面-控制程式」設計型樣

「模型-檢視畫面-控制程式」(MVC) 設計型樣可指定應用程式是由資料模型、呈現資訊與控制資訊所組成。此型樣會要求這些中的每一個需細分成不同的物件。

模型（例如：資料資訊）中只含單純的應用程式資料；而不含任何用以說明資料如何呈現在使用者面前的邏輯。

檢視畫面（例如：呈現資訊）則是將模型的資料呈現在使用者面前。檢視畫面知道如何存取模型的資料，但其並不清楚該資料的意義或哪位使用者可操作該資料。

控制程式（例如：控制資訊）則存在於檢視畫面與模型之間。它負責監聽檢視畫面（或另一個外部來源）所觸發的事件，並對這些事件作出適當反應。在大部份情況下，所謂反應是對模型呼叫一種方法。由於檢視畫面與模型是透過通知機制連接，此動作的結果會自動反映在檢視畫面中。

如今大部份的應用程式皆遵循此種型樣，但有許多會稍作變化。舉例來說，由於檢視畫面與控制程式關係密切，某些應用程式遂將檢視畫面與控制程式結合成一種類別。而所有的變化皆強烈促使資料與其呈現方式分隔開來。這不單讓應用程式的結構更爲簡化，也讓程式碼可以重複使用。

由於有許多書籍中都會談到型樣並提供各種範例，本書便不再進一步詳述型樣。

下圖顯示 MVC 設計型樣如何應用於 WebSphere Commerce 中。

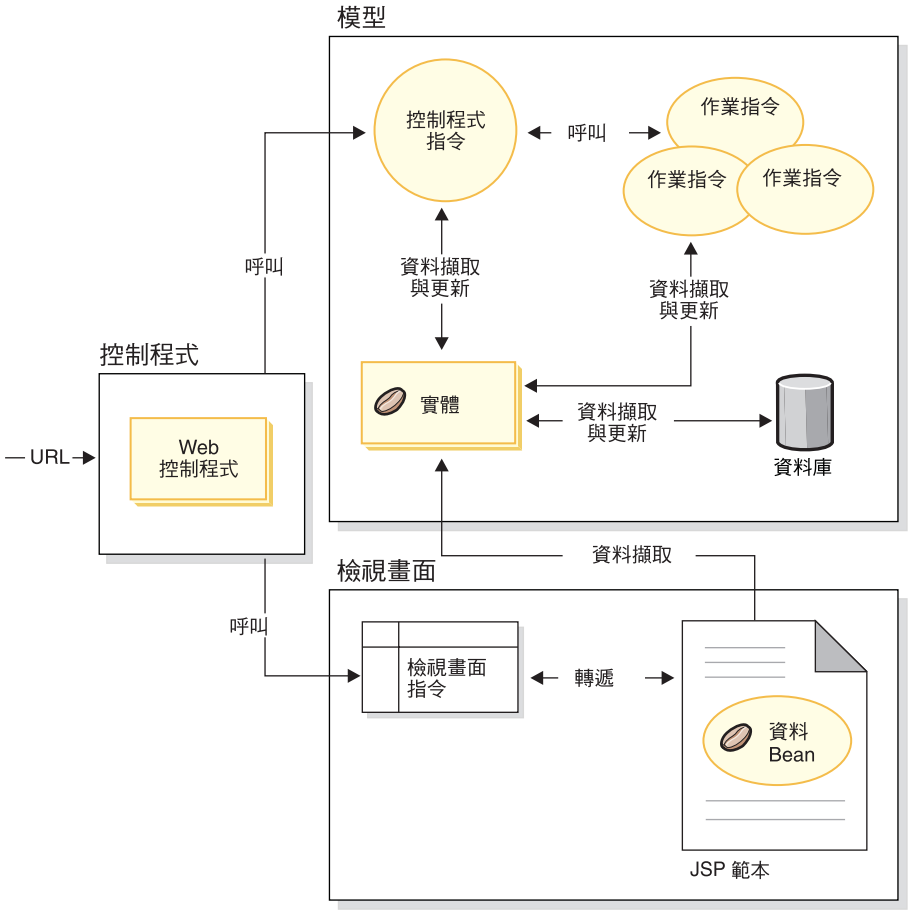

圖 6.

指令設計型樣

WebSphere Commerce Server 接受瀏覽器型小型用戶端（Thin Client）應用程式以及其他遠端應用程式所發的要求。舉例來說，出自遠端採購系統或另一商務伺服器的要求。

採用各種格式的所有要求會轉譯成配接器組織架構中之各配接器的共通格式。只要要求採用此種共通格式，即可被 WebSphere Commerce 指令瞭解。

指令爲一些會執行商業邏輯的 Bean。它們代表採高階程序邏輯或個別商業邏輯作業的程序性邏輯。程序型指令扮演著控制程式角色，亦即其橫跨多個實體與其他指令；而作業指令則負責執行一項特定作業，且可能只會存取一個物件。

指令組織架構

指令 Bean 會遵循一種特定的設計型樣。每一個指令皆含有一個介面類別（例如 `CategoryDisplayCmd`）與一個實作類別（例如 `CategoryDisplayCmdImpl`）。從呼叫端的角度來看，呼叫邏輯包括：設定輸入內容、呼叫 `execute()` 方法，以及擷取輸出內容。

而從指令實作端的角度來看，指令會遵循 WebSphere 指令組織架構，以實作標準指令設計型樣，並容許呼叫端與實作間存在一個間接層。此間接層中所啓用的關鍵機制包括：

1. 能呼叫存取控制原則管理程式，以判斷使用者能否呼叫該指令。
2. 能根據商店識別碼，以針對不同商店實作不同的指令。
3. 能根據要求端的裝置類型，執行不同的檢視畫面實作。

下圖顯示四種主要指令類型的介面概觀：

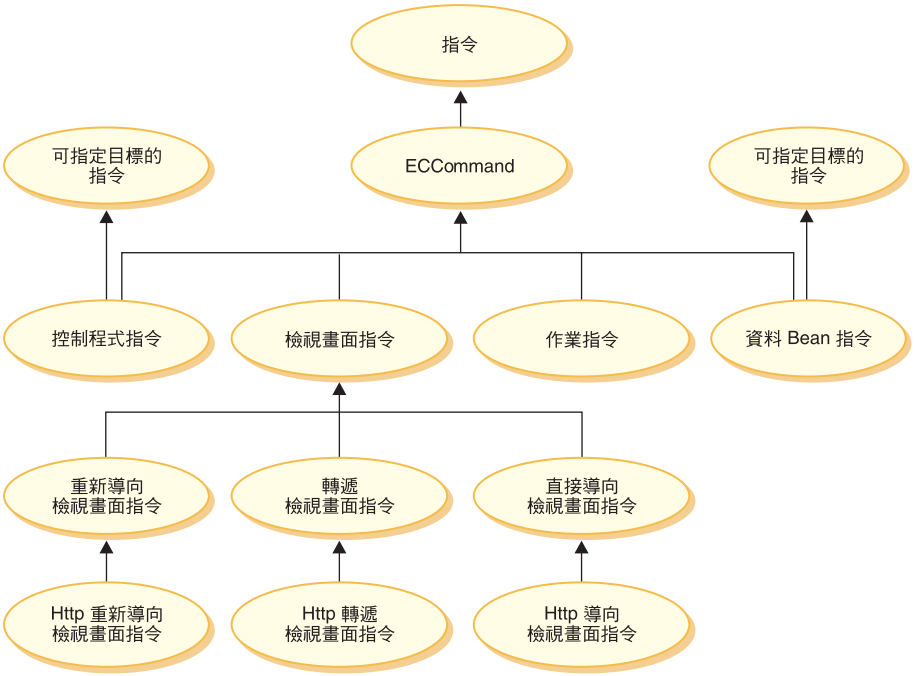

圖 7.

控制程式指令

控制程式指令概括了與特定商業程序有關的邏輯。控制程式指令的範例包括用於訂單處理的 *OrderProcessCmd* 指令，以及可容許使用者登入的 *LogonCmd*。一般而言，控制程式指令含有一些控制陳述式（例如：if、then、else），並會呼叫作業指令以執行商業程序中的個別作業。一旦完成，控制程式指令會傳回一個檢視畫面名稱。 Web 控制程式會根據檢視畫面名稱、商店 ID 和裝置類型，決定該檢視畫面指令的適當實作類別，並執行檢視畫面指令。

雖然控制程式指令爲一種可指定目標的指令，但目前只支援本端目標。

作業指令

作業指令是實作應用程式邏輯中的特定單元。通常，控制程式指令會結合一組作業指令，共同實作某個 URL 要求的應用程式邏輯。作業指令是在和控制程式指令相同的配置區中執行。

資料 Bean 指令

資料 Bean 指令是在某個資料 Bean 實例化時，由 JSP 頁面所呼叫。資料 Bean 指令的主要功能是將資料移入資料 Bean 的欄位中。

雖然資料 Bean 指令爲一種可指定目標的指令，但目前只支援本端目標。

檢視畫面指令

檢視畫面指令會撰寫一個檢視畫面，作爲用戶端要求的回應。呼叫檢視畫面指令的方法有下列三種：

- 控制程式指令會在順利完成要求時，指定一個檢視畫面指令名稱。
- 用戶端可直接要求檢視畫面。
- 控制程式或作業指令偵測到錯誤，且判斷必須執行錯誤作業以處理該錯誤，並擲出帶有檢視畫面指令名稱的異常狀況。當異常狀況傳播給 Web 控制程式時，它會執行檢視畫面指令並將回應傳回給用戶端。

檢視畫面指令的類型有下列三種：

重新導向檢視畫面指令

此種檢視畫面指令會使用重新導向通訊協定（如：URL 重新導向）來傳送檢視畫面。控制程式指令如果需要使用重新導向通訊協定，應會傳回此種檢視畫面類型的檢視畫面指令。當使用重新導向通訊協定時，將會變更瀏覽器中的 URL 堆疊。一旦輸入重新載入關鍵字時，將會執行重新導向的 URL，而非執行原來的 URL。

直接導向檢視畫面指令

此檢視畫面指令會將回應檢視畫面直接傳給用戶端。

轉遞檢視畫面指令

此檢視畫面指令會將檢視要求轉遞給另一個 Web 元件，像是 JSP 範本。

指令 Factory

爲了建立新指令物件，指令的呼叫端會使用*指令 Factory*。指令 Factory 爲一種 Bean，用來實例化指令。它是以 Factory 設計型樣爲基礎，而會延緩脫離呼叫類別而移往 Factory 類別（瞭解所要實例化的實作類別）之物件的實例化。

Factory 會提供一種智慧型方法來實例化新物件。在此情況下，當根據個別商店建立新指令物件時，指令 Factory 會提供一種方法以判斷正確的實作類別。一旦實例化，即會將指令介面名稱與特定的商店識別碼傳給新指令物件。

指定指令之實作類別的方法有兩種。預設的實作類別可藉由 `defaultCommandClassName` 變數直接指定於指令介面的程式碼中。舉例來說，下列程式碼存在於 CategoryDisplayCmd 介面中：

```
String defaultCommandClassName =
     "com.ibm.commerce.catalog.commands.CategoryDisplayCmdImpl"
```

指定實作類別的第二種方法是使用 WebSphere Commerce 指令登錄。當實作類別因商店而有所出入時，應一律會使用指令登錄。有關指令登錄的詳細資訊，可在第 27 頁找到。

當介面的程式碼中指定了預設實作類別，而在指令登錄中所指定的是不同的實作類別時，則會優先採用指令登錄。

使用指令 Factory 的語法如下：

```
cmd = CommandFactory.createCommand(interfaceName, storeId)
```

其中 *interfaceName* 是新指令 Bean 的介面名稱，而 *storeId* 是應該爲其實作指令的商店的 ID。通常，商店 ID 可以利用 commandContext.getStoreId() 方法來擷取。

註: 使用指令 Factory 來建立商業原則指令的相關語法，和前述的程式碼片段有所不同。有關使用指令 Factory 來建立商業原則指令的進一步資訊，請參閱第 170 頁的『呼叫新商業原則』。

巢狀控制程式指令

您會最常使用指令 factory 來建立作業指令的實例，然而，您也可以在某個控制程式指令中用它來建立另一個控制程式指令的實例。換言之，您可以在從另一個控制程式指令中呼叫某個控制程式指令時使用它。

將作業指令實例化的語法和控制程式指令的一樣。亦即，您必須在這兩個實務內容中同時指定指令的介面名稱和商店 ID。

如果您將某個控制程式指令巢狀在另一個控制程式指令內，請注意下列幾點：

- 一旦您將巢狀指令實例化，請呼叫其 setCommandContext 方法並傳入現行的指令環境定義。請注意，如果您傳入一組不同的要求內容到巢狀指令中，而這些參數會影響指令環境定義，您應該先複製指令環境定義，然後再將巢狀指令實例化。這樣就可以爲外部指令保留指令環境定義資訊。
- 理想的情況是呼叫巢狀指令的 setRequestProperties 方法，然後傳送包含輸入內容的 TypedProperties 物件。否則，您可以使用在指令介面中定義的個別 setter 方法，來設定必要的內容。
- 在設定輸入內容後，請呼叫巢狀指令的 execute 方法。
- 由於所有的控制程式指令在處理完成時都必須傳回檢視畫面，外部指令對於巢狀指令所傳回的檢視畫面將無法執行任何作業。
- 巢狀指令會在外部指令的交易範圍內執行。

請將下列程式碼片段視爲巢狀控制程式指令的範例。這個範例顯示外部指令中的其中一個方法，以及它如何使用指令 factory 來將第二個控制程式指令實例化。

```
yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommand()
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, this.getStoreId());
      ctrlCmd.setCommandContext(this.getCommandContext());
      ctrlCmd.setRequestProperties(this.getRequestProperties());
      ctrlCmd.execute();
    }
```

舉另一個範例來說，假設您正在爲與第一家商店不同的商店執行巢狀指令。在此情況下，就必須保留外部指令的指令環境定義，使它不會被內部指令改寫。

```
// 製作副本，以保留外部指令的指令環境定義
CommandContext cloneCmdCtx = (CommandContext)this.getCommandContext().clone();

//現在傳入一組新的要求內容到複製的指令環境定義
cloneCmdCtx.setRequestProperties(reqProp);

yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommandForOtherStore(int aStoreId)
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, aStoreId;
      ctrlCmd.setCommandContext(cloneCmdCtx);
      ctrlCmd.setRequestProperties(reqProp);
      ctrlCmd.execute();
    }
```

指令流程

本節提供指令與 WebSphere Commerce 資料庫間的邏輯流程概觀。下圖與說明將描述此流程。

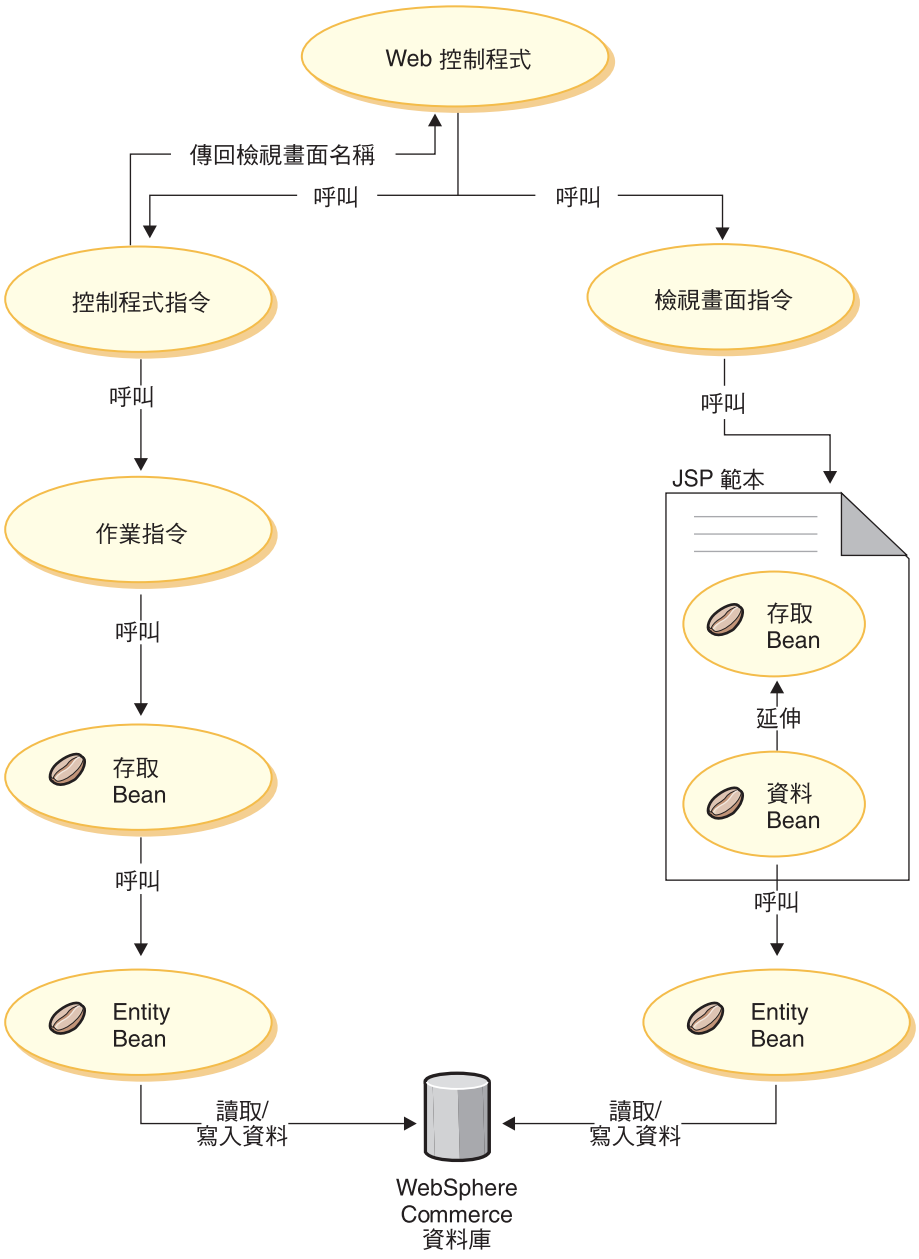

圖 8.

當 Web 控制程式收到要求時，它會判斷該要求所要呼叫的是控制程式指令或檢視畫面指令。不論哪一種情況，Web 控制程式皆會決定指令的實作類別並呼叫之。

請先看本圖左邊。由於控制程式指令概括了商業程序的邏輯，因而會經常呼叫個別的作業指令以執行商業程序中的特定工作單元。當必須擷取或更新資料庫中的資訊時，則會呼叫存取 Bean。作業或控制程式指令皆可呼叫存取 Bean。接著要求流程會從存取 Bean 移往 Entity Bean，而這類 Entity Bean 可從 WebSphere Commerce 資料庫讀取以及寫入資料庫中。

此時請看本圖右邊。當控制程式指令完成處理並傳回所要呼叫的檢視畫面指令名稱時，或者當發生錯誤而必須顯示錯誤檢視畫面時， Web 控制程式將會呼叫檢視畫面指令。

一般而言，檢視畫面指令會呼叫一個 JSP 範本以顯示回應給用戶端。在 JSP 範本中，會採用資料 Bean 將動態資訊移入頁面中。資料 Bean 是由資料 Bean 管理程式啓動。資料 Bean（從存取 Bean 延伸而來）會呼叫其對應的 Entity Bean。當從 JSP 範本間接存取時，Entity Bean 通常會擷取資料庫中的資訊（而非將資訊寫到資料庫中）。

指令登錄組織架構

WebSphere Commerce 控制程式與作業指令會登錄在指令登錄中。下列三種表格由指令登錄組成：

- URLREG
- CMDREG
- VIEWREG

註: Business 本節不適用於商業原則指令的登錄。有關登錄新商業原則指令的進一步資訊，請參閱第 154 頁的『登錄新商業原則與商業原則指令』。

URLREG 表格

URLREG 表格會將 URI（通用資源指示器）對映至控制程式指令介面。 URI 可在資源識別方面提供一種簡單而可延伸的機制。 URI 是一種相對的字元短字串，用來識別抽象或實體資源。在 WebSphere Commerce 中，URI 只含有指令資訊。在下列的 URL 中，URI 區段以粗體字顯示：

```
http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=store_Id&langId=-1
```

當 URI 與介面名稱間採一對一對映時，每一家商店可指定指令所需的是 HTTPS 或 AUTHENTICATION。就每一個入埠 URL 要求方面，Web 控制程式會查看控制程式指令的介面名稱，並採用該名稱來判斷登錄於 CMDREG 表格中的正確實作類別。

下表說明 URLREG 資料庫表格中所含的資訊。

直欄名稱	說明	備註
URL	URI 名稱	例如，MyNewCommand 或 com.ibm.commerce.catalog.commands.ProductDisplayCmd
STOREENT_ID	商店實體識別碼	可設爲 0，表示將指令用於所有商店上；或設成一個唯一的商店識別碼，表示該指令只用於特定的商店上。
INTERFACENAME	控制程式指令介面名稱	例 如 com.ibm.commerce.catalog. 指令。ProductDisplayCmdImpl
HTTPS	此 URL 要求是否需要安全 HTTP	當需要 HTTPS，請使用 1；若不需要，請使用 0。
DESCRIPTION	URI 的說明	例如，此指令作為測試用。
AUTHENTICATED	此 URL 要求是否要求使用者登入	當需要鑑別時，請使用 1；若不需要，請使用 0。
INTERNAL	指出該指令是否爲 WebSphere Commerce 的內部指令。	如果是內部指令，請使用 1，如果是外部指令，請使用 0。

當 Web 控制程式收到 URL 要求時，它會擷取所要求之控制程式指令的介面名稱，並透過該介面從 CMDREG 表格中查看實作類別名稱。此外，亦會檢查 URLREG 表格中的 HTTPS 直欄，判斷該 URL 要求是否需要 HTTPS。

只有因 URL 要求而呼叫的指令才需登錄在 URLREG 表格中。因此，只有控制程式指令才必須在此登錄，作業或檢視畫面指令則不必。

以下的 SQL 陳述式會建立 MyNewControllerCommand 的項目，且供特定商店使用（其商店識別碼爲 5）：

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCommand',
5,'com.ibm.commerce.commands.MyNewControllerCommand',0,
'This is a test command.',null)
```

insert 陳述式的一般語法如下：

```
 insert into table_name (column_name1,column_name2, ... ,column_namen)
values (column1_value,column2_value,...,columnn_value)
```

字串值應括上單引號。

CMDREG 表格

CMDREG 爲一種指令登錄表格。此表格提供一種將指令介面對映至其實作類別的機制。介面提供多種實作方式讓您根據商店來自訂指令。

只有控制程式指令與作業指令才能登錄在 CMDREG 表格中。檢視畫面指令則是登錄在 VIEWREG 表格中。

下表說明 CMDREG 資料庫表格中所含的資訊。

直欄名稱	說明	備註
STOREENT_ID	商店實體識別碼	可設爲 0，表示將指令用於所有商店上；或設成一個唯一的商店識別碼，表示該指令只用於特定的商店上。
INTERFACENAME	指令介面名稱	用以定義介面；請使用與您在 URLREG 表格中所定義的名稱。
DESCRIPTION	此指令的說明	例如，此指令作為測試用。
CLASSNAME	指令實作類別名稱	一般而言，是在介面名稱尾端加上 ″Impl″。
PROPERTIES	設爲指令之輸入內容的預設「名稱-值」對	其格式與 URL 查詢字串相同。例如 ″parm1=val1&parm2=val2″
LASTUPDATE	此指令項目前次更新的日期	
TARGET	指令的目標名稱。此爲實際執行指令的所在。	只支援本端目標。

一般而言，在您建立新控制程式或作業指令時，您應在 CMDREG 表格中建立對應項目。例如，下列 SQL 陳述式會爲特定商店（其商店 ID 爲 5）正在使用的 MyNewCommand 建立一個項目：

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION, CLASSNAME,
PROPERTIES, LASTUPDATE, TARGET) values
(5,'com.mycompany.commands.MyNewCommand', 'This is a test command',
'com.mycompany.commands.MyNewCommandImpl', 'myDefaulParm1=myDefaultVal1',
'0000-12-01', 'Local')
```

字串值必須以單引號括住。

如果您所撰寫的指令一律使用同樣的實作類別，您未必得將指令登錄於 CMDREG 表格中。在此情況下，您可以使用介面中的 defaultCommandClassName 屬性來指定實作類別。舉例來說，您可在介面的程式碼中包含下列：

```
String defaultCommandClassName =
     "com.ibm.commerce.command.MyNewCommandImpl"
```

如果您採用此方式來指定實作類別，您便無法將預設內容傳遞給實作類別，且必須將同一實作類別用於所有商店上。

已登錄之控制程式指令的範例說明

請假設下列情況：您的網站有兩家商店，分別是商店 A 與商店 B。每一家商店對於 MyUrl 控制程式指令各有不同的安全要求，且對於指令各有不同的實作方式。本節將顯示如何使用指令登錄來啓用此項自訂。

下表顯示商店 A 與商店 B 在 URLREG 表格中的項目：

直欄名稱	商店 A 的項目	商店 B 的項目
URL	MyUrl	MyUrl
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce.mycommands.myUrl	com.ibm.commerce.mycommands.myUrl
HTTPS	1	1
DESCRIPTION	在 URLREG 表格中的範例項目。	在 URLREG 表格中的範例項目。
AUTHENTICATED	1	0
INTERNAL	空值	空值

註: INTERFACENAME 值中的空格只爲了方便閱讀用。實際上每一個值皆爲一個連續字串。

Web 控制程式會根據 URLREG 表格中的項目，判斷出 MyURL URI 的介面名稱爲 com.ibm.commerce.mycommands.MyUrl。此外，亦判斷出商店 A 要求使用 HTTPS 與鑑別模型來執行指令，而商店 B 只要求使用 HTTPS。HTTPS 與鑑別模型的值是供 Web 控制程式使用，而非介面使用。

下圖顯示此項流程：

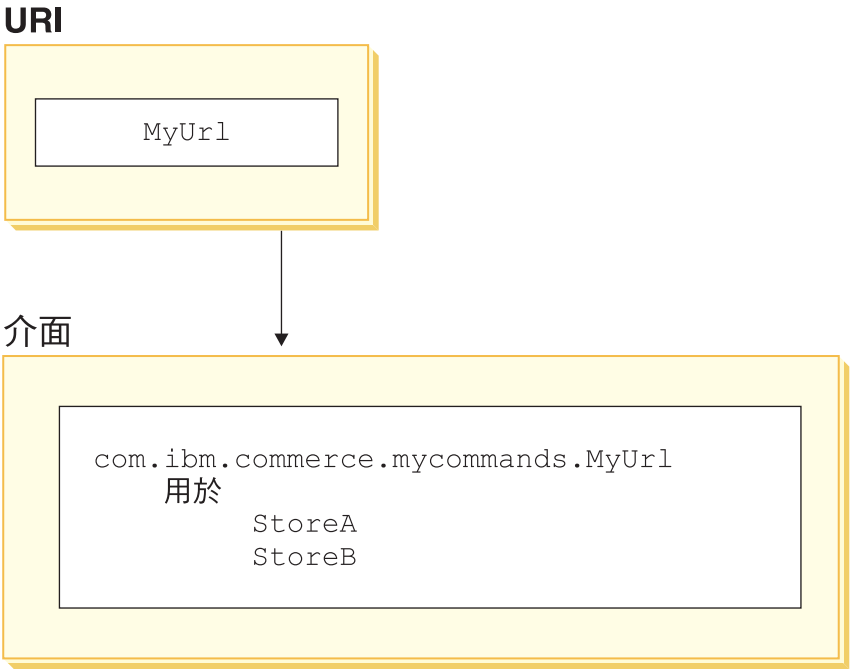

圖 9.

下表顯示 CMDREG 表格中的項目。在此僅顯示爲方便說明此範例而需用到的直欄：

直欄名稱	商店 A 的項目	商店 B 的項目
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce.mycommands.myUrl	com.ibm.commerce.mycommands.myUrl
CLASSNAME	com.ibm.commerce.mycommands.myUrlStoreAImpl	com.ibm.commerce.mycommands.myUrlStoreBImpl

註: INTERFACENAME 與 CLASSNAME 值中的空格只爲了方便閱讀用。實際上每一個值皆爲一個連續字串。

Web 控制程式會根據 CMDREG 表格中的項目，判斷出在商店 A 方面，com.ibm.commerce.mycommands.MyUrl 介面的實作類別爲 com.ibm.commerce.mycommands.MyUrlStoreAImpl。而在商店 B 方面，同一介面的實作類別爲 com.ibm.commerce.mycommands.MyUrlStoreBImpl。下圖顯示此項流程：

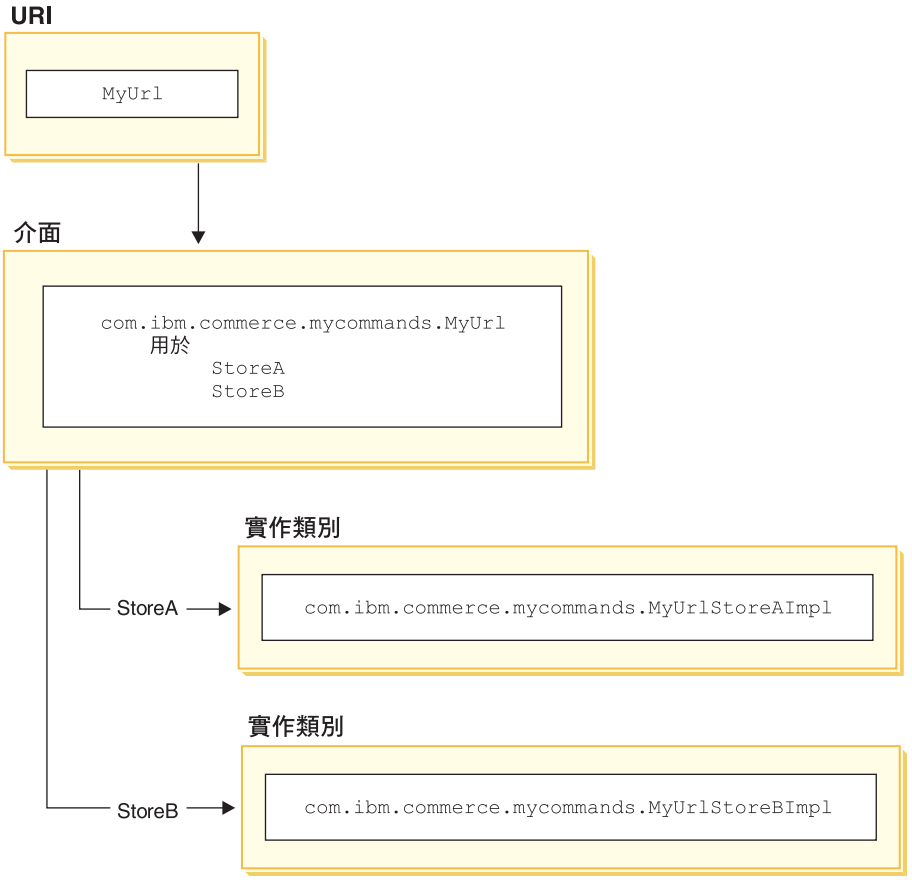

圖 10.

VIEWREG 表格

VIEWREG 表格可容許登錄裝置特定及商店特定的檢視畫面實作。透過此表格，可爲檢視畫面指令登錄多種實作方式。這樣，指令組織架構便能傳回不同的檢視畫面給各種裝置。

當控制程式指令傳回檢視畫面名稱，或者在異常狀況中指定檢視畫面名稱時， Web 控制程式會由 VIEWREG 表格來判斷檢視畫面實作。可以有多個檢視畫面名稱對映至同一個實作類別。

直欄名稱	說明	備註
VIEWNAME	檢視畫面名稱	例如 AddressForm

直欄名稱	說明	備註
DEVICEFMT_ID	裝置類型識別碼	可用的選項包括： • BROWSER（預設值） • I_MODE • E-mail • MQXML • MQNC
STOREENT_ID	商店實體識別碼	可設爲 0，表示將指令用於所有商店上；或設成一個唯一的商店識別碼，表示該指令只用於特定的商店上。
INTERFACENAME	檢視畫面指令介面名稱	預設選項有：ForwardView、DirectView 與 RedirectView。
CLASSNAME	檢視畫面指令的實作類別名稱	可使用預設實作方式。
PROPERTIES	設爲指令之輸入內容的預設「名稱-值」對	如果一律顯示同一頁面，請在此內容中設定 JSP 檔名稱 (docname=*jsp_name*.jsp)。 如果同一 JSP 範本用於所有商店上，請設定 storeDir=no，以防使用商店特定的目錄。 如果一般使用者可呼叫指令，請設定 isGeneric=true。
DESCRIPTION	此指令的說明	
HTTPS	此 URL 要求是否需要安全 HTTP	當需要 HTTPS，請使用 1；若不需要，請使用 0。
LASTUPDATE	此項目前次更新的日期	
INTERNAL	指出該指令是否爲 WebSphere Commerce 的內部指令。	如果是內部指令，請使用 1，如果是外部指令，請使用 0。

在您建立新檢視畫面時，您可能需要在 VIEWREG 表格中建立對應的項目。只要符合下列其中一個情況，就必須在 VIEWREG 表格中登錄檢視畫面：

- 檢視畫面是在存取控制下執行
- 該檢視畫面指令有多種實作方式
- 在 PROPERTIES 直欄中設定內容

已登錄的檢視畫面可透過檢視畫面登錄來存取，方法是使用檢視畫面名稱，或直接使用實際的顯示檔名稱。而未登錄在 VIEWREG 表格中的檢視畫面，則只能在用戶端使用實際的顯示檔名稱時才能存取到。

在下列範例中，假設檢視畫面名稱爲 *MyView*，且具有 VIEWREG 項目：

直欄名稱	項目
VIEWNAME	MyView
DEVICEFMT_ID	BROWSER
STOREENT_ID	0
INTERFACENAME	com.ibm.commerce.commands.ForwardViewCommand
CLASSNAME	com.ibm.commerce.commands.HTTPForwardViewCommandImp
PROPERTIES	docname=MyView.jsp
DESCRIPTION	此範例採用檢視畫面名稱來呼叫 JSP 範本，或直接從 URL 呼叫。
HTTPS	0
LASTUPDATE	2000-11-30
INTERNAL	0

由於 MyView 是已登錄的檢視畫面，用戶端可使用檢視畫面名稱，或將檢視畫面名稱替換成實際的顯示檔名稱，來存取檢視畫面。如果採用檢視畫面名稱，其範例 URL 如下：

```
http://hostname.com/webapp/wcs/stores/servlet/MyView
```

如果採用檔案名稱，其範例 URL 如下：

```
http://hostname.com/webapp/wcs/stores/servlet/MyView.jsp
```

如果用戶端有可能直接呼叫已登錄的檢視畫面（採用顯示檔名稱），則您必須如本範例（MyView 與 MyView.jsp）所示，使用與實際顯示檔名稱同名的檢視畫面名稱來登錄檢視畫面。

而未登錄於表格中的檢視畫面只能使用顯示檔名稱來呼叫。因此，假設有個未登錄的檢視畫面採用 MyUnregisteredView.jsp 檔，則用以存取此檢視畫面的 URL 如下：

```
http://hostname.com/webapp/wcs/stores/servlet/MyUnregisteredView.jsp
```

下列的範例 SQL 陳述式會爲某個特定商店使用的 *MyNewView* 建立一個項目：

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
CLASSNAME, PROPERTIES, DESCRIPTION,HTTPS, LASTUPDATE, INTERNAL) values
```

```
('MyNewView', -1, 5, 'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commmerce.command.HttpForwardViewCommandImpl',
'docname=MyNewView.jsp', 'A test view.', 0, '0000-12-01', 0)
```

下表提供另一個範例 VIEWREG 表格與主要資訊：

VIEW NAME	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplay View	轉遞檢視畫面指令	HttpForwardViewCommandImpl	docname= UserArea/ServiceSection/InterestItemListSubsection/WishListDisplay.jsp
通用應用程式錯誤	轉遞檢視畫面指令	HttpForwardViewCommandImpl	docname =GenericApplicationError.jsp&storeDir=no
GenericSystem 錯誤	轉遞檢視畫面指令	HttpForwardViewCommandImpl	docname = GenericSystemError.jsp&storeDir=no
LogonForm	轉遞檢視畫面指令	HttpForwardViewCommandImpl	docname = LoginForm.jsp&generic=true&storeDir=no

註: VIEWNAME、INTERFACENAME、 CLASSNAME 與 PROPERTIES 值中的任何空格都只爲了方便顯示。實際上每一個值皆爲一個連續字串。直欄名稱中的連字符號亦純粹爲了方便閱讀用。

上表所描述的情況如下：

- 控制程式指令（在本範例中爲 ProductDisplay）將 *ProductDisplayView* 檢視畫面名稱傳回給 Web 控制程式。 Web 控制程式透過 ProductDisplayView 檢視畫面指令名稱與其裝置識別碼，判斷出檢視畫面指令介面與類別名稱。檢視畫面指令可針對不同的商店與裝置識別碼，使用不同的實作類別。不過，介面名稱應維持相同，這是因爲它會定義檢視畫面指令類型。
- 如果控制程式或作業指令因某個使用者參數不正確而擲出 ECApplication 異常狀況，則可能發生下列情況：
  - 如果控制程式指令中有指定一個應在發生應用程式異常狀況時呼叫的檢視畫面，則會從 VIEWREG 表格中擷取該檢視畫面的項目，並加以處理。

- 若未指定檢視畫面，則會呼叫 GenericApplicationError 指令，並顯示登錄在資料庫中的 JSP 範本。以上面的表格爲例，這樣就會顯示 `GenericApplicationError.jsp` 範本。

- 如果控制程式或作業指令因系統異常狀況而擲出 ECSystem 異常狀況，則可能發生下列情況：
  - 如果控制程式指令中有指定一個應在發生系統異常狀況時呼叫的檢視畫面，則會從 VIEWREG 表格中擷取該檢視畫面的項目，並加以處理。
  - 若未指定檢視畫面，則會呼叫 GenericSystemError 指令，並顯示登錄在資料庫中的 JSP 範本。以上面的表格爲例，這樣就會顯示 `GenericSystemError.jsp` 範本。
- 瀏覽器用戶端可藉由輸入登入 URL 以呼叫登入頁面。由於 storeDir 內容設爲 "no"，因此在 JSP 範本的路徑中將不含商店特有的資訊。也因此，所有商店的客戶所看到的是相同的登入頁面。

顯示設計型樣

顯示頁面會傳回一則回應給用戶端。一般而言，顯示頁面採 JSP 範本方式實作（建議採用此方法），不過，您也可以將之直接寫成 Servlet。

爲了支援多種裝置類型，會存取檢視畫面指令的 URL 應使用檢視畫面名稱，而非使用實際 JSP 檔的名稱。

此間接層面背後的主要原理是 JSP 範本代表一個檢視畫面。較理想的結果是能夠選出適當的檢視畫面（例如，根據要求環境定義中的語言環境、裝置類型或其他資料），特別是單一要求中常有多種可能的檢視畫面。請想想下列情況：有兩位購物者要求顯示商店首頁，其中一位購物者使用典型的 Web 瀏覽器，另一位則使用行動電話。顯然地，這兩位購物者所看到的首頁應不會一樣。這是因爲 Web 控制程式將負責接受要求，並根據指令登錄組織架構中的資訊，來決定每位購物者將收到的檢視畫面。

JSP 範本與資料 Bean

資料 Bean 是一種在 JSP 範本中使用的 Java Bean，用以提供動態內容。資料 Bean 通常代表簡化的 WebSphere Commerce Entity Bean。資料 Bean 概括了一些可從 Entity Bean 中擷取或設於 Entity Bean 中的內容。因此，資料 Bean 簡化了將動態資料納入 JSP 範本的作業。

資料 Bean 具有一個 *BeanInfo* 類別，用以定義可用於顯示頁面上的內容。另外，BeanInfo 類別可讓資料 Bean 能用於多種文化型網站中，這是因爲它會提供採 WebSphere Commerce 所支援之所有語言的內容名稱。

資料 Bean 是藉由下列呼叫而啓動的：

```
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

其中 *data_bean* 是要啓用的資料 Bean，而 *request* 是 HTTPServletRequest 物件。

商店程式開發人員在開發 JSP 範本時，應將商店的內容與全球化的課題納入考慮。有關全球化的其他資訊，請參閱 *WebSphere Commerce 商店程式開發手冊*。

資料 Bean 安全注意事項

使用特定的程式碼撰寫習慣來使用資料 Bean 時，可將懷有惡意的使用者擅自存取您資料庫的可能性降至最低。 SQL 陳述式中的 Insert、select、update 與 delete 部份應在開發期間建立。您可使用插入參數方式來收集執行期間輸入資訊。

以下是使用插入參數方式來收集執行期間輸入資訊的範例：

```
select * from Order where owner =?
```

相對地，您應避免使用輸入字串作爲撰寫 SQL 陳述式的方法。以下是使用輸入字串的範例：

```
select * from Order where owner = "input_string"
```

資料 Bean 的類型

資料 Bean 是一種主要用來在 JSP 範本中提供動態資料的 Java Bean。資料 Bean 的類型有下列兩種：智慧型資料 Bean 與指令資料 Bean。

智慧型資料 Bean 採用*延緩提取*方式以擷取本身的資料。若在不需用到存取 Bean 中之所有資料的情況下，此種資料 Bean 類型可提供理想的效能，這是因爲它只在必要時才會擷取資料。而需要存取資料庫的智慧型資料 Bean 應是從對應 Entity Bean 的存取 Bean 延伸而來，且會實作 com.ibm.commerce.SmartDataBean 介面。舉例來說， ProductData 資料 Bean 爲 ProductAccessBean 存取 Bean 的延伸，並且會對應至「產品」 Entity Bean。

有些智慧型資料 Bean 不需進行資料庫存取。舉例來說， PropertyResource 智慧型資料 Bean 是從資源連結中擷取資料，而非從資料庫中。在不需存取資料庫時，智慧型資料 Bean 應爲 SmartDataBeanImpl 類別的延伸。

指令資料 Bean 是靠指令擷取其資料，可說是較輕量的資料 Bean。指令會立即擷取資料 Bean 的所有屬性，而不管 JSP 這些範本是否需要。因此，對於只會從資料 Bean 中挑選所要屬性的 JSP 範本而言，從效能時間觀點來看指令資料 Bean 頗爲費時。但對於需用到大部份甚至全部屬性的 JSP 範本而言，指令資料 Bean 相當方便。

指令資料 Bean 也可以從其對應的存取 Bean 延伸而來，且會實作 com.ibm.commerce.CommandDataBean 介面。

資料 Bean 介面

資料 Bean 會實作下列之一或所有的 Java 介面：

- com.ibm.commerce.SmartDataBean
- com.ibm.commerce.CommandDataBean
- com.ibm.commerce.InputDataBean（選用）

每一個 Java 介面用以說明資料 Bean 的移入資料來源。藉由實作多種介面，資料 Bean 可存取不同來源中的資料。以下是各介面的進一步說明。

SmartDataBean 介面: 實作 SmartDataBean 介面的資料 Bean 可擷取本身的資料，而不用透過相關的資料 Bean 指令。智慧型資料 Bean 通常是從對應 Entity Bean的存取 Bean 延伸而來。當啓動智慧型資料 Bean 時，資料 Bean 管理程式會呼叫資料 Bean 的移入方法。藉由移入方法，資料 Bean 可擷取到所有屬性，除了相關物件中的屬性外。舉例來說，假設資料 Bean 是從 Entity Bean 的存取 Bean 類別延伸而來，而該資料 Bean 會呼叫 refreshCopyHelper 方法。在此情況下，會自動將對應 Entity Bean 中的所有屬性移入到智慧型資料 Bean 中。不過，假設 Entity Bean 具有相關聯的物件，則不會擷取這些物件中的屬性。使用智慧型資料 Bean 的主要優點有：

- 實作方式較爲簡單，且不需撰寫資料 Bean 指令。
- 當 Entity Bean 中加入新欄位時，並不需要在資料 Bean 中進行變更。在 Entity Bean 被修改後，必須重新產生存取 Bean（採用 WebSphere Studio Application Developer 中的工具）。一旦重新產生存取 Bean，就會自動提供所有的新屬性給智慧型資料 Bean 使用。
- Entity Bean 中常含有一些代表相關聯物件的屬性。爲求效能，智慧型資料 Bean 並不會自動擷取這些屬性。反而它會延緩擷取這些屬性，直到有需要用到爲止；請見下圖：

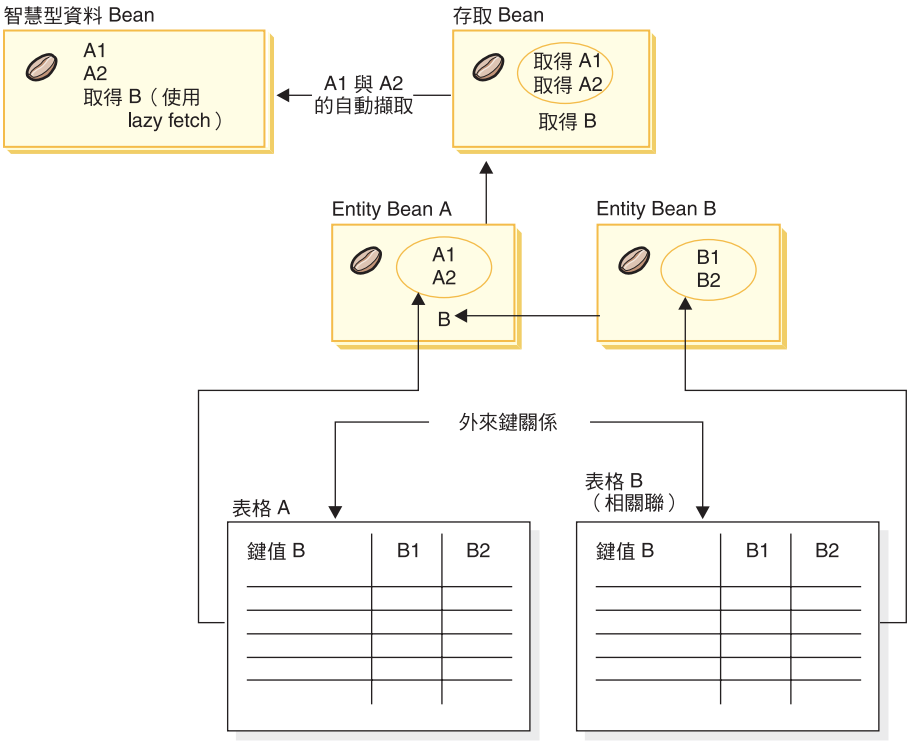

圖 11.

有關實作「延緩提取」擷取方式的詳細說明，請參閱第 41 頁的『延緩提取資料擷取方式』。

CommandDataBean 介面: 實作 CommandDataBean 介面的資料 Bean 則是從資料 Bean 指令擷取資料。此種類型的資料 Bean 屬於輕量物件；它仰賴資料 Bean 指令移入其資料。資料 Bean 必須實作 getCommandInterfaceName() 方法（由 com.ibm.commerce.CommandDataBean 介面所定義），而此方法將傳回資料 Bean 指令的介面名稱。

InputDataBean 介面: 實作 InputDataBean 介面的資料 Bean 則是從檢視畫面指令所設的 URL 參數或屬性中擷取資料。

定義於此介面中的屬性可作爲提取其他資料的主要鍵欄位。當呼叫 JSP 範本時，所產生的 JSP Servlet 程式碼中會移入所有符合 URL 參數的屬性，並藉由將資料 Bean 傳遞給資料 Bean 管理程式，以啓動資料 Bean。接著，資料 Bean 管理程式會呼叫資料 Bean 的 setRequestProperties() 方法（由 com.ibm.commerce.InputDataBean 介面所定義），以傳遞檢視畫面指令所設的所有屬性。請注意，必須有下列程式碼才能啓用資料 Bean：

`com.ibm.commerce.beans.DataBeanManager.activate(`*`data_bean`*`, `*`request`*`)`

其中 *data_bean* 是要啓用的資料 Bean，而 *request* 是 HTTPServletRequest 物件。

BeanInfo 類別

資料 Bean 若少了會實作 java.lang.Object.BeanInfo 介面的 BeanInfo 類別便不完整。BeanInfo 類別用來提供資料 Bean 之方法與內容的相關明確資訊。它可用來在資料 Bean 實作類別中隱藏公用的執行期間方法而不讓 Web 設計人員看到，或爲資料 Bean 的每一個屬性設定適當的顯示字串。

有關實作 BeanInfo 類別的詳細資訊，請參閱 Sun Microsystems 的 JavaBeans 規格。

啓動資料 Bean

您可使用 activate 或 silentActivate 方法（可在 com.ibm.commerce.beans.DataBeanManager 類別中找到）來啓動資料 Bean。 activate 方法爲一種全面的啓動方法；亦即，只有在所有屬性皆可用時，啓動事件才會成功。只要有一個屬性無法使用，即會針對整個啓動程序擲出異常狀況。

silentActivate 方法則在有個別屬性無法使用時並不會擲出異常狀況。

從 JSP 範本中呼叫控制程式指令

雖然從 JSP 範本內呼叫控制程式指令和將邏輯與顯示區隔開來並不一樣，您仍有可能會遇到需要採行這種方式的情況。當遇到此情況時，您可使用 `ControllerCommandInvokerDataBean`。

使用這個資料 Bean，您可以指定所要呼叫之指令的介面名稱，或者可直接設定所要呼叫的指令名稱。您也可以爲指令設定要求內容。

當資料 Bean 管理程式啓動這個資料 Bean 時，會執行控制程式指令，並爲 JSP 範本提供回應內容。

如果您在啓用這個資料 Bean 之前沒有使用 setRequestProperties 方法，則來自要求物件的參數會傳送到 Bean，同時也會傳送到控制程式指令。不過，如果您已經在啓用這個資料 Bean 之前呼叫 setRequestProperties 方法，則只會將指定內容（已傳送到 setRequestProperties 方法中的內容）以及 CMDREG 表格的 PROPERTIES 直欄中指定的任何預設內容提供給指令使用。

一旦執行控制程式指令，您可以執行檢視畫面。

您不應該重覆使用同一個資料 Bean 實例來呼叫其他控制程式指令，因爲它包含其原始用法的資料和狀態資訊。

延緩提取資料擷取方式

當啓動資料 Bean 時，可透過資料 Bean 指令或資料 Bean 的 populate() 方法，將資料移入該資料 Bean 中。所擷取的屬性乃取自資料 Bean 的對應 Entity Bean。Entity Bean 可能會有相關聯的物件，而這些物件本身亦有一些屬性。

一旦啓動時，若自動擷取所有相關聯物件中的屬性，則可能會出現效能問題。效能可能會因相關聯物件的增加而降低。

假設有一個產品資料 Bean 含有大量的交叉銷售產品、高級品推薦產品或配件產品（亦即相關聯的物件）。一旦啓動該產品資料 Bean 時，便有可能移入所有相關聯的物件。不過，以此方式移入可能需要多次查詢資料庫。如果頁面並不需用到所有屬性，則多次查詢資料庫可能使效率降低。

一般而言，頁面通常不會用到所有的屬性，因此較理想的設計型樣是按如下般執行「延緩提取」方式：

```
getCrossSellProducts () {
   if (crossSellDataBeans == null)
     crossSellDataBeans= getCrossSellDataBeans();
   return crossSellDataBean;
    }
```

設定 JSP 屬性 - 概觀

WebSphere Commerce 程式設計模型改良了 MVC 設計型樣。因此，在 URL 要求結果的呈現上，已有別於控制程式與作業指令。這些指令並不取決於裝置而定。它們會實作將傳回給用戶端的商業邏輯與產品資料，而不含有用戶端的相關資訊。相反地，檢視畫面指令則會取決於裝置而定。

儘管控制程式與作業指令並不直接撰寫檢視畫面，但仍會傳遞資訊給檢視畫面。因此瞭解如何將資訊傳遞給檢視畫面很重要。下圖示範內容如何在 Web 控制程式、指令登錄、控制程式指令與檢視畫面指令之間傳遞：

CMREG

INTERFACENAME		PROPERTIES
com.ibm.xxx. NewCommand		parm1=1&parm2=2

CCPd: parm1=1&parm2=2

VIEWREG

INTERFACENAME		PROPERTIES
com.ibm.command.ForwardViewCommand		docname=NewView.jsp

VPd: docName=NewView.jsp

URL: http://*hostname*/webapp/wcs/stores/servlet/NewCommand?storeId=1&...

CCPu: storeID=1&...

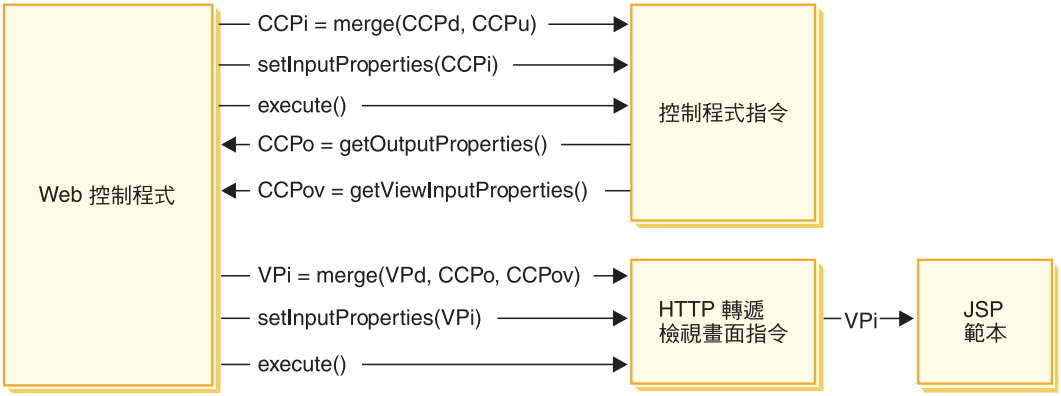

圖 12.

上圖顯示如下的互動：

- Web 控制程式合併了 URL 參數 (CCPu) 中的輸入內容以及控制程式指令 (CCPd) 在 CMDREG 表格中的項目。因而建立了 CCPi。
- Web 控制程式將合併後的內容 (CCPi) 傳給控制程式指令，並執行控制程式指令。

- 控制程式指令設定輸出內容 (CCPo)。這些是指令本身產生的輸出內容。將其中一個輸出內容 viewCommandName 設爲所要的檢視畫面指令名稱。Web 控制程式會使用 get 方法擷取這些內容。
- 控制程式指令設定另一組輸出內容 (CCPov)。在預設的情況下，會將這些設爲原來所合併的輸入內容 (CCPi)。這些內容可供自訂。舉例來說，不見得需將所有輸入參數傳給檢視畫面指令。
- Web 控制程式將 CCPo、CCPov 與 VPd（登錄於 VIEWREG 表格中的內容）這三組內容合併成檢視畫面指令的輸入內容 (VPi)。
- Web 控制程式設定合併後的內容 (VPi)，並執行檢視畫面指令。
- 檢視畫面指令藉由輸入內容將屬性設定到 JSP 範本中。

在您撰寫新指令時，您不必明確執行內容的合併。在抽象指令類別中即含有 mergeProperties 方法。有關這個方法的其他資訊，請參閱 WebSphere Commerce 正式作業與開發線上說明中的「參照」主題。

必要的內容設定

控制程式指令必須針對每一種檢視畫面指令類型設定下列內容。如果指令未設定內容，則必須定義於 VIEWREG 表格中。

- 如果採用 ForwardView 指令，請設定 `docname = view_file_name`；其中 *view_file_name* 爲顯示範本的名稱。例如，`docname=SearchResult.jsp`。
- 如果採用 DirectView 指令，請執行下列之一：
  - 設定 `textDocument = xxx`；其中 *xxx* 代表內含表單文字文件的 java.io.InputStream 物件。
  - 設定 `rawDocument = yyy`；其中 *yyy* 代表內含表單二進位文件的 java.io.InputStream 物件。

  在使用 DirectView 指令時，可以選擇設定 `contentType = ttt`，其中 *ttt* 爲文件內容類型
- 如果採用 RedirectView 指令，請設定 `url = uuu`；其中 *uuu* 爲重新導向 URL。

第 3 章 持續性物件模型

WebSphere Commerce 會處理大量的持續性資料。現行的資料庫綱目中已定義了相當多的表格。即使此綱目相當廣泛，您可能仍得延伸或自訂資料庫綱目，以符合自己的特定商業需求。

WebSphere Commerce 採用以 Enterprise JavaBeans (EJB) 1.1 版元件架構爲基礎的 Entity Bean 作爲持續性物件層。這些 Entity Bean 仿造了商務領域中的概念與物件，藉以呈現 WebSphere Commerce 資料。此持續層提供可延伸的組織架構。

WebSphere Studio Application Developer 提供了複雜的 EJB 工具與單元測試環境，可支援這個組織架構的開發。

下列各節是在 WebSphere Commerce 持續物件模型實作的實作環境定義內，該實作使用 EJB 1.1 規格。

WebSphere Commerce Entity Bean 的實作

WebSphere Commerce Entity Bean - 概觀

一如上述，WebSphere Commerce 結構中的持續層是根據 EJB 元件架構來實作的。EJB 架構定義兩種 Enterprise Bean 類型：Entity Bean 與 Session Bean。 Entity Bean 則進一步細分爲「由儲存區所管理的持續性 (CMP)」Bean 與「由 Bean 所管理的持續性 (BMP)」Bean。

大部份的 WebSphere Commerce Entity Bean 皆屬於 CMP Entity Bean。有少數無狀態式 Session Bean 會用來處理一些密集的資料庫作業，像是執行特定直欄中之所有列的加總。使用 CMP Entity Bean 的優點之一是程式開發人員可利用 WebSphere Studio Application Developer 中所提供的 EJB 工具。這些工具可讓開發人員定義 Java 物件與其資料庫表格的對映。這些工具會自動爲 Entity Bean 產生必要的持續器。持續器爲一種 Java 物件，它可讓 Java 欄位延伸到資料庫中，並在 Java 欄位中移入資料庫中的資料。

WebSphere Studio Application Developer 提供兩種 EJB 1.1 規格的延伸：EJB 繼承與連結。EJB 繼承特性可讓 Enterprise Bean 繼承同一群組中之另一個 Enterprise Bean 的內容、方法與方法層次的控制描述子屬性。連結特性是指存在於兩個 CMP Entity Bean 間的關係。

有些 WebSphere Commerce Entity Bean 會利用到 EJB 繼承特性。WebSphere Commerce Entity Bean 不會用到 WebSphere Studio Application Developer 所提供的連結特性。在您開發自己的 Entity Bean 時，建議您不要使用 WebSphere Studio Application Developer 的連結特性。這是希望將物件模型中的複雜程度降至最低。在不使用 WebSphere Studio Application Developer 所提供的連結特性下，您可以藉由在 Enterprise Bean 中新增明確的 getter 方法，來建立 Enterprise Bean 間的物件關係。

WebSphere Commerce 提供兩組 Enterprise Bean：私用與公用。私用 Enterprise Bean 是供 WebSphere Commerce 執行期間環境與工具使用。您不得使用或修改這些 Bean。

反過來說，公用 Enterprise Bean 可供商務應用程式使用與延伸。這些公用 Enterprise Bean 可分成下列幾個 EJB 模組：

- Catalog-ProductManagementData
- Enablement-RelationshipManagementData
- Marketing-CampaignsAndScenarioMarketingData
- Marketing-CustomerProfilingAndSegmentationData
- Member-MemberManagementData
- Merchandising-PromotionsAndDiscountsData
- Order-OrderCaptureData
- Order-OrderManagementData
- Trading-AuctionsAndRFQsData

前述清單中的某些 EJB 模組包含 Session Bean。爲了簡化日後的移轉工作，您最好不要修改 Session Bean 類別。必要時，您可以在 WebSphereCommerceServerExtensionsData EJB 模組中建立新 Session Bean。有關建立新 Session Bean 的詳細資訊，請參閱第 72 頁的『撰寫新 Session Bean』。

WebSphere Commerce Enterprise Bean 的部署描述子

EJB 部署描述子包含 Enterprise Bean 的部署設定。 WebSphere Studio Application Developer 提供了一個 EJB 部署描述子編輯程式，可用來修改這個部署資訊。

在建立新的 Enterprise Bean（Entity 或 Session Bean）時，部署描述子資訊是在 WebSphere Studio Application Developer 中的 J2EE 視景的「J2EE 階層」檢視畫面中設定的。您可以執行下列步驟來檢視 WebSphere Commerce Bean 的 EJB 部署描述子：

1. 開啓 WebSphere Studio Application Developer 並切換到 J2EE 視景。

2. 使用「J2EE 階層」檢視畫面，尋找您要檢視部署描述子資訊的 EJB 模組。
3. 以滑鼠右鍵按一下 *EJB_moduleName* EJB 模組，然後選取**開啓工具 > 部署描述子編輯程式**。
   這時會開啓「部署描述子編輯程式」。
4. 選取 Bean 標籤並注意下列項目：
   a. 從 Bean 的清單中選取一個 Bean。該 Bean 的資訊就會移入其他欄位中。
   b. Bean 應該顯示成「配置區所管裡的實體」1.x Bean。
   c. 請勿選取「重入」勾選框。
   d. 在「WebSphere 連結」區段中，會使用 JNDI 名稱的預設值。
   e. 在「WebSphere 延伸」區段中，並行控制並未啓動樂觀鎖定。
   f. 同時，您可以在「搜尋器」區段中檢視 Bean 的搜尋器。
5. 選取「組譯描述子」標籤，並注意下列項目：
   a. 在「方法許可權」區段中，WCSecurityRole 會指定給 Enterprise Bean 中的所有方法。
   b. 在「配置區交易」一節中，您必須爲 Enterprise Bean 中的所有方法指定“必要的”。
6. 選取「存取」標籤並注意下列項目：
   a. 在「實體 1.x 的存取含意」區段中，已經定義所有的唯讀方法。例如，_copyFromEJB() 和寫入程式碼中的 getter 方法都會指定成唯讀方法。當您建立自己的 Entity Bean 時，請確定您將適當的方法標示成唯讀。如果您未採用此方式來標示唯讀方法，則 EJB 儲存器會在交易結束時無謂地試著更新資料庫，因而在唯讀交易中造成交易回復錯誤。這會造成效能問題。
   b. 「隔離層次」是依照下列方式設定：
      - DB2 可重複讀取
      - Oracle 讀取已確定

延伸 WebSphere Commerce 物件模型

您可採用下列方式來延伸 WebSphere Commerce 物件模型：

- 延伸 WebSphere Commerce 的公用 Enterprise Bean
- 撰寫新 Entity Bean
- 撰寫新無狀態式 Session Bean

有關如何進行這些延伸的詳述，請見下列各節。

物件模型的延伸方法

應用程式的需求可能會促使您延伸現有的 WebSphere Commerce 物件模型。這類的需求像是在您應用程式中新增其他屬性。這可藉由下列方法之一來完成：

不修改現有的 WebSphere Commerce 公用 Entity Bean

建立新資料庫表格，然後爲該表格建立一個新 Entity Bean。視需要在該 Entity Bean 中新增欄位與方法，以操作新屬性。針對新 Entity Bean 產生部署程式碼與一個存取 Bean。當應用程式需要新屬性時，它會實例化存取 Bean 物件，並使用其方法來擷取、設定或操作屬性。

修改現有的 WebSphere Commerce 公用 Entity Bean

建立新資料庫表格，並在新表格與對應至您所修改之現有 Enterprise Bean 的現有表格間建立一項表格合併。在現有 WebSphere Commerce 公用 Entity Bean 中建立新欄位，並使用次要表格對映，將欄位對映至新表格中的對應直欄。新增任何必要的方法。針對現有 Entity Bean 重新產生部署程式碼與存取 Bean。當應用程式實例化存取 Bean 物件時，即可用到新屬性。

這兩種方法間有所取捨。一般而言，這些取捨關係到程式碼維護的效能與成效。

延伸範例: 在下列範例中，假設您的應用程式要求您擷取客戶的住宅類型。您建立了一個 USERRES 表格，內含客戶 ID 與住家類型，其中住家類型 (resType) 可爲自有住宅、公寓大廈或公寓。這種資訊類型屬於個人背景資訊，因而和現有 Commerce Suite USERDEMO 表格有關。在檢查 WebSphere Commerce 程式碼儲存庫時，您會發現 Member-MemberManagementData EJB 模組中有一個 ″Demographics″ Enterprise Bean。這個 Bean 將個人背景資訊的 getter 與 setter 儲存在 USERDEMO 表格中。

如果要自訂，您有兩項選擇。您可以建立一個會和 USERRES 表格互動的新 Entity Bean，或者在 Demographics Bean 中加入新欄位（外加適當的 getter 與 setter 方法）。

當使用第一種方法（建立全新的程式碼）時，您將建立一個新的 Userres Entity Bean，並將其欄位對映至 USERRES 表格中的直欄。當應用程式需要客戶的住家類型時，它必須實例化一個 Userres 存取 Bean 物件並擷取資料。如果應用程式同時需要其他的個人背景資訊時，亦必須實例化 Demographics 存取 Bean 物件，並擷取其他任何必要的屬性。在應用程式邏輯中，凡會試著擷取客戶之整組個人背景資訊的環節皆必須修改，以實例化新存取 Bean 以及原始的存取 Bean。下圖顯示此種延伸物件模型方法：

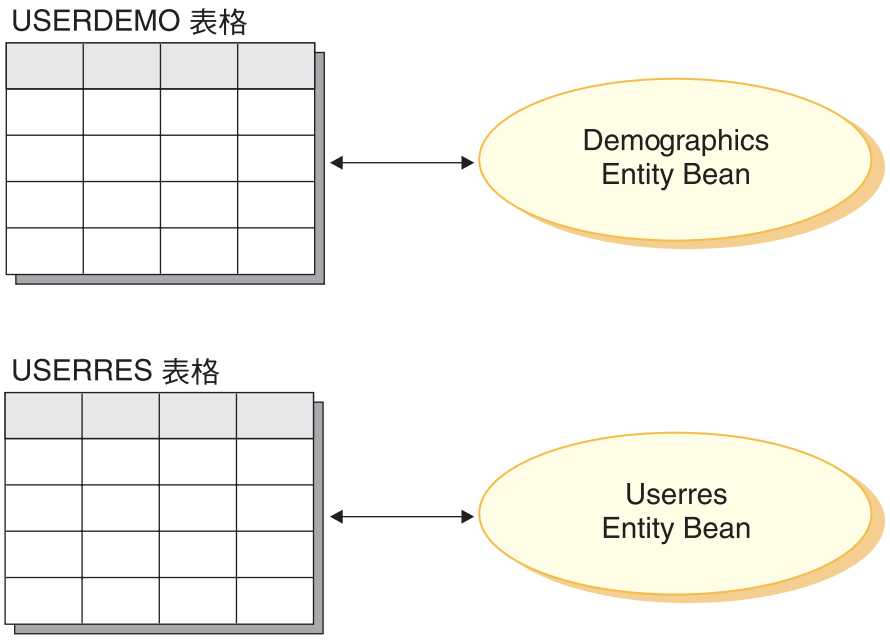

圖 13.

從顯示範本層面來看，資料 Bean 必須能夠存取新屬性，以便讓 JSP 範本能使用資訊。爲了提供統一的檢視畫面給建立 JSP 範本的 Web 程式開發人員，您應爲原來的現有 Entity Bean 建立一個延伸了存取 Bean 的新資料 Bean。資料 Bean 也應使用 delegation（委託）以便從新存取 Bean 移入屬性。下圖顯示這種資料 Bean 實作實務：

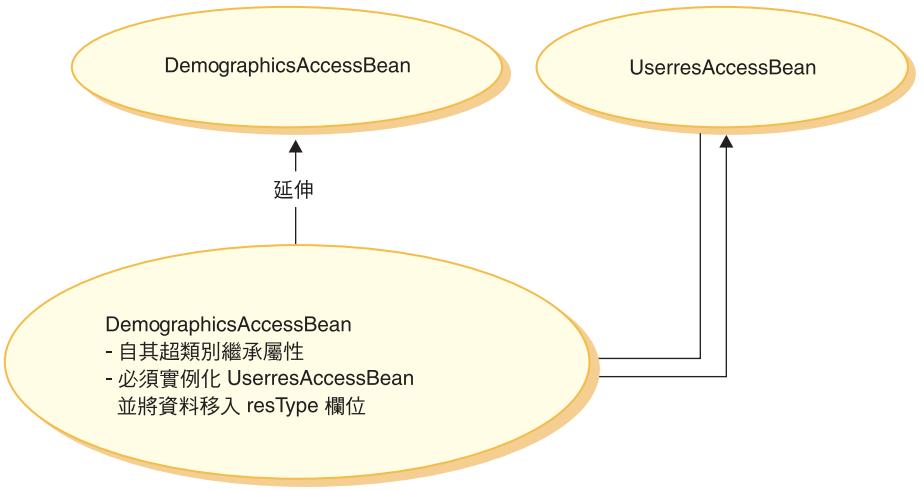

圖 14.

如果您使用第二種方法（修改現有程式碼），則您將新欄位加到 Demographics Entity Bean 中，並在新欄位與 USERRES 表格中的適當直欄間，建立次要的表格對映。當應用程式需要客戶的住家類型時，它會實例化 Demographics 存取 Bean 物件並擷取住家類型。如果應用程式需要客戶其他任何個人背景資訊時，即可在同一 Bean 呼叫中用到。下圖顯示此種 Enterprise Bean 修改方法：

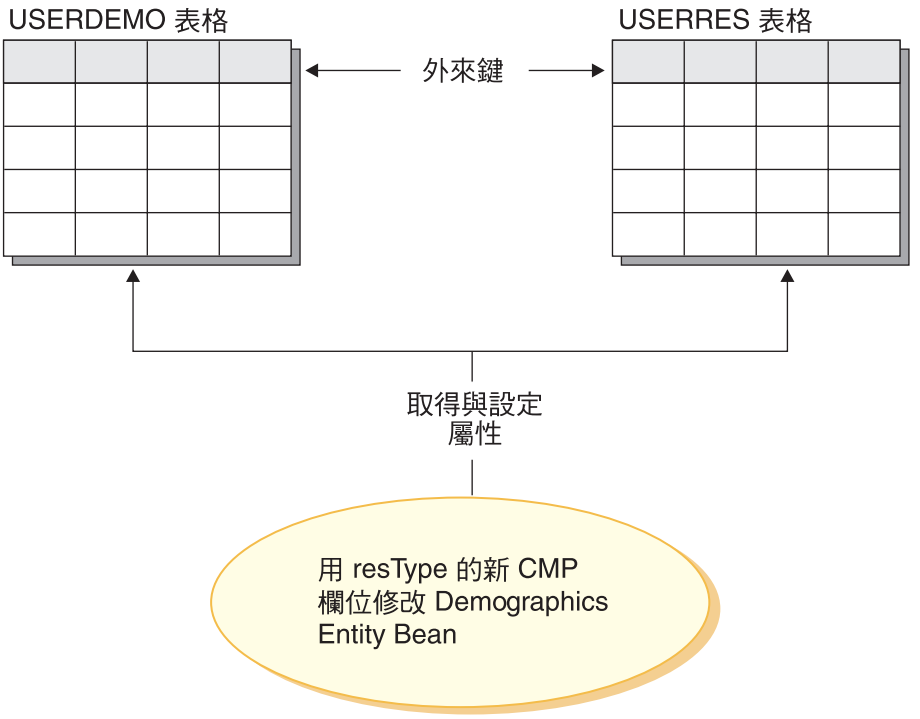

圖 15.

從顯示範本層面來看，只要重新產生 DemographicsAccessBean，新屬性 (resType) 即會自動提供於資料 Bean 中。

請注意，當您延伸物件模型時，*請勿*在現有 WebSphere Commerce 資料庫表格中新增直欄。您必須爲新屬性建立一個新表格。如果您試著在現有表格中新增直欄，當您移轉至未來的 WebSphere Commerce 版次時，新屬性將會遺失。

牽涉到效能與程式碼維護的問題: 第二種方法具有較好的執行期間效能。這是因爲在取得或設定新屬性時，只需實例化單一 Entity Bean，並且是使用單一提取來擷取所有必要的屬性。

由於第二種方法會修改現有的 WebSphere Commerce 程式碼，在發行新版的 WebSphere Commerce 時，將會出現移轉問題。您必須將您自訂的程式碼和新程式碼合併，但當您匯入新的 WebSphere Commerce 工作區時，將不會保留您新增到 Enterprise Bean 的欄位與新表格之間的對映資訊。因此，當您要移轉至新的 WebSphere Commerce 程式碼版次時，必須執行下列步驟：

1. 將您自訂的 EJB 程式碼版本化。
2. 匯入新的 WebSphere Commerce 程式碼版本。

3. 使用 WebSphere Studio Application Developer 中的工具，比較程式碼自訂版本與新的 WebSphere Commerce 程式碼版次。將您自訂的程式碼合併回您的工作區中。
4. 將您新增到 WebSphere Commerce 公用 Enterprise Bean 中的任何屬性，用手動方式重新對映至您資料庫中的適當直欄。
5. 爲您在步驟 4 中所修改的 Enterprise Bean，重新產生部署程式碼與存取 Bean。

爲了簡化此項移轉，請務必在開發期間完整記錄您物件模型的延伸。

如果要對物件模型進行多項延伸，您也可以混合使用這兩種方法。您可以針對較不會受效能降低影響的系統區域使用第一種方法，並針對會有效能問題的區域使用第二種方法。在此方式下，除了可將日後要作的移轉工作減至最少，還可維持理想的系統效能層次。

Session Bean 的建議用法

WebSphere Commerce 具備強大功能的原因之一是它能夠善用儲存器管理的持續 (CMP) Entity Bean。CMP Entity Bean 是一種分散、持續和交易式的伺服器端 Java 元件，並且可以利用 WebSphere Studio Application Developer 提供的工具來產生。在眾多情況中，就物件持續性而言，CMP Entity Bean 是一種相當好的選擇，您可讓它們的運作效率至少等於甚至超過其他「物件對關聯性」對映選項。基於這些原因，WebSphere Commerce 已使用 CMP Entity Bean 來實作核心商務物件。

不過，在某些情況下，建議您使用 Session Bean JDBC helper。這些情況包括：

- 當查詢傳回大量的結果集時。這可稱爲*大量結果集*情況。
- 當查詢擷取數個表格中的資料時。這可稱爲*聚集實體情況*。
- 當 SQL 陳述式執行需要大量使用資料庫的作業時。這可稱爲*任意 SQL* 情況。

下列各節將進一步詳細說明。

請注意，如果 Session Bean 是做爲 JDBC 外層用，以擷取資料庫中的資訊，則較難實作資源層次的存取控制。當在此種方式下使用 Session Bean 時，Session Bean 的程式開發人員必須在 “`select`” 陳述式中新增適當的 “`where`” 子句，以防未獲授權的使用者存取資源。

大量結果集情況: 在某些情況下查詢會傳回大量結果集，且所擷取的資料主要供讀取或顯示用。在此情況下，建議您使用無狀態 Session Bean，並在該 Session Bean 中，建立一個 Finder 方法，以執行和 Entity Bean 中之 Finder 方法相同的功能。亦即，無狀態 Session Bean 中的 Finder 方法應執行下列動作：

- 執行™ SQL select 陳述式
- 針對每個提取的列，實例化存取 Bean

- 針對每一個擷取的直欄，在存取 Bean 中設定對應的屬性

當傳回存取 Bean 時，指令並不清楚存取 Bean 是由 Session Bean 中的 Finder 方法所傳回，或是由 Entity Bean 中的 Finder 方法傳回。因此，在 Session Bean 中使用 Finder 方法並不會對程式設計模型造成任何變更。只有發出呼叫的指令知道它所呼叫的是 Session Bean 或 Entity Bean 中的 Finder 方法。它對於程式設計模型中的其他所有部份是透明化的。

聚集實體情況: 在此情況下檢視畫面是由幾個物件部份組成，且會在單一顯示頁面中移入出自數個資料庫表格中的資訊。以「我的帳戶」的概念爲例。這可由客戶資訊表格中的資訊（如：客戶名稱、年齡與客戶 ID）以及地址表格中的資訊（如：由路名與縣市構成的地址）組成。

您可以建構一個簡單的 SQL 陳述式，以藉由執行 SQL 結合，來擷取各種表格中的所有資訊。這可說是執行「深度提取」。下列是"我的帳戶"範例的 SQL select 陳述式範例，其中 CUSTOMER 表格爲 T1，ADDRESS 表格爲 T2：

```
select T1.NAME, T1.AGE, T2.STREET, T2.CITY
  from CUSTOMER T1, ADDRESS T2
  where (T1.ID=? and T1.ID=T2.ID)
```

WebSphere Studio Application Developer 中供 EJB 1.1 規格的 Enterprise Bean 使用的工具並不支援這種深度提取。相反地，它採用延緩提取方式，因此每一個相關的物件都有一個 SQL select。對於擷取此種類型的資訊而言，這並不是好方法。

如果要執行深度提取，建議您使用 Session Bean。在該 Session Bean 中，建立一個 Finder 方法以擷取所要的資訊。 Finder 方法應執行下列事項：

- 爲深度提取執行一個 SQL select 陳述式
- 針對主要表格中的每一列以及每一個相關物件，各實例化一個存取 Bean
- 針對每一個提取的直欄以及每一個提取的相關物件，在存取 Bean 中分別設定對應的屬性。

請注意，存取 Bean 不會快取擲出異常狀況的 getter 方法。在此情況下，您應使用下列型樣，爲存取 Bean 建立一個簡單的 wrapper 類別：

```
public class CustomerAccessBeanCopy extends CustomerAccessBean {
   private AddressAccessBean address=null;

   /* 下列方法是改寫 CustomerAccessBean 中的 getAddress 方法。
   */
   public AddressAccessBean getAddress() {
      if (address == null)
         address = super.getAddress();
      return address;
      }
```

```
    /* 下列方法是將地址設定到副本中。*/

    public void _setAddress(AddressAccessBean aBean) {
       address = aBean;
      }
}
```

延續 CUSTOMER 與 ADDRESS 範例， Session Bean Finder 方法會針對 CUSTOMER 表格中的每一列各實例化一個 CustomerAccessBean，以及針對 ADDRESS 表格中的每一個對應列各實例化一個 AddressAccessBean。接著，針對 ADDRESS 表格中的每一個直欄，在 AddressAccessBean 中設定屬性（路名與縣市）。針對 ADDRESS 表格中的每一個直欄，在 CustomerAccessBean 中設定屬性（姓名、年齡與地址）。請見下圖：

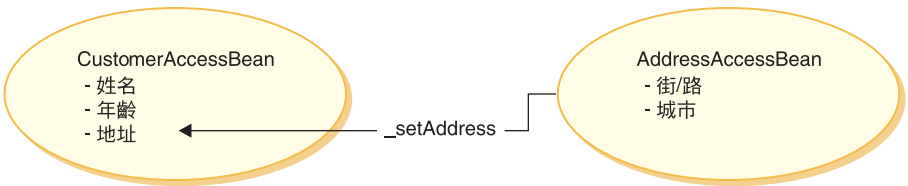

圖 16.

任意 SQL 情況: 在此情況中，會有一組任意 SQL 陳述式執行資料庫密集作業。舉例來說，總和表格中的所有列即可視爲資料庫密集作業。但可能所選之列並非全都對應至持續性模型中的某個 Entity Bean。

當客戶試著瀏覽一組大量資料時，便可能會建立任意 SQL 陳述式。舉例來說，如果客戶想查看線上五金商店中的所有膠帶，或者想查看線上服飾商店中的所有洋裝。這會產生相當龐大的結果集，但從這個結果集來看，很可能只需要每一列中的少數欄位。也就是說，一開始可能只會呈現顯示項目名稱、圖片與價格的摘要給客戶。

在此情況中，可建立一個 Session Bean helper 方法。此 Session Bean helper 方法會執行讀取或寫入作業。當執行讀取作業時，它會傳回一個顯示用的唯讀值物件。

藉由建立適當的資料模型，通常可將任意 SQL 陳述式的情況減至最少。

延伸公用 Entity Bean

本節說明 WebSphere Commerce 公用 Entity Bean 的設計型樣。此設計型樣可讓您進行延伸，像是加入新的持續性欄位、新商業方法或新 Finder 方法。

下圖顯示 Catalog（型錄）Entity Bean 的實作類別。

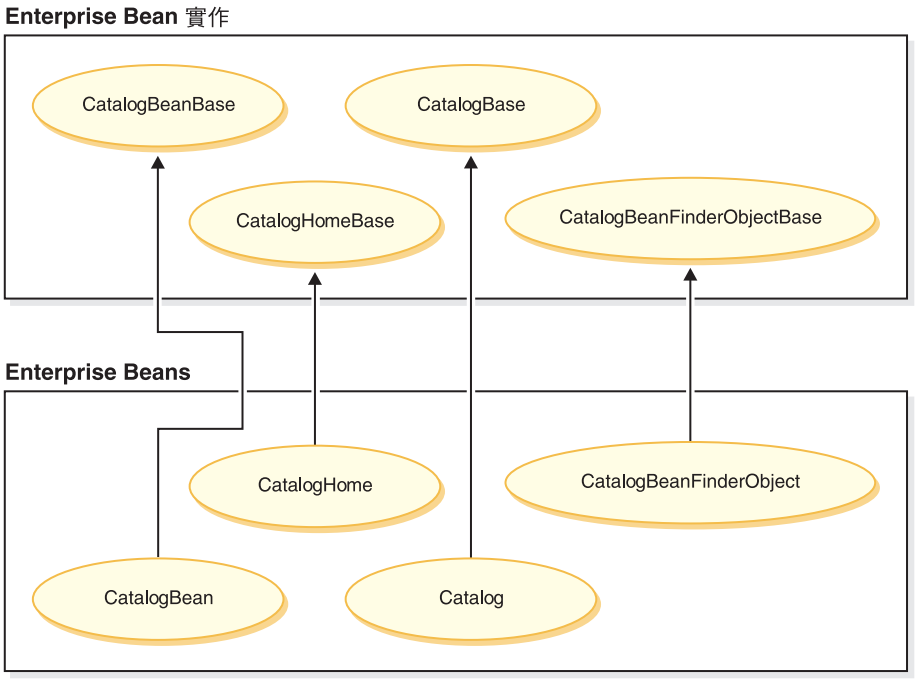

圖 17.

上圖亦適用於其他 Entity Bean，因爲這些 Entity Bean 是以類似方式建構而成，並遵循相同的命名慣例。如果要將此圖套用在其他 Entity Bean 上，請更換 Entity Bean 名稱 “Catalog”。例如， InterestItemBean 類別會延伸 InterestItemBeanBase 類別，而 InterestItem 介面會延伸 InterestItemBase 介面。

在本圖中，使用 Java 繼承特性將公用 Enterprise Bean 的實作類別或介面分成兩部份。超類別或介面中含有 WebSphere Commerce 實作程式碼。所有這些超類別與介面都是在與子項類別和介面不同的 Java 套件中定義。

WebSphere Commerce 工作區包含所有這些超類別與介面的二進位程式碼。您可以修改子類別與介面。通常，修改是在 com.ibm.commerce.*xxx*.objects 以及 com.ibm.commerce.*xxx*.objsrc 套件中進行（其中 *xxx* 是元件名稱）。

如果您在公用 Enterprise Bean 中加入新的 Finder 方法，您必須遵循該方法的特定命名慣例。請爲新方法 `findX`*`a_description`* 命名，其中 *`a_description`* 爲您自選的說明。舉例來說，名稱可以是 `findXByOwnerId` 與 `findXByOrderStatus`。使用此命名慣例可避免與 WebSphere Commerce Finder 方法發生名稱衝突（名稱重複）情況。在新增搜尋器時，會使用部署描述子編輯程式。

修改現有 WebSphere Commerce 公用 Entity Bean 的一個方法是新增其他欄位。在此情況下，在您加入新欄位後，您必須檢查 Bean 中的每一個 Finder 方法。如果 Finder 方法中的 `where` 子句部份含有任何資料庫別名（例如 T1. 或 T2.），則必須移除別名。

包含 "findForUpdate" 類型搜尋器的公用 Entity Bean: 如果 WebSphere Commerce 公用 Entity Bean 包含任何 "findForUpdate" 類型的搜尋器，您就無法藉由建立次要對映到您已經建立的新表格中，來新增欄位到 Bean 中。這是因爲如果您建立次要對映，則產生的 SQL 陳述式會變成無效，而 Bean 就不再會如預期般運作。如果您要延伸此類 Bean 所代表的物件模型的一部分，就必須建立一個新的 Entity Bean，然後在您的自訂程式碼中一起使用原始的 Bean 和新的 Bean。

建立新 CMP Enterprise Bean

當您具有一個必須新增到 WebSphere Commerce 物件模型中的新屬性後，您可以建立一個新資料庫表格，並內含一個必要屬性的直欄。您也必須將此屬性置於 Enterprise Bean 中，以便讓 WebSphere Commerce 指令可存取資訊。

將新屬性整合到 WebSphere Commerce 物件模型的方法之一是建立一個新 CMP Enterprise Bean。在這個 Bean 中，您將建立一個欄位來對應到新資料庫表格中的屬性。

由於 WebSphere Commerce 工作區會爲您的新 Enterprise Bean 提供預先定義的 EJB 專案，您並不需要建立額外的 EJB 專案。您新的 Enterprise Bean 應該放置到 WebSphereCommerceServerExtensionsData EJB 專案中。稍後，當您部署自訂 Bean 時，您可以建立一個 WebSphereCommerceServerExtensionsData.jar JAR 檔，並將現有的 JAR 檔放置到在 WebSphere Application Server 中執行的 WebSphere Commerce 企業應用程式中。透過這個包裝慣例，可以大幅簡化部署作業。

如果要建立新 CMP Enterprise Bean，您必須在 WebSphere Studio Application Developer 中執行下列步驟：

1. 使用「Enterprise Bean 建立」精靈來建立新的 CMP Enterprise Bean。針對對應資料庫表格中的每一個直欄，在 Bean 中分別新增一個新 CMP 欄位。
2. 設定新的 Bean 的交易隔離層次。
3. 設定新的 Bean 的安全身份。
4. 修改實體環境定義方法。
5. 必要時，使用 EJB 部署描述子編輯程式來定義新的搜尋器。
6. 必要時可建立新 ejbCreate 方法，並將 ejbCreate 方法引介到 Enterprise Bean 的發源介面中。如果新 Enterprise Bean 必須在其對應的資料庫表格中建立新項目，則必須執行此步驟。

7. 如果 Bean 是由 WebSphere Commerce 存取控制系統所保護，請實作 Bean 中所需要的存取控制方法。請參閱第 85 頁的第 4 章, 『存取控制』，以取得在 Enterprise Bean 中實作存取控制的其他明細。您也可以選擇在建立存取 Bean 之後才實作存取控制。
8. 將 Enterprise Bean 中的欄位對映至資料庫表格中的直欄。
9. 爲 Enterprise Bean 產生對應的存取 Bean。
10. 爲 Enterprise Bean 產生已部署的程式碼。
11. 使用 WebSphere Studio Application Developer 中的「通用測試用戶端」來測試 Bean。

有關上述每一個步驟的詳述，請參閱下列章節。在閱讀這些章節時，假設您有一個新表格 XUSERRES，其中指定使用者住家類型的部分相關資訊。此表格含有三個直欄： USERID 直欄、HOME 直欄（指定住宅類型）與 ROOMS 直欄（指定住家中的房間數）。

建立新 CMP Enterprise Bean: 如果要建立新的 CMP Enterprise Bean，您可以使用「Enterprise Bean 建立」精靈，方式如下：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 模組並選取**新建 >**（**其他 >**）**Enterprise Bean**。
   這時會開啓「Enterprise Bean 建立」精靈。
3. 從 **EJB 專案**下拉清單中，選取 **WebSphereCommerceServerExtensionsData**，然後按一下**下一步**。
4. 在「建立 Enterprise Bean」視窗中，執行下列步驟：
   a. 選取**具有儲存器所管理的持續 (CMP) 欄位的 Entity Bean**
   b. 在 **Bean 名稱**欄位中，爲您的 Bean 輸入一個適當的名稱。通常， Bean 名稱會與對應的資料庫表格的名稱相符。例如，您可以將 Bean 命名爲 XUserRes，以便對應到 XUSERRES 表格。
   c. 在**來源資料夾**欄位中，保留已指定的預設值 (ejbModule)。
   d. 在**預設套件**欄位中，輸入 `com.mycompany.mycomponent.objects`。
   e. 按一下**下一步**。
5. 在「Enterprise Bean 明細」視窗中，執行下列步驟：
   a. 按一下**新增**來爲資料庫表格中的直欄加入新的 CMP 屬性。
   這時會開啓「建立 CMP 屬性」視窗。請在此視窗中執行下列步驟：

1) 在**名稱**欄位中，爲新的 CMP 欄位輸入一個適當的名稱。請注意，當您稍後在將這個欄位對映到欄位在資料庫表格中的對應直欄時，如果要使用「比對名稱」功能，請將您的欄位命名爲直欄的名稱（不區分大小寫）。

2) 在**類型**欄位中，請爲欄位輸入一個適當的資料類型。請注意，您應該爲原生資料類型使用外層類別（例如，使用 *java.lang.Long* 資料類型，而不是 *long* 資料類型）。

3) 如果欄位是主要鍵，請選取**鍵值欄位**勾選框，然後按一下**套用**。

4) 如果欄位不是主要鍵，請選取**使用 getter 與 setter 方法來存取**勾選框。

5) 如果欄位不是主要鍵，請清除**將 getter 與 setter 方法提升至遠端介面**勾選框。**使 getter 成為唯讀**勾選框將會無法使用，然後按一下**套用**。

6) 再按一下**新增**並重覆這些步驟，爲資料庫表格中每一個需要 CMP 欄位的直欄加入新的欄位。

7) 按一下**關閉**來關閉視窗。

b. 清除**在鍵值類別中使用單一鍵值屬性類型**勾選框，然後按一下**下一步**。

6. 在「EJB Java 類別明細」視窗中，執行下列步驟：

a. 如果要選取 Bean 的超類別，請按一下**瀏覽**。
這時會開啓「類型選項」視窗。

b. 在**選取類別的方法：（任何）**欄位中，輸入 `ECEntityBean`，然後按一下**確定**。這樣就會選取 com.ibm.commerce.base.objects.ECEntityBean 作爲超類別。

c. 如果新的 Enterprise Bean 要由 WebSphere Commerce 存取控制組織架構來保護，請按一下**新增**來指定遠端介面應該延伸的介面。這時會開啓「類型選項」視窗。

d. 在**選取類別的方法：（任何）**欄位中，輸入 `Protectable`，然後按一下**確定**。這樣就會選取 com.ibm.commerce.security.Protectable。您需要這個介面才能使新的資源受到存取控制的保護。

e. 按下**完成**。

在 Bean 類別中，WebSphere Studio Application Developer 會建立專用欄位 `EntityContext`。WebSphere Commerce 會在 ECEntityBean 中提供本身的實體環境定義欄位，而您的新 Entity Bean 應使用該欄位，而不是產生的欄位。因此，您應移除新 Entity Bean 中所產生的 `EntityContext`。下一節中將說明這一點。

當您指定 `com.ibm.commerce.base.objects.ECEntityBean` 作爲超類別時，您的 Bean 會繼承特定的功能。以下是示範這些功能的程式碼範例：

```
public class myEJB extends com.ibm.commerce.base.objects.ECEntityBean {
   public void ejbLoad() {
      super.ejbLoad();--the super method will add EJB trace
       --your logic --
   }
   public void ejbStore() {
      super.ejbStore();--the super method will add EJB trace
      --your logic --
   }
}
```

設定交易隔離層次: 您必須將 Bean 的交易隔離層次設定成您的開發資料庫類型適用的正確值。如果要設定交易隔離層次，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 按兩下 **WebSphereCommerceServerExtensionsData** 專案，以使用「部署描述子編輯程式」來開啓它。
3. 按一下**存取**標籤。
4. 按一下「隔離層次」文字框旁邊的**新增**。
   這時會開啓「新增隔離層次」視窗。
5. DB2 選取**可重複讀取**，然後按一下**下一步**。
   Oracle 選取**讀取已確定**，然後按一下**下一步**
6. 從**找到的 Bean** 清單中，選取 *yourNewBean* Bean，然後按一下**下一步**。
7. 從**找到的方法**清單中，選取 *yourNewBean* 來選取其所有的方法，然後按一下**完成**。
8. 儲存您的工作 (Ctrl + S)，並保持編輯程式開啓。

設定 Bean 的安全身份: 接下來您需要設定 Bean 的安全身份，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，確定您已經選取「存取」標籤。
2. 按一下「安全身份」文字框旁邊的**新增**。
   這時會開啓「新增安全身份」視窗。
3. 選取**使用 EJB 伺服器的身份**，然後按一下**下一步**。
4. 從**找到的 Bean** 清單中，選取 *yourNewBean* Bean，然後按一下**下一步**。
5. 從**找到的方法**清單中，選取 *yourNewBean* 來選取其所有的方法，然後按一下**完成**。
6. 儲存您的工作 (Ctrl+S)。讓編輯程式維持開啓狀態。

設定 Bean 的安全職務: 接下來，您需要設定 Bean 中的方法的安全職務，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，選取「組譯描述子」標籤。
2. 在「方法許可權」區段中，按一下**新增**。
3. 選取 **WCSecurityRole** 作爲安全職務，然後按一下**下一步**。
4. 從找到的 Bean 清單中，選取 *yourNewBean*，然後按一下**下一步**。
5. 在「方法元素」頁面中，按一下**全部套用**，然後按一下**完成**。
6. 儲存您的工作 (Ctrl+S)，然後關閉「部署描述子」編輯程式。

刪除實體環境定義欄位及方法: 下一步是移除 WebSphere Studio Application Developer 產生的某些與實體環境定義相關的欄位和方法。這些欄位需要刪除是因爲 ECEntityBean 基本類別已提供自己對這些方法的實作。如果要刪除產生的實體環境定義欄位與方法，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **WebSphereCommerceServerExtensionsData** 專案。
2. 展開 *yourNewBean* Bean，然後按兩下 *yourNewBean* **Bean** 類別。
3. 在「大綱」檢視畫面中，執行下列步驟：
   a. 以滑鼠右鍵按一下 **myEntityCtx** 欄位並選取**刪除**。
   b. 以滑鼠右鍵按一下 **getEntityContext()** 方法並選取**刪除**。
   c. 以滑鼠右鍵按一下 **setEntityContext(EntityContext)** 方法並選取**刪除**。
   d. 以滑鼠右鍵按一下 **unsetEntityContext()** 方法並選取**刪除**。
4. 儲存您的工作 (Ctrl+S)。

加入新的搜尋器:

使用搜尋器的簡介: 搜尋器輔助程式的介面用法已經不合時宜。在任何新的開發工作中，您必須使用 EJB 部署描述子編輯程式來定義您的查詢和方法宣告，而不能再使用搜尋器輔助程式介面。

有關使用 EJB 查詢語言來建立搜尋器的詳細資訊、這個方法相較於其他搜尋器方法的優點，以及相關工具的其他明細，請參閱 WebSphere Studio Application Developer 線上說明。

加入新的搜尋器到新的 Bean 中: 如果您需要加入新的搜尋器到您的 Enterprise Bean 中，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 按兩下 **WebSphereCommerceServerExtensionsData** 專案來開啓「 EJB 部署描述子編輯程式」。
3. 按一下 **Bean** 標籤。

4. 在 Bean 窗格中，選取 *yourNewBean* Bean，然後在右邊的窗格中，向下捲動並展開 **WebSphere 延伸**。
5. 按一下**搜尋器**文字框旁邊的**新增**。
   這時會開啓「新增搜尋器描述子」視窗。
6. 選取**新建**，然後在**名稱**欄位中，輸入 `findByX`*`yourArg`*（其中 *yourArg* 是您要用來進行搜尋的引數名稱）。請使用 “findXBy” 命名慣例來命名您的欄位名稱，以確定您的欄位名稱在 WebSphere Commerce 欄位名稱中恆爲唯一。
7. 按一下**參數**文字框旁邊的**新增**，然後執行下列步驟
   a. 在**名稱**欄位中，輸入 *yourArg*。
   b. 在**類型**欄位中，輸入適當的資料類型。
   c. 按一下**確定**。
8. 在**傳回類型**欄位中，輸入下列其中一項，然後按一下**下一步**：
   - 如果 FinderHelper 方法使用主要鍵來查詢資料庫，且方法應傳回唯一記錄，請指定 EJB 物件作爲傳回類型。舉例來說，輸入 `UserRes`。
   - 如果 FinderHelper 方法所傳回的是結果集，而非唯一記錄，請將傳回類型指定爲 `java.util.Enumeration`。
9. 從**搜尋器類型**下拉清單中，選取 **WhereClauseFinderDescriptor**。
10. 在**搜尋器陳述式**欄位中，輸入一個適當的搜尋器。例如，輸入 `T1.MEMBERID = ?`，然後按一下**完成**。
11. 儲存您的工作，然後關閉「EJB 部署描述子」編輯程式。

基於安全理由，在爲新 Entity Bean 建立 FinderHelper 方法時，您應該使用前面的步驟中所顯示的插入參數方式。建議的理由是因它可防止使用者改變查詢。另一種方法是使用類似如下的建構：

```
T1.MEMBERID = "input_string ";
```

其中 *input_string* 爲 URL 傳來的字串值。此種方法並不理想，這是因爲心懷不軌的使用者可輸入像是 “‘123’ OR 1=1” 等會變更 SQL 陳述式的值。如果使用者能夠變更 SQL 陳述式，即有可能可擅自存取資料。因此，建議您使用插入參數方式。

如果您無法使用插入參數方式，因而得使用輸入字串來撰寫 SQL 陳述式，您必須對輸入字串進行參數檢查，以確保輸入參數沒有機會讓心懷不軌的使用者存取資料。

建立新 ejbCreate 方法: 在您建立 Enterprise Bean 時，會自動建立 ejbCreate 方法。此方法會引介到遠端介面中，以便能在存取 Bean 中使用。預設 ejbCreate 方法中只含有屬於主要鍵或主要鍵一部份的參數。也就是說，在實例化期間只有這些值才會實例化。

如果您的 Enterprise Bean 中含有不屬於主要鍵一部份的欄位，且這些欄位不可空值，您必須建立一個新 ejbCreate 方法，以便明確實例化這些欄位。在此情況下，每當建立新記錄時，才會在所有不可空值的欄位中移入適當資料。

如果要建立新 ejbCreate 方法，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，按兩下 ***yourNewBean*Bean** 類別來開啓它，並檢視其原始碼。
2. 您必須修改原始碼，以便將每一個不可爲空值的 CMP 欄位併入作爲方法的輸入參數，以使用適當的值將每一個 CMP 欄位實例化。在 UserRes 範例中（其中 UserId 爲主要鍵），一開始的原始碼爲：

   ```
   public void ejbCreate(int argUserId)
      throws javax.ejb.CreateException {
      _initLinks();
      userId = argUserId;
   }
   ```

   但您可能想確定房間數與住宅類型皆已起始設定。在此情況下，您將程式碼改爲：

   ```
   public void ejbCreate(int argUserId, String argHome, byte Rooms)
      throws javax.ejb.CreateException {
      _initLinks();
      // 此處所有的 CMP 欄位應已起始設定
      userId = argUserId;
      home = argHome;
      rooms = argRooms;
   }
   ```

   註: 如果您想使用系統產生的主要鍵，請參閱第 75 頁的『主要鍵』，取得詳細資訊。
3. 您必須將新的 ejbCreate 方法加到發源介面。這樣就能在產生的存取 Bean 中提供該方法。如果要新增方法到發源介面中，請執行下列步驟：
   a. 以滑鼠右鍵按一下「大綱」檢視畫面中的 **ejbCreate(***yourParameters***)** 方法，然後選取 **Enterprise Bean > 提升至發源介面**。

建立新的 ejbPostCreate 方法: 接下來，您必須建立一個新的 ejbPostCreate 方法，且其輸入參數與新的 ejbCreate 方法相同。如果要建立這個新方法，請執行下列步驟：

1. 按兩下 ***yourNewBean*Bean** 類別來開啓它，並檢視其原始碼。

2. 建立一個新的 ejbPostCreate 方法，且其輸入參數與新的 ejbCreate 方法中所用的輸入參數相同。如果要繼續以使用者居住地的範例來執行，就應該將下列程式碼寫入到類別中：

```
public void ejbPostCreate(int argUserId,
   String argHome, byte Rooms)
   {
   }
```

   儲存程式碼變更。

新增存取控制方法到 Bean 中: 如果新的 Bean 是在存取控制的保護下，您必須新增 getOwner 方法。另一個用於存取控制的選用方法是 fulfills 方法。關於必要及選用方法的明細，請參閱第 100 頁的『在 Enterprise Bean 中實作存取控制』。如果要將這些存取控制方法加入到新的 Bean 中，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，按兩下 ***yourNewBean*Bean** 類別來開啓它，並檢視其原始碼。
2. 在原始碼中，加入一個 getOwner 方法以併入用來傳回此資源擁有者的邏輯。例如，假設要傳回 UserRes Bean 的擁有者，您應該傳回使用者的成員 ID：

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. 在這個範例中，您會新增一個 fulfills 方法來指定在容許使用者對這個資源執行動作前，使用者必須符合的關係。在此情況下，您會指定只有這個 UserRes 物件的建立者可以採取動作。換言之，每一個使用者只能對本身的 UserRes 物件採取動作。下列程式碼片段顯示出這個關係基本需求：

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      return false;
   }
```

4. 儲存您的工作。

將資料庫表格對映至新 Enterprise Bean: 一旦您建立新 Enterprise Bean 後，您必須在 Bean 中的 CMP 欄位和資料庫表格中的直欄間建立對映。當 Enterprise Bean 及其對應資料庫表格都存在時，就會使用「上下同時進行」類型的對映。WebSphere Studio Application Developer 提供了許多工具來簡化這項作業。

如果要建立對映，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**產生 > EJB 至 RDB 對映**。
   這時會開啓「EJB 至 RDB 對映」視窗。
2. 選取**上下同時進行**，然後按一下**下一步**。
3. 在「資料庫連線」視窗中，執行下列步驟：
   a. 在**連線名稱**欄位中，輸入 `WebSphereCommerceServerExtensionsData`
   b. 在**資料庫**欄位中，輸入 *developmentDB*
   c. 在**使用者 ID** 欄位中，輸入 *dbuser*
   d. 在**密碼**欄位中，輸入 *dbpassword*
   e. 從資料庫下拉清單中，選取您的開發資料庫的資料庫供應商類型。
      - DB2 DB2 Universal Database 8.1
      - Oracle Oracle 9i
   f. Oracle 在**主機**欄位中，輸入您的資料庫伺服器的完整主機名稱。例如，輸入 `dbserver.yourcompany.com`
   g. Oracle 在「類別位置」欄位中，輸入 classes12.zip 檔的位置。例如，輸入 `D:\oracle\ora92\jdbc\lib\classes12.zip`
   h. 按一下**下一步**。一旦建立連線，就會顯示資料庫的表格清單。您也可以在稍後察看「資料」視景中的「資料庫伺服器」，來檢視「連線文件」。
4. 選取 ***yourNewTable*** 表格，然後按一下**下一步**。
5. 選取**比對名稱和類型**，然後按一下**完成**。這時會開啓「對映編輯程式」。
6. Oracle 如果有任何直欄的資料類型爲 “NUMBER”，您就必須修改資料類型。以滑鼠右鍵按一下 ***yourNewTable*** 表格，然後選取**開啓表格編輯程式**。在表格編輯程式中，執行下列步驟：
   a. 選取**直欄**標籤。
   b. 選取需要變更資料類型的直欄，然後將直欄類型從 NUMBER 變更爲更特定的類型。例如，將它變更爲 INTEGER。
   c. 儲存變更。
7. 在 Enterprise Bean 窗格中，展開 ***yourNewBean*** Bean。在「表格」窗格中，展開 ***yourNewTable*** 表格。
8. 將 *yourNewBean* Bean 中的欄位對映到 *yourNewTable* 表格中的直欄，步驟如下：
   a. 以滑鼠右鍵按一下 ***yourNewBean*** Bean 並選取**比對名稱**。

9. 儲存對 Map.mapxmi 檔所做的變更，然後關閉檔案。
10. Oracle 您必須使用文字編輯程式來編輯表格定義，方式如下：
    a. 在文字編輯程式中開啓 *yourNewBean*.xmi 檔。
    b. 將所有出現的 SQLNumeric6 換成 SQLNumeric3。
    c. 儲存您的變更，然後關閉檔案。

修改綱目名稱: 下一步是修改綱目名稱，使您的 Bean 可以轉移到其他資料庫。讓 Bean 可以用這種方式轉移的特殊值是 NULLID。如果要修改綱目名稱，請執行下列步驟：

1. 在「J2EE 視景」中，切換到「J2EE 階層」檢視畫面。
2. 展開**資料庫**，然後**展開 WebSphereCommerceServerExtensionsData**。
3. 以滑鼠右鍵按一下綱目節點（例如，db2user），然後選取**更名**。
4. 將值設定爲 NULLID。

建立存取 Bean: 存取 Bean 形同 Enterprise Bean 的外層，可簡化其他元件和 Enterprise Bean 間的互動方式。您必須爲您的新 Enterprise Bean 建立一個存取 Bean。 WebSphere Studio Application Developer 工具可讓您根據您所建立的實體，來產生此存取 Bean （尤其是存取 Bean 只能使用已引介到遠端介面中的方法）。

如果要建立存取 Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**新建 > 存取 Bean**。
   這時會開啓「新增存取 Bean」視窗。
2. 選取**複製輔助程式**，然後按一下**下一步**。
3. 選取 ***yourNewBean*** Bean，然後按一下**下一步**。
4. 從「建構子方法」下拉清單中，選取 **findByPrimaryKey(***yourPackageName*。*yourNewBean***Key)** 作爲建構子方法。
5. 選取「屬性輔助程式」區段中的所有屬性。
6. 按下**完成**。
7. 儲存您的工作。

產生部署程式碼: 程式碼產生公用程式會分析 Bean，以確定是否符合 Sun Microsystem 的 EJB 規格，並確定是否遵循 EJB 伺服器的特定規則。此外，在每一個所選的 Enterprise Bean 方面，程式碼產生工具會爲發源與遠端介面產生發源與 EJBObject（遠端）實作方式與實作類別，以及爲 CMP Bean 產生 JDBC persister

與 finder 類別。它同時會產生透過 IIOP 進行 RMI 存取時所需要的 Java ORB、存根與 Tie 類別，以及發源和遠端介面的存根。

如果要產生已部署的程式碼，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，再選取**產生 > 部署和 RMIC 程式碼**。
   這時會開啓「部署和 RMIC 程式碼」視窗。
2. 選取 ***yourNewBean*** Bean，然後按一下**完成**。

您可以切換到「J2EE 導覽器」檢視畫面，來檢視新產生的程式碼。您會發現下列內容：

表 1.

程式碼類型	類別名稱
配置區實作產生的程式碼	EJSCMP*yourNewBean*HomeBean.java
	EJSRemoteCMP*yourNewBean*.java
	EJSRemoteCMP*yourNewBean*Home.java
	EJSFinder*yourNewBean*Bean.java
JDBC 存取程式碼	EJSJDBCPersisterCMP*yourNewBean*Bean.java
RMI tie 和存根程式碼	_EJSRemoteCMP*yourNewBean*_Tie.java
	*_yourNewBean*_Stub.java
	_EJSRemoteCMP*yourNewBean*Home_Tie.java
	*_yourNewBean*Home_Stub.java

使用測試用戶端來測試 Enterprise Bean: WebSphere Studio Application Developer 提供一個測試用戶端，可用來測試 Enterprise Bean。如果要使用測試用戶端來測試您的新 Bean，請執行下列步驟：

1. 切換到「伺服器」視景。
2. 按兩下 **WebSphereCommerceServer** 伺服器，然後按一下**配置**標籤。
3. 選取**啓用通用測試用戶端**。儲存變更。
4. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**。
5. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**在伺服器上執行**。
   這時會將 Web 瀏覽器開啓在通用測試用戶端。測試 Enterprise Bean 的起點是 JNDI Explorer，您可以在這裡找到在這部伺服器的 EJB 配置區中執行的 Bean 的名稱。

6. 以滑鼠右鍵按一下 **JNDI Explorer**
7. 接下來您必須導覽至 *yourNewBean*Home 介面，方法是依照下面的方式來展開階層：**cell > nodes > localhost > servers > server1 > ejb >** *yourPackageStructure*。
8. 按一下 ***yourNewBean* 發源**介面。在「參照」窗格中，選取 **EJB 參照 > *yourNewBean* > *yourNewBean*Home**。按一下建立方法。
9. 在右窗格中，於建立方法的必要輸入參數相對應的欄位中，輸入適當的值。
10. 按一下**呼叫**，然後結果就會顯示在底端的窗格中。
11. 按一下**處理物件**來將遠端介面新增至「參照」窗格，然後您就會在「物件參照」下面看到您輸入的值。這時您的新表格中已經建立一筆新紀錄。
12. 使用適當的方式來測試其他的方法。
13. 關閉測試用戶端並停止伺服器。

程式碼撰寫慣例: 請注意如下的 Enterprise Bean 程式碼撰寫慣例：

- 請勿使用 BLOB 或 CLOB 資料類型。
- Enterprise Bean 程式碼不應該參照 Enterprise Bean 模組以外的任何東西。舉例來說，您不應參照 Enterprise Bean 程式碼中的指令或資料 Bean。
- 前面的幾節說明如何在一開始建立 Bean 的時候，在新的 Bean 中併入存取控制。您也可以在建立 Bean 以後才加入它，方法是新增 com.ibm.commerce.security.Protectable 介面。必要時，您也可以將 com.ibm.commerce.security.Groupable 介面新增至 Enterprise Bean 的遠端介面。您也必須在 Bean 中實作特定的方法。新增這些介面以及必要的方法後，請重新產生 Bean 的部署程式碼及存取 Bean。其他資訊請參閱第 100 頁的『在 Enterprise Bean 中實作存取控制』。

建立簡單的資料 Bean

資料 Bean 是一種在 JSP 範本中用來擷取 Enterprise Bean 中資訊的 Bean。簡單的資料 Bean 延伸了其對應的存取 Bean，並實作 SmartDataBean 介面。WebSphere Studio Application Developer 會爲資料 Bean 自動產生大部份的程式碼。

新的資料 Bean 是儲存在 WebSphereCommerceServerExtensionsLogic 專案中。

如果要建立簡單的資料 Bean，您必須執行下列步驟：

1. 建立一個套件來儲存資料 Bean 程式碼。
2. 建立一個延伸了對應的存取 Bean 並實作適當資料 Bean 介面的資料 Bean。
3. 爲資料 Bean 建立 set 方法。
4. 爲資料 Bean 建立 get 方法。

後續各節將詳細說明每一個步驟。

為資料 Bean 程式碼建立套件: 建立套件時，會建立一個空間來儲存您的資料 Bean 程式碼。

如果要建立新套件，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到 Java 視景。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，選取**新建 > 套件**。這時會開啓「新建 Java 套件」精靈。
3. **來源資料夾**的值會預先移入成 `WebSphereCommerceServerExtensionsLogic/src` 這個值。請保留這個值。
4. 在**名稱**欄位中，爲您的新套件輸入一個適當名稱。舉例來說，輸入 `com.mycompany.mydatabeans`。
5. 按一下**完成**。

建立資料 Bean: 資料 Bean 是一種 Java Bean，可在 JSP 範本中提供動態內容給頁面。通常它會藉由延伸存取 Bean 提供 Entity Bean 的簡單呈現（間接）。資料 Bean 概括了一些可從 Entity Bean 中擷取或設於 Entity Bean 中的內容。

如果要建立資料 Bean，請執行下列步驟：

1. 用滑鼠右鍵按一下將用來儲存資料 Bean 的套件，然後選取**新建 > 類別**。這時會開啓「新建 Java 類別」精靈。
2. 專案與套件名稱欄位皆已移入資料。
3. 在**名稱**欄位中，輸入新資料 Bean 的名稱。舉例來說，如果要建立一個延伸 UserResAccessBean 的資料 Bean，請輸入 `UserResDataBean`。
4. 從**修飾元**清單中，選取 **Public**。
5. 如果要指定超類別，請按一下**瀏覽**，然後在型樣欄位中輸入對應存取 Bean 的名稱。舉例來說，輸入 `UserResAccessBean`，然後按一下**確定**。
6. 如果要指定資料 Bean 應實作的介面，請按一下**新增**。在「介面」視窗中，執行下列步驟：
   a. 在**型樣**欄位中，輸入 `com.ibm.commerce.beans.SmartDataBean`，然後按一下**新增**。
   b. 在**型樣**欄位中，輸入 `com.ibm.commerce.beans.InputDataBean`，然後按一下**新增**。
   c. 按一下**確定**。
7. 按一下**完成**。

新增必要欄位到資料 Bean 中: 本節說明如何修改您的新資料 Bean 中的必要欄位。其中有兩個必要的欄位是針對下列資訊類型：

- 指令環境定義
- 要求內容

如果要修改 iCommandContext 欄位，請執行下列步驟：

1. 按兩下新的資料 Bean（例如，UserResDataBean），來檢視其原始碼。
2. 尋找 getCommandContext 方法。其內容如下：

   ```
   public CommandContext getCommandContext() {
         return null;
   ```

   將下列程式碼新增至原始碼中：

   ```
   private CommandContext iCommandContext = null;
    public com.ibm.commerce.command.CommandContext getCommandContext()
   {
    return iCommandContext;
   }
   ```

3. 修改 setCommandContext 方法。這個方法剛開始是顯示成下面的樣子：

   ```
   public void setCommandContext(CommandContext arg0) {
    }
   ```

   修改程式碼，使它變成下面的樣子：

   ```
   public void setCommandContext(com.ibm.commerce.command.CommandContext
      aCommandContext)
   {
    iCommandContext = aCommandContext;
   }
   ```

4. 修改 getCommandContext 方法。這個方法剛開始是顯示成下面的樣子

   ```
   public CommandContext getCommandContext() {
         return null;
    }
   ```

   依下列所示修改原始碼：

   ```
   public com.ibm.commerce.command.CommandContext getCommandContext ()
   {
      return iCommandContext;
   }
   ```

5. 儲存您的工作。

如果要修改 iRequestProperties 欄位，請執行下列步驟：

1. 按兩下新的資料 Bean（例如，UserResDataBean），來檢視其原始碼。
2. 尋找 getRequestProperties 方法。這個方法剛開始是顯示成下面的樣子：

```
public TypedProperty getRequestProperties() {
      return null;
 }
```

依下列所示修改原始碼：

```
private com.ibm.commerce.datatype.TypedProperty
   requestProperties;

public TypedProperty getRequestProperties() {
 return requestProperties;
}
```

3. 修改 setRequestProperties 方法。這個方法剛開始是顯示成下面的樣子：

```
public void setRequestProperties(TypedProperty arg0) throws Exception {
 }
```

依下列所示修改原始碼：

```
      public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
   aParam)
   throws Exception {
   // 將輸入 TypedProperteis 複製到本端

  requestProperties = aParam;
}
```

4. 修改 getRequestProperties 方法。這個方法剛開始是顯示成下面的樣子：

```
public TypedProperty getRequestProperties() {
      return null;
 }
```

依下列所示修改原始碼：

```
public TypedProperty getRequestProperties() {
 return requestProperties;
}
```

5. 儲存您的工作。

將資料移入對應存取 Bean 的主要鍵： 請注意，您可能想修改原始碼，以便在對應存取 Bean 的主要鍵中移入資料。建議您使用資料 Bean 管理程式間接設定此值。此間接方法旨在確定取自 URL 內容的主要鍵值不會改寫主要鍵（如果先前有設定的話）。如果要讓您的 setRequestProperties 方法遵循此模型，請以類似如下程式碼片段的形式來撰寫程式。請注意，在下列範例中，主要鍵爲使用者 ID。這將視情況而異（因此下列程式碼在您應用程式中不見得能立即編譯）。

```
public void setRequestProperties(
 com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception {
      iRequestProperties = arg1;
try {
          if (// check for nulls
```

```
            getDataBeanKeyUserId() == null) {
          super.setInitKey_UserId(aUserId);
           }
    } catch (com.ibm.commerce.exception.ParameterNotFoundException e) {}
  }
```

您也可以採用下列兩種方式爲存取 Bean 設定主要鍵。您可在資料 Bean 外（例如：在 JSP 範本中）來執行此動作。在此情況下，在您於 JSP 範本中啓動資料 Bean 前，請明確針對主要鍵呼叫資料 Bean 的 set 方法。舉例來說， JSP 可含有類似如下的程式碼（其中 db 爲資料庫資料 Bean 物件）：

```
db.setInitKey_UserId(/*input parameter*/)
db.activate();
```

或者，您可直接設定主要鍵。亦即，讓 JSP 範本中只含有 db.activate 方法，而使用資料 Bean 管理程式明確在存取 Bean 中設定主要鍵。舉例來說，資料 Bean 之 setRequestProperties 方法的程式碼可類似如下：

```
public void setRequestProperties(
 com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception {
      iRequestProperties = arg1;
try
            {
          super.setInitKey_UserId(aUserId);
           }
    } catch (com.ibm.commerce.exception.ParameterNotFoundException e) {}
     }
```

請注意，在設定主要鍵時，建議的程序是使用間接的方法。

修改 populate() 方法: 您必須修改 populate 方法；其步驟如下：

1. 展開您的新資料 Bean，以檢視其欄位與方法。
2. 在「大綱」檢視畫面中，選取 **populate()** 方法來檢視其原始碼。
   原始碼剛開始的內容如下：

   ```
   public void populate () throws Exception {}
   ```

3. 修改原始碼，讓方法如下所示：

   ```
   try {
      super.refreshCopyHelper();
        } catch (javax.ejb.FinderException e) {
        throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        "UserResDataBean", "populate");
   ```

   儲存您的工作 (Ctrl+S)。

請注意，如果新的資料 Bean 所延伸的存取 Bean 在實例化時需要額外的輸入參數，您就必須同時在資料 Bean 的 populate 方法中設定這些值。

撰寫新 Session Bean

在建立新的 Session Bean 時，請在 WebSphereCommerceServerExtensionsData 專案中建立它們。

您的新 Session Bean 應爲 com.ibm.commerce.base.helpers.BaseJDBCHelper 類別的延伸。超類別會提供許多方法，讓您從 WebSphere Commerce Server 所用的資料來源物件中取得 JDBC 連線物件，以便讓 Session Bean 能參與和其它 Entity Bean 相同的交易。以下的程式碼範例是示範超類別所提供的函數：

```
public class mySessionBean extends com.ibm.commerce.base.helpers.BaseJDBCHelper
  implements SessionBean {

    public Object myMethod () throws javax.naming.NamingException,
      SQLException {

       ////////////////////////////////////////////////
      //  -- 起始設定的邏輯 --                        //
      ////////////////////////////////////////////////

try {
             // 從 WebSphere Commerce 資料來源取得連線
             makeConnection();
       PreparedStatement stmt = getPreparedStatement("your sql string");
             ////////////////////////////////////////////////////////////
             // -- 將參數設為已備妥陳述式的                              //
             // 相關邏輯 --                                              //
             ////////////////////////////////////////////////////////////
     ResultSet rs = executeQuery(stmt, false);

             ////////////////////////////////////////////////
             // -- 您處理結果集的邏輯 --                    //
            ////////////////////////////////////////////////

                }
       finally {
             // 將連線傳回給 WebSphere Commerce 資料來源
       closeConnection();
          }

        ////////////////////////////////////////////////
       // -- 您傳回結果的相關邏輯 ---                   //
       ////////////////////////////////////////////////

   }

}
```

在前述程式碼範例中，executeQuery 方法採用兩個輸入參數。第一個是已備妥的陳述式，第二個是一個和快取沖寫作業有關的 boolean 旗號。如果您需要在執行查詢前讓儲存器沖寫快取中現行交易的所有實體物件，請將此旗號設爲 true。如果您已更新某些實體物件，且您需要查詢以搜尋這些已更新的物件，則需要如此做。如果您將旗號設爲 false，則在交易結束前並不會將這些實體物件更新寫到資料庫中。

您應限制使用這種沖寫作業，除非眞的必要，在平常時請將旗號設爲 false。沖寫作業是一種資源密集作業。

物件生命週期

物件模型中的 Enterprise Bean 同時含有*獨立*與*相依*物件。獨立物件擁有自己的生命週期，而受呼叫該物件之商業邏輯的建立或移除要求直接控制。相依物件的生命週期則依附於另一個物件，即所謂的*擁有者物件*（可能亦爲相依物件，只是在更高的連結階層上另存在一個獨立物件）。當刪除擁有者物件時，亦會連帶刪除所有相依物件。實際的刪除是藉由連帶刪除資料庫中的規格所控制。

舉例來說，某個使用者物件會傳回通訊錄物件與訂單物件清單，假設您刪除該使用者物件，則亦會刪除其通訊錄物件（這是因爲該通訊錄屬於該使用者的）以及該通訊錄中的所有地址物件（因爲這些地址屬於該通訊錄中的）。不過，並不會刪除訂單物件，這是因爲訂單的擁有者爲商店物件，而非使用者物件。

在建立相依物件時會採用特定的設計型樣。相依物件的 create 方法必須提供其擁有者物件的參照；因此，擁有者物件必須存在，才能建立其相依物件。

交易

Enterprise JavaBeans 1.1 版的架構在實例狀態方面指定了三種替代的確定時間選項。在規格文件中是以選項 A、B 與 C 表示之。有關這些選項的完整詳述，請參閱 Sun Microystem 的 Enterprise JavaBean 1.1 版的規格文件。

雖然 WebSphere Application Server 會實作選項 A 與 C，但選項 A 會假設資料庫不共用。

在選項 C 中， Enterprise Bean 儲存器不會快取各交易之間的 “備妥” 實例。一旦交易完成，即會將實例傳回到可用實例儲存池中。WebSphere Commerce 採用選項 C，這是因爲資料庫是供多個 WebSphere Commerce 應用程式共用。在此種實作方式下，在每次交易開始時，儲存器即會載入 Entity Bean 的持續性資料，而在交易期間只會快取 Entity Bean。儲存器會啓動 Entity Bean 的多個實例，且會存取該實體的每一項交易各一個。而資料庫會執行交易的同步化。

每一個 Enterprise Bean 的交易屬性會設爲 TX_REQUIRED。由於 Web 控制程式在執行會存取 Enterprise Bean（透過其對應的存取 Bean）的指令之前會先啓動一項交易，因而在此交易的環境定義中將會呼叫 Enterprise Bean 的商業方法。

Entity Bean 的其他注意事項

尋找以進行更新

當多個應用程式同時存取資料庫中之同一列，以更新該列的情形稱爲*並行更新*。在某些情況下可容許進行並行更新，但在某些情況下應用程式則絕不希望發生並行更新現象。

如果資料庫更新指的是改寫，亦即新值與資料庫中的目前值沒有關係，則可容許進行並行更新。假設容許並行更新，當有多個應用程式試著更新資料庫中的同一列時，則最後一次嘗試會讓資料庫中發生更新。

如果資料庫更新取決於資料庫中的目前值而定，便不應使用並行更新方法。舉例來說，如果應用程式正在更新產品的庫存量，則一次應只讓一個應用程式更新庫存量。

影響是否容許使用並行更新方法的因素包括：資料庫鎖定以及 Enterprise Bean 的隔離層次。

爲了避免第二個應用程式同時更新同一列，第一個存取該列的應用程式必須使用 “find for update” 選項來提取列。當使用 “for update” 選項時，會對該列採取寫入*鎖定*（亦稱爲排外鎖定）。藉由對列採取寫入鎖定，任何試著使用 “find for update” 存取該列的應用程式都會遭到阻擋。

如果您的應用程式容許並行更新，它可單純提取資料而不鎖定列。

請想像 OrderProcess 情況，其中 UpdateInventory 需要尋找訂單中所含的所有產品，從而更新庫存量。由於其他許多訂單中可能亦含有相同產品，因此應使用 *find for update*，且應在交易範圍內儘早使用以降低死鎖的可能性。因此，可使用下列的虛擬程式碼來表示 UpdateInventory 演算法：

```
UpdateInventory
 find all the order items in the order
 for each order item
  fetch its inventory using "find for update"
  ...
```

在長時間執行的企業消費型商務情況中，由於訂單可能含有眾多項目，因此應儘早使用 find for update。其邏輯可能變成：

```
find for update the inventory of all the products in an order
for each product
 if (total quantity ordered for that product < inventory)
        deduct quantity from inventory
 else
        error
```

沖寫遠端方法

除非到了交易確定時間，否則 WebSphere Application Server 並不會將您對 Entity Bean 所作的變更寫到資料庫中，因此資料庫可能會與 Entity Bean 儲存器中所快取的資料暫時失去同步。

您可使用沖寫遠端方法（位於 `com.ibm.commerce.base.helpers.BaseJDBCHelper` 類別中）；此方法會寫入所有交易中所有已確定的變更（亦即，它會採用 Enterprise Bean 快取中的資訊）並更新資料庫。您可透過指令來呼叫此遠端方法。請在有絕對必要的情況下才使用此方法，這是因爲此方法相當耗費資源，而會對效能造成負面影響。

假設某登入指令中含有下列的程式碼：

```
UserAccessBean uab = ...;
 uab.setRegisteredTimestamp(currentTimestamp);
 uab.commitCopyHelper();
```

在確定交易前，並不會以現行的時間戳記來更新 USERS 表格中的 REGISTRATIONUPDATE。只有在交易確定期間才會進行更新。必須使用沖寫方法，以便讓同一交易中的任何直接 JDBC 查詢（如 *select from user where registeredstamp ...*）能以指定的登錄時間戳記傳回給使用者。

保護 Enterprise Bean 的安全

如果您使用 WebSphere Application Server 來保護 Enterprise Bean 的安全，您必須將 `WCSecurityRole` 職務指定到任何新的 Enterprise Bean 的方法中。 WebSphere Application Server「應用程式組譯工具」就是用來執行這個作業。請在部署新的 Enterprise Bean 時執行這個步驟。另外，如果您修改現有 WebSphere Commerce Entity Bean，就必須將 `WCSecurityRole` 職務指定到已修改的 EJB 專案中的每一個 Entity Bean 中的每一個方法。

有關自訂程式碼的部署程序說明，請參閱第 185 頁的第 9 章, 『部署明細』。

主要鍵

主要鍵是一種唯一鍵值，爲表格定義的一部份。它可用來區別記錄。所有記錄皆必須有一個主要鍵。當您在表格中建立新記錄時，您可能得爲該記錄產生一個唯一的主要鍵。

在 WebSphere Commerce 程式設計模型中，持續層含有會和資料庫互動的 Entity Bean。從而當實例化某個 Entity Bean 時，可能會建立資料庫記錄。因此，實例化 Entity Bean 時所用的 ejbCreate 方法中可能需含有產生新記錄之主要鍵的相關邏輯。

當應用程式需要資料庫中的資訊時，會實例化 Entity Bean 的對應存取 Bean，並取得或設定各欄位，從而間接使用 Entity Bean。應用程式會針對資料庫中某特定記錄（例如：某特定的使用者設定檔）實例化存取 Bean，並使用主要鍵從資料庫中選取正確的資訊。

下節說明如何建立唯一的主要鍵，以及如何以主要鍵來選取。

建立主要鍵: ejbCreate 方法用以實例化 Entity Bean 的新實例。此方法會自動產生，但產生的方法中僅含有將主要鍵起始設定成靜態值的相關邏輯。

您可能需要確定主要鍵爲一個新的唯一值。在此情況下，您可能有一個類似下列程式碼片段的 ejbCreate 方法：

```
public void ejbCreate(int argMyOtherValue)
   throws javax.ejb.CreateException {
      //起始設定 CMP 欄位
      MyKeyValue = com.ibm.commerce.key.ECKeyManager.
         singleton().getNextKey("table_name");
      MyOtherValue = argMyOtherValue;
}
```

在上述的程式碼片段中，getNextKey 方法會爲主要鍵產生一個唯一整數。方法的 *table_name* 輸入參數必須完全和定義於 KEYS 表格中的 TABLENAME 值相符。請確定字元與大小寫皆完全相符。

除了讓 ejbCreate 方法中包含上述的程式碼外，您亦必須在 KEYS 表格中建立一個項目。下列的 SQL 陳述式範例，是在 KEYS 表格中建立項目：

```
insert into KEYS (TABLENAME, COUNTER, KEYS_ID)
   values ("table_name", 0, 1)
```

請注意，在上述的 SQL 陳述式中，會接受 KEYS 表格中其他直欄的預設值。COUNTER 值代表計數應從該值算起。 KEYS_ID 值應爲任何正數值。

如果您的主要鍵是定義成長資料類型（若爲 DB2®，則是 BIGINT，若爲 Oracle，則是 NUMBER(38, 0)），請使用 getNextKeyAsLong 方法。

以主要鍵選取: 在存取 Bean 中，您必須藉由主要鍵來選取適當的資料庫記錄。下列程式碼片段是示範如何執行此項選擇。其中亦包含額外的邏輯（將在稍後解釋）。

```
UserProfileAccessBean abUserProfile = new UserProfileAccessBean();
abUserProfile.setInitKey_UserId(getUserId().toString());
abUserProfile.refreshCopyHelper();
```

上述程式碼片段中的第一行是實例化一個新的 UserProfileAccessBean，稱爲 ″abUserProfile″。第二行則是在存取 Bean 中設定主要鍵。 WebSphere Studio Application Developer 使用 `setInitKey_`*xxx*（其中 *xxx* 是主要鍵欄位名稱）命名慣例來命名主要鍵的 set 方法。當實例化存取 Bean 時，您應確定在使用 refreshCopyHelper 方法前，以 setInitKey_xxx 方法設定的所有欄位皆已起始設定。setInitKey_xxx 方法的呼叫順序不重要。

在呼叫所有的 setInitKey_xxx 方法後，您便已起始設定所有必要的欄位，而可使用 refreshCopyHelper 方法來擷取資料庫中的資訊。

如果您有更新存取 Bean 之本端快取中的值，您也必須包含 commitCopyHelper 呼叫，以便使用更新過的資訊更新資料庫。舉例來說，如果您在使用 refreshCopyHelper 方法擷取資料後，更新了客戶的名稱（藉由設定名稱值），您必須呼叫 `abUserProfile.commitCopyHelper()`，以便以新資訊更新資料庫。

使用 Entity Bean

採用 Enterprise Bean 的程式必須處理 Java Naming and Directory Interface (JNDI) 以及 Enterprise Bean 的起源與遠端介面。爲了簡化程式設計模型，會爲每一個 Enterprise Bean 產生一個存取 Bean。在您建立自己的 Enterprise Bean 時，請使用 WebSphere Studio Application Developer 中的工具來產生此種存取 Bean。

WebSphere Commerce 指令是與存取 Bean 互動，而非直接與 Entity Bean 互動。如圖所示，使用存取 Bean 具有下列優點：

- 程式設計介面較爲簡單。存取 Bean 的行爲類似 Java Bean，並且會隱藏 Enterprise Bean 所有特定的程式設計介面就像 JNDI、起源與遠端介面，而不讓用戶端看到。
- 在執行期間，存取 Bean 會快取 Enterprise Bean 的起源物件，這是因爲從時間與資源的使用觀點來看，查看起源物件頗費時費工。
- 存取 Bean 會實作一種 copyHelper 物件，可在指令取得並設定 Enterprise Bean 的屬性時減少呼叫 Enterprise Bean 的次數。因而在讀取或寫入多個 Enterprise Bean 屬性時，只需呼叫一次 Enterprise Bean 即可。

下圖顯示指令、存取 Bean、Entity Bean 與資料庫之間的互動。

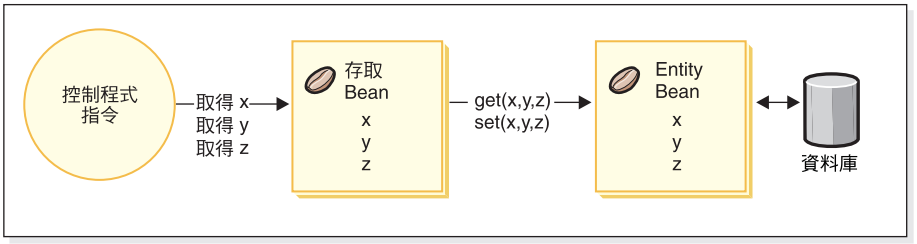

圖 18.

資料庫注意事項

在您自訂您的 e-commerce 應用程式時，您可能會建立新資料庫表格。在您建立這些表格時，建議您遵循一組使用慣例，以便讓您的表格與 WebSphere Commerce 表格取得一致。

命名資料庫綱目物件時的注意事項

下列各節提供命名資料庫綱目物件的相關指示。

表格與檢視畫面的命名慣例

以下提供命名新表格與檢視畫面時的相關指示。

- 爲了避免與未來新版中的 WebSphere Commerce 表格與檢視畫面名稱相衝突（名稱重複），表格或檢視畫面名稱的第一個字元應爲 X。例如：XMYTABLE。
- 表格或檢視畫面名稱不應超過 10 個字元。如果您所選的名稱超過此上限，請移除名稱尾端的母音以維持 10 個字元。
- 表格或檢視畫面名稱不應含有任何特殊字元（像是 “_”、“+”、“$”、“%”）或空格。
- 請勿使用資料庫的保留字作爲表格或檢視畫面名稱。
- 檢視畫面名稱的結尾應是 VW。
- 表格與檢視畫面名稱應爲單數名詞。

直欄的命名慣例

通常，當您建立的新表格是遵循先前的表格名稱慣例時，您可以在這些表格的直欄中實作您自己的命名慣例。這時會假設您在 SQL 陳述式中一律使用完整的直欄名稱。然而，如果您想要執行與現有 WebSphere Commerce 表格的結合，而您不要使用完整的直欄名稱，就必須遵循在這一節中說明的直欄命名慣例。

以下提供命名新表格中之直欄的相關指示。

- 爲了避免與未來新版中之 WebSphere Commerce 表格內的直欄名稱相衝突（名稱重複），直欄名稱的第一個字元應爲 X。例如：XMYCOLUMN。
- 直欄名稱不應超過 18 個字元。如果您所選的名稱超過此上限，請移除名稱尾端的母音，使其不超過 18 個字元。
- 直欄名稱（外來鍵以外）不應含有任何特殊字元（像是 “_”、“+”、“$”、“%”）或空格。
- 請勿使用資料庫的保留字作爲直欄名稱。
- 合併字可使用主動語法組合作爲直欄名稱。例如 COMBINERESULT。
- 產生之主要鍵直欄名稱的格式應爲 *table*_id。舉例來說，USERS 表格的主要鍵爲 USERS_ID。
- 產生的外來鍵直欄名稱不應變更。
- 任何保留供日後自訂用的直欄其名稱應爲 field*x*，其中 *x* 爲從 1 開始的數值。

索引的命名慣例

以下提供命名新表格中之索引的相關指示。

- 索引名稱長度不應超過 18 個字元。
- 索引名稱不應含有空格。
- 索引名稱不應含有任何資料庫保留字。
- 非唯一索引名稱的格式應爲 I_*tablex*；其中 *table* 爲表格的名稱，*x* 爲從 1 開始的數字。例如：USERS 表格的非唯一索引爲 I_USERS1。
- 唯一索引名稱的格式應爲 UI_*tablex*；其中 *table* 爲表格的名稱，*x* 爲從 1 開始的數字。例如：USERS 表格的唯一索引爲 UI_USERS1。
- 索引合計大小不應超過 254 個位元組。
- 索引名稱在整個資料庫綱目中必須是唯一的。

主要鍵的命名慣例

以下提供命名新表格之主要鍵的相關指示。

- 主要鍵名稱長度不應超過 18 個字元。
- 主要鍵名稱不應含有空格。
- 主要鍵名稱不應含有任何資料庫保留字。
- 主要鍵名稱的格式應爲 P_*table*；其中 *table* 爲表格名稱。舉例來說， USERS 表格的主要鍵爲 P_USERS。
- 主要鍵名稱在整個資料庫綱目中必須是唯一的。

外來鍵的命名慣例

以下提供命名新表格之外來鍵的相關指示。

- 外來鍵長度名稱不應超過 18 個字元。
- 外來鍵名稱不應含有空格。
- 外來鍵名稱不應含有任何資料庫保留字。
- 外來鍵名稱的格式應爲 F_*table*；其中 *table* 爲表格名稱。舉例來說， USERS 表格的外來鍵爲 F_USERS1。
- 外來鍵名稱在整個資料庫綱目中必須是唯一的。

資料庫觸發指令的命名慣例

以下提供命名資料庫觸發指令時的相關指示。

- 資料庫觸發指令名稱長度不應超過 18 個字元。
- 資料庫觸發指令名稱不應含有空格。
- 資料庫觸發指令名稱不應含有任何資料庫保留字。
- 資料庫觸發指令名稱的格式應爲 T_*table*；其中 *table* 爲表格名稱。舉例來說，USERS 表格的資料庫觸發指令名稱爲 T_USERS1。
- 資料庫觸發指令名稱在整個資料庫綱目中必須是唯一的。

資料庫直欄資料類型的注意事項

本節介紹在您建立新表格時所能使用的直欄資料類型。我們將採用 DB2 術語來說明各種資料類型。如果您所用的是其他資料庫，第 81 頁的『各資料庫間的資料類型差異』會說明其差異處。

BIGINT

爲 64 位元的正負號整數，其範圍從 -9223372036854775807 到 9223372036854775807。若與 INTEGER 相對照，INTEGER 只有 BIGINT 的一半大小。

INTEGER

爲 32 位元的正負號整數，其範圍從 -2147483647 到 2147483647。一般而言，您應使用 INTEGER 作爲預設的有限數值資料類型，而非使用 BIGINT。除了您有強烈的商業理由必須使用 BIGINT，基於效能，建議您使用 INTEGER 作爲數值資料類型。較常使用 BIGINT 資料類型的是系統產生的鍵值。

最不建議您使用 SMALLINT 或 SHORT 資料類型，因爲這些資料類型會對映至非物件式 Java 資料類型，且這些非物件式資料類型會在某些 Enterprise Bean 物件實例化時造成問題。

TIMESTAMP

此值共分 7 部份（年、月、日、時、分、秒以及毫秒），除了細微到包含毫秒規格的時間外，還指出日期與時間。時間戳記的內部表示法爲一個 10 位元組的字串，每一個時間戳記由二位壓縮十進位數組成。前 4 個位元組代表日期，隨後的 3 個位元組代表時間，最後的 3 個位元組代表毫秒。

CHAR 爲 INTEGER 長度的固定長度字串，其長度範圍可從 1 到 254 個字元。如果您省略長度規格，則會假設其長度爲 1 個字元。由於 CHAR 爲一種「固定長度」資料庫直欄，任何未用的尾隨字元空格會變成空白空格。除非基於效能原因，不建議您使用 CHAR 資料類型，因爲 CHAR 沒有彈性，且往後無法變更長度。依照經驗，如果您的字串直欄的長度少於 64 個字元，且會定期擷取或更新該直欄，則使用 CHAR 反而效能較好。

VARCHAR

爲最大長度整數的可變長度字串，其範圍可從 1 到 32672。不過，不像 CHAR 其直欄資料是隨表格儲存， VARCHAR 在內部環境中代表資料庫頁面內的一個參照指標。因此在建立後 VARCHAR 直欄的長度可隨時改變。

LONG VARCHAR

爲可變長度字串，可在無法於同一資料庫頁面中建立 VARCHAR 時使用。LONG VARCHAR 與 VARCHAR 非常相似，只不過它可橫跨多個資料庫頁面。請在有絕對必要時才使用 LONG VARCHAR 資料類型，這是因爲從效能的角度來看 LONG VARCHAR 物件通常較爲耗能。

CLOB 爲另一個可變長度字串，可在直欄長度必須超過 LONG VARCHAR 的 32KB 上限時使用。 CLOB 物件的長度可達 1 GB，且不需修改資料庫配置。當在不同系統間移動時，儲存成 CLOB 的文字資料將可適當轉換。

BLOB 爲可變長度的二進位字串，用以儲存資料庫中的非結構性資料。BLOB 物件最多可儲存 4 GB 的二進位資料。一般而言，除非絕對必要，您應儘量不要以 BLOB 作爲直欄資料類型。從效能的角度來看，在任何資料庫中 BLOB 物件一向被視爲最耗能的物件。

DECIMAL(20,5)

此種資料類型特別被定義成用於大多數的固定小數點數字上，像是貨幣單位。而在其他浮點十進位數方面，則可改用 FLOAT。

各資料庫間的資料類型差異

下表列出 WebSphere Commerce 資料庫綱目中所用的資料類型，並顯示不同資料庫實作的相對資料類型方面。

JDBC 物件	Windows AIX Solaris Linux DB2	Windows AIX Solaris Oracle®	400 DB2
Hashtable	BLOB()	BLOB	BLOB()
Timestamp	TIMESTAMP	DATE	TIMESTAMP
Integer	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER(38,0)	BIGINT
Double	FLOAT	NUMBER(38,0)	FLOAT
String	CHAR()	VARCHAR2()	GRAPHIC() CCSID 13488
byte[]	CHAR()（若爲位元資料）	RAW()	CHAR()（若爲位元資料）
String	VARCHAR()	VARCHAR2()	VARGRAPHIC() CCSID 13488
String	LONG VARCHAR	VARCHAR2()（詳細說明請參閱表格後的附註。）	VARGRAPHIC(4000) ALLOCATE() CCSID 13488
byte[]	LONG VARCHAR（若爲位元資料）	LONG RAW	VARCHAR(8000) ALLOCATE()（若爲位元資料）
String	CLOB()	CLOB()	DBCLOB() CCSID 13488

註:

當 Oracle JDBC 驅動程式處理 LONG 資料類型的資訊時，由於成功的比例不定，建議您盡量避免使用 LONG 資料類型。此情況中最常報告的錯誤是「串流已關閉」錯誤。

如果您必須使用這種資料類型，則每個資料庫表格中只能有一個直欄使用 LONG 類型。此外，在您建構 select 陳述式時，請勿讓 LONG 直欄成爲 select

中的第一個或最後一個元素。在大量負載下運作時的另一個解決辦法是避免此特定直欄對映至 Entity Bean 中的 CMP 欄位。相反地，請使用 Session Bean 來執行此直欄的擷取與更新。

第 4 章 存取控制

認識存取控制

WebSphere Commerce 應用程式的存取控制模型有下列三個主要概念：使用者、動作與資源。使用者爲使用系統的人。資源爲在應用程式中維護或由應用程式維護的實體。舉例來說，資源可以是產品、文件或訂單。而代表人員的使用者設定檔亦爲資源。動作是指使用者可對資源執行的活動。存取控制是電子商務應用程式中的元件，用以決定給定的使用者是否能對給定的資源執行給定的動作。

在 WebSphere Commerce 應用程式中，存取控制有兩個主要層次。存取控制的第一個層次由 WebSphere Application Server 執行。在此層面中，WebSphere Commerce 使用 WebSphere Application Server 來保護 Enterprise Bean 與 Servlet。存取控制的第二個層次則爲 WebSphere Commerce 中細密的存取控制系統。

WebSphere Commerce 存取控制組織架構採用存取控制原則來判斷給定使用者能否對給定的資源執行給定的動作。這種存取控制組織架構提供的是細密的存取控制。它會結合使用 WebSphere Application Server 所提供的存取控制但不會予以取代。

WebSphere Application Server 中的資源保護概觀

以下的 WebSphere Commerce 資源由 WebSphere Application Server 的存取控制保護：

- Entity Bean
  這些 Bean 仿造了電子商務應用程式中的物件。它們屬於分散式物件，可供遠端用戶端存取。
- JSP 範本
  WebSphere Commerce 在顯示頁面上採用了 JSP 範本。每一個 JSP 範本可含有一或多個會從 Entity Bean 中擷取資料的資料 Bean。用戶端可藉由撰寫 URL 要求來要求 JSP 頁面。
- 控制程式與檢視畫面指令
  用戶端可藉由撰寫 URL 要求來要求控制程式與檢視畫面指令。此外，顯示頁面中可藉由使用 JSP 檔名稱或檢視畫面名稱（視 VIEWREG 表格中的登錄而定），以包含移往另一個頁面的鏈結。

一般而言會將 WebSphere Commerce Server 配置成使用如下的 Web 路徑：

- `/webapp/wcs/stores/servlet/*`
  用於送往「要求 Servlet」的要求方面。
- `/webapp/wcs/stores/*.jsp`
  用於送往 JSP Servlet 的要求方面。

下圖是就上述的 Web 路徑配置，顯示要求在存取 WebSphere Commerce 資源時可能的遵循途徑。

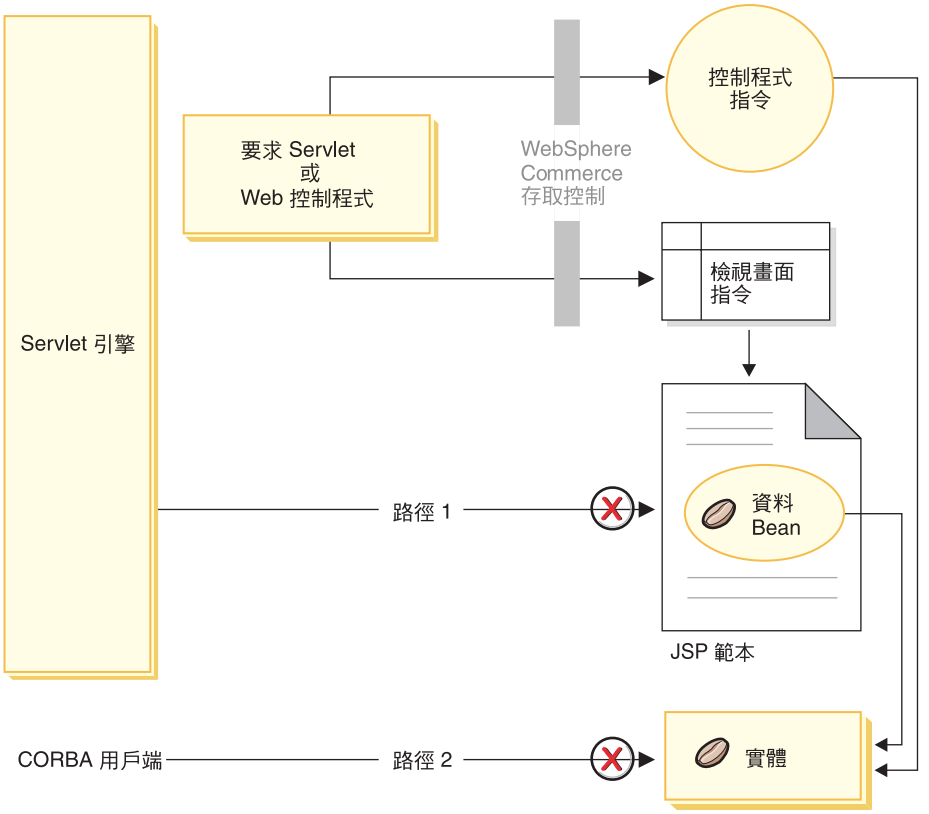

圖 19.

所有合格的要求應導向到「要求 Servlet」，再由「要求 Servlet」將之導向到 Web 控制程式。Web 控制程式會針對控制程式指令與檢視畫面實作存取控制。不過上述的 Web 路徑卻可讓某些蓄意的使用者有機會直接存取 JSP 範本（路徑 1）與 Entity Bean（路徑 2）。爲了讓這些蓄意攻擊無法得逞，在執行期間必須將之拒絕在外。

您可以使用下列一種方法，以防直接存取 JSP 範本與 Entity Bean：

WebSphere Application Server 安全特性

WebSphere Application Server 提供許多安全特性。WebSphere Commerce 會使用其中一個特性來確保所有的 Enterprise Bean 方法與 JSP 範本都是配置成只能由選擇的身份來呼叫。如果要存取這些 WebSphere Commerce 資源，則必須先將 URL 要求遞送到要求 Servlet。要求 Servlet 會先設定現行執行緒中的選定身份，然後在將它傳送到 Web 控制程式。接著 Web 控制程式會先確定呼叫端具有必要的權限，然後才將要求傳遞給對應的控制程式指令或檢視畫面。任何試著直接存取 JSP 範本與 Entity Bean （亦即未使用 Web 控制程式）的動作，都會遭 WebSphere Application Server 安全元件拒絕。

有關配置 WebSphere Application Server 以保護 WebSphere Commerce 資源的資訊，請參閱 *WebSphere Commerce Security Guide*。有關 WebSphere Application Server 中之安全的說明，請參閱 WebSphere Application Server 文件中的「系統管理」主題。

防火牆保護

當 WebSphere Commerce Server 在防火牆後面執行時，網際網路用戶端將無法直接存取 Entity Bean。當您使用此方法時，是由頁面中所含的資料 Bean 來提供 JSP 範本的保護。資料 Bean 是由資料 Bean 管理程式所啓動。資料 Bean 管理程式會偵測 JSP 範本是否由檢視畫面指令所轉遞。若不是由檢視畫面指令轉遞，則會擲出異常狀況，並拒絕該項 JSP 範本要求。

URL 參數的安全注意事項

URL 參數提供了一個有用的方式來傳送特定的要求資訊。爲了使懷有惡意的使用者以未經授權的方式存取您的資料庫的機率降至最低，您應該遵循特定的程式碼撰寫方式。SQL 陳述式中的 Insert、select、update 和 delete 部分，應該在開發時建立。您應該使用插入參數的方法來蒐集執行期間輸入資訊。

以下是使用插入參數方式來收集執行期間輸入資訊的範例：

```
select * from Order where owner =?
```

相對地，您應避免使用輸入字串作爲撰寫 SQL 陳述式的方法。以下是使用輸入字串的範例：

```
select * from Order where owner = "input_string"
```

避免使用輸入字串來撰寫 select 的原因是輸入字串容易操作。懷有惡意的使用者可以將這類輸入字串設定成非預期的值，藉以取得未經授權的存取權，或對資料庫執行不當的作業。

WebSphere Commerce 存取控制原則的簡介

本節提供 WebSphere Commerce 存取控制組織架構的主要元件的簡要說明。提供這些說明的目的是爲了讓您以正確的環境定義來檢視與程式設計作業相關的存取控制。如果您需要在正式作業系統上設定存取控制的相關資訊，或者需要存取控制組織架構的其他明細，請參閱 *WebSphere Commerce Security Guide*。

WebSphere Commerce 存取控制模型是以實作存取控制原則爲基礎。存取控制原則容許存取控制規則和商業邏輯程式碼分開，因而不用將存取控制陳述式硬寫在程式碼中。例如，您不必包含如下的程式碼：

```
if (user.isAdministrator())
   then {}
```

存取控制原則是透過存取控制原則管理程式實施。一般而言，當使用者嘗試存取受保護的資源時，存取控制原則管理程式會先判斷該受保護的資源有哪些存取控制原則適用，然後再依據適用的存取控制原則，來判斷使用者能否存取所要的資源。

存取控制原則是一種 4 變數值組的原則，且儲存在 ACPOLICY 表格中。每一個存取控制原則的格式如下：

```
AccessControlPolicy [UserGroup, ActionGroup, ResourceGroup, Relationship]
```

4 變數值組型存取控制原則中的元素會指出屬於特定使用者群組的使用者，可針對屬於指定資源群組的資源，執行指定動作群組中的動作，但前提是使用者得符合該資源的相關關係或關係群組中指定的條件。例如，`[AllUsers, UpdateDoc, doc, creator]` 表示只要使用者是文件的建立者，便能更新文件。

使用者群組爲一種特定的成員群組類型，其定義在 MBRGRP 資料庫表格中。使用者群組必須連結成員群組類型 `-2`。值 `-2` 代表一種存取群組，且定義於 MBRGRPTYPE 表格中。使用者群組與成員群組類型間的連結關係儲存在 MBRGRPUSG 表格中。

使用者與特定使用者群組間的成員關係可明確或隱含陳述。當 MBRGRPMBR 表格中有指出使用者隸屬於某特定成員群組時，即屬明確指定。而如果使用者符合 MBRGRPCOND 表格中的陳述條件（例如，執行「產品經理」職務的所有使用者），即屬隱含指定。另外，亦可能存在合併條件（例如，執行「產品經理」職務並擔任該職務至少 6 個月的所有使用者）或明確排除。

大部份用以將使用者納入使用者群組中的條件，是以使用者執行某特定職務爲基礎。例如，可以用一個存取控制原則來容許擔任「產品經理」職務的所有使用者執行型錄管理作業。在此情況下，凡在 MBRROLE 表格中被指定爲「產品經理」職務的使用者，即隱含納於使用者群組中。

有關成員群組子系統的其他詳細資料，請參閱 WebSphere Commerce 正式作業與開發線上說明。

ActionGroup 元素出自 ACACTGRP 表格。動作群組會參照某個明確指定的動作群。動作清單儲存在 ACACTION 表格中，而每個動作和其動作群組間的關係則儲存在 ACACTACTGP 表格中。例如像 ″OrderWriteCommands″ 動作群組即爲一種動作群組。此動作群組中含有下列各種用以更新訂單的動作：

- com.ibm.commerce.order.commands.OrderDeleteCmd
- com.ibm.commerce.order.commands.OrderCancelCmd
- com.ibm.commerce.order.commands.OrderProfileUpateCmd
- com.ibm.commerce.order.commands.OrderUnlockCmd
- com.ibm.commerce.order.commands.OrderScheduleCmd
- com.ibm.commerce.order.commands.ScheduledOrderCancelCmd
- com.ibm.commerce.order.commands.ScheduledOrderProcessCmd
- com.ibm.commerce.order.commands.OrderItemAddCmd
- com.ibm.commerce.order.commands.OrderItemDeleteCmd
- com.ibm.commerce.order.commands.OrderItemUpdateCmd
- com.ibm.commerce.order.commands.PayResetPMCmd

資源群組是一種將特定資源類型集結在一起的機制。資源與群組資源間的成員關係可用下列兩種方法之一指定：

- 使用 ACRESGRP 表格中的條件直欄
- 使用 ACRESGPRES 表格

在大部份情況下，要讓資源連結資源群組使用 ACRESGPRES 表格即可。當使用此方法時，資源是以其 Java 類別名稱定義在 ACRESCGRY 表格中。然後透過 ACRESGPRES 連結表格將這些資源連結適當的資源群組（ACRESGRP 表格）。如果單使用 Java 類別名稱尚不足以定義資源群組的成員（例如，如果您需要依照資源的屬性，進一步限制此類別的物件），您可使用 ACRESGRP 表格的條件直欄完整定義資源群組。請注意，如果要依照屬性來分組資源，則資源亦必須實作「可分組」介面。

下圖舉例說明資源分組的指定。在本例中，資源群組 10023 含有 ACRESGPRES 表格中所有和其連結的資源。資源群組 10070 則以 ACRESGRP 表格中的 Conditions 欄位直欄定義出。此資源群組中含有「訂單」遠端介面的實例，其狀態爲 ″Z″（代表共用的需求項目清單）。

註: 有關 ACRESGRP 表格的「條件」直欄的 XML 資訊明細，可以在*WebSphere Commerce Security Guide*中找到。

ACRESGRP

AcResGrp_Id	GrpName	條件
10023	AccountRepresentatives CmdResourceGroup	空值
10070	SharedRequisitionList ResourceGroup	<profile> <andListCondition> <simpleCondition> <variable name="Status"/> <operator name="="/> <value data="Z"/> </simpleCondition> <simpleCondition> <variable name="classname"/> <operator name="="/> <value data="com.ibm.commerce.order.objects.Order"/> </simpleCondition> </andListCondition> </profile>

ACRESGRPES

AcResGrp_Id	AcResCgry_Id
10023	10246
10023	10247
10023	10248
10023	10249
10023	10250

ACRESCGRY

AcResCgry_Id	ResClassname
10246	com.ibm.commerce.contract.commands.ContractCreateCmd
10247	com.ibm.commerce.contract.commands.ContractUpdateCmd
10248	com.ibm.commerce.contract.commands.ContractDeleteCmd
10249	com.ibm.commerce.contract.commands.ContractCancelCmd
10250	com.ibm.commerce.contract.commands.ContractCloseCmd

圖 20.

ACACTGRP、ACRESGRP 與 ACRELGRP 表格中的 MEMBER_ID 直欄值應爲 -2001（根組織）。

存取控制原則可選擇性地包含 Relationship 或 RelationshipGroup 元素做爲其第四個元素。

如果您的存取控制原則使用 Relationship 元素，則其出自 ACRELATION 表格。相反地，如果其包含 RelationshipGroup 元素，則其出自 ACRELGRP 表格。請注意，不一定要包含這兩者，但如果要包含則只能擇一。ACRELGRP 表格中的 RelationshipGroup 規格會比 ACRELATION 表格中的 Relationship 資訊優先採用。

ACRELATION 表格用以指出存在於使用者與資源間的關係類型。例如，關係類型可爲：creator（建立者）、submitter（提交者）與 owner（擁有者）。而會用到 relationship 元素的情況像是：用來確定訂單的建立者恆可更新訂單。

ACRELGRP 表格用以指定可連結特定資源的關係群組類型。關係群組是由一或多個關係鏈組成的群組。關係鏈爲一系列一個以上的關係。舉例來說，關係群組可指出使用者必須是資源的建立者，且隸屬於資源中所參照的買方組織實體。

關係群組（或關係）規格是存取控制原則中的一個選用部份。通常是在您建有自己的指令且未限制這些指令僅供某些職務使用時使用。在這些情況下，您可能會想在使用者與資源間設立一項關係。一般而言，如果指令被限制只有某些職務才能使用，通常是透過存取控制原則的 UserGroup 元素來達到此目的，而非使用 Relationship 元素。

與存取控制原則相關的另一個重要概念是存取控制原則群組的概念。 WebSphere Commerce 5.5 版可支援各種商業模型，而每一種商業模型都有自己的一組存取控制原則。如果要將模型內的原則集組合起來，可以使用原則群組。將原則明確地指定至適當的原則群組後，組織就可以訂閱一或多個原則群組。在先前的 WebSphere Commerce 版本中，原則會套用到該原則的擁有者組織的下一代所擁有的全部資源。

在 WebSphere Commerce 5.5 版中，如果組織訂閱一或多個原則群組，只有這些原則群組中的原則會套用到該組織的資源中。如果擁有資源的組織並未訂閱任何原則群組，則存取控制原則管理程式會搜尋組織階層，直到發現至少訂閱一個原則群組以及最接近的上一代組織，一旦找到之後，就會套用屬於這些原則群組的原則。

請看下列圖解：

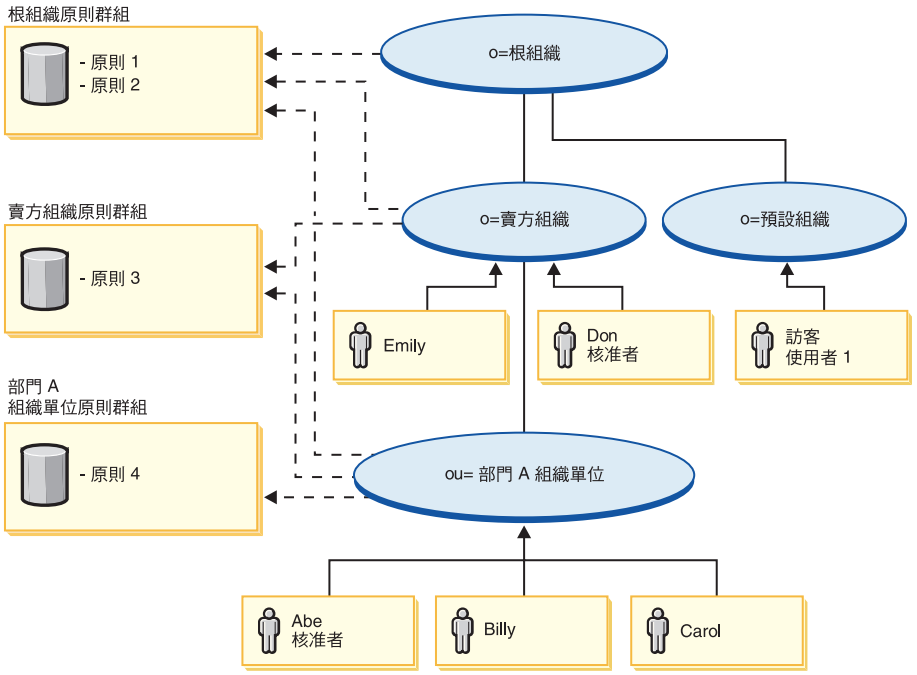

圖 21.

對「賣方組織」所擁有的任何資源，會套用「賣方組織」原則群組以及「根組織」原則群組中的原則。這些原則是適用的，因爲「賣方組織」已經明確訂閱這兩個原則群組（帶點的箭頭表示訂閱）。總計這個組織會套用三個原則。在這個範例中，就是已明確訂閱原則群組的資源所隸屬的組織。圖解也顯示組織並未明確訂閱任何原則群組的一個範例，所以存取控制組織架構會往階層上方繼續搜尋。至於「預設組織」所擁有的資源，就必須尋找階層上方的「根組織」。「根組織」只會訂閱一個原則群組，所以這些是唯一適用於「預設組織」的資源的原則（原則 1 和 2）。

關係群組

關係群組可讓您指定多重關係。關係可以是使用者與該資源間的直接關係，也可以是間接讓使用者連結資源的一個關係鏈。

註： 在下列和關係群組有關的各節中，要強調的是 WebSphere Commerce Professional Edition 中的可用組織只有：RootOrganization、DefaultOrganization 與 SellerOrganization。凡提及其他組織的範例則僅適用於 WebSphere Commerce Business Edition。

比較關係與關係群組： 存取控制原則可指出使用者和所要存取的資源間必須存在某種特定關係，或者可指出使用者必須具備關係群組中的指定條件。

在大部份情況下，當您指定關係時，應符合您應用程式的存取控制需求。不過，如果因該原則使得您必須在使用者與資源之間指定一個間接關係，但是實際上是使用者與資源之間的一系列關係，則必須使用關係群組。

舉例來說，如果您必須在使用者與買方組織間指定一項連結，而其中的關係要求使用者必須擔任該組織的某個特定職務，或該使用者必須是買方組織的成員，則您必須用到關係群組與關係鏈。

如果您純粹只想直接在使用者與該資源間設立一項連結，您可使用一個單純關係。舉例來說，當您想指出該使用者必須是資源的建立者時。

如果您組合了多個單純關係（舉例來說，使用者必須是建立者或提交者），便成爲一種關係鏈，且您必須使用關係群組。如果您使用的是 WebSphere Commerce Professional Edition 或 WebSphere Commerce Business Edition，便可能出現此種單純關係組合現象。

關係群組的一般資訊: 關係鏈爲一系列一個以上的關係。關係鏈的長度取決於所含的關係數目而定。您可檢查關係鏈 XML 表示法中的 `<parameter name="aName" value="aValue" />` 元素數即可判定。

只有最後的 `<parameter name="Relationship" value="aValue" />` 元素才必須由資源的 fulfills() 方法處理。其餘的則由存取控制原則管理程式內部處理。

當關係鏈的長度爲 2 時，第一個 `<parameter name="aName" value="aValue" />` 元素是在使用者與組織實體之間。最後的 `<parameter name="aName" value="aValue" />` 元素是在組織實體與資源之間。

如果您需要定義關係群組，必須藉由在 XML 檔中定義關係群組資訊來完成。您可以根據 `defaultAccessControlPolicies.xml` 檔來建立您自己的 XML 檔。有關建立 XML 型資訊的其他資訊，請參閱 *WebSphere Commerce Security Guide*。

存取控制類型

存取控制有兩種類型，這兩種類型皆以原則爲基礎：指令層次的存取控制與資源層次的存取控制。

指令層次（亦稱爲「職務型」）的存取控制採用廣泛的原則類型。您可以指出凡具備特定職務的使用者可執行某些指令類型。舉例來說，您可以指定具備「帳戶代表」職務的使用者可執行 AccountRepresentativesCmdResourceGroup 資源群組中的任何指令。或者，以下圖所述的另一原則爲例，指出所有商店管理者可對 StoreAdminCmdResourceGrp 指定的任何資源執行 ExecuteCommandAction 群組中指定的任何動作。

註: MBRGRPCOND 表格之 Conditions 直欄的 XML 資訊是在您使用管理主控台來設置存取群組時產生。有關使用管理主控台來設定存取群組的資訊，請參閱 WebSphere Commerce 正式作業線上說明。

ACPOLICY

PolicyName	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdministrators CmdResourceGroup	-2001	-8	10052	10018	空值

MBRGRP

MbrGrp_Id	MbrGrpName
-8	StoreAdministrators

MBRGRPCOND

MbrGrp_Id	條件
-8	<profile> <simpleCondition> <variable name="role"/> <operator name="="/> <value data="Store Administrator"/> </simpleCondition> </profile>

ACACTGRP

AcActGrp_Id	GroupName
10052	ExecuteCommandActionGroup

ACRESGRP

AcResGrp_Id	GrpName
10018	StoreAdministratorsCmdResourceGroup

圖 22.

指令層次的存取控制原則恆有 `ExecuteCommandActionGroup` 以作爲控制程式指令的動作群組。就檢視畫面而言，資源群組恆爲 `ViewCommandResourceGroup`。

所有控制程式指令必須受指令層次的存取控制保護。此外，凡是可直接呼叫的檢視畫面，或者可藉由另一個指令重新導向而啓動的檢視畫面（相對於藉由轉遞至檢視畫面而啓動）皆必須受指令層次的存取控制保護。

指令層次的存取控制不會考慮到指令所要執行的資源對象。它只會判斷該使用者能否執行特定的指令。假設使用者能執行該指令，接著便會套用資源層次的存取控制原則，以判斷使用者能否存取考量中的資源。

假設某商店管理者試著執行某項管理作業。第一層的存取控制檢查會判斷該使用者能否執行特定的商店管理指令。一旦它判斷出該使用者確能如此做時（因爲商店管理者能執行 storeAdminCmds 群組中的指令），即可呼叫資源層次的存取控制原則。此原則可能指出：商店管理者能執行管理作業的商店，僅限於該使用者在其中扮演商店管理者職務之組織所擁有的商店。

總結來說，在指令層次的存取控制中，「資源」爲指令本身，而「動作」純爲執行指令（換句話說，實例化指令物件）。存取控制檢查會判斷使用者能否執行指令。相對地，在資源層次的存取控制中，「資源」可爲指令或 Bean 所存取之任何可保護的資源，而「動作」則爲指令本身。

存取控制的互動

本節以互動圖解來說明存取控制在 WebSphere Commerce 存取控制原則組織架構下如何運作。

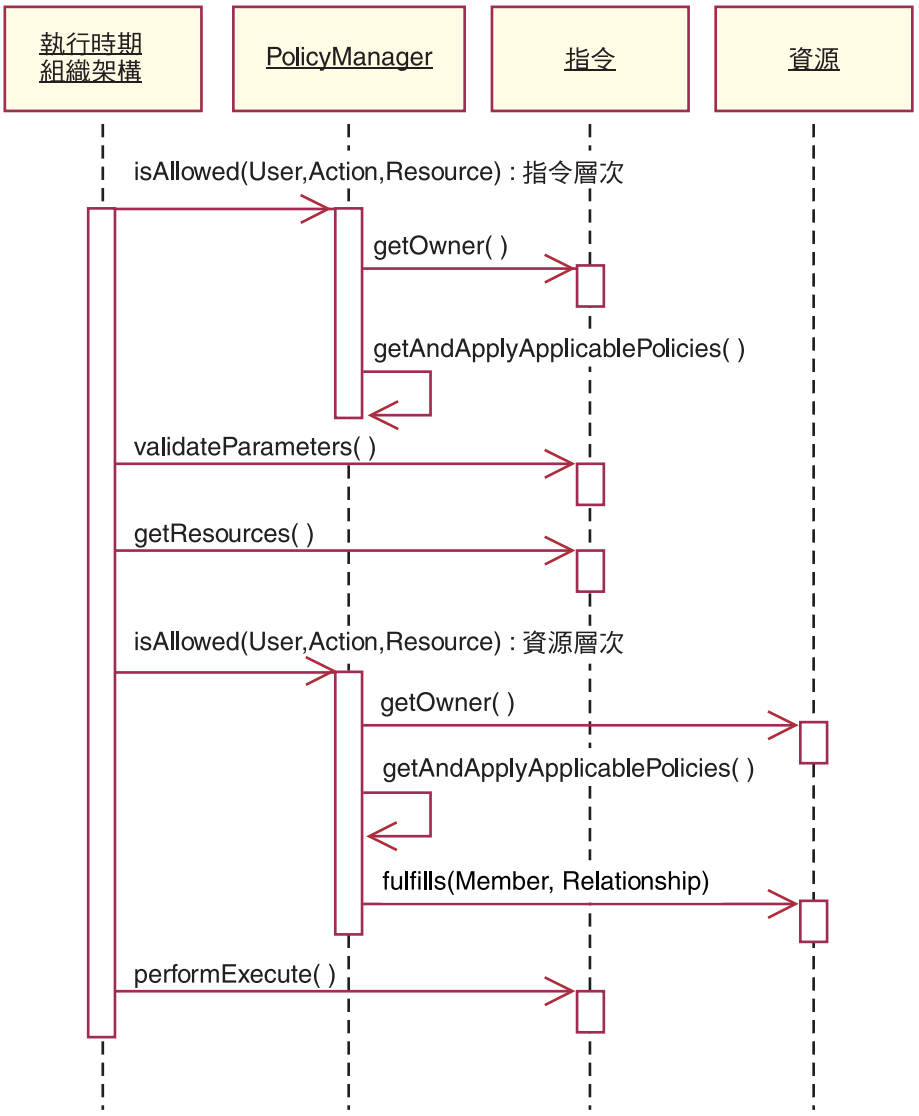

圖 23.

上圖顯示存取控制*原則管理程式*所執行的動作。存取控制原則管理程式是一種存取控制元件，會判斷現行使用者能否對指定資源執行指定動作。它是藉由搜尋資源擁有者所訂閱的群組中的原則，來作出判斷。如果資源擁有者沒有訂閱任何原則群組，就會搜尋資源擁有者最接近的上一代所訂閱的群組中的原則。只要至少有一項原則授與存取，即授與許可權。

下列說明上述互動圖中的各項動作。且從圖中的上到下依序說明。

1. isAllowed()
   執行期間元件會判斷使用者是否具備控制程式指令或檢視畫面的指令層次存取權。
2. getOwner()
   存取控制原則管理程式判斷指令層次資源的擁有者。就預設實作而言，會傳回指令環境定義中的商店 (storeId) 擁有者的成員識別碼 (memberId)。如果指令環境定義中沒有商店識別碼，則會傳回根組織 (-2001)。
3. getAndApplyApplicablePolicies()
   存取控制原則管理程式會根據指定的使用者、動作與資源，來尋找及處理適用的原則。如果至少有一個適用的原則可授與存取權，指令層次的存取權會檢查通過狀況，而原則管理程式會繼續下一步，以開始檢查資源層次的授權。相反的，如果沒有適用的原則可授與指令層次的存取權，這時原則管理程式就會返回並拒絕存取。
4. validateParameters()
   起始參數的檢查與解析。
5. getResources()
   傳回一個存取向量（即「資源/動作」對的向量）。

   假設未傳回任何項目，則不會執行資源層次的存取控制檢查。如有資源應受保護，則應會傳回一個存取向量（由「資源/動作」對組成）。

   每一項*資源*皆爲可保護物件（即實作 com.ibm.commerce.security.Protectable 介面的物件）的實例。在眾多情況下，資源爲一種存取 Bean。

   存取 Bean 不見得實作 com.ibm.commerce.security.Protectable 介面，不過只要其對應的 Enterprise Bean 受到保護，仍可進行存取控制檢查（相關資訊請參閱第 100 頁的『在 Enterprise Bean 中實作存取控制』）。

   *動作*爲一種字串，代表對資源所執行的作業。在大部份情況下，動作爲指令的介面名稱。
6. isAllowed()
   執行期間元件會判斷使用者對於 getResources() 指定的所有資源動作配對是否具備資源層次的存取權。
7. getOwner()
   資源傳回其擁有者的 memberId。這是判斷要套用哪些原則。
8. getAndApplyApplicablePolicies()
   存取控制原則管理程式搜尋適用的原則，然後加以套用。如果每個「資源/動作」對至少找到一項原則可授與使用者存取資源的許可權，則授與存取，否則則拒絕存取。

9. fulfills()
   如果適用的原則已有指定關係或關係群組，就會對資源執行檢查，以察看成員是否符合對於資源的指定關係。
10. performExecute()
    指令的商業邏輯。

可保護介面

若要讓資源受 WebSphere Commerce 存取控制原則保護，其關鍵因素在於資源必須實作 com.ibm.commerce.security.Protectable 介面。此介面最常搭配 Enterprise Bean 與資料 Bean 使用，但只有這些需要保護的特定 Bean 才需實作此介面。

在「可保護」介面下，資源必須提供下列兩個關鍵方法：getOwner() 與 fulfills(Long member, String relationship)。

存取控制原則爲組織或組織實體所擁有。 getOwner 方法會傳回可保護資源之擁有者的 memberId。在存取控制原則管理程式判斷出資源擁有者後，亦會取得成員階層中該擁有者之每一個上層的 memberId。所有隸屬於原始 getOwner 要求中之擁有者的存取控制原則，以及隸屬於擁有者之上層的所有存取控制原則皆會套用。

適用於指定擁有者的存取控制原則，以及適用於成員階層中擁有者之任何上層的存取控制原則皆會套用。

只有在給定成員符合所要的資源關係時，fulfills 方法才會傳回 true。通常成員爲單一使用者，不過也可以是一個組織。如果您在存取控制原則中有使用關係群組，則會是組織。

可分組介面

存取控制原則的應用將隨資源群組而定。您可根據屬性（像是：類別名稱、訂單狀態或 storeId 值）來進行資源分組。

如果您爲了套用存取控制原則，而依屬性（而非其類別名稱）來分組資源，則其必須實作 com.ibm.commerce.grouping.Groupable 介面。

下列的程式碼片段顯示「可分組」介面：

```
Groupable interface {
 Object getGroupingAttributeValue (String attributeName, GroupContext context)
}
```

舉例來說，假設您要實作一項僅套用於處於擱置狀態（status = P（擱置））的訂單之原則， Order Entity Bean 的遠端介面會實作「可分組」介面，且 attributeName 值會設爲 "status"。

通常很少使用「可分組」介面。

尋找存取控制的詳細資訊

有關 WebSphere Commerce 存取控制模型的其他資訊，請參閱 *WebSphere Commerce Security Guide*。此手冊會提供存取控制的詳細概觀，並說明如何使用管理主控台來建立或修改原則、動作群組與資源群組。

實作存取控制

本節說明如何以自訂程式碼來實作存取控制。

識別可保護的資源

一般而言，Enterprise Bean 與資料 Bean 可能是您想保護的資源。不過，並非所有的 Enterprise Bean 與資料 Bean 皆應保護。在現有的 WebSphere Commerce 應用程式中，需要保護的資源已實作可保護介面。通常，當您建立新的 Enterprise Bean 與資料 Bean 時，便會出現該保護哪些資源的問題。請根據您的應用程式來決定所要保護的資源。

如果指令在 `getResources` 方法中傳回 Enterprise Bean，則該 Enterprise Bean 必須加以保護，這是因爲存取控制原則管理程式會對該 Enterprise Bean 呼叫 getOwner 方法。如果對應的資源層次存取控制原則中有指定關係，亦會呼叫 `fulfills` 方法。

如果您針對自己所有的 Enterprise Bean 與資料 Bean 實作可保護介面（因而讓資源受到保護）您的應用程式可能會要求許多原則。原則越多，效能可能降低，且原則管理亦趨複雜。

主要資源與相依資源有理論上的區分。*主要資源*可獨立存在。*相依資源*則必須在其相關主要資源存在時才存在。舉例來說，在「開箱即用」WebSphere Commerce 應用程式碼中， Order Entity Bean 爲可保護的資源，OrderItem Entity Bean 則不是。其原因在於 OrderItem 的存在與否端視 Order 而定 -- Order 爲主要資源，而 OrderItem 爲相依資源。如果使用者應具備 Order 的存取權，則亦應具備訂單中各項目的存取權。

同樣地，User Entity Bean 爲可保護的資源，但 Address Entity Bean 則不是。在此情況中，地址的存在與否端視使用者而定，因此只要能夠存取該使用者，便應能夠存取該地址。

主要資源應受保護，但相依資源通常不需保護。如果使用者能存取主要資源，在預設的情況下，該使用者也應能存取其相依資源。

在 Enterprise Bean 中實作存取控制

如果您新建立的 Enterprise Bean 需要受存取控制原則保護，您必須執行下列步驟：

1. 建立新 Enterprise Bean，並確定它是從 `com.ibm.commerce.base.objects.ECEntityBean` 延伸而來。
2. 確定 Bean 的遠端介面是 `com.ibm.commerce.security.Protectable` 介面的延伸。
3. 如果您爲了套用存取控制原則而依照屬性來將資源分組，而不是依照其 Java 類別名稱，則 Bean 的遠端介面也必須延伸 `com.ibm.commerce.grouping.Groupable` 介面。
4. Enterprise Bean 類別繼承 com.ibm.commerce.base.objects.ECEntityBean 中的下列方法的預設實作：
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   請改寫任何您要的方法。您至少必須改寫 `getOwner` 方法。

   如果有存取控制原則在其資源群組中包含這個資源，並且指定關係或關係群組，就必須實作 fulfills 方法。如果存取控制原則的隱含資源群組包含這個資源的特定實例，就必須根據特定的屬性值來實作 getGroupingAttributeValue 方法（比方說，如果有一個存取控制原則只和狀態爲 'P'（擱置）的「訂單」相關。）

   請注意，如果唯一需要的關係是「擁有者」，就不需要改寫 fulfills 方法。在此情況下，原則管理程式會利用 getOwner() 方法的結果。

   下列程式碼片段顯示這些方法的預設實作。這些實作是來自 ECEntityBean 類別。

```
*****************************************************************************
public Long getOwner() throws Exception
{
      return null;
}
*****************************************************************************
*****************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception {
      return false;
}
*****************************************************************************
```

```
*************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception
{
      return null;
}
*************************************************************************
```

以下是這些方法的範例實作（以 OrderBean Bean 中所使用的實作爲基礎）：

- 對於 getOwner 方法而言，所提供的方法的邏輯是：

```
*************************************************************************
   com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB =
   new com.ibm.commerce.common.objects.StoreEntityAccessBean();
   storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
   return storeEntAB.getMemberIdInEJBType();
*************************************************************************
```

- 對於 fulfills 方法而言，所提供的方法的邏輯是：

```
*************************************************************************
if ("creator".equalsIgnoreCase(relationship))
{
   return member.equals(bean.getMemberId());
}
else if ("BuyingOrganizationalEntity".equalsIgnoreCase(relationship))
{
   return (member.equals(bean.getOrganizationId()));
}
else if ("sameOrganizationalEntityAsCreator".
   equalsIgnoreCase(relationship))
{
   com.ibm.commerce.user.objects.UserAccessBean creator =
     new com.ibm.commerce.user.objects.UserAccessBean();
   creator.setInitKey_MemberId(bean.getMemberId().toString());
   com.ibm.commerce.user.objects.UserAccessBean ab =
     new com.ibm.commerce.user.objects.UserAccessBean();
           ab.setInitKey_MemberId(member.toString());
           if (ab.getParentMemberId().equals(creator.getParentMemberId()))
     return true;
}
      return false;
*************************************************************************
```

- 對於 getGroupAttributeValue 方法而言，所提供的方法的邏輯是：

```
*************************************************************************
      if (attributeName.equalsIgnoreCase("Status"))
         return getStatus();
      return null;
 *************************************************************************
```

5. 建立（或重建）Enterprise Bean 的存取 Bean 與產生的程式碼。

請注意，如果您檢查其他的 WebSphere Commerce 公用 Entity Bean，以瞭解 getOwner、fulfills 和 getGroupingAttributeValue 方法的實作方式，就會注意到這些

方法是在 Bean 的存取輔助程式類別中實作。因此，方法是在存取輔助程式類別中實作，而不是直接在 Bean 類別中實作，所以方法簽章會略有差異。特別是取用額外的輸入參數，以便將物件本身傳送到存取輔助程式中的方法。

您*必須*確定當您在建立新的 Bean 時，您是直接在 Bean 類別中實作這些方法。另外，您不可以修改 WebSphere Commerce 公用 Entity Bean 的存取輔助程式類別中的任何方法。

在資料 Bean 中實作存取控制

如果您想保護某個資料 Bean，則可由存取控制原則直接或間接保護。如果資料 Bean 受直接保護，則會有一個套用在該特定資料 Bean 的存取控制原則。如果資料 Bean 受間接保護，則是委由有套用存取控制原則的另一資料 Bean 來保護。

如果要決定資料 Bean 是否需要保護，請考慮下列事項：

1. 資料 Bean 資訊中的資訊是否需要保護？例如，資訊是不是專用的？如果不是，就不需要藉由存取控制來提供直接保護。如果是的話，請繼續。
2. 資料 Bean 是由檢視畫面來實例化。檢視畫面是否會使用從指令環境定義取得的資訊（或某些其他預先決定的資訊）來將資料 Bean 實例化？如果是，而且存取控制已經決定容許存取，則不需要對這個資料 Bean 提供直接保護。如果不是，請繼續。
3. 檢視畫面是否會使用從某些輸入參數所取得的資訊來將資料 Bean 實例化？在此情況下，由於您並不確定存取控制是否已經決定讓使用者存取這項資訊，因此您應該保護新的資料 Bean。

如果您想建立一個直接由存取控制原則保護的新資料 Bean，則資料 Bean 必須執行下列步驟：

1. 實作 `com.ibm.commerce.security.Protectable` 介面。在此情況下， Bean 必須提供 `getOwner()` 與 `fulfills(Long member, String relationship)` 方法的實作。

   當資料 Bean 實作「可保護的」介面時，資料 Bean 管理程式會呼叫 `isAllowed` 方法，根據目前的存取控制原則來判斷使用者是否具備適當的存取控制專用權。下列程式碼片段說明 `isAllowed` 方法：

   ```
   isAllowed(Context, "Display", protectable_databean);
   ```

   其中 *protectable_databean* 是要保護的資料 Bean。
2. 如果和 Bean 互動的資源是依屬性（而非資源的 Java 類別名稱）分組，則 Bean 必須實作 `com.ibm.commerce.grouping.Groupable` 介面。
3. 實作 `com.ibm.commerce.security.Delegator` 介面。此介面是以下列程式碼片段來說明：

```
Interface Delegator {
 Protectable getDelegate();
}
```

註: 如果要直接保護，getDelegate 方法應傳回資料 Bean 本身（亦即，資料 Bean 基於存取控制而委任給自己）。

哪些資料 Bean 應直接保護以及哪些資料 Bean 應間接保護的區別，和主要與相依資源間的區別類似。如果資料 Bean 物件可獨立存在，則應直接保護。如果資料 Bean 的存在與否取決於另一資料 Bean 的存在而定，則應委由另一資料 Bean 來保護。

例如像 Order 資料 Bean 即爲直接保護的資料 Bean。而像 OrderItem 資料 Bean 則爲間接保護的資料 Bean。

如果您想建立一個由存取控制原則間接保護的新資料 Bean，則資料 Bean 必須執行下列：

1. 實作 com.ibm.commerce.security.Delegator 介面。此介面是以下列程式碼片段來說明：

   ```
   Interface Delegator {
    Protectable getDelegate();
   }
   ```

   註: getDelegate 傳回的資料 Bean 必須實作「可保護」介面。

如果資料 Bean 未實作 Delegator 介面，則會在沒有存取控制原則保護下移入資料。

在控制程式指令中實作存取控制

在建立新控制程式指令時，新指令的實作類別應該繼承 com.ibm.commerce.commands.ControllerCommandImpl 類別，而且其介面應該繼承 com.ibm.commerce.command.ControllerCommand 介面。

就控制程式指令的指令層次存取控制原則來說，會以指令的介面名稱作爲資源。爲了讓資源能受保護，必須要實作 Protectable 介面。根據 WebSphere Commerce 程式設計模型，您可藉由讓指令的介面延伸自 com.ibm.commerce.command.ControllerCommand 介面，以及讓指令的實作延伸自 com.ibm.commerce.commands.ControllerCommandImpl 介面，來完成這個步驟。ControllerCommand 介面繼承 com.ibm.commerce.command.AccCommand 介面，後者又繼承 Protectable。爲了可以受指令層次存取控制的保護，AccCommand 介面是指令要實作的最小介面。

如果指令會存取應受保護的資源，請建立一個 AccessVector 類型的專用實例變數，以放置資源。然後改寫 getResources 方法，因爲這個方法的預設實作是傳回空值，因此不會進行任何資源檢查。

在新 getResources 方法中，您應傳回指令所能執行的資源陣列或「資源/動作」對陣列。如果未明確指定動作，將會執行指令介面名稱的預設動作。

當指令對同一個資源類別的不同實例執行讀取和寫入作業時，只需要指定動作。例如，在 OrderCopy 指令中，它可以讀取來源訂單以及寫入目標訂單。在此情況下，兩個動作之間必須有區別。這是藉由爲來源訂單指定 "-Read" 動作以及爲目標訂單指定 "-Write" 動作來完成。當存取控制組織架構偵測到這些動作時，就會自動在它們前面加上指令的介面名稱，然後才搜尋適用的原則。在此情況下，在原則中最後會使用的動作是 "com.ibm.commerce.order.commands.OrderCopyCmd-Read" 和 "com.ibm.commerce.order.commands.OrderCopyCmd-Write" 動作。

此外，建議您由方法來判斷是否必須實例化資源，或者可否使用內含資源參照的現有實例變數。檢查資源物件是否存在，有助提昇系統效能。必要時，您可以在新控制程式指令的 performExecute 方法中使用相同的實例變數。

如果要決定是否必須改寫 getResources 方法，請考慮下列事項：

- 如果您是根據預先定義的資訊來源（例如指令環境定義）來衍生資源，就不需要改寫 getResources 方法。例如，WebSphere Commerce UserRegistrationUpdate 指令會從指令環境定義中衍生使用者的 ID。在此情況下，使用者已經被授權處理本身的登錄資訊，所以 getResources 方法並不需要改寫。
- 如果您的指令會隨機指定新的資源（而這個資源是專用的），就必須改寫 getResources 方法。例如，WebSphere Commerce OrderItemUpdate 指令會採用訂單 ID 作爲輸入參數。在此情況下，當將訂單資源實例化時，您並不知道使用者是否有權對該特定資源採取動作。在此情況下，就會改寫 getResources 方法。

下列是 getResources 方法的範例：

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {

   if (resources == null) {
      OrderAccessBean orderAB = new OrderAccessBean();
      orderAB.setInitKey_orderId(getOrderId().toString());
      resouces = new AccessVector(orderAB);
      }
   return resources;
}
```

以 OrderItemUpdate 指令爲例。這個指令的 getResources 方法在更新現有的訂單時，會傳回一或多個「訂單」物件（可保護的）。由於未指定動作，會將動作預設爲 OrderItemUpdate 指令的介面。

getResources 方法所傳回的資源可能有多個。如果發生此情況，當執行動作時，必須能夠找到容許使用者存取所有指定資源的原則。假設使用者有權存取三個資源中的兩個資源，則動作可能無法繼續（三個資源都是必要的）。

如果您需要檢查控制程式指令中的其他參數或解析其中的參數，您可以使用 validateParameters() 方法。這個方法是選用的。

其他的資源層次檢查

當呼叫控制程式指令中的 getResources 方法時，不一定都會判斷所有需要保護的資源。

必要時，作業指令也可實作 getResources 方法，以傳回指令所能執行的資源清單。

另一種呼叫資源層次檢查的方法是，使用 checkIsAllowed(Object resource, String action) 方法直接呼叫存取控制原則管理程式。此方法適用於任何延伸自 com.ibm.commerce.command.AbstractECTargetableCommand 類別的類別。舉例來說，下列類別是延伸自 AbstractECTargetableCommand 類別：

- com.ibm.commerce.command.ControllerCommandImpl
- com.ibm.commerce.command.DataBeanCommandImpl

checkIsAllowed 方法亦適用於延伸自 com.ibm.commerce.command.AbstractECCommand 類別的類別。舉例來說，下列類別延伸自 AbstractECCommand 類別：

- com.ibm.commerce.command.TaskCommandImpl

下面顯示 checkIsAllowed 方法的宣告方式：

```
void checkIsAllowed(Object resource, String action)
   throws ECException
```

如果現行使用者不允許對指定資源執行指定的動作，則此方法會擲出 ECApplicationException。如果授與存取權，則方法只是返回。

“create” 指令的存取控制

由於在指令中 getResources 方法是在 performExecute 方法之前呼叫，您必須針對尚未建立之資源的存取控制採取不同的方式。舉例來說，假設您有 WidgetAddCmd，getResources 方法無法傳回即將建立的資源。在此情況下，

getResources 方法應該會傳回新資源的配置區。比方說，如果正在建立訂單，就會在商店資源內部執行；並且在組織資源內建立一個新使用者。

指令層次之存取控制的預設實作

在指令層次的存取控制方面，getOwner() 方法的預設實作會傳回商店擁有者的 memberId（如果有指定 storeId 的話）。如果未指定 storeId，則會傳回根組織的 memberId（memberId = -2001）。

就 getResources() 方法的預設實作而言，是傳回 null。

就 validateParameters() 的預設實作而言，則不執行任何動作。

在檢視畫面中實作存取控制原則

檢視畫面的資源層次存取控制是由資料 Bean 管理程式所執行。在下列情況下將會呼叫資料 Bean 管理程式：

1. 當 JSP 範本含有 <useBean> 標籤，且屬性清單中沒有資料 Bean 時。
2. 當 JSP 範本含有下列的啓動方法時：

```
DataBeanManager.activate(xyzDatabean, request);
```

註: 任何必須保護的資料 Bean（不論直接或間接）皆必須實作 Delegator 介面。凡受直接保護的資料 Bean 將委任給自己，因而亦必須實作「可保護」介面。而受間接保護的資料 Bean 則應委任給實作「可保護」介面的資料 Bean。

下列情況下將會略過存取控制檢查（不過不建議您如此做）：

1. 如果 JSP 範本直接呼叫存取 Bean，而非使用資料 Bean 時。
2. 如果 JSP 範本直接呼叫資料 Bean 的 populate() 方法時。

如果控制程式指令的結果會轉遞至檢視畫面（使用 ForwardViewCommand），則不會對檢視畫面執行指令層次的存取控制。再者，如果控制程式指令將移入資料的資料 Bean（用於檢視畫面中）置於回應內容的屬性清單中，並轉遞給檢視畫面，則 JSP 範本可直接存取資料，而不需透過資料 Bean 管理程式。不過 JSP 範本中得使用 <useBean> 標籤。此方法可讓 JSP 範本更有效率，這是因爲它可針對使用者透過控制程式指令而有權存取的資源（資料 Bean），略過對這些資源冗餘的資源層次的存取控制檢查。

修改現有 WebSphere Commerce 資源的存取控制

本節提供許多資訊來引導您修改現有 WebSphere Commerce 資源的存取控制。尤其是將會檢閱下列實務內容：

- 新增關係是已經由存取控制保護的現有 WebSphere Commerce Entity Bean。

- 新增存取控制保護至*尚未*受到存取控制保護的現有 WebSphere Commerce Entity Bean。
- 瞭解在延伸現有控制程式指令時對存取控制的含意。

新增關係至現有 WebSphere Commerce Entity Bean

實作「可保護的」介面的 WebSphere Commerce Entity Bean 已經受到存取控制的保護。存取控制需求的條款是取決於在 WebSphere Commerce 的 out-of-the-box 特性與功能中使用 Bean 的方式而定。有時候，您可能需要新增額外的關係到這類 Bean 的存取控制中。比方說，如果您在某些自訂程式碼中使用現有的 Bean，或者您修改現有的 WebSphere Commerce 公用 Entity Bean，您可能需要新增額外的關係到 Bean 中。

下列清單提供新增關係至已經由存取控制保護的現有 WebSphere Commerce Entity Bean 的高階步驟：

1. 檢查 Entity Bean 的現有 fulfills 方法。它是位於 Bean 的存取輔助程式類別中。請勿修改這個類別，而僅用它來決定您是否要新增一或多個新關係到這個邏輯中，或者您是否需要改寫這個方法。例如，下面的 fulfills 方法會出現在 com.ibm.commerce.fulfillment.objsrc.FulfillmentCenterBeanAccessHelper 類別中：

```
public boolean fulfills(Object obj, Long member, String relationship)
   throws Exception {

 FulfillmentCenterBean bean = (FulfillmentCenterBean) obj;

 if ("ShippingArrangementOrganizationalEntity".
   equalsIgnoreCase(relationship))
 {
      FulfillmentJDBCHelperAccessBean ffmJDBCAB =
         new FulfillmentJDBCHelperAccessBean();
      int count = ffmJDBCAB.
         checkFulfillmentCenterByMemberIdAndFulfillmentCenterId(
         member,bean.getFulfillmentCenterId());
      if(count>0)
     return true;
 }
      return false;
}
```

2. 下一步是在 Bean 類別中建立一個新的 fulfills 方法。例如，您可以在 com.ibm.commerce.fulfillment.objects.FulfillmentBean.java 類別中建立一個新的 fulfills 方法。方法的宣告方式如下：

```
public boolean fulfills(Long member, String relationship)
   throws Exception {
      // 關係資訊的預留位置
 }
```

3. 如果您新增一個額外的關係到現有的關係中，方法中的第一行應該是從超類別來呼叫 fulfills 方法。然後，如果該方法傳回 false，則依照下列方式檢查您的新關係：

```
public boolean fulfills(Long member, String relationship)
   throws Exception {
      if (super.fulfills().equals(false))
      {
         // 檢查是否符合新關係
     return true;
      }

      return false;
 }
```

4. 如果您要取代整個原始實作的關係，就不應該呼叫 super.fulfills 方法，方式如下：

```
public boolean fulfills(Long member, String relationship)
   throws Exception {
         // 檢查是否符合新關係
      // 如果是，則傳回 true；

     // 如果關係不符，則傳回 false；
 }
```

5. 儲存變更。爲 Bean 重新產生已部署和 RMIC 程式碼，以及對應的存取 Bean。

新增存取控制至尚未受到保護的現有 WebSphere Commerce Entity Bean

如果您使用現有的 WebSphere Commerce Entity Bean，而您的應用程式需要由存取控制來保護 Bean，您可以新增這項保護。

下列清單提供將現有的 WebSphere Commerce Entity Bean 交由 WebSphere Commerce 存取控制系統來保護的高階步驟：

1. 開啓 *BeanName*.java 類別。這是遠端介面。修改它，使它延伸 com.ibm.commerce.security.Protectable 介面。
2. 如果您爲了套用存取控制原則而依照屬性來將資源分組，而不是依照其 Java 類別名稱，則 Bean 的遠端介面也必須延伸 com.ibm.commerce.grouping.Groupable 介面。
3. 將您的變更儲存到遠端介面中。
4. 開啓 *BeanName*Bean.java 類別，其中 *BeanName* 是您要新增存取控制保護的 Entity Bean 的名稱。
5. Enterprise Bean 類別會繼承 com.ibm.commerce.base.objects.ECEntityBean 中的下列方法的預設實作：
   - getOwner

- fulfills
- getGroupingAttributeValue

請改寫任何您要的方法。您至少必須改寫 getOwner 方法。有關這些方法的其他資訊，請參閱第 100 頁的『在 Enterprise Bean 中實作存取控制』。

6. 儲存變更。爲 Bean 重新產生已部署和 RMIC 程式碼，以及對應的存取 Bean。

瞭解延伸控制程式指令時的存取控制含意

根據 WebSphere Commerce 程式設計模型，您可以建立自己的現有控制程式指令實作。在此情況下，您可以建立一個新的實作類別，然後藉由更新指令登錄，將新的實作類別關聯到現有的介面。

執行此類延伸時，有三種與存取控制相關的可能含意：

1. 對 getResources 方法的影響。
2. 對指令層次的存取控制原則的影響
3. 對資源層次的存取控制原則的影響

後續各節將詳細說明上述每一點。

對 getResources 方法的影響

如果您要延伸現有的控制程式指令，就表示會執行指令現有的邏輯以及您的新自訂邏輯，您的新指令就會成爲現有指令的子類別。新實作的 performExecute 方法會呼叫超類別的 performExecute 方法。如果您的新指令沒有存取任何需要保護的新資源，就不必改寫 getResources 方法。然而，如果有存取新的可保護資源，新指令就應該實作本身的 getResources 方法，並且在這個方法中從超類別呼叫 getResources 方法，然後才實作您自己的 getResources 邏輯。

超類別的 getResources 方法所產生的結果應該儲存在類型 AccessVector 的專用實例變數中。然後本端 getResources 方法的結果應該附加到這個向量結尾。例如，新的實作類別應該包含與下面的虛擬程式碼類似的程式碼：

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {
   // 首先，從原始實作取得資源
   resources = super.getResources();

   // 然後，附加新資源

   ////////////////////////////////////////
   //  取得新資源                        //
   //  並附加至向量的邏輯。              //
```

```
    ////////////////////////////////////////
    return resources;
}
```

如果您不要延伸現有控制程式指令的邏輯，但是您要完全取代邏輯，就不應該在新實作中呼叫 super.performExecute 方法。然後您就不需要從自己的實作中呼叫 super.getResources 方法。相反的，您只需要根據需要來實作您自己的 getResources 方法。

對指令層次的存取控制原則的影響

控制程式的指令層次原則是由在指令資源上作業的 "Execute" 動作所組成。指令資源是由其介面名稱所指定。當您延伸現有指令時，指令仍然會實作原始的介面，因此，現有的指令層次原則仍足以維持先前的指令層次的存取控制。所以不需要任何變更。

對資源層次的存取控制原則的影響

如果您已經建立現有指令的新實作，並且將此實作關聯到該現有指令的介面，就不需要變更資源層次的存取控制原則。

相反的，如果您建立一個新的實作類別來延伸現有的實作，並且在這個類別中呼叫 super.performExecute 方法，而同時您也實作新的介面，就需要變更資源層次的存取控制原則。

在後面的情況，如果新指令實作 getResources 方法或者從基本指令繼承重要的 getResources 方法實作，就需要變更資源層次的原則。通常，資源層次的原則是由在商業物件資源上作業的指令動作所組成。由於動作只是指令介面的字串表示法，通常需要將新指令加到基本指令的動作群組中。然而，如果新指令只是傳回空值來改寫 getResources 方法的基本實作，就不需要爲這個新指令執行存取控制檢查。請注意，如果沒有小心進行這個步驟，可能會對懷有惡意的使用者開啓新指令。

在修改資源層次的存取控制原則時，高階步驟如下：

1. 在下列目錄中尋找 defaultAccessControlPolicies.xml 檔：
   - Studio *WCStudio_installdir*\Commerce\xml\policies\xml
2. 製作這個檔案的副本。例如，將檔案命名爲 *myDefaultAccessControlPolicies.xml*。
3. 在這個新檔案中，您必須將新的介面定義成新的動作。例如，您會新增下列項目：

```
<Action Name="yourNewInterface"
   CommandName="yourNewInterface">
   </Action>
```

其中 *yourNewInterface* 是新介面的名稱。

4. 接下來，您必須搜尋檔案，找出原始動作所屬的全部動作群組，然後加入您的新動作。
5. 如果新指令是在先前並未由基本指令指定的資源上作業，則必須同時變更對應存取控制原則的資源群組，以容納新的資源。
6. 一旦您完成對 XML 檔案的變更，就可以移除尚未修改的區段。請務必保留 <Policies> 標籤前面的文字。
7. 根據*WebSphere Commerce Security Guide*中的指示，將新的原則資訊載入資料庫中。

用於開發的範例存取控制原則

本節提供許多非常簡單的存取原則，讓您在開發環境中使用，以便快速測試新的資源。這些範例的設計並不是爲了在任何 WebSphere Commerce 正式作業環境中使用，因爲它們並沒有提供足夠的資源保護。

有關如何載入這些原則的資訊，請參閱 *WebSphere Commerce Security Guide*。

新檢視畫面的範例存取控制原則

如果您建立新的檢視畫面，就可以使用下列存取控制原則，這樣就能在開發環境中測試您的檢視畫面（針對您的環境來修改原則，然後使用 acpload 指令來載入它）：

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="YourNewView"
        CommandName="YourNewView">
   </Action>
   <ActionGroup Name="AllSiteUsersViews"
     OwnerID="RootOrganization">
      <ActionGroupAction Name="YourNewView"/>
   </ActionGroup>
</Policies>
```

其中 *YourNewView* 是新建立的檢視畫面的名稱。前面的存取控制原則會將新的檢視畫面新增至現有的 AllSiteUsersViews 動作群組中。這個原則可讓任何使用者存取新的檢視畫面。

新控制程式指令的範例指令層次存取控制原則

控制程式指令需要存取控制原則才能符合存取控制組織架構的需求。如果您建立新的控制程式指令，就會將指令的介面名稱指定成資源。您可以爲新指令修改下列 XML 片段，然後使用 acpload 指令來載入它：

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>

   <Action Name="ExecuteCommand"
         CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory"
         ResourceBeanClass="com.yourcompany.yourpackage.commands.
         YourControllerCmd">
        <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
     OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory" />
   </ResourceGroup>

</Policies>
```

其中：

- *com.yourcompany.yourpackage.commands* 代表您的包裝結構
- *YourControllerCmd* 代表您的新控制程式指令的名稱

舉例來說，下列 XML 檔案是用來爲本書指導教學中建立的新控制程式指令載入存取控制原則。

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="ExecuteCommand"
         CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.sample.commands.
          MyNewControllerCmdResourceCategory"
          ResourceBeanClass="com.ibm.commerce.sample.commands.
          MyNewControllerCmd">
         <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
```

```
     OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.ibm.commerce.sample.commands.
         MyNewControllerCmdResourceCategory" />
   </ResourceGroup>

</Policies>
```

新指令與 Enterprise Bean 的範例資源層次存取控制原則

下列 XML 檔是取自本手冊中的指導教學。它可以在您建立新的 Entity Bean 時，作爲存取控制需求的範本。在下面的檔案中，新 Entity Bean 稱爲 Bonus Bean，它會對應到 XBONUS 資料庫表格，並且供 MyNewControllerCmd 控制程式指令使用。在這個存取控制原則中，只有 Bonus Bean 物件的建立者才能對該物件執行 MyNewControllerCmd 動作。

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="MyNewControllerCmd"
      CommandName="com.ibm.commerce.sample.commands.MyNewControllerCmd">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.extension.objects.
       BonusResourceCategory"
       ResourceBeanClass="com.ibm.commerce.extension.objects.Bonus" >

         <ResourceAction Name="MyNewControllerCmd" />
   </ResourceCategory>

   <ActionGroup Name="MyNewControllerCmdActionGroup"
     OwnerID="RootOrganization">
   <ActionGroupAction Name="MyNewControllerCmd"/>
   </ActionGroup>

   <ResourceGroup Name="BonusResourceGroup"  OwnerID="RootOrganization" >
      <ResourceGroupResource Name="com.ibm.commerce.extension.objects.
            BonusResourceCategory" />
   </ResourceGroup>
   <Policy Name="AllUsersUpdateBonusResourceGroup"
      OwnerID="FashionFlowMemberId"
UserGroup="AllUsers"
      UserGroupOwner="RootOrganization"
      ActionGroupName="MyNewControllerCmdActionGroup"
      ResourceGroupName="BonusResourceGroup"
      RelationName="creator"
      PolicyType="groupableStandard">
</Policy>

   <PolicyGroup Name="ManagementAndAdministrationPolicyGroup"
     OwnerID="RootOrganization">
   <!-- Define policies in this policy group -->
   <PolicyGroupPolicy  Name="AllUsersUpdateBonusResourceGroup"
```

```
            PolicyOwnerID="FashionFlowMemberId" />

   </PolicyGroup>

</Policies>
```

其中 *FashionFlowMemberId* 是使用新資源的商店的成員 ID。

在前面的存取控制原則中，控制程式指令的介面名稱是指定成動作，而沒有使用其套件名稱來完整定義它。如果您的應用程式有多個介面使用相同的名稱，在將它們指定成存取控制原則中的動作時，您必須使用其套件名稱來完整定義它們。舉例來說，如果介面名稱不明確，則需要依照下列方式來變更前面的存取控制原則（請注意，此處僅顯示變更的程式行，而修改的部分是以粗體字顯示）：

```
<Action Name="com.ibm.commerce.sample.commands.MyNewControllerCmd"
      CommandName="com.ibm.commerce.sample.commands.MyNewControllerCmd">
.
.
.
<ResourceAction Name="com.ibm.commerce.sample.commands.MyNewControllerCmd" />
.
.
.
<ActionGroupAction Name="com.ibm.commerce.sample.commands.MyNewControllerCmd"/>
```

第 5 章 錯誤的處理與訊息

指令錯誤處理

WebSphere Commerce 採用一種定義明確的指令錯誤處理組織架構，可方便您在自訂程式碼中使用。以設計來說，此種組織架構是在支援多種文化型商店下處理錯誤。下列各節說明指令所能擲出的異常狀況類型，如何處理異常狀況，如何儲存與利用訊息文字，如何記載異常狀況，以及如何在您自己的指令中使用本產品提供的組織架構。

異常狀況類型

指令可擲出下列一種異常狀況：

ECApplicationException

如果錯誤與使用者有關，則會擲出此異常狀況。舉例來說，當使用者輸入的參數無效時，則會擲出 ECApplicationException。一旦擲出此異常狀況，Web 控制程式便不再重試指令，即使該指令被設爲可重試型指令。

ECSystemException

只要偵測到執行期間異常狀況或 WebSphere Commerce 架構錯誤，即會擲出此異常狀況。像空指標異常狀況、交易回復異常狀況，皆屬於此種異常狀況類型。一旦擲出此種異常狀況類型，只要該指令屬於可重試型，且該異常狀況是因資料庫死鎖或資料庫回復所造成，則 Web 控制程式即會重試該指令。

上述兩種異常狀況是由 ECException 類別延伸而來的類別，而可在 com.ibm.commerce.exception 套件中找到。

如果要擲出這些異常狀況之一，則必須指定下列資訊：

- 錯誤檢視畫面名稱
  Web 控制程式會在 VIEWREG 表格中查看此名稱。
- ECMessage 物件
  此值會對應至內容檔中所含的訊息文字。
- 錯誤參數
  這些「名稱-值」對用以換成輸入到錯誤訊息中的資訊。舉例來說，某訊息中含有一個參數用以保留擲出異常狀況的方法名稱。當擲出異常狀況時會設定此參數，而當記載錯誤訊息時，日誌檔中將含有實際的方法名稱。

- 錯誤資料
  這些是選用屬性，可透過錯誤資料 Bean 提供給 JSP 範本使用。

異常狀況的處理已與日誌記載系統密切整合。一旦擲出系統異常狀況，即會自動記載。

錯誤訊息內容檔

爲了簡化錯誤訊息的維護，並支援多種語言型商店，會將錯誤訊息的文字儲存在內容檔中。WebSphere Commerce 訊息文字儲存在 ecServerMessages_*XX_XX*.properties 檔中，其中 *_XX_XX* 爲語言環境指示碼（例如 _zh_TW）。

指令環境定義會傳回一個識別碼，指出用戶端所用的語言。當需要訊息時，Web 控制程式會根據語言識別碼判斷出所要使用的內容檔。

在 ecServerMessages*XX_XX*.properties 檔中定義了下列兩種訊息類型：使用者訊息與系統訊息。使用者訊息會顯示在客戶的瀏覽器中。而系統與使用者訊息皆會被自動擷取到訊息日誌中。

當擲出錯誤時，其中一個必要參數爲訊息物件。就 ECSystemExceptions 而言，訊息物件中必須含有兩個關鍵字，一個爲系統訊息的，另一個爲使用者訊息的。就 ECApplicationExceptions 而言，訊息物件中則含有使用者訊息的關鍵字（未用到系統訊息）。

所有系統訊息皆爲預先定義的。您無法建立自己的系統訊息。因此當自訂的程式碼擲出 ECSystemException 時，其必須指出其中一個預先定義的系統訊息訊息關鍵字。自訂的使用者訊息則可讓您建立。新使用者訊息必須儲存在個別的內容檔中。

異常狀況的處理流程

下圖顯示發生異常狀況時的資訊流程。而各步驟的說明如下。

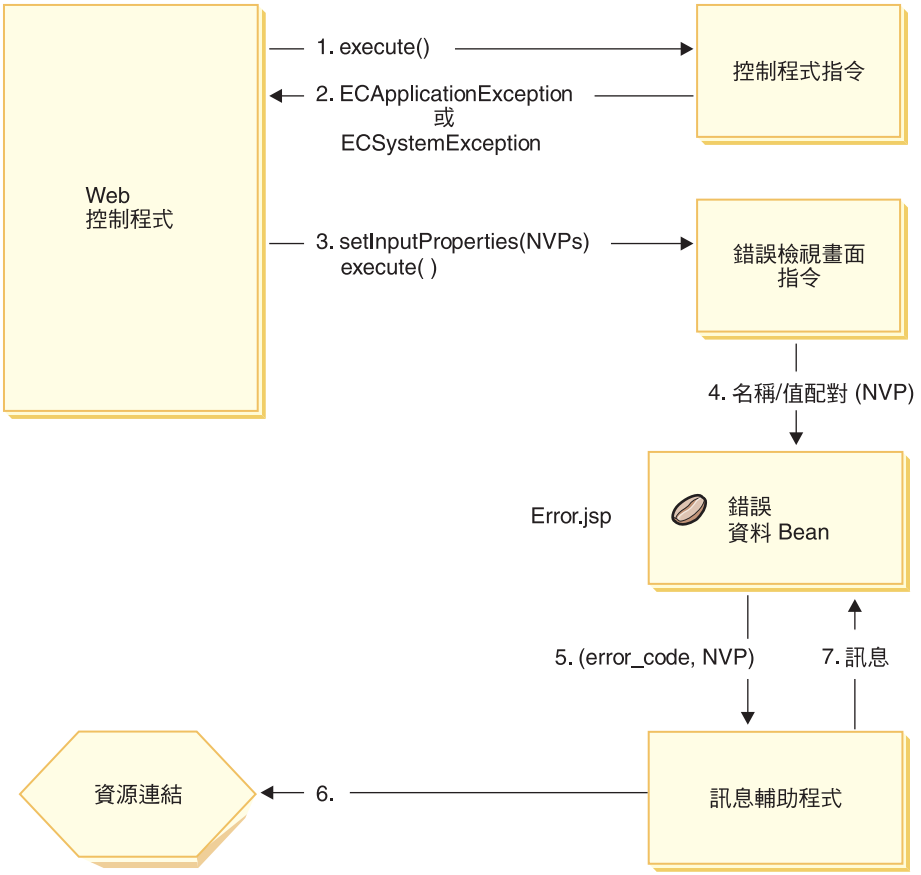

圖 24.

1. Web 控制程式呼叫控制程式指令。
2. 指令擲出異常狀況，而被 Web 控制程式捕捉到。此異常狀況可以是 ECApplicationException 或 ECSystemException。異常狀況物件中含有下列資訊：
   - 錯誤檢視畫面名稱
   - ECMessage 物件
   - 錯誤參數
   - （選用）錯誤資料
3. Web 控制程式會判斷 VIEWREG 表格中的錯誤檢視畫面名稱，並呼叫指定的錯誤檢視畫面指令。 Web 控制程式在呼叫指令時，會撰寫一組出自 ECException 物件的內容，並使用檢視畫面指令的 setInputProperties 方法將之設定爲檢視畫面指令。

4. 檢視畫面指令呼叫錯誤 JSP 範本（在本範例中爲 Error.jsp），並將「名稱-值」對傳給 JSP 範本。
5. ErrorDataBean 將錯誤參數傳給訊息 Helper 物件。
6. 訊息 Helper 物件從適當的內容檔中取得必要的訊息（採用訊息物件與錯誤參數）。
7. 錯誤資料 Bean 將訊息傳給 JSP 範本。

自訂程式碼中的異常狀況處理

在您建立新指令時，請記得包含適當的異常狀況處理。您可以在捕捉異常狀況時指定必要資訊，以善用 WebSphere Commerce 中的錯誤處理與傳訊組織架構。

在您撰寫自己的異常狀況處理邏輯時，將涉及下列步驟：

1. 捕捉需要特殊處理之指令中的異常狀況。
2. 根據捕捉到的異常狀況類型，來建構 ECApplicationException 或 ECSystemException。
3. 如果 ECApplicationException 使用新訊息，請在新內容檔中定義該訊息。

捕捉與建構異常狀況

爲了描述前兩個步驟，下列的程式碼片段顯示捕捉指令中之系統異常狀況的範例：

```
try {
// 您的商業邏輯
}
catch(FinderException e) {
     throw new ECSystemException (ECMessage._ERR_FINDER_EXCEPTION,
          className, methodName, new Object [] {e.toString()}, e);
}
```

上述 _ERR_FINDER_EXCEPTION ECMessage 物件的定義如下：

```
public static final ECMessage _ERR_FINDER_EXCEPTION =
new ECMessage (ECMessageSeverity.ERROR, ECMessageType.SYSTEM,
     ECMessageKey._ERR_FINDER_EXCEPTION);
```

_ERR_FINDER_EXCEPTION 訊息文字是定義於 ecServerMessages_*xx_XX*.properties 檔中（其中 *_xx_XX* 爲語言環境指示碼，像是 *_zh_TW*）例如：

```
_ERR_FINDER_EXCEPTION  =
在處理期間發生如下的搜尋器異常狀況："{0}"。
```

在捕捉系統異常狀況時，有一組預先定義的訊息可供使用。下表將說明這些訊息：

訊息物件	說明
_ERR_FINDER_EXCEPTION	當 EJB finder 方法呼叫傳回錯誤時擲出。
_ERR_REMOTE_EXCEPTION	當 EJB remote 方法呼叫傳回錯誤時擲出。
_ERR_CREATE_EXCEPTION	當建立 EJB 實例而發生錯誤時擲出。
_ERR_NAMING_EXCEPTION	當名稱伺服器傳回錯誤時擲出。
_ERR_GENERIC	當發生非預期的系統錯誤時擲出。例如：空指標異常狀況。

在捕捉應用程式異常狀況時，您可以使用指定於適當 ecServerMessages_xx_xx.properties 檔中的現有訊息，或建立一個新訊息（儲存在新內容檔中）。只要先前已指定，即不得再修改任何 ecServerMessages_xx_XX.properties 檔。

下列的程式碼片段顯示捕捉指令中之應用程式異常狀況的範例：

```
try {
// 您的商業邏輯

}
// 捕捉一些新的應用程式異常狀況類型
catch(//新的異常狀況)
 {
     throw new ECApplicationException (MyMessages._ERR_CUSTOMER_INVALID,
          className, methodName, errorTaskName, someNVPs);
}
```

上述 _ERR_CUSTOMER_INVALID ECMessage 物件的定義如下：

```
public static final ECMessage _ERR_CUSTOMER_INVALID =
     new ECMessage (ECMessageSeverity.ERROR, ECMessageType.USER,
     MyMessagesKey._ERR_CUSTOMER_INVALID, "ecCustomerMessages");
```

在您建構新使用者訊息時，應按如下爲這些訊息指定 USER 類型：

ECMessageType.USER

_ERR_CUSTOMER_INVALID 訊息的文字則儲存在 ecCustomerMessages.properties 檔中。此檔案必須常駐在位於類別路徑的目錄中。文字的定義如下：

```
_ERR_CUSTOMER_INVALID = 無效 ID "{0}"
```

建立訊息

如果您的指令擲出採用新訊息的 ECApplicationException，您必須建立此新訊息。建立新訊息將涉及下列步驟：

1. 建立一個內含訊息關鍵字的新類別。
2. 建立一個內含 ECMessage 物件的新類別。
3. 建立資源連結。

下列各節將進一步說明每一個步驟。

建立訊息關鍵字的類別

建立新使用者訊息的第一個步驟是建立一個內含新訊息關鍵字的類別。訊息關鍵字爲一種唯一的指示碼，供日誌記載服務程式用以在資源連結中找到對應的訊息文字。這個新類別必須建立在您自己的套件中，並且儲存在 WebSphereCommerceServerExtensionsLogic 專案中。

舉例來說，假設您將建立新類別 MyNewMessages 以及 MyMessageKeys，其中含有 _ERR_CUSTOMER 與 _ERR_CUSTOMER_INVALID_ID 訊息關鍵字，且您將此類別置於 com.mycompany.messages 套件中。 在此情況下，類別的定義類似如下：

```
public class MyMessageKeys
{
    public static String _ERR_CUSTOMER="_ERR_CUSTOMER";
    public static String _ERR_CUSTOMER_INVALID_ID="_ERR_CUSTOMER_INVALID_ID";
}
```

假設訊息關鍵字的 String 封套容許編譯器檢查其眞確與否。

建立 ECMessage 物件的類別

請在您建立訊息關鍵字之類別的同一套件中，另建一個內含 ECMessage 物件的類別。ECMessage 類別用以定義訊息物件的結構。 它可用來擷取與留存與語言環境有關的文字訊息。

訊息物件具有下列屬性：嚴重程度、類型、關鍵字、資源連結與相關聯的資源連結。此類別有數個建構子方法。 有關完整詳述，請參閱 WebSphere Commerce 線上說明中的「參照」區段。

延續範例，您何以依照下列的說明在 com.mycompany.messages 套件中建立一個新類別 MyMessages：

```
import com.ibm.commerce.ras.*;

public class MyMessages
    {
        static String myResourceBundle = "ecCustomerMessages";

        public static ECMessage _ERR_CUSTOMER = new ECMessage
             (ECMessageSeverity.ERROR,ECMessageType.USER,
                  MyMessageKeys._ERR_CUSTOMER, myResourceBundle);
        public static ECMessage _ERR_CUSTOMER_INVALID_ID = new ECMessage
             (ECMessageSeverity.ERROR, ECMessageType.USER,
```

```
                MyMessageKeys._ERR_CUSTOMER_INVALID_ID,
                myResourceBundle);

    }
```

在上述的程式碼片段中，在建立 ECMessage 物件時需用到 import 陳述式。MyMessage._ERR_CUSTOMER 物件爲一種使用者訊息，其嚴重程度爲 ERROR。MyMessageKeys._ERR_CUSTOMER 是供 WebSphere Commerce 日誌記載服務程式用以在 *ecCustomerMessages* 內容檔中找到訊息文字。

建立使用者訊息的資源連結

您必須建立新資源連結，以便儲存訊息關鍵字與對應的訊息文字。此資源連結可當成 Java 物件或內容檔來實作。建議您採用內容檔，這是因爲轉換與維護都較爲容易。內容檔是用於 WebSphere Commerce 訊息上。

爲了延續 MyNewMessages 範例，您必須建立一個文字檔 ecCustomerMessages.properties。如果訊息要提供給單一商店 Servlet 使用，請將這個檔案置於下列目錄中：

Studio *workspace_dir*\Stores\Web Content\WEB-INF\classes*storeDir*

其中 *storeDir* 是您的商店名稱。

如果訊息要提供給 WebSphere Commerce Accelerator 使用，請將這個檔案置於下列目錄中：

Studio *workspace_dir*\CommerceAccelerator\Web Content\WEB-INF\classes

如果訊息要提供給管理主控台使用，請將這個檔案置於下列目錄中：

Studio *workspace_dir*\SiteAdministration\Web Content\WEB-INF\classes

Business 如果訊息要提供給「組織管理主控台」使用，請將這個檔案置於下列目錄中：

Studio *workspace_dir*\OrganizationAdministration\Web Content\WEB-INF\classes

如果訊息要提供給企業應用程式中的任何 Servlet 廣泛使用，請將這個檔案置於下列目錄中：

Studio *workspace_dir*\WebSphereCommerceServer\properties

前述的目錄是在開發環境的環境定義中指定。一旦完成該環境的測試之後，請參閱第 185 頁的第 9 章, 『部署明細』，以取得關於部署至目標 WebSphere Commerce Server 的資訊。

由於內容檔含有「訊息關鍵字/對應的訊息文字」對，ecCustomerMessages.properties 檔將含有下列各行：

```
_ERR_CUSTOMER_MESSAGE = The customer message "{0}".
_ERR_CUSTOMER_INVALID_ID = Invalid ID "{0}".
```

執行流程追蹤

當您需要透過 Commerce 應用程式來追蹤執行流程時，應該要使用 WebSphere Application Server JRas 機能。這是可供應用程式使用的訊息日誌記載以及診斷追蹤 API。

有關如何在您的自訂程式碼中使用這個機能的資訊，請參閱 WebSphere Application Server InfoCenter。

有關在開發環境中配置元件追蹤的資訊，請參閱第 357 頁的附錄 A, 『在 WebSphere Commerce Studio 中配置 WebSphere Commerce 元件追蹤』。

JSP 範本錯誤處理

您可採用下列方式來處理 JSP 範本的錯誤：

- 從頁面進行錯誤處理
  如果 JSP 需要較複雜的錯誤處理與復原，則可將檔案撰寫成直接從資料 Bean 來處理錯誤。 視如何啓動資料 Bean 而定，JSP 檔可攫取資料 Bean 擲出的異常狀況，或者可查看各資料 Bean 中有無設定錯誤碼。接著 JSP 檔可根據所收到的錯誤以採取適當的復原動作。 請注意，JSP 檔可使用下列錯誤處理範圍內的任何組合。
- 頁面層次的錯誤 JSP
  JSP 檔可透過 JSP 錯誤標籤指定自己的預設錯誤 JSP 範本，以便在本身發生異常狀況時出現。這可讓 JSP 程式指定自己的錯誤處理方式。若 JSP 檔未指定 JSP 錯誤標籤，則錯誤將屬於應用程式層次的 JSP 錯誤範本。在頁面層次下的錯誤 JSP 中，它必須呼叫 JSP helper 類別 (com.ibm.server.JSPHelper)，以回復目前的交易。
- 應用程式層次的錯誤 JSP
  WebSphere 下的應用程式可指定一個預設的錯誤 JSP 範本，以便在其 Servlet 或 JSP 檔中發生異常狀況時出現。應用程式層次的錯誤 JSP 範本可當成商場層次或商店層次的（單一商店模式方面）錯誤處理程式。在應用程式層次的錯誤 JSP 範本中，必須呼叫 Servlet helper 類別，以回復目前的交易。這是因爲 Web 控制程式不會出現在回復交易的執行路徑中。只要可能的話，您應盡量使用前兩種 JSP 錯誤處理類型。而應用程式層次錯誤處理策略應只在必要時才使用。

第 6 章 指令的實作

本章提供如何撰寫新控制程式、作業與資料 Bean 指令的相關資訊。此外亦說明如何延伸現有的控制程式、作業與資料 Bean 指令。

註: Business 本章不說明商業原則指令。有關商業原則指令資訊，請參閱第 145 頁的第 7 章, 『交易協定與商業原則（Business Edition）』。

新指令 - 簡介

WebSphere Commerce 程式設計模型定義了四種指令類型：控制程式、作業、檢視畫面與資料 Bean 指令。在您爲電子商務應用程式建立新商業邏輯時，您或許想建立新控制程式、作業與資料 Bean 指令。您應該不需要建立新檢視畫面指令。本節後面會進一步說明檢視畫面指令。

新指令必須實作其對應的介面（理應從現有介面延伸而來）。如果要簡化指令的撰寫，WebSphere Commerce 會針對每一種指令類型各提供一種抽象的實作類別。新指令應從這些類別延伸而來。

如同概觀所述，下表說明新指令應從哪個實作類別延伸而來，以及其應實作哪一種介面：

指令類型	範例指令名稱	延伸自	實作範例介面
控制程式指令	MyControllerCmdImpl	com.ibm.commerce.command.ControllerCommandImpl	MyControllerCmd
作業指令	MyTaskCmdImpl	com.ibm.commerce.command.TaskCommandImpl	MyTaskCmd
資料 Bean 指令	MyDataBeanCmdImpl	com.ibm.commerce.command.DataBeanCommandImpl	MyDataBean

註: 實作類別名稱中的空格僅爲了方便閱讀用。

下圖說明新控制程式指令（採用現有的抽象實作類別與介面）之介面與實作類別間的關係。抽象類別與介面皆可在 com.ibm.commerce.command 套件中找到。

新控制程式指令

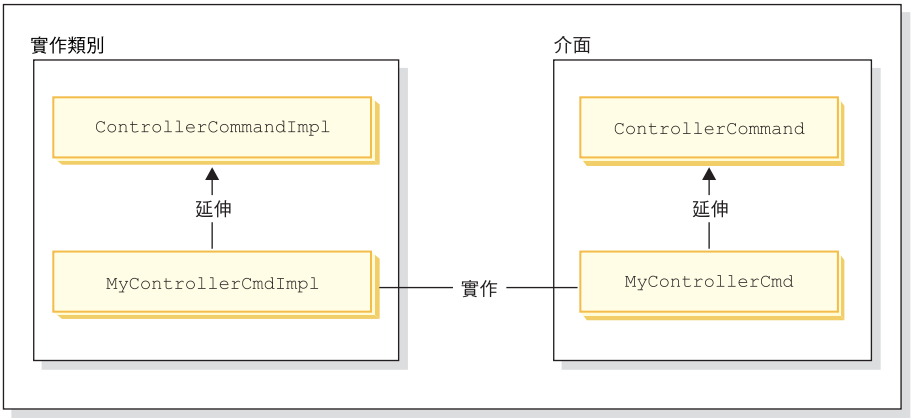

圖 25.

下圖說明新作業指令（採用現有的抽象實作類別與介面）之介面與實作類別間的關係。抽象類別與介面皆可在 com.ibm.commerce.command 套件中找到。

新作業指令

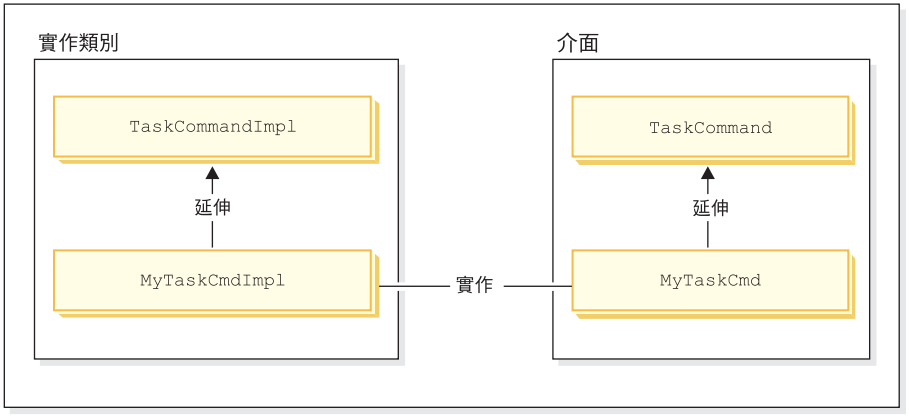

圖 26.

下圖說明新資料 Bean 指令（採用現有的抽象實作類別與介面）之介面與實作類別間的關係。抽象類別與介面皆可在 com.ibm.commerce.command 套件中找到。

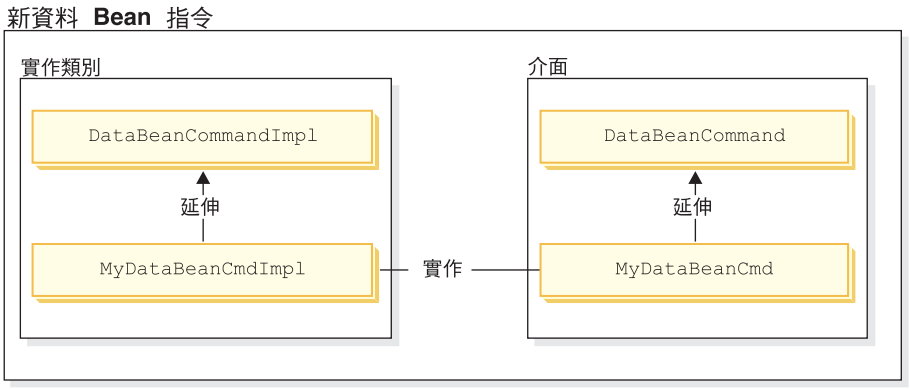

圖 27.

檢視畫面指令有下列兩種主要功能：設定回應的格式以及傳送回應給用戶端。產品中會提供許多的通用檢視畫面指令，供您以不同的通訊協定將回應給用戶端。一般而言，格式設定是由呼叫 JSP 範本的檢視畫面指令所處理。舉例來說，RedirectViewCommand 檢視畫面指令會將用戶端導向到某個 URL，以取得回應（接著由指定的 JSP 範本來設定回應的格式）。 ForwardViewCommand 檢視畫面指令會將要求轉遞給 JSP 範本，以設定格式，並將頁面顯示給用戶端。

透過此種檢視畫面指令模型，您可以藉由建立新 JSP 範本來建立新的*檢視畫面*（傳給用戶端的回應）。不過，JSP 範本應由現行檢視畫面指令所呼叫。

組裝自訂程式碼

在您建立自訂程式碼時，必須遵循特定程式碼的組織結構。一般而言，自訂程式碼是保存在爲您的自訂程式碼預先定義的 WebSphere Commerce 工作區的專案中。目前提供兩個預先定義的專案： WebSphereCommerceServerExtensionsLogic 專案和 WebSphereCommerceServerExtensionsData 專案。第一個專案是供指令與資料 Bean 邏輯使用，第二個是供您建立的任何 Enterprise Bean 使用。

在您建立新指令時，您必須將之放在其名稱適合您商業需求的套件中。亦即，假設某些指令適用於特定商店，則您應將之組裝於一個專屬於該商店的套件中。如果這些指令適用於多家商店，請適當組裝之。舉例來說，假設您有下列套件：

- `com.bigbusiness.storeA.commands`
- `com.bigbusiness.storeB.commands`
- `com.bigbusiness.commands`

前述的套件結構容許商店層次下之商業邏輯間的差異。

在建立新資料 Bean 時，必須將它們保存在與指令邏輯不同的套件中，然而，這個套件應該保存在用來儲存指令套件的專案內 (WebSphereCommerceServerExtensionsLogic)。從前面的範例來看，您應該將 com.bigbusiness.databeans 套件放在 WebSphereCommerceServerExtensionsLogic 專案內。

在建立新 Entity Bean 時，必須將它們儲存在 WebSphereCommerceServerExtensionsData 專案中。因此，您可能會有一個包含 com.bigbusiness.objects 套件的 WebSphereCommerceServerExtensionsData 專案。

爲了部署程式碼，此套件策略爲必要的。

指令環境定義

指令可使用指令環境定義取得 Web 控制程式中的資訊。可用資訊的範例包括使用者的 ID、使用者物件、語言 ID 以及商店 ID。

在您撰寫指令時，您可以藉由呼叫指令超類別的 getCommandContext() 方法，以存取指令環境定義。當 Web 控制程式呼叫指令時，指令環境定義是設定成控制程式指令。控制程式指令應將指令環境定義傳達給在處理程序期間任何會呼叫到的作業或控制程式指令。指令可從指令環境定義中取得下列的關鍵資訊：

getUserId() 與 getUser()

取得現行使用者 ID 或使用者物件。現行階段作業的使用者 ID 會儲存在階段作業環境定義中。階段作業環境定義可透過下列兩種方式之一延伸：採用 WebSphere Commerce cookie 或 WebSphere Application Server 持續性階段作業物件。指令環境定義可讓指令避開階段作業管理的複雜性。

getStoreId()、getStore() 與 getStore(storeId)

取得與現行要求有關的商店。Web 控制程式會傳回 URL 中的商店 ID。如果 URL 中未指定商店 ID，則可從在前次要求中所儲存的階段作業物件中擷取。 WebSphere Commerce 執行期間環境會維護一組經常存取的物件。舉例來說，它會維護一組商店物件。指令應一律從指令環境定義中取得商店物件，以善用 Web 控制程式中的物件快取特性。您可以藉由呼叫 getStore() 方法取得現行商店，或藉由呼叫 getStore(storeId) 方法從指令環境定義中取得特定的商店物件。

getLanguageId()

傳回現行要求應使用的語言 ID。 Web 控制程式會實作「全球化組織架構」。此種組織架構背後的概念是決定使用者所偏好以及商店所支援的語言。如果 URL 中含有語言 ID，Web 控制程式會判斷商店是否支援此語言，若有支援，則會成爲 getLanguageId() 方法將傳回的語言 ID。如果

URL 中沒有語言 ID，則 Web 控制程式會瀏覽決策樹，以判斷現行階段作業物件中或使用者登錄的喜好設定中是否有商店支援的語言 ID，或者是傳回商店的預設語言 ID。

getCurrency()

傳回現行要求要使用的貨幣。由於貨幣爲「全球化組織架構」的一部分，此方法背後的邏輯和 getLanguageId() 方法的邏輯類似。

getCurrentTradingAgreements() 與 getTradingAgreement(tradingAgreementId)

傳回現行階段作業所用的交易協定集。此交易協定集可以是授與使用者資格的所有交易協定，或者是 ContractSetInSession 指令所定義的一個子集。指令應固定從指令環境定義中取得交易協定物件，以善用 Web 控制程式中的物件快取特性。您可以藉由呼叫 getCurrentTradingAgreements() 方法取得現行交易協定，或藉由呼叫 getTradingAgreement(tradingAgreementId) 方法從指令環境定義中取得特定的交易協定物件。

指令環境定義應當成一個唯讀物件。您不應呼叫其 setter 方法。setter 方法保留供 WebSphere Commerce 執行期間環境使用，在未來版次中可能會更換之。

有關指令環境定義 API（應用程式設計介面）的完整明細，請參閱 WebSphere Commerce 正式作業與開發線上說明中的「參照」主題。

URL 指令的環境定義資訊的暫時變更

您可以改寫某些指令環境定義資訊以及在另一家商店的環境定義內執行 URL 指令，或者代表另一個使用者來執行。URL 指令具有下列 URL 輸入參數，可容許在指令環境定義中使用這個暫時切換：

- forStoreId
- forUser
- forUserId

forStoreId URL 輸入參數可讓您指定此特定 URL 要求要使用的商店 ID。這個 ID 在生效之後，會將指令環境定義中的 storeId 值暫時變更爲指定的商店的 ID，但是這個變更只有在 URL 指令期間有效。

forUser 和 forUserId 這兩個 URL 輸入參數都能讓您爲指定的使用者指定要執行的指令，即使目前登入的使用者可能不同。當客戶服務代表需要協助客戶時，這一點特別有用。例如，客戶服務代表可以代表客戶來更新該客戶的地址資訊，方法是利用 URL 輸入參數來指定客戶的使用者名稱或使用者 ID。對於使用者資訊的變更，只有在指定 URL 要求的期間有效。

新控制程式指令

一如前述，新控制程式指令應從抽象控制程式指令類別 (com.ibm.commerce.command.ControllerCommandImpl) 延伸而來。在您撰寫新控制程式指令時，應改寫下列方法（出自抽象類別）：

- isGeneric()
- isRetriable()
- setRequestProperties(com.ibm.commerce.datatype.TypedProperty reqParms)
- validateParameters()
- getResources()
- performExecute()

下列各節將進一步說明上述這些方法。

isGeneric 方法

在標準的 WebSphere Commerce 實作中會有多種使用者類型。包括：一般、訪客與已登錄使用者。在已登錄使用者的分組中，還分爲客戶與管理者。

一般使用者會有一個用於整個系統中的共通使用者 ID。這個共通使用者 ID 支援網站上的一般瀏覽，而在此方式下，可將系統資源的使用降至最少。 使用此一共通使用者 ID 進行一般性瀏覽最有效率，這是因爲 Web 控制程式不需要爲可供一般使用者呼叫的指令擷取使用者物件。

isGeneric 方法會傳回一個 boolean 值，用來指出是否可供一般使用者呼叫。控制程式指令之超類別的 isGeneric 方法會將此值設爲 `false`（意指呼叫端必須是一位已登錄使用者或來賓使用者）。 如果您的新控制程式指令可供一般使用者呼叫，請改寫此方法以傳回 `true`。

如果您的新指令不會提取或建立使用者的相關資源，您應改寫此方法以傳回 `true`。像 ProductDisplay 指令即爲可供一般使用者呼叫的指令。在容許任何使用者檢視產品方面應該愼重。像 OrderItemAdd 指令即爲使用者必須是來賓或已登錄使用者（因而 isGeneric 所傳回的是 `false`）才能呼叫的指令。

當 isGeneric 傳回 `true` 值時，Web 控制程式不會爲現行階段作業建立新使用者物件。因此，可供一般使用者呼叫的指令執行速度較快，這是因爲 Web 控制程式不必擷取使用者物件。

下列語法顯示如何使用此方法讓一般使用者能呼叫指令：

```
public boolean isGeneric()
{
     return true;
}
```

isRetriable 方法

isRetriable 方法會傳回一個 boolean 值，用來指出一旦發生交易回復異常狀況，是否要重試該指令。新控制程式指令之超類別的 isRetriable 方法所傳回的是 false 值。如果您的指令可在發生交易回復異常狀況時重試，您應改寫此方法以傳回 true 值。

像 OrderProcess 指令即爲一個在發生交易異常狀況時不應重試的指令。此指令會呼叫第三方付款授權程序。此指令無法重試，這是因爲授權無法撤銷。而可以重試的指令如 ProductDisplay 指令。

下列語法說明如何讓指令可在發生交易回復異常狀況時重試：

```
public boolean isRetriable()
{
     return true;
}
```

setRequestProperties 方法

setRequestProperties 方法是由 Web 控制程式所呼叫，用以傳遞所有輸入內容給控制程式指令。控制程式指令必須剖析輸入內容，並明確設定此方法中的每一個個別內容。這種由控制程式指令本身明確設定內容的方式，主要是提倡類型安全內容的概念。

下列語法說明如何使用此方法：

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty reqParms)
{

     // 剖析輸入內容並明確設定每一個參數

}
```

validateParameters 方法

validateParameters 方法用來執行起始的參數檢查以及任何必要的參數解析。舉例來說，它可用來解析 orderId=*。此方法會在 getResources 與 performExecute 方法之前先行呼叫。有關此順序的詳細資訊，請參閱第 95 頁的『存取控制的互動』。

getResources 方法

此方法用來實作資源層次的存取控制。它會傳回指令所要執行之「資源/動作」對的向量。假設未傳回任何項，則不會執行任何資源層次的存取控制。有關存取控制的詳細資訊，請參閱第 85 頁的第 4 章, 『存取控制』。

performExecute 方法

performExecute 方法含有您指令的商業邏輯。在執行任何新商業邏輯前，此指令應呼叫指令之超類別的 performExecute 方法。而在結束時，此指令必須傳回一個檢視畫面名稱。

以下是新控制程式指令中之 performExecute 方法的範例語法。在本範例中，回應採用的是重新導向檢視畫面指令，不過，它也可以採用轉遞檢視畫面指令或直接導向檢視畫面指令。

```
public void performExecute() throws ECException
{
super.performExecute();

//////////////////////////////////////////////
//  您的商業邏輯                            //
//////////////////////////////////////////////

// 為回應內容建立新的 TypedProperty。
  TypedProperty rspProp = new TypedProperty();

// 設定回應內容
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
//////////////////////////////////////////////////
//  下列一行為選用的。VIEWREG 表格              //
//  可指定重新導向 URL。                        //
//////////////////////////////////////////////////

rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

//////////////////////////////////////////////////////////////////////////
//   如果您使用轉遞檢視畫面，您可以按如下                                //
//   設定回應內容：                                                     //
//   TypedProperty rspProp = new TypedProperty();                       //
//   rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");                //
//   rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");              //
//                                                                      //
//   再者，您可以選擇性地設定 JSP 範本的名稱。                         //
//   VIEWREG 表格可指定 JSP 範本。                                                //
//////////////////////////////////////////////////////////////////////////

  setResponseProperties(rspProp);
}
```

如果您在 performExecute 方法中指定重新導向 URL，且 VIEWREG 表格中存在相關項目，則會優先採用程式碼中的指定值，而非優先採用 VIEWREG 表格中的值。對程式碼中的 JSP 範本規格而言，亦爲同樣的優先順序。

長時間執行的控制程式指令

如果控制程式指令的執行時間頗長，您可以將該指令分割成兩個指令。 因 URL 要求而執行的第一個指令會將第二個指令新增到排程器中，因此將之當成背景工作執行。其說明請見下圖：

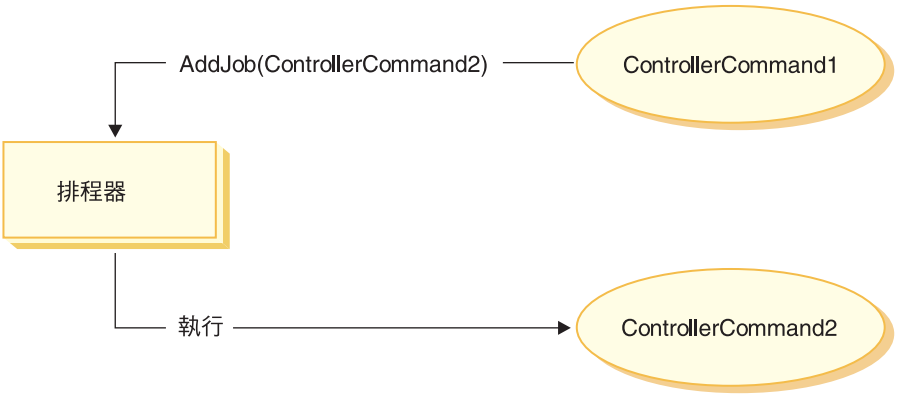

圖 28.

前圖中之流程的說明如下：

1. 因 URL 要求而執行了 ControllerCommand1。
2. ControllerCommand1 在排程器中新增一項工作。此工作爲 ControllerCommand2。ControllerCommand1 在新增工作到排程器中後隨即傳回一個檢視畫面。
3. 排程器將 ControllerCommand2 當成背景工作執行。

在本情況中，用戶端通常會輪詢 ControllerCommand2 產生的結果。ControllerCommand2 應會將工作狀態寫到資料庫中。

檢視畫面指令輸入內容的設定格式

當控制程式指令完成時，會傳回應執行的檢視畫面名稱。此檢視畫面可能會要求需傳給它一些輸入內容。這些輸入參數的來源可爲三種，如下所示：

- 儲存在 CMDREG 表格之 PROPERTIES 直欄中的預設內容
- VIEWREG 表格之 PROPERTIES 直欄中的預設內容
- URL 中的輸入內容

有關這些內容如何合併與設定在 JSP 範本之屬性中的詳細資訊，請參閱第 41 頁的『設定 JSP 屬性 - 概觀』。本節說明如何格式化檢視畫面指令的輸入內容。

在重新導向檢視畫面指令方面，請查看下列兩個主題：

- 將查詢字串純文字化，以支援 URL 重新導向
- 處理重新導向 URL 的長度限制

在轉遞檢視畫面指令方面，請查看「輸入參數的列舉」主題以及「將之當成參數設定在 HttpServletRequestObject」主題。

將輸入參數純文字化為 HttpRedirectView 的查詢字串

所有要傳給重新導向檢視畫面指令的輸入參數，會純文字化爲查詢字串以進行 URL 重新導向。舉例來說，假設重新導向檢視畫面指令的輸入中含有下列內容：

```
URL = "MyView?p1=v1&p2=v2";
ip1 = "iv1";   // 原始控制程式指令的輸入
ip2 = "iv2" ;  // 原始控制程式指令的輸入
op1 = "ov1";
op2 = "ov2";
```

根據上述的輸入參數，最終的 URL 爲

```
MyView?p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2
```

請注意，假設指令採用 SSL，則會加密參數，而其最終 URL 會是：

```
MyView?krypto=encrypted_value_of"p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2"
```

處理長度有限的重新導向 URL

在預設的情況下，控制程式指令的所有輸入參數會傳達給重新導向檢視畫面指令。如果有限制重新導向 URL 中的字元數，則可能會造成問題。舉例來說，如果用戶端使用 Internet Explorer 瀏覽器，便有長度上的限制。對此瀏覽器而言， URL 不能超過 2083 個位元組。如果 URL 超過此限制，則會截斷 URL。因此，假設輸入參數的數目龐大，或者您使用加密，則可能會遇到問題，這是因爲加密字串通常是未加密字串的兩到三倍長。

處理長度有限之重新導向 URL 的方法有兩種。

1. 改寫控制程式指令中的 getViewInputProperties 方法，而只傳回必須傳給重新導向檢視畫面指令的參數集。
2. 在 URL 參數中使用指定的特殊字元，指出哪些參數可從輸入參數字串中移除。

爲了示範上述每一種方法，請注意下列之控制程式指令的輸入參數集：

```
URL="MyView";
// 下列全為原始控制程式指令的輸入。
ip1="ipv1";
ip2="ipv2";
ip3="ipv3";
iq1="iqv1";
iq2="iqv2";
ir1="ipr1";
ir2="ipr2";
is="isv";
```

如果您改寫 getViewInputProperties 方法，您可撰寫新方法，以便僅將下列參數傳給檢視畫面指令：

```
ir2="ipr2";
is="isv";
```

當使用第二種方法時，您可在呼叫檢視畫面指令時，使用特殊參數指出一些應移除的輸入參數。舉例來說，您可以將下列指定成 URL 參數以達到相同結果：

```
URL="MyView?ip*=&iq*=&ir1="
```

此 URL 參數將下列告知 WebSphere Commerce 執行期間組織架構：

- `ip*=` 規格表示應移除名稱開頭爲 `ip` 的所有參數。
- `iq*=` 規格表示應移除名稱開頭爲 `iq` 的所有參數。
- `ir1=` 規格表示應移除 `ir1` 參數。

在 HttpForwardView 的 HttpServletRequest 物件中設定屬性

預設 HttpForwardViewCommandImpl 會列舉所有傳給指令的參數，並將之當成屬性設於 HttpServletRequest 物件中。

舉例來說，假設傳給轉遞檢視畫面指令的 requestProperties 物件中含有下列參數：

```
p1="pv1";
p2="pv2";
p3=pv3;   // pv3 為物件
```

然後使用 request.setAttribute() 方法將下列屬性傳給 JSP 範本。

```
request.setAttribute("p1", "pv1");
request.setAttribute("p2", "pv2");
request.setAttribute("p1", pv1);
request.setAttribute("RequestProperties", requestProperties);
request.setAttribute("CommandContext", commandContext);
```

其中 `requestProperties` 爲傳給指令的 TypedProperty 物件， `commandContext` 爲傳給指令的指令環境定義物件， `p1`、`p2` 與 `p3` 爲定義在 requestProperties 物件中的參數。

控制程式指令的資料庫確定與回復

在控制程式指令的整個執行過程中，經常會建立或更新資料。在許多情況中，當交易結束時必須以新資訊更新資料庫。交易是由 Web 控制程式所管理。

Web 控制程式在呼叫控制程式指令前會標示交易的開始。當控制程式指令執行完畢時，控制程式指令會傳回一個檢視畫面名稱給 Web 控制程式。Web 控制程式負責標示交易的結束。實際的交易結束點（呼叫檢視畫面之前或之後）視所用的檢視畫面類型而定。

檢視畫面指令的類型有下列三種：

- 轉遞檢視畫面指令
- 重新導向檢視畫面指令
- 直接導向檢視畫面指令

Web 控制程式會查閱 VIEWREG 表格中的檢視畫面名稱，以判斷檢視畫面所要使用的檢視畫面指令。

如果 VIEWREG 表格中的項目指定使用 ForwardViewCommand，則 Web 控制程式會將控制程式指令的結果轉遞給對應的 ForwardViewCommand 實作類別（亦指定於 VIEWREG 中）。檢視畫面指令會在現行交易的環境定義中執行。 在此情況下，在檢視畫面指令完成前不會進行資料庫確定或回復。

如果 VIEWREG 表格中的項目指定使用 RedirectViewCommand，則 Web 控制程式會將控制程式指令的結果轉遞給對應的 RedirectViewCommand 實作類別。檢視畫面指令會在現行交易範圍外運作，而資料庫確定或回復會在呼叫重新導向檢視畫面指令前進行。

如果 VIEWREG 表格中的項目指定使用 DirectViewCommand，則 Web 控制程式會將控制程式指令的結果轉遞給對應的 DirectViewCommand 實作類別。檢視畫面指令會在現行交易的環境定義中執行。 在此情況下，在檢視畫面指令完成前不會進行資料庫確定或回復。（請注意，ForwardViewCommand 與 DirectViewCommand 類似。ForwardViewCommand 會轉遞結果給 JSP 範本。相對地，DirectViewCommand 則將結果當成輸入串流接收，並將之當成輸出串流傳遞。它採用將資料視爲位元組的 getRawDocument 方法，或採用將資料視爲文字的 getTextDocument。）

如果檢視畫面指令是在和控制程式指令相同的交易範圍下執行，則檢視畫面指令中的錯誤會造成整個交易回復。視您的商業邏輯而定，這不見得是所要的結果。

控制程式指令的交易範圍範例說明

爲了說明控制程式指令之交易範圍中的差異，視所用的檢視畫面指令類型而定，請注意下列範例。

情況 1：在控制程式指令交易範圍中執行檢視畫面

假設您建立一個新控制程式指令 YourControllerCmdA。指令的 performExecute 方法中含有：

```
.
.
// 建立輸出的新 TypedProperty 物件。
  TypedProperty rspProp = new TypedProperty();

//////////////////////
// 商業邏輯          //
//////////////////////

// 傳回檢視畫面
rspProp.put(ECConstants.EC_VIEWTASKNAME, "YourView");
SetResponseProperties(rspProp);
```

在上述程式碼片段中，控制程式指令傳回 “YourView” 做爲檢視畫面。YourView 登錄在 VIEWREG 表格中。以下是登錄 YourView 的 insert 陳述式範例。

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView.jsp');
```

其中 XX 爲商店識別碼。由於檢視畫面使用 com.ibm.commerce.command.HttpForwardViewCommandImpl 實作類別，Web 控制程式使用通用的轉遞檢視畫面指令。

根據上述的指令登錄，Web 控制程式在控制程式指令交易範圍中啓動 YourView.jsp 檔。如果 YourView.jsp 中有錯誤，則交易失敗，並進行資料庫回復。也因此，整個控制程式指令執行失敗。

情況 2：在控制程式指令交易範圍外執行檢視畫面

假設您想將資訊確定到資料庫中，即使檢視畫面中發生錯誤。爲了讓檢視畫面在控制程式指令的交易範圍外執行，檢視畫面必須當成重新導向來執行。

爲了將檢視畫面當成重新導向來執行，控制程式指令的 performExecute 方法以下列方式傳回檢視畫面：

```
.
.
// 建立輸出的新 TypedProperty 物件。
  TypedProperty rspProp = new TypedProperty();
```

```
//////////////////////
// 商業邏輯          //
//////////////////////

// 傳回檢視畫面
rspProp.put(ECConstants.EC_VIEWTASKNAME, EC_GENERIC_REDIRECTVIEW);
rspProp.put(EC_Constants.EC_REDIRECTURL, "YourView2");
```

下列的範例 SQL 陳述式支援重新導向策略：

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView2',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView2.jsp');
```

其中 XX 爲商店識別碼。

由於指令將 EC_GENERIC_REDIRECTVIEW 值當成回應內容參數傳遞， Web 控制程式會採用通用的重新導向檢視畫面指令。通用的重新導向檢視畫面以下列資訊登錄在 VIEWREG 表格中：

- ViewName = RedirectView
- DeviceFmt_Id = -1
- InterfaceName = com.ibm.commerce.command.RedirectViewCommand
- ClassName = com.ibm.commerce.command.HttpRedirectViewCommandImpl

Web 控制程式呼叫通用的重新導向檢視畫面指令，它會將重新導向 URL 當成輸入內容。其回應是重新導向至重新導向 URL。在發生重新導向後，會呼叫 YourView2。然後當成同屬轉遞檢視畫面來實作。

新作業指令

新作業指令應從抽象作業指令類別 (com.ibm.commerce.command.TaskCommandImpl) 延伸而來，並且實作一個延伸 com.ibm.commerce.command.TaskCommand 介面的介面。如同第 124 頁中之圖所示，新作業指令應定義成：

```
public class MyTaskCmdImpl extends com.ibm.commerce.command.TaskCommandImpl
     implements MyTaskCmd {

}
```

作業指令的所有輸入與輸出內容必須定義在指令介面中，例如：MyTaskCmd。呼叫端所規劃的是作業指令介面，而非作業指令的實作類別。這可讓您擁有作業指令的多種實作方式（每一家商店一種）而呼叫端不需顧慮要呼叫哪個實作類別。

定義於介面中的所有方法必須置於實作類別中實作。由於指令環境定義應由呼叫端（即控制程式指令）設定，作業指令不必設定指令環境定義。不過作業指令可透過指令環境定義從 Web 控制程式取得資訊。

除了實作定義於作業指令介面中的方法外，您應改寫出自 com.ibm.commerce.command.TaskCommandImpl 類別中的 performExecute 方法。

performExecute 方法中含有作業指令所執行之特定工作單元的商業邏輯。在執行商業邏輯前，此指令應呼叫作業指令之超類別的 performExecute 方法。下列程式碼片段顯示作業指令的範例 performExecute 方法。

```
public void performExecute() throws ECException
{
super.performExecute();

    // 請在此納入您的商業邏輯。

    // 請設定輸出以便讓控制程式指令
    // 可擷取此作業指令產生的結果。
}
```

執行期間組織架構會呼叫控制程式指令的 getResources 方法，以判斷指令將存取哪些可保護的資源。不過有可能發生下列情況：在控制程式指令範圍期間執行作業指令，且該指令試著存取控制程式指令之 getResources 方法並未傳回的資源。如果發生此情況，作業指令本身可實作 getResources 方法，以確定有提供存取控制給可保護的資源。

請注意，在預設的情況下，getResources 爲作業指令所傳回的是 null，且不會執行資源層次的存取控制檢查。因此，如果作業指令要存取可保護的資源，您必須加以改寫。

自訂現有指令

本節提供各種方法供您自訂現有的控制程式、作業與資料 Bean 指令。

自訂現有的控制程式指令

控制程式指令概括了商業程序的相關商業邏輯。商業程序中的個別工作單元可交由作業指令執行。因此，自訂控制程式指令的方法有許多，而其中有些會牽涉到作業指令的自訂。

在您自訂控制程式指令時，您可以達成下列事項：

- 新增其他的處理程序與邏輯到現有的控制程式指令中。您可加在現有商業邏輯之前、之後或前後。
- 取代一或多個作業指令。這可讓您修改商業程序中之特定步驟的執行方式。

- 更換控制程式指令所呼叫的檢視畫面。

下節詳述如何進行上述的修改。

將新的商業邏輯加到控制程式指令中

假設有個現有 WebSphere Commerce 控制程式指令名爲 ExistingControllerCmd。根據 WebSphere Commerce 命名慣例，此控制程式指令的介面類別名稱爲 ExistingControllerCmd，實作類別名稱爲 ExistingControllerCmdImpl。現在假設因爲商業需求，您必須將新的商業邏輯加到這個現有的指令中。邏輯中有一部份必須在現有指令邏輯前執行，另有一部份必須在現有指令邏輯後執行。

加入新的商業邏輯的第一步是建立新實作類別，且此實作類別是從原始實作類別延伸而來。在本例中，您將建立從 ExistingControllerCmdImpl 延伸而來的新 ModifiedControllerCmdImpl 類別。新實作類別應實作原始介面 (ExistingControllerCmd)。

在新實作類別中，您必須建立新的 performExecute 方法，以改寫現有指令的 performExecute。在新 performExecute 方法中，插入新商業邏輯的方式有下列兩種：直接在控制程式指令中包含程式碼，或者建立新作業指令以執行新商業邏輯。如果您要建立新作業指令，您必須從控制程式指令中實例化新作業指令物件。

下列的程式碼片段示範如何藉由在控制程式指令中直接包含邏輯，以便將新的商業邏輯加到現有控制程式指令的開頭與尾端：

```
public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
         /* 插入必須在原始指令之前執行的
            新商業邏輯。
            */

         // 執行原始的指令邏輯。
super.performExecute();

         /* 插入必須在原始指令之後執行的
            新商業邏輯。
            */
          }
    }
```

下列的程式碼片段示範如何藉由從控制程式指令中實例化新作業指令，以便將新的商業邏輯加到現有控制程式指令的開頭。此外，您也將建立新作業指令介面與實作類別，並將作業指令登錄在指令登錄中。

```
// 以 CommandFactory 匯入套件
import com.ibm.commerce.command.*;

public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
     MyNewTaskCmd cmd = null;
          cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.mycompany.mycommands.MyNewTaskCommand",
               getStoreId());

            /*
          設定作業指令的輸入參數，
          呼叫其執行方法，並擷取輸出參數
          （若有需要的話）。
            */

super.performExecute();
         }
   }
```

不論您是在控制程式指令中包含新商業邏輯，或者建立新作業指令來執行邏輯，您都必須更新 WebSphere Commerce 指令登錄中的 CMDREG 表格，以便讓新控制程式指令實作類別連結現有的控制程式指令介面。下列 SQL 陳述式顯示範例更新：

```
update CMDREG
set CLASSNAME='ModifiedControllerCmdImpl'
where INTERFACENAME='ExistingControllerCmd'
```

更換控制程式指令所呼叫的作業指令

通常控制程式指令會呼叫一些執行個別作業的作業指令。這些作業共同組成了控制程式指令所代表的商業程序。您或許需改變程序中某特定步驟的執行方式（但不是將新的商業邏輯加到控制程式指令的開頭或尾端）。在此情況下，您必須將想要改寫的作業指令的實作取代成新作業指令的實作，而以您要的方式來執行作業。

根據 WebSphere Commerce 程式設計模型的設計方式，您不必建立新控制程式指令實作類別來更換作業指令。控制程式指令會藉由呼叫指令 factory 的 createCommand 方法，來實例化作業指令。指令 factory 會使用作業指令的介面名稱，然後根據指令登錄，決定正確的實作類別。因此，如果要更換已實例化的作業指令，您必須建立新作業指令實作類別，然後更新指令登錄，以便讓原來的作業指令介面名稱連結新作業指令實作類別。相關資訊請參閱第 141 頁的『自訂現有的作業指令』。

更換控制程式指令所呼叫的檢視畫面

如果要更換控制程式指令所呼叫的檢視畫面，您必須爲控制程式指令建立新實作類別。舉例來說，建立一個由 ExistingControllerCmdImpl 延伸而來並實作 ExistingControllerCmd 介面的新 ModifiedControllerCmdImpl。

在 ModifiedControllerCmdImpl 類別中，改寫 performExecute 方法。在新 performExecute 方法中，呼叫 super.performExecute，以確定所有指令處理程序皆會進行。在執行指令邏輯後，您可使用回應內容來改寫呼叫的檢視畫面。下列的程式碼片段顯示在將檢視畫面當成重新導向執行的情況下，如何改寫檢視畫面：

```
// 匯入內含 TypedProperty 的套件以及 ECConstants。
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
         // 執行原始的指令邏輯。
super.performExecute();

// 為回應內容建立新的 TypedProperty。
  TypedProperty rspProp = new TypedProperty();

// 設定回應內容
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
          ////////////////////////////////////////////////
//  下列一行為選用的。VIEWREG 表格                    //
//  可指定重新導向 URL。                              //
          ////////////////////////////////////////////////

rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

  setResponseProperties(rspProp);

         }
   }
```

下列的程式碼片段顯示在將檢視畫面當成轉遞檢視畫面執行的情況下，如何改寫檢視畫面：

```
// 匯入內含 TypedProperty 的套件以及 ECConstants。
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
```

```
        // 執行原始的指令邏輯。
super.performExecute();

// 為回應內容建立新的 TypedProperty。
  TypedProperty rspProp = new TypedProperty();

// 設定回應內容
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");

          ////////////////////////////////////////////////
          //  選擇性地明確設定 JSP 範本的名稱。           //
          //  VIEWREG 表格可指定 JSP 範本。               //
          ////////////////////////////////////////////////

      rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");

  setResponseProperties(rspProp);

        }
    }
```

如果要判斷現有控制程式指令所用的是哪個檢視畫面，請參閱 WebSphere Commerce 正式作業與開發線上說明中的「參照」主題。

自訂現有的作業指令

修改現有 WebSphere Commerce 作業指令有兩種標準方法。藉由這些修改方法，可讓您達成下列事項：

- 新增其他的處理程序與邏輯到現有的作業指令中。您可加在現有商業邏輯之前、之後或前後。
- 將現有商業邏輯全面換成您自己的商業邏輯。

如果要完成上述修改，您將實際建立一個新作業指令實作類別。下列各節將進一步詳述。

將新的商業邏輯加到作業指令中

假設有個現有 WebSphere Commerce 作業指令名爲 ExistingTaskCmd。根據 WebSphere Commerce 命名慣例，此作業指令的介面類別名稱爲 ExistingTaskCmd，實作類別名稱爲 ExistingTaskCmdImpl。現在假設因爲商業需求，您必須將新的商業邏輯加到這個現有的指令中。邏輯中有一部份必須在現有指令邏輯前執行，另有一部份必須在現有指令邏輯後執行。

加入新的商業邏輯的第一步是建立新實作類別，且此實作類別是從原始實作類別延伸而來。在本例中，您將建立從 ExistingTaskCmdImpl 延伸而來的新 ModifiedTaskCmdImpl 類別。新實作類別應實作原始介面 (ExistingTaskCmd)。

在新指令中，您將改寫現有 performExecute 方法，並在呼叫 super.performExecute 方法前後加上新邏輯。

下列程式碼片段示範如何將新的商業邏輯加入到現有作業指令中：

```
public class ModifiedTaskCmdImpl extends ExistingTaskCmdImpl
     implements ExistingTaskCmd {


         /* 插入必須在原始指令之前執行的
            新商業邏輯。
         */

         // 執行原始的指令邏輯。
super.performExecute();

         /* 插入必須在原始指令之後執行的
            新商業邏輯。
         */
}
```

您也必須更新 CMDREG 表格，以便讓新實作類別連結現有介面。下列 SQL 陳述式顯示範例更新：

```
update CMDREG
set CLASSNAME='ModifiedTaskCmdImpl'
where INTERFACENAME='ExistingTaskCmd'
```

更換現有作業指令的商業邏輯

如果要更換現有作業指令的商業邏輯，您必須爲作業指令建立一個新實作類別。這個新的實作類別必須繼承自現有的作業指令，但是不要實作現有的介面。此外，在新的實作類別中，請不要呼叫超類別的 performExecute 方法。

在繼承自您要取代眞正指令時可能會看起來與直覺相違背，採取這種方式的原因是有關未來版本的 WebSphere Commerce 的支援。這種方式可保護您的程式碼不會受到未來的 WebSphere Commerce 版本在指令介面變更的影響。

例如，假設您要取代 OrderNotifyCmdImpl 作業指令的商業邏輯。在此情況下，您會建立名稱爲 CustomizedOrderNotifyCmdImpl 的新作業指令。這個指令繼承 OrderNotifyCmdImpl。在新的 CustomizedOrderNotifyCmdImpl 中，您建立新的商業邏輯，但不要從超類別呼叫 performExecute 方法。如果未來的 WebSphere Commerce 版本在介面中引入的新方法，名稱爲 newMethod，對應的 OrderNotifyCmdImpl 指令版本將會包含 newMethod 方法的預設實作。然後，既然您的新指令是繼承自 OrderNotifyCmdImpl，編譯器會在 OrderNotifyCmdImpl 指令中尋找此新方法的預設實作，您的新指令即可不受介面變更的影響。

請參閱 WebSphere Commerce 正式作業與開發線上說明的「參照」區段，以確定新的實作類別提供的是和現有類別相同的外部規則。

自訂資料 Bean

資料 Bean 通常是存取 Bean 的延伸。存取 Bean（可由 WebSphere Studio Application Developer 產生）可提供一種簡單方法來存取 Entity Bean 中的資訊。由於資料 Bean 爲存取 Bean 的延伸，資料 Bean 會自動繼承新屬性。由於此關係，因而不需撰寫程式碼讓資料 Bean 採用 Entity Bean 中的新屬性。

如果您想在非從 Entity Bean 衍生而來的資料 Bean 中加入新屬性，您可以使用 Java 繼承特性來延伸現有資料 Bean。舉例來說，如果您想將新欄位加到 OrderDataBean 中，請依照下列的方法定義 MyOrderDataBean：

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  在此實作新欄位
     }

}
```

新資料 Bean 亦必須具有 BeanInfo 類別。此類別的宣告範例如下：

```
public class MyOrderDataBeanInfo extends java.beans.SimpleBeanInfo
{

}
```

WebSphere Studio Application Developer 會提供工具供您產生此 BeanInfo 類別。

第 7 章 交易協定與商業原則（Business Edition）

本章僅適用於 WebSphere Commerce Business Edition。

簡介

B2B（企業消費型商務）商務中的關鍵元素之一是關係管理。交易協定用以管理買方與賣方組織間的商業關係。 WebSphere Commerce Business Edition 所用的交易協定模型可支援各種交易協定類型，像是「合約」及「RFQ」（報價要求）。

交易協定的主要元素是一組條款。每一項條款分別定義出在交易期間所要使用的特定商業規則。透過 WebSphere Commerce Business Edition，您可使用 RFQ 線上程序來協議一組條款，或者離線協議，然後使用 WebSphere Commerce Accelerator 中的商業關係管理介面來扣款。

制定條款的方法有下列幾種：

- 選取預先定義之商業原則（像是：價格清單與退貨原則）之一的條款。或者，可選取您所建的商業原則。一項條款物件也可參照多個商業原則物件。
- 將特定調整套用在商業原則上的條款，像是：標準計價的調整。
- 定義一組用以支配商業程序之參數的條款。舉例來說，可指出某特定供貨中心是供某特定合約使用。

合約是由一組條款組成。請見下圖：

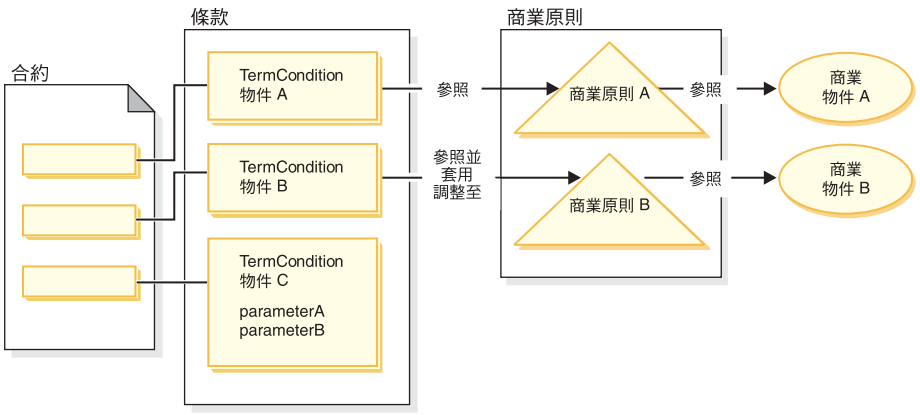

圖 29.

在先前的圖解中，注意下列事項：

- “調整”一詞是指商業原則的修正。例如，可以用來套用折扣到某個商業原則的結果，而使得標準價格打九折。也可以用一組參數來影響商業原則。
- 例如，在圖解中，TermCondition 物件 A 可能代表出貨條款物件。在此情況下，商業原則 A 可能代表出貨模式商業原則，商業物件 A 代表貨運公司 XYZ 的出貨模式 “A3”。
- 另一個範例是，在圖解中，TermCondition 物件 B 可能代表價格條款物件，以套用商業原則 B 所定義的價格的 50% 折扣。在此情況下，商業原則 B 是價格原則，而商業物件 B 是定義主要型錄的交易狀態的交易狀態儲存區。

本章提供程式設計師如何建立新商業原則與新條款的相關準則。

「工具屋」範例商店示範其商業流程中的出貨條款物件和價格條款物件。有關支援這些範例的合約資料的其他資訊，請參閱第 147 頁的『「工具屋」範例合約資料』。

商業原則物件與指令

商業原則物件含有下列資訊：

- 原則 ID
  爲商業原則物件的主要鍵。
- 原則類型
  定義商業原則類型。舉例來說，像 `Price` 與 `ProductSet` 皆爲原則類型。
- 原則名稱
  每個商業原則的名稱皆必須是唯一的。
- 商店實體
  將部署商業原則的商店或商店群組。
- 內容
  一組可傳給商業原則指令的預設內容。商業原則物件的相關指令儲存在 BusinessPolicyCmd 表格中。
- 有效期
  商業原則物件的生效期間。
- 商業原則指令
  用以實作商業原則的零或多個商業原則指令。商業原則指令通常是被商業程序來呼叫，可以是作業或控制程式指令。例如，`getContractPrice()` 指令會取得條款的價格。此價格條款會參照特定的價格原則指令，而且此價格原則指令會用來計價。

多個商業原則指令可連結一個商業原則物件。每個商業原則指令必須實作商業原則類型物件所定義的相同介面。下圖顯示新商業原則指令的結構：

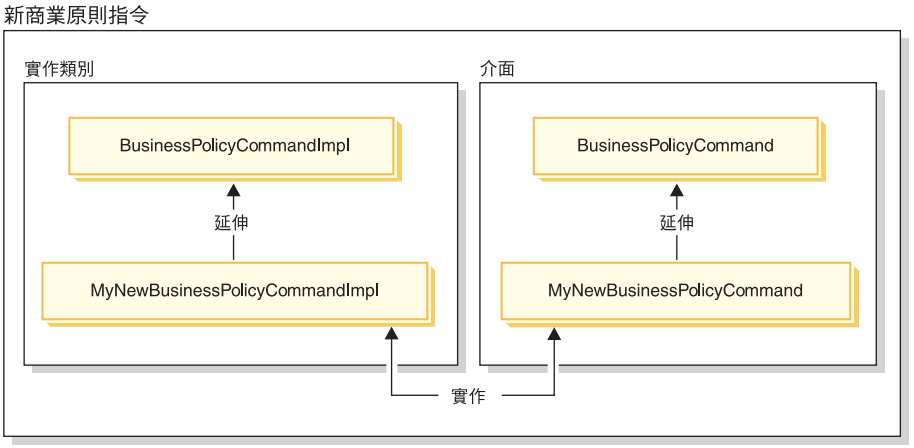

圖 30.

如上圖所示，如果要建立新商業原則指令，您將建立新實作類別，且這個實作類別是從 WebSphere Commerce BusinessPolicyCmdImpl 實作類別延伸而來。您也將建立從 BusinessPolicyCmd 介面延伸而來的新介面。

「工具屋」範例合約資料

本節提供「工具屋」範例商店中使用的部份合約資料簡介。

下列各節中的範例資料會按資料庫表格來組織。只會顯示相關的列和欄。另外，請注意在安裝範例之後，任何唯一的 ID（例如 CONTRACT_ID）的值可能會與此處顯示的值不同。

CONTRACT 表格範例資料

下表顯示 CONTRACT 資料庫表格中的相關範例資料。請注意，爲了方便顯示，資料庫直欄標題會顯示在第一欄，表格的範例資料列是顯示於第二欄。

直欄名稱	範例資料
CONTRACT_ID	10007
MAJORVERSION	1
MINORVERSION	0
NAME	ToolTechContractNumber 4567
MEMBER_ID	-2001

直欄名稱	範例資料
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

TERMCOND 表格範例資料

下表顯示 TERMCOND 資料庫表格中的相關範例資料。請注意，爲了方便顯示，資料庫直欄標題會顯示在第一欄，表格的範例資料列是顯示於第二欄和第三欄。

直欄名稱	範例資料列 1	範例資料列 2
TERMCOND_ID	10025	10030
TCSUBTYPE_ID	PriceTCPriceListWithSelectiveAdjustment	ShippingTCShippingMode
TRADING_ID	10007	10007
STRINGFILED1	ProductSet2	
INTEGERFIELD2	10002	
INTEGERFIELD3	1	
BIGINTFIELD1	10051	
FLOATFIELD1	-50.0	
SEQUENCE	1	6

POLICYTC 表格範例資料

下表顯示 POLICYTC 資料庫表格中的相關範例資料。此表格建立原則與條款物件之間的關係。

	直欄名稱	
	POLICY_ID	TERMCOND_ID
範例資料列 1	10053	10025
範例資料列 2	10056	10030

POLICY 表格範例資料

下表顯示 POLICY 資料庫表格中的相關範例資料。

直欄名稱	範例資料列 1	範例資料列 2
POLICY_ID	10053	10056
POLICYNAME	MasterCatalogPriceList	A3
POLICYTYPE_ID	Price	ShippingMode
STOREENT_ID	10051	10051
PROPERTIES	n a m e = T o o l T e c h & member_id=-2001	shippingMode=A3
STARTTIME	空值	空值
ENDTIME	空值	空值

TRADEPOSCN 表格範例資料

下表顯示 TRADEPOSCN 資料庫表格中的相關範例資料。

	直欄名稱			
	TRADEEPOSCN_ID	MEMBER_ID	NAME	TYPE
範例資料列	10051	-2001	ToolTech	S

SHIPMODE 表格範例資料

下表顯示 SHIPMODE 資料庫表格中的相關範例資料。

	直欄名稱			
	SHIPMODE_ID	STOREENTITY_ID	CODE	CARRIER
範例資料列	10053	10051	A3	XYZ 貨運公司

延伸現有的合約模型

合約可以由一或多個條款物件所組成，每一個物件都參照一個原則。後續各節說明建立新商業原則以及整合至商業流程的必要步驟。

下列是執行這項作業的高階步驟，作爲簡短的總覽：

1. 建立新的商業原則。
   下列作業是與建立新商業原則指令有關：

a. 建立新商業原則類型 (如果需要)。
   會提供數種商業原則類型，但是如果標準類型不能適合您的商業需求，請建立新的商業原則類型。
b. 建立新商業原則指令。
c. 登錄新商業原則和商業原則指令。

2. 您可以將條款物件關聯至新的商業原則。
   您可以將現有的條款物件關聯到新商業原則，或是建立新的條款物件，來完成這個步驟。如果您建立新的條款物件，則必須執行下列步驟：
   a. 在資料庫中登錄新條款
   b. 在合約 XSD（XML 綱目定義）中登錄新條款
   c. 爲條款建立新 CMP Enterprise Bean
   d. 更新 WebSphere Commerce Accelerator 以反映新條款
3. 在商業流程期間呼叫新的商業原則。

WebSphere Commerce 5.5 版包含許多新的合約類型。有關可用合約類型的資訊，請參閱 WebSphere Commerce 正式作業線上說明中「概念」主題的「商業帳戶」子主題。

下列各節使用 BuyerContract 作爲延伸範例中的合約類型。類似的延伸方法也用於其他合約類型中。

建立新商業原則

建立新的商業原則時，通常必須在資料庫中登錄一個唯一的商業原則，並且建立一個新的商業原則指令。

建立新商業原則指令涉及下列各高階步驟：

1. 建立新商業原則類型 (如果需要)。
2. 撰寫新商業原則指令。
3. 將新商業原則與商業原則指令登錄於資料庫中。

上述每一個步驟將在後續各節中詳細說明。

建立新商業原則類型

本節說明如何建立新商業原則類型。商業原則類型用以指出將套用原則的交易領域。商業原則類型的範例如下：

- Price
- ProductSet

- ShippingMode
- ShippingCharge
- Payment
- ReturnCharge
- ReturnApproval
- ReturnPayment
- InvoiceFormat

如果現有商業原則類型不能滿足您的商業需求，您應建立新商業原則類型。建立新商業原則類型包括：定義與登錄商業原則類型。

在您定義與登錄新原則類型時，您必須更新下列的資料庫表格：

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

POLICYTYPE 表格用以指定您要建立的商業原則類型。此表格含有單一直欄 POLICYTYPE_ID（爲主要鍵）。舉例來說，其值可爲 Price。如果您要建立新商業原則類型，請確定您有指定一個唯一的 POLICYTYPE_ID。

PLCYTYCMIF 表格爲「商業原則類型/指令介面」關係規格表。亦即，在每個商業原則類型方面，它會指定商業原則物件的 Java 指令介面。雖然實作商業原則的商業原則指令可有零或多個，但每個商業原則指令必須實作此處指定的介面。

PLCYTYPDSC 表格用以指定商業原則類型的說明。此表格含有說明的語言識別碼以及商業原則類型的說明。

如果要建立新商業原則類型，請在這些表格中分別建立新商業原則類型的項目。下列 SQL 陳述式提供範例：

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('MyNewPolicyType');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('MyNewPolicyType',
   'com.mycompany.mybusinesspolicycommands.MyNewPolicy');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('MyNewPolicyType', -1,
   'My new policy type for example purposes.');
```

在建立新商業原則類型的最後一個步驟時，您可以撰寫一或多個新的商業原則類型介面。然後在商業原則類型範圍內的任何商業原則指令就會實作這些介面。例如，在「工具屋」範例商店中，「價格」是定義成商業原則類型。因此，其中包含所有與價格相關的商業原則指令所實作的

com.ibm.commerce.price.commands.ResolvePriceListsCmd 和 com.ibm.commerce.price.commands.RetrievePricesCmd 介面。

如果您將不會有對新商業原則類型執行作業的商業原則指令，就不需要建立新的介面。這種情況很少見，因爲在大部分的情況下，當您在建立新的商業原則類型時，必須同時建立新的商業原則類型介面。

當您建立商業原則類型介面時，新的介面必須延伸 com.ibm.commerce.command.BusinessPolicyCommand 介面。

撰寫新商業原則指令

如果要建立新的商業原則指令，您必須建立一個新的指令來實作與指令相關的商業原則類型的介面。新指令也必須延伸 com.ibm.commerce.command.BusinessPolicyCommandImpl 實作類別。這點和建立新控制程式或作業指令相當類似。

您可採用下列兩種方式將輸入內容傳遞給商業原則指令。第一種方式是在 POLICY 表格的 PROPERTIES 直欄中指定預設輸入內容。有關此表格的詳細資訊，請參閱下節。

第二種方式是在指令中爲每個輸入內容各建一個新欄位。然後針對每一個欄位，分別新建一對 getter 與 setter 方法。

在商業原則指令中設定 requestProperties

將 requestProperties 設定在商業原則指令物件中的方式有兩種。第一種方式是使用 POLICY 表格中的 PROPERTIES 直欄來設定預設內容。這是使用 setRequestProperties 方法來達成。第二種設定內容的方式是讓會呼叫商業原則指令的指令（控制程式或作業）明確設定其他必要的內容。

在您建立新商業原則指令時，您應改寫預設 setRequestProperties 方法而包含邏輯，以明確設定 requestProperties 物件中所含的每一個參數。

舉例來說，新商業原則指令的介面名稱爲 MyNewBusinessPolicyCmd，實作類別名稱爲 MyNewBusinessPolicyCmdImpl。

假設在 POLICY 表格中，此新商業原則指令項目於 PROPERTIES 直欄中含有下列值：

- defaultProperty1=apple
- defaultProperty2=orange
- defaultProperty3=banana

此商業原則指令的介面被定義成：

```
public interface MyNewBusinessPolicyCmd extends
   com.ibm.commerce.command.BusinessPolicyCmd {
     java.lang.String defaultCommandClassName =
         'com.mycompany.mycommands.MyNewBusinessPolicyCmdImpl';
      public void setProperty1();
      public void setProperty2();
}
```

此商業原則指令的實作類別被定義成：

```
public class MyNewBusinessPolicyCmdImpl extends
   com.ibm.commerce.command.BusinessPolicyCmdImpl
   implements com.mycompany.mycommands.MyNewBusinessPolicyCmd {
      // 建立儲存在 POLICY 表格中的預設內容

      private java.lang.String defaultProperty1;
      private java.lang.String defaultProperty2;
      private java.lang.String defaultProperty3;

      // 開始建立必須由呼叫指令
     // 來設定的內容。
      //  *** property1   ***
      private java.lang.String property1;
      public java.lang.String getProperty1() {
         return property1;
         }
      public void setProperty1(java.lang.String newProperty1) {
         property1 = newProperty1;
      }

      //  *** property2   ***
      private java.lang.String property2;
      public java.lang.String getProperty2() {
         return property2;
         }
      public void setProperty1(java.lang.String newProperty2) {
         property2 = newProperty2;
         }

      // 結束建立必須由呼叫指令
     // 來設定的內容。
      /* 一旦實例化，商業原則指令會將 POLICY 表格
         中的所有預設內容設定到 requestProperties 物件中。
         呼叫指令負責設定其他任何必要內容。
      */

      public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
         requestProperties) {
         //  取得定義於 POLICY 表格中的預設內容
         setDefaultProperty1(requestProperties.get("defaultProperty1"));
         setDefaultProperty2(requestProperties.get("defaultProperty2"));
```

```
            setDefaultProperty3(requestProperties.get("defaultProperty3"));
            }

}
```

用來呼叫新商業原則指令的指令可使用下列方式來定義：

```
public class MyCallerCommandImpl
   extends com.ibm.commerce.command.TaskCommandImpl
   implements com.mycompany.mycommands.MyCallerCommand {

   /* 包含作業指令所需的所有元素與處理程序。
   */

   // 判斷原則 ID 與 setPolicyId

   // 呼叫商業原則指令。

   cmd = (MyNewBusinessPolicyCmd) CommandFactory
            createPolicyCommand(policyId);

   // 設定必要內容

   cmd.setProperty1("Fruit salad");
   cmd.setProperty2("Favorite food");

 cmd.execute();
}
```

登錄新商業原則與商業原則指令

在您建立新商業原則指令後，您必須將商業原則與商業原則指令同時登錄在資料庫中。

商業原則是登錄在 POLICY 表格中。此表格中含有下列直欄：

- POLICY_ID
  主要鍵。此爲原則識別碼。
- POLICYNAME
  唯一的原則名稱。
- POLICYTYPE_ID
  原則類型識別碼。此爲 POLICYTYPE 表格的外來鍵。
- STOREENT_ID
  將套用原則的商店或商店群組。
- PROPERTIES
  可設定於商業原則指令中的預設內容。請指定成「名稱-值」對；例如：`parm1=val1&parm2=val2`。

- STARTDATE
  原則的起始日期（指定成時間戳記）。若爲空值，表示此刻即爲起始日期。
- ENDDATE
  原則的結束日期（指定成時間戳記）。若爲空值，表示沒有結束日期。

一旦您將新原則登錄在 POLICY 表格中後，您必須登錄原則與實作商業原則之商業原則指令間的關係。 POLICYCMD 表格即用來完成此目的。POLICYCMD 表格中含有下列直欄：

- POLICY_ID
  POLICY 表格的外來鍵參照。
- BUSINESSCMDCLASS
  實作原則的商業原則指令。
- PROPERTIES
  可設定於商業原則指令中的預設內容。請指定成「名稱-值」對；例如：`parm1=val1&parm2=val2`。

關聯條款物件至新的商業原則

在 WebSphere Commerce 合約與原則組織架構中，條款（亦稱爲條文）可讓您用來說明買方與賣方間的協定。條款可用於各種不同的交易協定類型中，像是：合約與 RFQ（報價要求）。通常條款物件會參照商業原則，並且可選擇性調整。舉例來說，藉由挑選一個價格原則物件，來建立價格條款物件。在價格條款中，帳戶經理可調整商店的標準價格，像是：

- 對標準價格表採取某個百分比的折扣
- 對指定的一組產品採取某個百分比的折扣

每一項調整皆可指定成一項條款。

當您建立新的商業原則時，如果原則是用於合約之中，最少必須有一個條款物件參照此商業原則。您可以將現有的條款物件關聯到新的商業原則（藉由擷取 XSD（XML 綱目定義）檔中的現有條款物件與新商業原則之間的關係來完成），或是建立與新的商業原則相關的新條款物件。

建立新條款

在 WebSphere Commerce 結構中，可藉由執行下列步驟來建立新條款物件：

1. 更新資料庫綱目，以包含新條款。
2. 更新 XSD 檔案以反映新的條款。
3. 爲條款建立新 Enterprise Bean。

4. 更新 WebSphere Commerce Accelerator 以反映新條款，或者使用 contract load 指令，以新條款來建立合約。

在下列各節中，MyTC 範例爲新條款物件。

在資料庫中登錄新條款

在您建立新條款物件時，您必須更新資料庫綱目以包含此物件。必須更新的資料庫表格爲 TCTYPE 與 TCSUBTYPE。

下列的 SQL 陳述式顯示如何在資料庫中登錄新條款的範例：

```
insert into TCTYPE (TCTYPE_ID) values ('MyTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME, DEPLOYCOMMAND)
values ('MySubTC', 'MyTC',
   'com.ibm.commerce.contract.objects.MySubTCAccessBean',
'packagename.MySubTCDeployCmd');
```

在合約 XSD 中登錄新的條款

如果要在合約中使用新的條款，您必須建立一個新的 XSD 檔案來定義新的條款。您也必須更新 Package.xsd 檔，以併入新的 XSD 檔。

如果要建立新的 XSD 檔案，請執行下列步驟：

1. 導覽至下列目錄：
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。
2. 在這個目錄中，建立一個新的 XSD 檔。以下顯示將用於範例 MyTC 的 XSD。檔案是 CustomizedBuyerContract.xsd：

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->

  <include schemaLocation="BuyerContract.xsd" />
  <complexType name="MyTCType">
        <complexContent>
            <extension base="wc:TermConditionType"/>
            </complexContent>
    </complexType>
    <element name="MySubTC" substitutionGroup="wc:AbstractCustomizedTC">
```

```
            <complexType>
            <complexContent>
                    <extension base="wc:MyTCType">
                        <sequence>
                            <element ref="wc:ProductSetPolicyRef"/>
                        </sequence>
                        <attribute name="attr1" type="normalizedString"
                             use="required"/>
                        <attribute name="attr2" type="int" use="required"/>
                    </extension>
                </complexContent>
        </complexType>
        </element>
    </schema>
```

3. 儲存新的檔案。

接下來，您必須更新 Package.xsd 來移除 BuyerContract.xsd，然後併入 CustomizedBuyerContract.xsd，方式如下：

1. 導覽至下列目錄：
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。
2. 在文字編輯程式中開啓 Package.xsd 檔。
3. 尋找關於 BuyerContract.xsd 的區段，然後依照下列方式來修改它：

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="CustomizedBuyerContract.xsd"/>
```

為條款建立新 CMP Enterprise Bean

您必須爲條款物件建立一個新的 CMP Enterprise Bean。這個 Bean 是針對條款子類型而建立的。

請注意，通常在建立新的 Enterprise Bean 時，您會將 Bean 放置到 WebSphereCommerceServerExtensionsData 專案中，而不是將它們併入到其中一個內含 WebSphere Commerce Entity Bean 的 EJB 群組。然而，在這個情況下，因爲條款所有新的 Entity Bean 都必須繼承自 WebSphere Commerce TermCondition Bean，因此您必須將新的條款 Bean 放置到 Enablement-RelationshipManagementData 專案中。下列各節說明如何使用 WebSphere Studio Application Developer 中的工具來建立新的 Enterprise Bean。

建立新的 Enterprise Bean: 如果要爲新條款建立新的 CMP Enterprise Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 以滑鼠右鍵按一下 **Enablement-RelationshipManagementData** 模組，然後選取**新建 >**（**其他 >**）**Enterprise Bean**。
   這時會開啓「Enterprise Bean 建立」精靈。
3. 在 **EJB 專案**下拉清單中，已經選取 **Enablement-RelationshipManagementData**。按一下**下一步**。
4. 在「建立 Enterprise Bean」視窗中，執行下列步驟：
   a. 選取**具有儲存器所管理的持續 (CMP) 欄位的 Entity Bean**
   b. 在 **Bean 名稱**欄位中，爲您的 Bean 輸入一個適當的名稱。在這個範例中，請輸入 `MySubTC`
   c. 在**來源資料夾**欄位中，保留已指定的預設值 (ejbModule)。
   d. 在**預設套件**欄位中，輸入 `com.ibm.commerce.contract.objects`。
   e. 按一下**下一步**。
5. 在「Enterprise Bean 明細」視窗中，執行下列步驟：
   a. 從 **Bean 超類型**下拉清單中，選取 **TermCondition**。
   b. 按一下**新增**來加入新的 CMP 屬性。
      這時會開啓「建立 CMP 屬性」視窗。請在此視窗中執行下列步驟：
      1) 在**名稱**欄位中，爲新的 CMP 欄位輸入一個適當的名稱。在這個範例中，請輸入 `attr1`。
      2) 在**類型**欄位中，請爲欄位輸入一個適當的資料類型。在這個範例中，請輸入 `String`。
      3) 選取**使用 getter 與 setter 方法來存取**勾選框。
      4) 選取**將 getter 與 setter 方法提升至遠端介面**勾選框。
      5) 清除**使 getter 成為唯讀**勾選框。
      6) 按一下**套用**。
      7) 建立另一個屬性。在**名稱**欄位中，輸入 `attr2`。
      8) 在**類型**欄位中，請爲欄位輸入一個適當的資料類型。在這個範例中，請輸入 `Integer`。
      9) 清除**使 getter 成為唯讀**勾選框。
      10) 按一下**套用**。
      11) 按一下**關閉**來關閉視窗。
6. 按下**完成**。

將新的 Bean 中的欄位對映到 TERMCOND 表格中: 下一步是將新的 Bean 中的欄位對映到 TERMCOND 表格中的直欄。如果要建立這個對映，請執行下列步驟：

1. 切換到「J2EE 導覽器」檢視畫面。
2. 展開下列資料夾：**Enablement-RelationshipManagementData > ejbModule > META-INF**。
3. 按兩下 **Map.mapxmi** 檔。
4. 在 Enterprise Bean 窗格中，展開 **TermCondition** Bean，然後展開 **MySubTC** Bean，以便檢視其屬性。
5. 在「表格」窗格中，展開 **TERMCOND** 表格，以便檢視其直欄。
6. 將 **attr1** 欄位從 MySubTC 拖曳到 TERMCOND 表格中的 **STRINGFIELD3** 直欄。
7. 將 **attr2** 欄位從 MySubTC 拖曳到 TERMCOND 表格中的 **INTEGERFIELD3** 直欄。
8. 儲存變更。

加入新的 ejbCreate 方法: 在這個步驟中，您會執行下列步驟來加入一個新的 ejbCreate 方法到 MySubTC Bean 中：

1. 在「J2EE 階層」檢視畫面中，按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 新增下列程式碼到類別中，以建立一個新的 ejbCreate(Long, Element) 方法：

```
public com.ibm.commerce.contract.objects.TermConditionKey
   ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
     javax.naming.NamingException,
     javax.ejb.RemoveException {
    _initLinks();
    super.ejbCreate (argTradingId, argElement);
    this.attr1= null;
           this.attr2 = null;
return null;
}
```

   儲存程式碼變更。

3. 您必須將新的 ejbCreate(Long, Element) 方法加入到發源介面。這樣就能在產生的存取 Bean 中提供該方法。如果要新增方法到發源介面中，請執行下列步驟：
   a. 以滑鼠右鍵按一下「大綱」檢視畫面中的 **ejbCreate(Long, Element)** 方法，然後選取 **Enterprise Bean > 提升至發源介面**。

加入新的 ejbPostCreate 方法： 接下來，執行下列步驟來建立一個新的 ejbPostCreate(Long, Element) 方法，使它具備和 ejbCreate(Long, Element) 方法相同的參數：

1. 按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 新增下列程式碼到類別中，以建立一個新的 ejbPostCreate(Long, Element) 方法：

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
     javax.naming.NamingException,
     javax.ejb.RemoveException
        {
           parseXMLElement(argElement);
        }
```

儲存程式碼變更。

新增 parseXMLElement 方法： 在這個步驟中，您必須在 MySubTCBean 中建立 parseXMLElement 方法，方式如下：

1. 按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 依下列方式來更新方法：

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
    javax.ejb.CreateException,
    javax.ejb.FinderException,
    javax.naming.NamingException,
    javax.ejb.RemoveException
{
      super.parseXMLElement(argElement);
      if (argElement == null)
  return;
      String nodeName = argElement.getNodeName();
      if (nodeName.equals("TCCopy"))
  return;

 this.attr1 = argElement .getAttribute("attr1").trim();
 this.attr2 = new Integer (argElement.
     getAttribute("attr2").trim());
 // 從 "MySubTC" 取得元素 "ProductSetPolicyRef"
 Element ePolicyReference = null;
 ePolicyReference = ContractUtil.getElementByTag(
      argElement, "ProductSetPolicyRef");

      parseElementPolicyReference(ePolicyReference);

}
```

3. 儲存您的工作。

新增 createNewVersion 方法： 在這個步驟中，您必須在 MySubTCBean 中建立一個新的 createNewVersion 方法，方式如下：

1. 按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 依下列方式來更新方法：

```
public Long createNewVersion(Long argNewTradingId) throws
  javax.ejb.CreateException,
  javax.ejb.FinderException,
  javax.naming.NamingException,
  javax.ejb.RemoveException,
  org.xml.sax.SAXException,
  java.io.IOException
{
   // 由於 tcSequence 不能空值，合約為 seqElement
   Element seqElement = ContractUtil.
      getSeqElementFromTCSequence(this.tcSequence);
   MySubTCAccessBean newTC = new MySubTCAccessBean(
      argNewTradingId, seqElement);
   Long newTCId = newTC.getReferenceNumberInEJBType();
   newTC.setInitKey_referenceNumber(newTCId);
   newTC.setMandatoryFlag(this.mandatoryFlag);
   newTC.setChangeableFlag(this.changeableFlag);
   // 設定此特定 TC 的直欄
   newTC.setAttr1(this.attr1);
   newTC.setAttr2(this.attr2);
   newTC.commitCopyHelper();
   return newTCId;
}
```

3. 儲存您的工作。

新增 getXMLString 方法: 在這個步驟中，您必須在 MySubTCBean 中建立一個新的 getXMLString 方法，方式如下：

1. 按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 依下列方式來改寫方法：

```
public String getXMLString() throws javax.ejb.CreateException,
   javax.ejb.FinderException,
   javax.naming.NamingException
{
 return getXMLString(false);
}


public String getXMLString(boolean tcdata) throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
 String xmlTC = " <MySubTC %TC_DATA% " +
  "  attr1=\"" + this.attr1 +
  "\" attr2=\"" + this.attr2.toString() + "\">" +
  "%TC_DESC%" +
  "%PARTICIPANT%" +
                        "%XML_POLICYREFERENCE%" +
```

```
 " </MySubTC>";
xmlTC = ContractUtil.replace( xmlTC, "%TC_DATA%",
     getXMLStringForTCData(tcdata));
     String xmlPolicy = getXMLStringForElementPolicyReference(
        "ProductSet") ;
     xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
        xmlPolicy );
xmlTC = ContractUtil.replaceAll( xmlTC, "%POLICY_REF_TYPE%",
    "ProductSetPolicyRef");
     return xmlTC;
}
```

3. 儲存您的工作。

新增 markForDelete 方法: 在這個步驟中，您必須在 MySubTCBean 中建立一個新的 markForDelete 方法，方式如下：

1. 按兩下 **MySubTCBean** 類別來開啓它並檢視其原始碼。
2. 依下列方式來改寫方法：

```
public void markForDelete() throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
      // 程式碼：將無法因連帶刪除關係而刪除的項目，
      // 從相關聯的表格中移除
```

3. 儲存您的工作。

更新遠端介面: 您必須確定下列方法已經新增至遠端介面，方式如下：

1. 切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Enablement-RelationshipManagementData** 專案。
3. 展開 **com.ibm.commerce.contract.objects** 套件。
4. 按兩下 **MySubTCBean** Bean。
5. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **getXMLString()** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**
6. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **getXMLString(boolean) tcdata)** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**
7. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **parseXMLElement(org.w3c.dom.Element argElement)** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**
8. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **createNewVersion(Long argNewTradingId)** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**
9. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **markForDelete()** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**

10. 儲存變更。

為 MySubTC 建立存取 Bean: 如果要建立存取 Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **Enablement-RelationshipManagementData**，然後選取**新建 > 存取 Bean**。這時會開啓「新增存取 Bean」視窗。
2. 選取**複製輔助程式**，然後按一下**下一步**。
3. 選取 ***MySubTC*** Bean，然後按一下**下一步**。
4. 從「建構子方法」下拉清單中，選取 **findByPrimaryKey(com.ibm.commerce.contract.objects.*MySubTC*Key)** 作爲建構子方法。
5. 選取「屬性輔助程式」區段中的所有屬性。
6. 按下**完成**。
7. 儲存您的工作。

加入新的字串轉換器: 您必須加入新的字串轉換器，方式如下：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組 > Enablement-RelationshipManagementData > ejbModule > META-INF**。
2. 按兩下 **ibm-ejb-access-bean.xmi** 檔。
3. 尋找關於 MySubTC 的區段。
4. 在每一個 copyHelperProperties 元素中，新增下列屬性：

   ```
   converterClassName="com.ibm.commerce.base.objects.WCSStringConverter"
   ```

5. 儲存您的工作。
6. 接下來，您必須重新產生 MySubTC 的存取 Bean，方式如下：
   a. 以滑鼠右鍵按一下 **Enablement-RelationshipManagementData**，然後選取**存取 Bean > 重新產生存取 Bean**。
   b. 選取 **MySubTC**，然後按一下**完成**。

產生部署程式碼: 您必須同時爲 MySubTC Bean 以及 TermCondition Bean 產生已部署的程式碼，方式如下：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 以滑鼠右鍵按一下 **Enablement-RelationshipManagementData** 並選取**產生 > 部署和 RMIC 程式碼**。
3. 選取 **MySubTC**，然後按一下**完成**。
4. 以滑鼠右鍵按一下 **Enablement-RelationshipManagementData** 並選取**產生 > 部署和 RMIC 程式碼**。

5. 按一下**全選**，然後按一下**完成**。

   註: 就技術上而言，您必須爲母項 Bean (TermCondition Bean) 以及所有的同輩 Bean （名稱中有 “TC” 的 Enablement-RelationshipManagementData 群組中的所有其他 Bean）重新產生已部署的程式碼。請注意，如果您已經加入新的欄位或修改現有的 TermCondition Bean 的遠端介面，則您必須爲母項 Bean 本身及其所有子項 Bean 重新產生存取 Bean。爲了簡化，前述的指示會選取這個專案中所有的 Bean。

改寫 validateContract 作業指令中的方法: 下一步是改寫 ValidateContractCmd 作業指令中的方法。在這個指令中，爲支援新條款物件，有三個方法您可改寫。包括：

- validateTCType()
  此方法會檢查可在合約中使用的條款類型。舉例來說，InvoiceTC 隸屬於帳戶，因此，不能出現在合約中。
- validateTCOccurrence()
  此方法會檢查條款的出現與否。舉例來說，就此方法的預設實作而言，合約中至少需有一項 PriceTC。
- otherValidateCheck()
  就此方法的預設實作而言爲空的。您可以另外新增其他不屬於前兩種方法範圍的檢驗。

有關如何進行這項修改的詳細資訊，請參閱第 141 頁的『自訂現有的作業指令』。

建立新的部署指令: 如果條款必須部署，您必須建立新部署指令，並將此指令登錄在資料庫中。必要時，請執行下列步驟：

1. 在本例中，新部署指令介面爲 MySubTCDeployedCmd，實作類別爲 MySubTCDeployedCmdImpl。此外，指令是包裝於 packagename 套件中。如果要登錄此指令，請發出下列的 SQL 指令：

   ```
   insert into CMDREG (STOREENT_ID, INTERFACENAME, CLASSNAME, TARGET)
   values (0, 'packagename.MySubTCDeployCmd',
   'packagename.MySubTCDeployCmdImpl', 'Local');
   ```

2. 在 packagename 套件中，建立新 MySubTCDeployedCmd 介面。此介面必須從 com.ibm.commerce.contract.commands.DeployTCCmd 指令介面延伸而來。以下說明新指令介面：

   ```
   public interface MySubTCDeployCmd extends
      com.ibm.commerce.contract.commands.DeployTCCmd
      {
          // 自訂的程式碼
      }
   ```

DeployTCCmd 中有一個受保護的參數 abTC 以及一個 getTargetStoreId() 方法。abTC 的值是 MySubTCAccessBean，而 getTargetStoreId() 方法會傳回正在部署合約的商店的 ID。

3. 在相同範例中，建立 MySubTCDeployCmdImpl 實作類別。此實作類別必須從 com.ibm.commerce.contract.commands.DeployTCCmdImpl 延伸而來。以下說明新指令實作類別：

```
public class MySubTCDeployCmdImpl
   extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
   implements MySubTCDeployCmd
   {
       // 客戶代碼
   }
```

更新 WebSphere Commerce Accelerator 以使用新的條款

一旦您建立新條款後，就可以更新 WebSphere Commerce Accelerator，以便用它來建立內含這些新條款的新合約。因爲這個原因而更新 WebSphere Commerce Accelerator 時，包括下列步驟：

1. 爲新條款建立一個新 JavaScript 檔。就本節中的範例而言，此檔案爲 Extensions.js。
2. 建立一個內含 HTML 區段的新 JSP 範本，以便讓使用者可在此區段中輸入新條款的必要資訊。就本節中的範例而言，此檔案爲 ContractMyTC.jsp。
3. 爲新條款建立一個新資料 Bean。就本節中的範例而言，此檔案爲 MyTCDataBean。
4. 將新檢視畫面登錄在 VIEWREG 表格中。
5. 更新 ContractRB_*locale*.properties 檔，以包含新資源。
6. 編輯 ContractNotebook.xml 檔，以包含新頁面。

下列各節將進一步說明上述每一個步驟。

建立新 JavaScript 檔： 更新 WebSphere Commerce Accelerator 以使用新條款的第一步是爲這些條款建立新 JavaScript 檔。如有需要，可參考下列的範例檔：

- Windows *WC_installdir*\samples\contract\Extensions.js
- Studio *WCStudio_installdir*\samples\contract\Extensions.js
- AIX Solaris Linux 400 *WC_installdir*/samples/contract/Extensions.js

若要使用這個範例檔，請將之複製到下列目錄中：

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\CommerceAccelerator.war\tools\contract

- Studio *workspace_dir*\CommerceAccelerator\Web Content\tools\contract
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/tools/contract
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/tools/contract

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱，而 *cell_name* 是 WebSphere Application Server 資料格名稱。

在這個新檔案中，您必須建立一個 JavaScript 物件以儲存新條款的資料。請見下列的程式碼片段：

```
function ContractMyTCModel() {

   this.tcReferenceNumber = "";
   this.policyReferenceNumber = "";

   this.attr1 = "";
   this.attr2 = "";

   this.policyList = new Array();
   this.selectedPolicyIndex = "0";
}
```

您也應建立一個用以提交新條款的新 JavaScript 物件。作法必須與您對 XSD 檔案所作的延伸一致。請見下列的程式碼片段：

```
function submitMyTC(contract) {

   var tcModel = get("ContractMyTCModel");

   if (tcModel != null) {

 var myTC = new Object();
 myTC.attr1 = tcModel.attr1;
 myTC.attr2 = tcModel.attr2;

 myTC.ProductSetPolicyRef = new Object();
 myTC.ProductSetPolicyRef.policyName = tcModel.policyList[
      tcModel.selectedPolicyIndex].policyName;
 myTC.ProductSetPolicyRef.StoreRef = new Object();
 myTC.ProductSetPolicyRef.StoreRef.name = tcModel.policyList[
      tcModel.selectedPolicyIndex].storeIdentity;
 myTC.ProductSetPolicyRef.StoreRef.Owner = new Object();
 myTC.ProductSetPolicyRef.StoreRef.Owner = tcModel.policyList[
      tcModel.selectedPolicyIndex].member;

      if (tcModel.tcReferenceNumber != "") {
         // 變更條款
         myTC.action = "update";
```

```
        myTC.referenceNumber = tcModel.tcReferenceNumber;
}
else {
        // 建立新條款
        myTC.action = "new";
}

contract.MySubTC = myTC;
 }

    return true;
}
```

建立新 JSP 範本： 接下來的步驟是建立一個內含 HTML 區段的新 JSP 範本，以便讓使用者可在此區段中輸入新條款所需的資訊。如有需要，可參考下列的範例檔：

- Windows *WC_installdir*\samples\contract\ContractMyTC.jsp
- Studio *WCStudio_installdir*\samples\contract\ContractMyTC.jsp
- AIX Solaris Linux 400
  WC_installdir/samples/contract/ContractMyTC.jsp

若要使用這個範例檔，請將之複製到下列目錄中：

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\CommerceAccelerator.war\ javascript\tools\contract
- Studio *workspace_dir*\CommerceAccelerator\Web Content\tools\contract
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/ javascript/tools/contract
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/CommerceAccelerator.war/javascript/tools/contract

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱，而 *cell_name* 是 WebSphere Application Server 資料格的名稱。

下列的程式碼片段顯示可用於 MyTC 上之 JSP 範本中的 HTML 區段範例。

```
<!--
////////////////////////////////////
// HTML 區段
////////////////////////////////////
-->

<BODY onLoad="onLoad()" class="content">
```

```
   <H1>
   <%= contractsRB.get("MyTCHeading") %>
   </H1>

<FORM NAME="MyTCForm">

   <%= contractsRB.get("MyTCAttr1Label") %>
   <BR>
   <INPUT type=text name=Attr1 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCAttr2Label") %>
   <BR>
   <INPUT type=text name=Attr2 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCPolicyLabel") %>
   <BR>
   <SELECT NAME="PolicyList" SIZE="1">
   </SELECT>

</FORM>
```

建立新資料 Bean: 在這個步驟中，您將建立一個會從 MySubTC 存取 Bean 中載入必要資料的新資料 Bean。下列的程式碼片段顯示程式碼中的相關區段：

```
public class MyTCDataBean extends MySubTCAccessBean
   implements SmartDataBean, Delegator {
      private java.lang.Long contractId;
      private boolean hasMyTC = false;
      private CommandContext iCommandContext;


      /**
       * MyTCDataBean 預設建構子。
       */
      public MyTCDataBean() {
      }
      /**
       * MyTCDataBean 建構子。
       */
      public MyTCDataBean(Long newContractId) {
       contractId = newContractId;
      }


      /*
       * 從 TermConditionAccessBean 移入屬性
       */
      public void populate() throws Exception {

                Enumeration myTCEnum = new TermConditionAccessBean().
                     findByTradingAndTCSubType(contractId, "MySubTC");
                if (myTCEnum != null) {
                        // 假設在此範例中合約只有一個 MyTC
```

```
                    setEJBRef(((TermConditionAccessBean)
                          myTCEnum.nextElement()).getEJBRef());
                    refreshCopyHelper();
                hasMyTC = true;
            }

}
```

將新檢視畫面登錄在 VIEWREG 表格中: 您必須將新建的檢視畫面登錄在 VIEWREG 表格中。下列的 SQL 陳述式範例顯示如何登錄新檢視畫面。

```
insert into VIEWREG(VIEWNAME,DEVICEFMT_ID,STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, HTTPS, INTERNAL)
values ('ContractMyTCPanelView', -1, 0,
   'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
   'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
   'docname=tools/contract/ContractMyTC.jsp', 1, 1)
```

更新 *ContractRB_locale.properties* 檔: 您必須以新條款的特定資訊更新下列的內容檔：

- Windows *WAS_installdir*\installedApps*cell_name*\WC_*instanceName*.ear\properties\com\ibm\commerce\tools\contract\properties\ ContractRB_*locale*.properties
- Studio *workspace_dir*\WebSphereCommerceServer\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*locale*.properties
- AIX Solaris Linux *WAS_installdir*/installedApps/*cell_name*/WC_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- 400 *WAS_userdir*/installedApps/*cell_name*/WC_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱，而 *cell_name* 是 WebSphere Application Server 資料格的名稱。

以下是您將新增到檔案中的資訊範例。

```
MyTCHeading=My TC
attr1Empty=Attribute One must be entered.
attr2Empty=Attribute Two must be entered.
attr1TooLong=Attribute One is too long.
attr2TooLong=Attribute Two is too long.
MyTCAttr1Label=Attribute One (required)
MyTCAttr2Label=Attribute Two (required)
MyTCPolicyLabel=Policy
```

編輯 *ContractNotebook.xml* 檔： 將新條款併入 WebSphere Commerce Accelerator 中的最後一個步驟是更新下列檔案，以包含新頁面。

- Windows *WC_installdir*\xml\tools\contract\ContractNotebook.xml
- Studio *WCStudio_installdir*\Commerce\xml\tools\contract\ContractNotebook.xml
- AIX Solaris Linux *WC_installdir*/xml/tools/contract/ContractNotebook.xml
- 400 *WC_installdir*/xml/tools/contract/ContractNotebook.xml

下列的程式碼片段範例顯示如何將新頁面併入到此範例中。

```
<panel name="MyTCHeading"
    url="ContractMyTCPanelView"
    parameters="contractId,accountId"
    helpKey="MC.contract.MyTCPanel.Help" />
```

匯入使用新條款的新合約

更新 WebSphere Commerce 工具以使用新條款的另一項做法是，使用合約 import 指令（有關此指令的說明，請參閱 WebSphere Commerce 線上說明）來匯入內含此條款的新合約。在匯入之後，Contract.xml 檔中的相關區段會是：

```
<MySubTC attr1="abc" attr2="123">
       <ProductSetPolicyRef policyName = "Product Set 1">
           <StoreRef name = "StoreGroup1">
               <Owner>
                   <OrganizationRef distinguishName = "o=Root Organization"/>
               </Owner>
           </StoreRef>
       </ProductSetPolicyRef>
   </MySubTC>
```

呼叫新商業原則

一旦您建立了新商業原則，且此商業原則已關聯至少一個條款物件，即可更新您的應用程式邏輯來呼叫新的商業原則指令。

商業原則指令是從控制程式與作業指令中呼叫。

指令 factory 用以呼叫商業原則指令。下列兩種 create 方法可用來呼叫商業原則指令。當只有一個商業原則指令連結該商業原則時，可使用第一種方法來呼叫商業原則指令。請見下列的程式碼片段：

```
CommandFactory createBusinessPolicyCommand(Long policyId);
```

當有多個商業原則指令連結該商業原則時，可使用第二種方法來呼叫商業原則指令。請見下列的程式碼片段：

```
CommandFactory createBusinessPolicyCommand(Long policyId, String cmdIfName);
```

在上例中，cmdIfName 用來指定所要建立之商業原則指令的介面名稱。

指令 factory 會在 POLICYCMD 表格中查閱原則物件，以判斷實作該原則的指令。此外，亦會提取表格中的任何預設內容，並在商業原則指令中將之設成 requestProperties。

下列程式碼片段顯示呼叫退款原則的範例：

```
RefundPolicyCmd  cmd;

   ////////////////////////////////////////////////////////
   // 從 refundTC 物件取得退款原則 ID，                    //
   // 並用來建立原則指令。                                 //
   ////////////////////////////////////////////////////////
 cmd = (RefundPolicyCmd) CommandFactory
         createPolicyCommand (refundTC.getRefundPolicy);

  cmd.execute()
```

建立合約

將合約模型的延伸完全整合到您的商業程序的下一步是建立一個合約，來包含參照新商業原則的條款。合約可以使用 WebSphere Commerce Accelerator 或使用其中一種合約 URL 指令 (ContractImportApprovedVersion 和 ContractImportDraftVersion) 來建立。有關建立合約的其他資訊，請參閱 WebSphere Commerce 正式作業與開發線上說明。

合約自訂實務

本節提供下列合約自訂實務內容所需要的步驟概觀：

- 啓用折讓

折讓實務內容

在這個範例實務內容中，會建立一個單一費率的折讓。由於「工具屋」範例商店並未包含與折讓實務內容相符的任何條款或原則類型，因此必須建立它們。此外，您必須建立新的商業原則以及資料庫表格來儲存折讓代碼。

實作這個折讓實務內容時，包含下列高階步驟：

1. 建立 XREBATECODE 資料庫表格以及用來從這個表格存取資訊的對應 XRebateCodeBean Entity Bean。

2. 執行下列子作業來建立新的 5DollarRebate 商業原則：
   a. 建立對應的新商業原則類型。這樣會定義新的商業原則指令要實作的介面 (RebatePolicyCmd)。
   b. 建立新的 CalculateRebateCmdImpl 商業原則指令。
   c. 在資料庫中登錄新的商業原則指令和商業原則類型。
3. 執行下列子作業來爲折讓建立一個新的條款 (RebateTC)：
   a. 在資料庫中登錄 RebateTC 條款。
   b. 更新 XSD 檔案以反映新的 RebateTC。
   c. 爲 RebateTC 建立新的 Enterprise Bean。
   d. 更新 WebSphere Commerce Accelerator 以反映新的 RebateTC。
4. 建立使用 RebateTC 的新合約。
5. 將新的商業原則整合到購物流程中。

後續各節將詳細說明上述每一個步驟。

步驟 1：建立新的表格和 Enterprise Bean

由於現有的資料庫綱目不包含折讓金額與代碼的規格，因此必須建立新的表格。一般而言，在建立新表格時，也會建立一個新的 Entity Bean，以便在存取這個表格中的資訊時使用。

在這個範例中，假設建立下面的 XREBATECODE 資料庫表格。

表 2. XREBATECODE 資料庫表格

	直欄名稱		
	REBATECODE_ID	AMOUNT	CURRENCY
範例資料	201	5	CAD
	202	10	CAD

此外，也會建立一個新的 CMP Entity Bean (XRebateCodeBean)。有關建立這個 Bean 的詳細資訊，請參閱第 56 頁的『建立新 CMP Enterprise Bean』。

步驟 2：建立 “5DollarRebate” 商業原則

如果要建立這個新的商業原則，您必須執行下列步驟：

1. 建立新的商業原則類型介面。這是 `CalculateRebateCmdImpl` 將要實作的 `RebatePolicyCmd` 介面。
2. 建立新的 `CalculateRebateCmdImpl` 商業原則指令。
3. 在資料庫中登錄新的商業原則和商業原則指令。

建立"折讓"商業原則類型: 由於沒有和折讓相對應的現有商業原則類型，因此必須建立新的商業原則類型。建立新的商業原則類型時，必須在資料庫中定義及登錄一個原則類型。同時必須更新下面的表格：

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

在這個實務內容中，如果要建立新的 REBATE 原則類型，就會使用下面的 SQL 陳述式：

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('Rebate');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('Rebate',
   'com.mycompany.mybusinesspolicycommands.RebatePolicyCmd');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('Rebate', -1,
   'Rebate policy type.');
```

因此，下表顯示 PLCYTYCMIF 表格（用來顯示原則類型及其相關商業原則指令之間的關係）的相關直欄。

表 3. 對 PLCYTYCMIF 表格所做的更新

	直欄名稱	
	POLICYTYPE_ID	BUSINESSCMDIF
範例資料	折讓	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

您也必須撰寫新的 `RebatePolicyCmd` 介面。這個介面必須延伸 `com.ibm.commerce.command.BusinessPolicyCommand` 介面。根據前一個表格的建議，請將這個介面包裝到您自己的套件中。

建立 CalculateRebateCmdImpl 商業原則指令: 如果要建立新的商業原則指令，您必須建立一個稱爲 `CalculateRebateCmdImpl` 的新指令，以延伸 `com.ibm.commerce.command.BusinessPolicyCommandImpl` 實作類別。這個指令應該實作前一個步驟中建立的 `RebatePolicyCmd` 介面。

請注意，在這個範例中，介面名稱和指令名稱並不相同。選擇這些名稱是爲了顯示出可能有許多商業原則指令實作折讓類型的商業原則。而每一個實作（亦即，每一個商業原則指令）會以唯一的方式實作折讓。

指令的邏輯是根據客戶在取貨時的特定實作而定。此外，這個 `CalculateRebateCmdImpl` 應該由您的應用程式中的個別控制程式或作業指令來呼叫。

登錄新的商業原則和新的商業原則指令: 新的商業原則必須登錄到資料庫中。您也必須登錄新的商業原則與新的商業原則指令之間的關係。

如果要登錄這項資訊，您可以使用 `com.ibm.commerce.contract.commands.PolicyAddCmd` 指令。下面是在這個實務內容中使用 PolicyAdd 指令的範例：

```
http://localhost:8080/webapp/wcs/stores/servlet/PolicyAdd?
   type=Rebate&name=5DollarRebate&plcyStoreId=-1
   &cmd_1=com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl
   &startDate=2002-05-08%2000:00:00&endDate=2003-05-09%2000:00:00
   &commonProps=rebatecode_id%3D501&URL=aRedirectURL
```

請注意，URL 保留字元必須換成其輸入內容的 ASCII 碼。因此，一般的 =（等號）符號會換成 “%3D”， &（’&’ 符號）會換成 “%26”，而空格字元會換成 “%20”。前述範例中使用的日期格式是 yyyy-mm-dd hh:mm:ss，並使用 ASCII 碼來取代 URL 保留字元。

下表顯示在執行更新之後，受影響的資料庫表格的相關直欄。

表 4. 對 POLICY 表格所做的更新

	直欄名稱				
	POLICY _ID	POLICY NAME	POLICYTYPE _ID	STOREENT _ID	PROPERTIES
範例資料	301	5 Dollar Rebate	折讓	-1	rebatecode_id= 201

請注意，該範例也假設開始日期和結束日期值是設定成 `null`。

表 5. 對 POLICYCMD 表格所做的更新

	直欄名稱		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
範例資料	301	com.mycompany. mybusinesspolicycommands. CalculateRebateCmdImpl	空值

因此，您現在已經有一個與 CalculateRebateCmd 商業原則指令相關的新商業原則，叫做 “5DollarRebate”。

步驟 3：建立 “RebateTC” 條款

建立 “RebateTC” 條款時，需要執行下列步驟：

1. 在資料庫中登錄 RebateTC 條款。

2. 更新 XSD 檔案以反映新的 RebateTC。
3. 爲 RebateTC 建立新的 Enterprise Bean。
4. 更新 WebSphere Commerce Accelerator 以反映新的 RebateTC。

在資料庫中登錄 "RebateTC" 條款: 在您建立新條款物件時，您必須更新資料庫綱目以包含此物件。必須更新的資料庫表格爲 TCTYPE 與 TCSUBTYPE。

下列的 SQL 陳述式範例顯示如何在資料庫中登錄 RebateTC：

```
insert into TCTYPE (TCTYPE_ID) values ('RebateTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME, DEPLOYCOMMAND)
values ('RebateTC', 'RebateTC',
   'com.ibm.commerce.contract.objects.RebateTCAccessBean',
    null);
```

下表顯示 TCTYPE 及 TCSUBTYPE 表格中的相關直欄的擷取內容。

表 6. 對 TCTYPE 表格所做的更新

	直欄名稱
	TCTYPE_ID
範例資料	RebateTC

表 7. 對 TCSUBTYPE 表格所做的更新

	直欄名稱			
	TCSUBTYPE _ID	TCTYPE _ID	ACCESSBEAN NAME	DEPLOY COMMAND
範例資料	RebateTC	RebateTC	com.ibm.commerce. contract.objects. RebateTCAccessBean	空值

在合約文件類型定義中登錄折讓條款: 如果要在合約中使用折讓條款，您必須建立一個新的 XSD 檔案，來定義新的條款。您也必須更新 Package.xsd 檔，以併入新的 XSD 檔。

如果要建立新的 XSD 檔案，請執行下列步驟：

1. 導覽至下列目錄：
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

2. 在這個目錄中，建立一個新的 XSD 檔。以下顯示將用於範例 RebateTC 的 XSD。檔案是 RebateCustomizedBuyerContract.xsd：

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->

  <include schemaLocation="BuyerContract.xsd" />
  <complexType name="RebateTCType">
            <complexContent>
            <extension base="wc:TermConditionType"/>
            </complexContent>
        </complexType>
    <element name="RebateTC" substitutionGroup="wc:AbstractCustomizedTC">
        <complexType>
            <complexContent>
                <extension base="wc:RebateTCType">
                    <sequence>
                        <element ref="wc:RebatePolicyRef"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>

    <element name="RebatePolicyRef" type="wc:BusinessPolicyRef" />

</schema>
```

3. 儲存新的檔案。

接下來，您必須更新 Package.xsd 來移除 BuyerContract.xsd，然後併入 RebateCustomizedBuyerContract.xsd，方式如下：

1. 導覽至下列目錄：
   - Windows *WC_installdir*\xml\trading\xsd
   - AIX Solaris Linux *WC_installdir*/xml/trading/xsd
   - 400 *WC_userdir*/instances/*instanceName*/xml/trading/xsd

   其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。
2. 在文字編輯程式中開啓 Package.xsd 檔。
3. 尋找關於 BuyerContract.xsd 的區段，然後依照下列方式來修改它：

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="RebateCustomizedBuyerContract.xsd"/>
```

為 RebateTC 建立新的 Enterprise Bean: 您必須爲新 RebateTC 建立一個新的 Enterprise Bean。這個新的 Bean 應該繼承自 WebSphere Commerce TermCondition Bean。

條款的新 Enterprise Bean 通常是以子類型命名。請注意，在這種情況下，條款子類型與條款類型相同，因此，Bean 的名稱與條款類型相同。

下表顯示關於必須建立的新 Bean 的一般資訊。有關 Bean 的其他明細（包括要改寫的方法），請參閱第 157 頁的『爲條款建立新 CMP Enterprise Bean』。

表 8.

屬性	值
EJB 專案	Enablement-RelationshipManagementData
Bean 類型	具有儲存器所管理的持續欄位的 Entity Bean
Bean 名稱	RebateTC
套件	com.ibm.commerce.contract.objects
Bean 超類型	TermCondition
Bean 類別	RebateTCBean

在新的 Bean 中，建立下面的值要使用的三個 CMP 欄位：

- 折讓代碼 ID
- 金額
- 貨幣

更新 WebSphere Commerce Accelerator 以併入 RebateTC: 一旦您建立新條款後，就可以更新 WebSphere Commerce Accelerator，以便用它來建立內含這些新條款的新合約。有關如何更新這個工具的資訊，請參閱第 165 頁的『更新 WebSphere Commerce Accelerator 以使用新的條款』。

步驟 4：建立新合約

您必須建立一個新合約，以併入參照 “5DollarRebate” 商業原則的 “RebateTC” 條款。您可以使用 WebSphere Commerce Accelerator 或 XML 來建立新合約。WebSphere Commerce 正式作業與開發線上說明中說明了每一種建立新合約的方法。

下表顯示在建立合約之後，對 TERMCOND 和 POLICYTC 資料庫表格的相關直欄所做的更新。

表 9. 對 TERMCOND 表格所做的更新

	直欄名稱		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
範例資料	25	901	RebateTC

表 10. 對 POLICYTC 表格所做的更新

	直欄名稱	
	POLICY _ID	TERMCOND_ID
範例資料	301	901

步驟 5：將新的商業原則整合到購物流程中

在這個實務內容中，會假設在商店中加入一個新的頁面，以便讓客戶登入並要求折讓。當客戶按一下以要求折讓時，應該會呼叫一個指令來呼叫新的 `RebatePolicyCmd` 介面。例如，會有一個新的 ClaimRebateCmd 控制程式指令來呼叫 `RebatePolicyCmd`。然後會找到正確的商業原則，並且（在此情況下）套用 “5DollarRebate” 商業原則。

第 3 篇 開發環境

第 8 章 開發環境

本章介紹用於自訂 WebSphere Commerce 應用程式的主要開發工具。

典型的開發環境

如果要建立自訂程式碼以搭配 WebSphere Commerce Business Edition 使用，我們所推薦的開發套裝軟體爲 WebSphere Commerce Studio, Business Developer Edition 產品。如果要建立自訂程式碼以搭配 WebSphere Commerce Professional Edition 使用，我們所推薦的開發套裝軟體爲 WebSphere Commerce Studio Professional Developer Edition 產品。這兩種套裝軟體包含您建立自訂程式碼與執行 Web 開發作業所需的所有工具。通常本書會將這些產品通稱爲 WebSphere Commerce Studio。

WebSphere Commerce Studio 有四個主要元件：

1. 在 WebSphere Studio Application Developer 中使用的 WebSphere Commerce 工作區
2. 開發資料庫
3. 檔案系統資產
4. WebSphere Studio Application Developer 的 WebSphere Commerce 外掛程式

在這個開發環境中，您可以建立自訂程式碼，並且在 WebSphere 測試環境的環境定義中測試它。

如果要在開發環境中建立商店，只需要啓動管理主控台（在 WebSphere Studio Application Developer 的本端環境中定義的 WebSphereCommerceServer 測試伺服器上執行），然後根據其中一個範例商店，使用該工具來公佈商店。另外，您也可以建立自己的商店。

除了已提供的 WebSphere Commerce 工作區外，WebSphere Commerce Studio 也提供額外的工具和外掛程式。其中包含下列外掛程式：

- 「配置管理程式」外掛程式，可協助您管理 WebSphere Commerce（和 WebSphere Commerce Payments）實例。
- WebSphere Commerce 線上說明外掛程式，可讓您從 WebSphere Studio Application Developer 內部來存取 WebSphere Commerce 線上說明。此外，透過這個外掛程式，您可以在執行 WebSphere Commerce 工具（例如，管理主控台）時按下 F1 來啓動與 WebSphere Commerce 環境定義相關的線上說明。

- WebSphere Commerce API 參照資訊外掛程式，可讓您從 WebSphere Studio Application Developer 內部來存取 WebSphere Commerce API 參照資訊。

另外也提供了一個 WebSphere Commerce Enterprise Bean 轉換工具。您可以利用這個工具，使用與目標正式作業資料庫不同的開發資料庫來開發 Enterprise Bean。例如，這個工具可讓您使用本端 DB2 資料庫來進行開發，但是您的正式作業資料庫可以是在 iSeries 平台上，或甚至是 Oracle 資料庫中。

WebSphere Studio Application Developer

WebSphere Commerce Studio 套件包含 WebSphere Studio Application Developer，這是 IBM 提供的核心開發環境。它可以提供最佳慣例、範本、程式碼產生以及同等級中最廣泛的開發環境，來協助您最佳化及簡化 Java2 Enterprise Edition (J2EE) 與 Web 服務的開發。這個複雜的整合開發環境 (IDE) 可以在一個開發環境中包含對 Java 元件、Enterprise Bean、Servlet、JSP 檔、HTML、XML 和 Web 服務的整合支援。

在許多其他優秀的特性中，同時也包含許多本端測試工具，讓您快速產生測試用戶端。另外還包括一個完整的 WebSphere Application Server 測試環境，讓您在本端環境中以端對端的方式測試您的程式碼。

iSeries 的開發環境

簡單來說，您並不需要一個特殊的開發環境來爲 iSeries 建立自訂程式碼。設定開發工作站的程序與 *WebSphere Commerce Studio 安裝手冊*中所列出的程序相同。您使用的本端開發資料庫應該是 DB2。使用這個配置時，您可以使用本端 DB2 資料庫與本端 WebSphereCommerceServer 測試伺服器來建立及測試自訂程式碼。

在測試判斷出自訂程式碼功能在測試伺服器的環境定義中符合您的需求之後，您必須將它部署到在 iSeries 平台上執行的目標 WebSphere Commerce Server 上。爲了解決 Windows 和 iSeries 平台上的資料庫之間的差異，同時也提供了一個 WebSphere Commerce Enterprise Bean 轉換工具。第 183 頁的『WebSphere Commerce Enterprise Bean 轉換工具的概觀』中有關於這個工具的其他資訊。

當正式作業環境使用 Oracle 資料庫時使用本端 DB2 資料庫來進行開發

即使正式作業環境的資料庫將會是 Oracle 資料庫，程式開發人員仍然可以在開發機器上使用本端 DB2 資料庫。在此情況下，會使用 WebSphere Commerce Enterprise Bean 轉換工具將 Bean 的 meta 資料從 DB2 格式轉換成 Oracle 格式。第 183 頁的『WebSphere Commerce Enterprise Bean 轉換工具的概觀』中有關於這個工具的其他資訊。

WebSphere Commerce Enterprise Bean 轉換工具的概觀

一般而言，在兩種情況下會使用 Enterprise Bean 轉換工具：

- 如果目標正式作業環境是在 iSeries 平台上執行。
- 如果目標正式作業資料庫是 Oracle 資料庫，而開發機器使用本端 DB2 資料庫。

這個 WebSphere Commerce 特有的工具可讓您針對一種類型的資料庫來進行開發，然後部署到另一種資料庫類型。利用這個工具，就可以將 Enterprise Bean 的 meta 資料 轉換成目標資料庫適用的格式和資訊，而已部署的程式碼也是利用這個新的 meta 資料 來產生。有關如何使用這個工具的逐步明細，請參閱第 188 頁的『使用轉換來建立 EJB JAR 檔』。

開發環境內的付款選項

在先前的 WebSphere Commerce Studio 版本中，提供了一個測試付款方法讓程式開發人員可以在測試環境商店中完成購買，而不必呼叫遠端付款提供者。現在，在 WebSphere Commerce Studio 5.5 版中， WebSphere Commerce Payments 元件可以在測試環境中執行。

這表示您現在可以選擇使用本端 WebSphere Commerce Payments 實例，或者您可以將您的 WebSphere Commerce 開發實例配置成使用遠端 WebSphere Commerce Payments 實例。

在預設的情況下，當您安裝 WebSphere Commerce Studio 時，會建立本端 WebSphere Commerce Payments 實例。另外，如果這個實例在您公佈範例商店時正在執行，商店就會自動配置成使用該本端 WebSphere Commerce Payments 實例。

有關配置付款選項的資訊，可以在 *WebSphere Commerce Studio 安裝手冊*中取得。

第 9 章 部署明細

當您在 WebSphere Commerce Studio 中建立自訂程式碼，並且在 WebSphere 測試環境中完成測試後，就必須將自訂程式碼部署到在 WebSphere 測試環境外部執行的目標 WebSphere Commerce Server。這個目標 WebSphere Commerce Server 可以在部署機器的本端環境中執行，也可以在另一部機器上執行（使用相同或不同的作業系統）。

本章說明將自訂程式碼部署到於 WebSphere 測試環境外部執行的目標 WebSphere Commerce Server 時所需要的步驟。

本章分爲許多小節，說明如何部署您的自訂應用程式可能包含的各種程式碼類型。其中包含下列作業：

- 部署 Enterprise Bean
- 部署指令與資料 Bean
- 部署商店資產
- 更新目標資料庫

上述每一個作業都將在後續各節中詳細說明。

部署步驟的使用者許可權需求

AIX Solaris Linux 您應該使用您在準備 WebSphere Commerce 安裝程序時建立的非 root 使用者 ID，對目標 WebSphere Commerce Server 執行所有的部署步驟（存取控制更新除外）。此外，請確定您的檔案資產（例如 JAR 檔）以及放置這些資產的目錄已授與此使用者讀取、寫入與執行檔案的許可權。

有關執行存取控制更新所需要的使用者許可權，請參閱 *WebSphere Commerce Security Guide*中的「將您的 XML 變更載入資料庫中」主題。

遞增式部署

本章中所說明的部署類型是*遞增*式部署。在遞增式部署中，您必須已經在目標 WebSphere Commerce 伺服器上安裝 WebSphere Commerce 企業應用程式。然後在部署程序中，您只能將下列資產（指令、資料 Bean、Enterprise Bean 等）部署到現有的企業應用程式中。

如果要在您的目標 WebSphere Commerce 伺服器上建立起始企業應用程式，就必須安裝 WebSphere Commerce，然後使用「配置管理程式」來建立您的 WebSphere Commerce 實例（藉以建立 WebSphere Commerce 企業應用程式）。

部署 Enterprise Bean

本節說明如何部署 Enterprise Bean。這些 Bean 可以是您爲 e-commerce 應用程式新建立的 Enterprise Bean，也可以是您已經修改的 WebSphere Commerce Entity Bean。在任何一種情況下，部署步驟基本上都是相同的。

瞭解 WebSphere Commerce 應用程式的部署程序的其中一個主要因素，是爲了瞭解用於自訂 WebSphere Commerce 程式碼的包裝綱目。尤其是您不需要在 WebSphere Commerce 工作區中建立新的 EJB 專案。新的 Enterprise Bean 會放置到 WebSphereCommerceServerExtensionsData 專案中，而已修改的 WebSphere Commerce Enterprise Bean 的自訂程式碼仍然會保留在原始的 WebSphere Commerce EJB 專案中。

部署程序有兩個主要步驟：

- 建立 EJB JAR 檔
- 更新目標 WebSphere Commerce Server 中的 EJB JAR 檔

建立 EJB JAR 檔

視您的部署實務內容，建立 EJB JAR 檔有下面兩種不同的方法：

- 如果您要建立的 EJB JAR 檔是要部署到與您的開發環境使用相同資料庫類型的目標 WebSphere Commerce Server，請遵循第 186 頁的『不使用轉換來建立 EJB JAR 檔』中的指示。
- 如果您要建立的 EJB JAR 檔是要部署到與您的開發環境使用不同資料庫類型的目標 WebSphere Commerce Server，請遵循第 188 頁的『使用轉換來建立 EJB JAR 檔』中的指示。

不使用轉換來建立 EJB JAR 檔

如果要建立 EJB JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**），然後切換到「J2EE 導覽器」檢視畫面。
2. 展開內含您要部署的 Bean 的 EJB 專案，方式如下：
   - 如果您已經建立新的 Enterprise Bean，請展開 **WebSphereCommerceServerExtensionsData** EJB 專案。

- 如果您已經修改 WebSphere Commerce Entity Bean，請展開內含已修改的 Bean 的專案。比方說，如果您已經修改「使用者」Bean，請展開 **Member-MemberManagementData** EJB 專案。

3. 按兩下 **EJB 部署描述子**。
4. 在選取「概觀」標籤之後，捲動至窗格底端，尋找 **WebSphere 連結**區段。
5. 在**資料來源 JNDI** 名稱欄位中，輸入目標 WebSphere Commerce Server 的資料來源 JNDI 名稱。以下是範例值：

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 DB2 資料庫，而 WebSphere Commerce 實例名稱是 “demo”

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 Oracle 資料庫，而 WebSphere Commerce 實例名稱是 “demo”。

「資料來源 JNDI」名稱的值是藉由新增 “jdbc/” 到目標 WebSphere Commerce Server 的資料來源名稱所建立的。您可以開啓目標 WebSphere Commerce Server 上的 *instanceName*.xml 檔，並搜尋檔案中的 `DatasourceName=`，來驗證資料來源名稱。

6. 儲存您的部署描述子變更 (Ctrl + S)。
7. 在「J2EE 導覽器」檢視畫面中，以滑鼠右鍵按一下 EJB 專案（WebSphereCommerceServerExtensionsData 或包含已修改的 WebSphere Commerce Entity Bean 的專案），然後選取**匯出**。
   這時會開啓「匯出」精靈。
8. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **EJB JAR 檔**，然後按一下**下一步**。
   b. **您要匯出的資源**？的值會預先移入 EJB 專案的名稱。請保留這個值。
   c. 在**您要將資源匯出至**？欄位中，輸入要使用的完整 JAR 檔名。例如，輸入 `C:\ExportTemp\`*`JarFileName`*`.jar`，其中 *JarFileName* 是您的 JAR 檔名稱。如果您已經建立新的 Enterprise Bean，您應該輸入 *`yourDir`*`\WebSphereCommerceServerExtensionsData.jar`。如果您已經修改現有的 WebSphere Commerce 公用 Entity Bean，就必須爲這個 EJB 群組使用預先定義的 JAR 檔名。比方說，如果您的修改是在 Member-MemberManagementData EJB 模組中，請輸入 *`yourDir`*`\Member-MemberManagementData.jar`。
   d. 確定**匯出原始檔**並未選取。
   e. 按一下**完成**。

9. 在建立 JAR 檔之後，請開啓 EJB 部署描述子，並將您在步驟 5 中的修改還原成本端測試伺服器所需要的設定。儲存變更。

使用轉換來建立 EJB JAR 檔

如果要轉換 meta 資料 並建立 EJB JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**），然後切換到「J2EE 導覽器」檢視畫面。
2. 展開內含您要部署的 Bean 的 EJB 專案，方式如下：
   - 如果您已經建立新的 Enterprise Bean，請展開 **WebSphereCommerceServerExtensionsData** 專案。
   - 如果您已經修改 WebSphere Commerce Entity Bean，請展開內含已修改的 Bean 的專案。比方說，如果您已經修改「使用者」Bean，請展開 **Member-MemberManagementData** EJB 專案。
3. 按兩下 **EJB 部署描述子**。
4. 在選取「概觀」標籤之後，捲動至窗格底端，尋找 **WebSphere 連結**區段。
5. 在**資料來源 JNDI** 名稱欄位中，輸入目標 WebSphere Commerce Server 的資料來源 JNDI 名稱。以下是範例值：

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 DB2 資料庫，而 WebSphere Commerce 實例名稱是 “demo”

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 Oracle 資料庫，而 WebSphere Commerce 實例名稱是 “demo”。

「資料來源 JNDI」名稱的值是藉由新增 “jdbc/” 到目標 WebSphere Commerce Server 的資料來源名稱所建立的。您可以開啓目標 WebSphere Commerce Server 上的 *instanceName*.xml 檔，並搜尋檔案中的 `DatasourceName=`，來驗證資料來源名稱。

6. 儲存您的部署描述子變更 (Ctrl + S)。
7. 關閉 WebSphere Studio Application Developer。
8. 在指令提示下，切換至下列目錄：

   WCStudio_installdir\Commerce\bin

9. 輸入下列指令：

   ejbDeploy.bat *projName outputJarName mapFile workspace_dir eclipseDir ejbDeployXmlFile WCStudio_commercedir*

其中：

- *projName* 是要轉換的 EJB 專案的名稱。
- *outputJarName* 是輸出 JAR 檔的完整名稱。請注意，您應該使用已經在 WebSphere Commerce 企業應用程式中的 JAR 檔。請見下列範例：

  ```
  C:\ExportTemp\WebSphereCommerceServerExtensionsData.jar
  ```

- *mapFile* 是對映檔的名稱。下列檔案是對映檔的範例：

  ```
  WCStudio_installdir\Commerce\properties\com\ibm\commerce\
     metadata\conversion\oracle.mapping
  WCStudio_installdir\Commerce\properties\com\ibm\commerce\
     metadata\conversion\as400.mapping
  ```

- *workspace_dir* 是您現行開發工作區的目錄。
- *eclipseDir* 是 eclipse 目錄的路徑。在預設的情況下是

  ```
  WCStudio_installdir\Studio5\eclipse
  ```

- *ejbDeployXmlFile* 是完整的 ejbDeploy.xml 檔。在預設的情況下是

  ```
  WCStudio_installdir\Commerce\xml\ejbDeploy.xml
  ```

  註： 執行這個指令時，您可能會看到錯誤，指出無法展開某些檔案。這個問題並不嚴重。

- *WCStudio_commercedir* 是安裝 WebSphere Commerce Studio 時所建立的 Commerce 目錄的路徑。在預設的情況下是

  ```
  WCStudio_installdir\Commerce
  ```

  詳細的目錄則是

  ```
  C:\WebSphere\CommerceStudio55\Commerce
  ```

以下是指定所有值的這個指令的用法範例：

```
ejbDeploy.bat WebSphereCommerceServerExtensionsData
WebSphereCommerceServerExtensionsData.jar
C:\WebSphere\CommerceStudio55\Commerce\properties\com\ibm\
   commerce\metadata\conversion\oracle.mapping
C:\WebSphere\workspace_db2 C:\WebSphere\Studio5\eclipse
C:\WebSphere\CommerceStudio55\Commerce\xml\ejbDeploy.xml
C:\WebSphere\CommerceStudio55\Commerce
```

請注意，換行只是爲了方便顯示。

10. 開啓 WebSphere Commerce Studio 並開啓 EJB 部署描述子，然後將您在步驟 5 中的修改還原成本端測試伺服器所需要的設定。儲存變更。
11. 如果您正在收集要部署到單一目錄（例如，C:\ExportTemp）的所有資產，請將新建立的 EJB JAR 檔部署到該目錄中。

更新目標 WebSphere Commerce Server 上的 EJB JAR 檔

下一步是將新建立的 EJB JAR 檔複製到目標 WebSphere Commerce Server 的適當位置。

如果要更新目標 WebSphere Commerce Server 上的 EJB JAR 檔，請執行下列步驟：

1. 停止在 WebSphere Application Server 中執行的 WebSphere Commerce 實例。請參閱您的平台與資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於如何停止這個實例的詳細資料。
2. 在 WebSphere Commerce 實例中找出原始的 EJB JAR 檔。例如，尋找下列檔案：

   - Windows *WAS_installdir*\installedApps*cellName*\WC_*instance_name*.ear*JarFileName*.jar
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/*JarFileName*.jar
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/*JarFileName*.jar

   其中

   - *instance_name* 是您的 WebSphere Commerce 實例的名稱。
   - 400 *WAS_node_name* 代表安裝 WebSphere Application Server 產品的 iSeries 系統。
   - *JarFileName* 是包含自訂程式碼的 JAR 檔的名稱。
3. 製作原始 EJB JAR 檔的副本。
4. 將新的 EJB JAR 檔從開發機器複製到步驟 2 中的位置。
5. 如果您已經修改 EJB 部署描述子，請執行下列步驟：
   a. 尋找這個 WebSphere Application Server 資料格的部署儲存庫（META-INF 目錄）。目錄的格式通常如下：

      - Windows *WAS_installdir*\config\cells*cellName*\applications\WC_*instance_name*.ear\deployments\WC_*instance_name**EJBModuleName*.jar\META-INF
      - AIX Solaris Linux *WAS_installdir*/config/cells/*cellName*/applications/WC_*instance_name*.ear/deployments/WC_*instance_name*/*EJBModuleName*.jar/META-INF

- 400 *WAS_userdir*/config/cells/*WAS_node_name*/applications/WC_*instance_name*.ear/deployments/WC_*instance_name*/*EJBModuleName*.jar/META-INF

其中

- *instance_name* 是您的 WebSphere Commerce 實例的名稱。
- 400 *WAS_node_name* 代表安裝 WebSphere Application Server 產品的 iSeries 系統。
- *EJBModuleName* 是已經修改的 EJB 模組的名稱。

以下是 META-INF 目錄的平台特定範例：

- Windows *WAS_installdir*\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\Member-MemberManagementData.jar\META-INF
- AIX Solaris Linux *WAS_installdir*/config/cells/myCell/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF
- 400 *WAS_userdir*/config/cells/myNode/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF

其中

- myCell 是 WebSphere Application Server 資料格的名稱
- demo 是 WebSphere Commerce 實例的名稱
- Member-MemberManagementData 是已經修改的 EJB 模組的名稱
- 400 myNode 是 WebSphere Application Server 節點的名稱

b. 備份前述目錄中的所有檔案。

c. 使用工具來開啓 *JarFileName*.jar 檔並檢視其內容。

d. 將 META-INF 目錄的內容從 *JarFileName*.jar 檔擷取到步驟 5a 中的目錄裡。

6. 根據您的平台與資料庫所適用的 *WebSphere Commerce 安裝手冊*，重新啓動 WebSphere Commerce 實例。

部署指令與資料 Bean

當您執行下列任何作業時，應該將自訂程式碼包裝到 WebSphereCommerceServerExtensionsLogic 專案中：

- 建立新指令
- 建立新的資料 Bean
- 修改現有的 WebSphere Commerce 指令
- 修改現有的 WebSphere Commerce 資料 Bean

您不可以直接修改指令或資料 Bean 的現有類別（或包含現有類別的專案）。

部署自訂的指令和資料 Bean 時，必須執行下列步驟：

- 建立 JAR 檔
- 更新目標 WebSphere Commerce Server 中的 JAR 檔

上述步驟將在後續各節中詳細說明。

如果您的自訂程式碼需要更新指令登錄、新的存取控制原則或其他資料庫更新，請務必參閱第 195 頁的『更新目標資料庫』。

建立 JAR 檔

如果要建立 JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**），然後切換到「J2EE 導覽器」檢視畫面。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**匯出**。
   這時會開啓「匯出」精靈。
3. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **JAR 檔**，然後按一下**下一步**。
   b. **選取要匯出的資源**？下面的左窗格會預先移入專案名稱。請保留這個值。
   c. 在**選取要匯出的資源**？下面的右窗格中，確定只有選取下列資源：
      - .classpath
      - .project
      - .serverPreference
   d. 確定**匯出產生的類別檔和資源**已經選取。
   e. *請勿*選取**匯出 Java 原始檔和資源**。

f. 在**選取匯出目的地**欄位中，輸入要使用的完整 JAR 檔名。例如，輸入 `C:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar`。請注意，JAR 檔名必須是 WebSphereCommerceServerExtensionsLogic.jar。

g. 按一下**完成**。

一旦順利建立 JAR 檔之後，下一步就是將檔案轉送到目標 WebSphere Commerce Server 的適當位置中。

更新 WebSphere Commerce Server 上的 JAR 檔

下一步是將新建立的 JAR 檔複製到目標 WebSphere Commerce Server 的適當位置。

如果要更新 JAR 檔，請執行下列步驟：

1. 停止在 WebSphere Application Server 中執行的 WebSphere Commerce 實例。請參閱您的平台與資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於如何停止這個實例的詳細資料。
2. 在 WebSphere Commerce 實例中找出原始的 EJB JAR 檔。例如，尋找下列檔案：
   - Windows *WAS_installdir*\installedApps*cellName*\WC_*instance_name*.ear\WebSphereCommerceServerExtensionsLogic.jar
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/WebSphereCommerceServerExtensionsLogic.jar
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/WebSphereCommerceServerExtensionsLogic.jar

   其中
   - *instance_name* 是您的 WebSphere Commerce 實例的名稱。
   - 400 *WAS_node_name* 代表安裝 WebSphere Application Server 產品的 iSeries 系統。
3. 製作原始 JAR 檔的副本並放置到在備份位置中。
4. 將 JAR 檔從開發機器複製到步驟 2 中的位置。
5. 根據您的平台與資料庫所適用的 *WebSphere Commerce 安裝手冊*，重新啓動 WebSphere Commerce 實例。

部署商店資產

商店資產包含下列這類資產：

- JSP 範本
- HTML 檔
- 影像檔
- XML 檔
- 內容檔和資源連結

這些資產必須從您的開發環境部署到目標 WebSphere Commerce Server 上。其中包括下列步驟：

- 從 WebSphere Studio Application Developer 匯出商店資產
- 將資產轉送到目標 WebSphere Commerce Server

上述步驟將在後續各節中詳細說明。

匯出商店資產

如果要從開發環境匯出商店資產，請執行下列步驟：

1. 開啓 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**），然後切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Stores** 資料夾。
3. 以滑鼠右鍵按一下 **Web Content** 資料夾，然後選取**匯出**。
   這時會開啓「匯出精靈」。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取**檔案系統**，然後按一下**下一步**。
   b. 選取您要部署的所有資源。亦即，選取所有的 JSP 範本、HTML 檔、影像檔、內容檔和其他需要部署的商店資產。
   c. 選取**為選取的檔案建立目錄結構**。
   d. 在**目錄**欄位中，輸入要放置這些資源的暫時目錄。例如，輸入 `C:\ExportTemp\StoreAssets`
   e. 按下**完成**。

下一步是將這些資源複製到目標 WebSphere Commerce Server 上的適當位置。

轉送商店資產

如果要將商店資產從您的開發機器轉送到目標 WebSphere Commerce Server，請執行下列步驟：

1. 根據您要部署的資產類型以及您的特定配置明細而定，您可能需要停止正在 WebSphere Application Server 中執行的 WebSphere Commerce 實例。如果您不確定您的特定部署實務內容是否需要重新啓動，就應該停止實例。請參閱您的平台與資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於如何停止這個實例的詳細資料。
2. 在目標機器上，尋找 Stores.war 目錄。以下是這個目錄的範例：
   - Windows *WAS_installdir*\installedApps*cellName*\ WC_*instance_name*.ear\Stores.war
   - AIX Solaris Linux *WAS_installdir*/installedApps/*cellName*/WC_*instance_name*.ear/Stores.war
   - 400 *WAS_userdir*/installedApps/*WAS_node_name*/WC_*instance_name*.ear/Stores.war

   其中
   - *instance_name* 是您的 WebSphere Commerce 實例的名稱。
   - 400 *WAS_node_name* 代表安裝 WebSphere Application Server 產品的 iSeries 系統。
3. 將在"匯出商店資產"匯出的檔案複製到 Stores.war 目錄中。
4. 如果您之前已經停止您的 WebSphere Commerce 實例，請啓動實例。請參閱您的平台與資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於如何啓動這個實例的詳細資料。。

更新目標資料庫

若您的目標 WebSphere Commerce Server 所用的資料庫和您開發機器所用的不同，您必須在目標 WebSphere Commerce Server 所用的資料庫上執行當初對開發資料庫所做的全部更新。這包括：新增或已修改指令或檢視畫面的任何更新，您已建立的其他表格的更新，以及爲已經建立的任何新資源建立存取控制原則的更新。

400 您需要有一個公用程式來執行 SQL 陳述式。其中一種做法是使用 IBM iSeries Access for Windows。如果要開啓此公用程式，請執行下列步驟：

1. 開啓 **iSeries 導覽器**。
2. 在開啓「作業導引器」後，會要求您簽入到特定系統中。確定您已選取目標 iSeries 機器，並使用 WebSphere Commerce 實例使用者設定檔和密碼。這可確保 WebSphere Commerce 實例使用者設定檔擁有您所建的所有新表格。
3. 在左邊的畫面中，展開您的 iSeries 系統，然後展開**資料庫**。以滑鼠右鍵按一下「關聯式資料庫」，然後從下拉清單中選取**執行 SQL Script**。

這時會開啓「執行 SQL Script」視窗。透過這個視窗，您可以在 SQL 陳述式中剪貼，或開啓 SQL Script。您可以使用**連線**選項下面的 **JDBC 設定**選項來設定您的預設綱目。

存取控制更新

存取控制資訊是包含在資料庫中，這種特殊資訊類型不一定是從開發環境直接複製到目標環境。尤其是在開發環境中，您可能會決定使用非常自由、但卻不適用於正式作業（或下一階段的測試）環境的存取控制原則。舉例來說，在開發環境的限制下，新指令的原則應該設定成讓所有使用者都能執行指令，但是這種設定在其他地方可能不適合。

因此，在將存取控制資訊從開發環境複製到目標環境之前，您應該考慮新環境的存取控制需求，藉以調整您的原則。

有關載入存取控制原則（包括各平台的指令語法以及目錄許可權需求）的資訊，請參閱 *WebSphere Commerce Security Guide*。

第 4 篇 指導教學

下列指導教學將介紹爲 WebSphere Commerce 應用程式建立自訂程式碼時的各種相關作業。與開發相關的步驟是在 WebSphere Commerce Business Edition 或 WebSphere Commerce Studio Professional Developer Edition 開發環境中執行。部署步驟是針對在 Windows 2000 上執行的 WebSphere Commerce（與您的開發環境對應的 Business 或 Professional Edition）來執行。

第 10 章 指導教學：建立新商業邏輯

本指導教學的設計是爲了告訴您在建立新的商業邏輯時所需要的步驟。其中建立的資產類型包括新的檢視畫面、新的控制程式指令、新的作業指令、新的資料 Bean 以及新的 Entity Bean。本指導教學使用開發小型介面的實務內容，讓使用者修改紅利積點的結餘。此實務內容僅供示範，並不會影響反映出建置 loyalty program 應用程式所需要的邏輯。相反的，從這個指導教學中，您將會學習在建立先前列出的每一種程式碼資產類型時常用的開發步驟。

本指導教學可分爲下列子作業：

1. 準備您的工作區
2. 建立新的檢視畫面
3. 建立新的控制程式指令
4. 將資訊從控制程式指令傳送到檢視畫面
5. 剖析及驗證控制程式指令中的 URL 參數
6. 建立新的作業指令
7. 修改新的作業指令
8. 建立新的 Enterprise Bean：
   a. 建立新的表格
   b. 建立新的 CMP Enterprise Bean
   c. 將新的 Bean 對映到表格中以及建立綱目
   d. 建立相關的存取 Bean
   e. 產生已部署的程式碼
   f. 使用通用測試用戶端來測試 Bean
9. 將 Bonus Bean 整合到 MyNewControllerCmd 中
   a. 修改 MyNewTaskCmdImpl 類別的 performExecute 方法，以計算新紅利積點，並將點數儲存到 XBONUS 表格中。
   b. 新增 getResources 方法到 MyNewControllerCmdImpl 類別中，以傳回指令所用的資源清單。包含此方法其目的在於存取控制。
   c. 建立 BonusDataBean，以便在 JSP 範本中輕易顯示紅利積點。
   d. 爲新資源建立新的存取控制原則。
   e. 修改 MyNewJSPTemplate.jsp 檔，讓使用者輸入紅利積點以及顯示結果。

f. 測試整合的程式碼。

10. 將前述的所有程式碼、存取控制原則、JSP 範本、影像檔和資源連結部署到在 WebSphere Application Server 中執行的目標 WebSphere Commerce Server。

上述每一個步驟都將在後續各節中逐步詳細說明。

找出範例程式碼

在開始本指導教學前，請下載 WC_SAMPLE_55.zip 套件，這個套件中包含這些程式設計指導教學的起點。請將這個檔案儲存在您的開發機器上。例如，您可以將檔案儲存到 *WCStudio_installdir* 目錄中：

這個套件與 *WebSphere Commerce 程式設計與指導教學*都位於下列網站中：

http://www.ibm.com/software/commerce/library/

準備您的工作區

在這個步驟中，您會將範例程式碼匯入到您的 WebSphere Commerce 工作區。這個範例程式碼是指導教學的起點。

如果要準備您的工作區，請執行下列步驟：

1. 確定您已經安裝 WebSphere Commerce Studio 5.5 版，而且已經完成開發環境的配置。您也應該已經根據開發環境中的「流行館」範例商店（這是消費者市場模型的範例）來公佈了一家商店。有關如何公佈商店的指示，可以在 WebSphere Commerce 正式作業與開發線上說明中找到。
2. 執行下列步驟，將範例程式碼匯入到您的工作區中：
   a. 啓動 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**）。
   b. 切換到「J2EE 視景」（**視窗 > 開啓視景 > J2EE**），然後選取「J2EE 導覽器」檢視畫面。
   c. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
   d. 用滑鼠右鍵按一下 **src** 資料夾，然後選取**匯入**。
      這時會開啓「匯入」精靈。
   e. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
   f. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案是位於下列位置：
      yourDirectory\WC_SAMPLE_55.zip
      其中 *yourDirectory* 是您下載套件的目錄。

g. 按一下**取消全選**，然後展開目錄，選取要匯入下列檔案：
   - com\ibm\commerce\sample\commands\ MyNewControllerCmd.java
   - com\ibm\commerce\sample\commands\ MyNewControllerCmdImpl.java
   - com\ibm\commerce\sample\commands\MyNewTaskCmd.java
   - com\ibm\commerce\sample\commands\MyNewTaskCmdImpl.java
   - com\ibm\commerce\sample\databeans\MyNewDataBean.java

h. 在**資料夾**欄位中，已經指定 `WebSphereCommerceServerExtensionsLogic/src` 資料夾。請保留這個值。

i. 按下**完成**。

3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**重新建置專案**。

4. 將指導教學的 JSP 範本匯入到適當的目錄中，步驟如下：

   a. 在「J2EE 導覽器」檢視畫面中，展開 **Stores** 專案。然後展開 **Web Content > *FashionFlow_name*** 其中 *FashionFlow_name* 是您根據「流行館」範例商店所建立的商店名稱。

如果您剛剛才公佈商店，則可能不會顯示 *FashionFlow_name* 目錄。如果是這種情況，請以滑鼠右鍵按一下 Stores 專案，然後選取**重新整理**。*FashionFlow_name* 目錄就會出現在工作區中。

   b. 以滑鼠右鍵按一下 ***FashionFlow_name*** 目錄，然後選取**匯入**。這時會開啓「匯入」精靈。

   c. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。

   d. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案是位於下列位置：
   yourDirectory\WC_SAMPLE_55.zip
   其中 *yourDirectory* 是您下載套件的目錄。

   e. 按一下**取消全選**，然後選取 MyNewJSPTemplate_All.jsp 檔。

   f. 在**資料夾**欄位中，已經指定 `Stores/Web Content/`*`FashionFlow_name`* 資料夾。請保留這個值。

   g. 按下**完成**。

5. 將指導教學的內容檔匯入到適當的目錄中，步驟如下：

   a. 在「J2EE 導覽器」檢視畫面中，展開下列目錄：
   Stores > Web Content> WEB-INF> classes> *FashionFlow_name* 目錄。

b. 以滑鼠右鍵按一下 ***FashionFlow_name*** 目錄，然後選取**匯入**。
這時會開啓「匯入」精靈。

c. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。

d. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
yourDirectory\WC_SAMPLE_55.zip
其中 *yourDirectory* 是您下載套件的目錄。

e. 按一下**取消全選**，然後選取 Tutorial_All_en_US.properties 檔。

f. 在**資料夾**欄位中，已經指定
`Stores/Web Content/WEB-INF/classes/`*`FashionFlow_name`* 資料夾。請保留這個值。

g. 按下**完成**。

6. 有四個檔案是用來載入您在指導教學中所建立的新資源的存取控制原則。這四個檔案是：

   - MyNewViewACPolicy.xml -- 這個 XML 檔案包含在建立新檢視畫面時所使用的存取控制原則。
   - MyNewControllerCmdACPolicy.xml-- 這個 XML 檔案包含在建立新的控制程式指令時所使用的存取控制原則。
   - SampleACPolicy_template.xml-- 這個 XML 檔案包含在建立新的 Enterprise Bean 時所使用的存取控制原則。
   - SampleACPolicy_template_en_US.xml-- 這個 XML 檔包含在建立新的 Enterprise Bean 時的存取控制原則說明。

   請將前面這些檔案複製到適當的目錄中，步驟如下：

   a. 在檔案系統上，導覽至下列目錄：
   yourDirectory\WC_SAMPLE_55.zip。

   b. 展開 Zip 檔並將前面這四個檔案解壓縮到下列目錄中：
   WCStudio_installdir\Commerce\xml\policies\xml

7. 執行下列步驟來測試您的環境，以確定您可以開始進行指導教學：

   a. 在 WebSphere Studio Application Developer 中，切換到「伺服器」視景。

   b. 啓動您的 Payment Server。如果您是在本端 Payment Server 中執行，以滑鼠右鍵按一下 **WebSphereCommercePaymentsServer**，然後選取**啓動**（或**重新啓動**）。

   c. 以滑鼠右鍵按一下 **WebSphereCommerceServer**，然後選取**啓動**（或**重新啓動**）。

d. 觀察主控台，看看 WebSphereCommerceServer 伺服器何時完成其啓動程序。當伺服器啓動後，您會在看到類似下面的資訊：

```
[4/2/03 12:56:06:286 EST] 66adf8d1 ApplicationMg A WSVR0221I:
   應用程式已啓動：WebSphereCommerceServer
[4/2/03 12:56:06:777 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 http 正在接聽埠 9,080。
[4/2/03 12:56:10:742 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 https 正在接聽埠 9,443。
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 http 正在接聽埠 8,080。
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 http 正在接聽埠 80。
[4/2/03 12:56:11:123 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 https 正在接聽埠 443。
[4/2/03 12:56:11:183 EST] 66adf8d1 HttpTransport A SRVE0171I:
   傳輸 https 正在接聽埠 9,043。
[4/2/03 12:56:11:653 EST] 66adf8d1 RMIConnectorC A ADMC0026I:
   RMI 連接器可於埠 2809 使用
[4/2/03 12:56:12:575 EST] 66adf8d1 WsServer
   A WSVR0001I：伺服器 server1 開始電子商業
```

e. 在「J2EE 導覽器」檢視畫面中，展開 **Stores** 專案。然後展開 **Web Content > *FashionFlow_name***。

f. 以滑鼠右鍵按一下 **index.jsp** 檔，然後選取**在伺服器上執行**。這時會開啓範例商店。

g. 選取一個產品，並確定您可以購買它。

您現在已經可以繼續進行指導教學。

建立新的檢視畫面

本指導教學的第一步就是建立新的檢視畫面。這個新的檢視畫面稱爲 MyNewView，並且有一個對應的 JSP 範本，叫做 MyNewJSPTemplate.jsp。

在本節指導教學中，您將學習下列內容：

- 用來放置要套用到特定商店的 JSP 範本及圖形檔的地方。
- 如何使用 WebSphere Studio Application Developer 來建立 JSP 範本。
- 如何建立包含 JSP 範本的文字的內容檔。
- 如何使用新的 "MyNewView" 來更新檢視畫面登錄（VIEWREG 表格）。
- 如何設定新的檢視畫面的存取控制。
- 如何使用 WebSphere 測試環境 (WTE) 來測試新的檢視畫面。

通常，建立新的檢視畫面必須執行下面的步驟：

1. 將檢視畫面命名並登錄到檢視畫面登錄中。

2. 建立新的內容檔案，以儲存 JSP 範本中可翻譯的文字。
3. 爲新的檢視畫面建立新的 JSP 範本。
4. 爲檢視畫面建立及載入存取控制原則。

登錄 MyNewView

在這個實務內容中，您建立的檢視畫面稱爲 MyNewView。這個檢視畫面必須登錄到 VIEWREG 表格中，而該表格爲指令登錄的一部分。如果要登錄新的檢視畫面，您只需要使用一個簡單的 SQL 陳述式在 VIEWREG 表格中建立一個新的項目。

在繼續執行這個步驟之前，您必須知道商店的唯一識別碼（ID）。您可以對開發資料庫執行下面的 SQL 查詢，來判斷這個 ID：

```
select STOREENT_ID from STOREENT where IDENTIFIER = 'FashionFlow_name'
```

其中 *FashionFlow_name* 是您的商店名稱。請在此處記下這個值，因爲下一節中將會需要使用它：

_____

DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyNewView：

1. 開啓 DB2 指令中心（**開始 > 程式集 >IBM DB2 > 指令行工具 > 指令中心**）。
2. 從**工具**功能表中選取**工具設定**。
3. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
4. 關閉工具設定。
5. 選取「Script」標籤後，在 Script 視窗中輸入下列資訊，以便在 VIEWREG 表格中建立必要的項目：

   ```
   connect to developmentDB user dbuser using dbpassword;
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
   values ('MyNewView', -1, FF_storeent_ID,
      'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp', 'This is my new view for tutorial one',
   0, null)
   ```

   其中

   - *developmentDB* 是您的開發資料庫名稱
   - *dbuser* 是資料庫使用者
   - *dbpassword* 是您的資料庫使用者的密碼
   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

按一下**執行**圖示。

您應該會看到一則訊息，指出 SQL 陳述式已經順利完成。

Oracle 如果您所用的是 Oracle 資料庫，請執行下列步驟，以便將檢視畫面登錄在資料庫中：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   其中

   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

   按 Enter 鍵以執行 SQL 陳述式。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

這時 MyNewView 已經登錄完成。

建立指導教學的內容檔

這個步驟中，您必須建立新的內容檔，以保留任何在 JSP 範本中使用的可翻譯的文字。將可翻譯的文字與 JSP 範本本身隔開時，可以使翻譯的作業更簡單，這也是採用全球化網站的重要關鍵。

如果要建立內容檔，請在 WebSphere Studio Application Developer 中執行下列步驟：

1. 開啓「Web」視景（**視窗 > 開啓視景 > Web**）。
2. 在 **Stores** Web 專案內，展開 **Web Content > WEB-INF > classes > *FashionFlow_name*** 資料夾。您會找到 Tutorial_All_en_US.properties 檔。

3. 以滑鼠右鍵按一下 **Tutorial_All_en_US.properties**，然後選取**開啓方式 > 內容檔編輯程式**。
4. 以滑鼠右鍵按一下 ***FashionFlow_name*** 資料夾，然後選取**新建 > 其他 > 簡易 > 檔案 > 下一步**來建立新的內容檔。
   這時會開啓「新建檔案」視窗。
5. 在**檔名**欄位中，輸入 TutorialNLS_en_US.properties，然後按一下**完成**。
   這時會開啓新的空白檔案。
6. 將區段 1 從 Tutorial_All_en_US.properties 檔複製到新的 TutorialNLS_en_US.properties 檔。這樣會將下列名稱-值配對引入到 TutorialNLS_en_US.properties 檔案中：

```
# -- 區段 1 -- #

ProgrammerGuide=程式設計手冊
Tutorial=指導教學：建立新的商業邏輯
ParametersFromCmd= 從控制程式指令傳送來的參數-值配對的清單
CalledByControllerCmd=MyNewView 是由控制程式指令所呼叫
CalledByWhichControllerCmd=MyNewView 是由控制程式指令所呼叫
    ，也就是 -
ControllerParm1=ControllerParm1=
ControllerParm2=ControllerParm2=
Example=這是使用 JSP 標準標籤程式庫 (JSTL) 中的 <if> 標籤的範例

UserName=UserName=
Points=Points=
Greeting=Greeting=
UserId=UserId=
FirstInput=您的第一個輸入參數
RegisteredUser=是已登錄的使用者
ReferenceNumber=這個使用者的成員參考號碼是
NotRegisteredUser=不是已登錄的使用者
BonusAdmin=紅利管理
PointBeforeUpdate=更新前的紅利點數是
PointAfterUpdate=更新後的紅利點數是
EnterPoint=請輸入點數，然後將它提交給控制程式
指令

# -- 區段 1 結束 -- #
```

   請注意，值中的換行僅是爲了方便顯示。
7. 儲存 TutorialNLS_en_US.properties 檔 (Ctrl+S)。

新的 TutorialNLS_en_US.properties 檔會儲存在 Stores\Web Content\WEB-INF\classes*FashionFlow_name* 目錄下，而您的新 JSP 範本會將它當作資源連結來使用。

註: 變更內容檔的內容時，必須先重新啓動伺服器，才能在測試中看到變更。

建立 MyNewJSPTemplate

在這個步驟中，您會使用 WebSphere Studio 中的 Page Designer 工具來建立新的 JSP 範本。尤其是，您會建立與 MyNewVeiw 一起使用的 MyNewJSPTemplate.jsp。在建立這個範本時，您會建立一個新的空白 JSP 範本，然後從另一個 JSP 範本 (MyNewJSPTemplate_All.jsp) 來加入適當的區段。

如果要建立您的新 JSP 範本，請執行以下步驟：

1. 在 Web 視景中，切換到「J2EE 導覽器」檢視畫面。
2. 以滑鼠右鍵按一下 **Stores** Web 專案，然後選取**內容**。
3. 在左窗格中選取 **Web**，然後從「可用的 Web 專案特性」清單中，選取**包含 JSP 標準標籤程式庫**。按一下**套用**。當更新完成時，按一下**確定**來關閉內容編輯程式。
4. 展開 **Web Content*FashionFlow_name*** 目錄。
5. 以滑鼠右鍵按一下 **MyNewJSPTemplate_All.jsp** 檔，然後選取**開啓方式 > Page Designer**。
6. 以滑鼠右鍵按一下 ***FashionFlow_name*** 資料夾，然後選取**新建 > JSP 檔**，以便在這個資料夾中建立新的 JSP 範本。
   這時會開啓「新建 JSP 檔」視窗。
7. 依照下列方式來指定新檔案的值：
   a. 在**檔名**欄位中，輸入 `MyNewJSPTemplate.jsp`。
   b. 從**標記語言**下拉清單中，選取 **XHTML**，然後按一下**下一步**。
   c. 按一下**新增標籤程式庫**。
      這時會開啓「選取標籤程式庫」視窗。
   d. 選取下列標籤程式庫：
      - http://java.sun.com/jstl/core
      - http://java.sun.com/jstl/fmt

      按一下**確定**，然後按一下**下一步**。
   e. 按一下**下一步**。
   f. 清除（**使用工作台預設值**）**工作台編碼**勾選框。
   g. 從**編碼**下拉清單中，選取 **ISO Latin -1**。
   h. 從**文件類型**下拉清單中，選取 **XHTML 1.0 移轉**。
   i. 按一下**完成**。
      這時會開啓 MyNewJSPTemplate.jsp 檔。按一下檔案不同檢視畫面的「設計」、「來源」和「預覽」標籤。

8. 在選取「設計」標籤時，請按一下**將 MyNewJSPTemplate.jsp 的內容置於此處**文字。將這段文字換成 Hello world!。
9. 切換至「來源」，然後切換至「預覽」標籤。請注意，文字已經改變。
10. 現在您必須將準備區段從 MyNewJSPTemplate_All.jsp 檔複製到新的 MyNewJSPTemplate.jsp 檔。這個區段會設定您將來在對這個檔案進行更新時所需要的預留位置。請將 <%--準備區段和準備區段結束 --%> 標記之間的文字複製到新的 JSP 範本中。在將這些文字複製到您的 JSP 範本時，請改寫下列文字：

```
<title> MyNewJSPTemplate.jsp </title>
</head>
      <body>
<p> Hello World! </p>
      </body>
</html>
```

註: 請勿將 <%--準備區段和準備區段結束 --%> 標記複製到新的 JSP 範本中。您只需要複製這些標記之間的文字。

11. 將 MyNewJSPTemplate_All.jsp 檔的區段 1A 和 2 複製到新的 MyNewJSPTemplate.jsp 檔中。將新文字放置在 <!-- 區段 1A -->、<!-- 區段 1A 結束 -->、<!-- 區段 2 --> 以及 <!-- 區段 2 結束 --> 標記之間。這樣會將下列文字引入到 MyNewJSPTemplate.jsp 中：

```
<!-- 區段 1A -->

    <%@ include file="include/EnvironmentSetup.jsp"%>

<!-- 區段 1A 結束 -->

<!-- 區段 2 -->

<fmt:setLocale value="${CommandContext.locale}" />
<fmt:setBundle basename="${sdb.directory}/TutorialNLS" var="tutorial" />

<!-- 區段 2 結束 -->
```

第一個區段包括用來設定環境變數的 EnvironmentSetup.jsp 檔。第二個區段是用來建立資源連結物件，這個物件是用來從內容檔擷取資訊，而且它會設定語言環境。

12. 現在您要新增圖形和文字到 JSP 範本中。同樣的，您必須將文字從 MyNewJSPTemplate_All.jsp 檔複製到 MyNewJSPTemplate.jsp 檔中，來完成這個步驟。這一次，請將 MyNewJSPTemplate_All.jsp 檔的區段 3 複製到 MyNewJSPTemplate.jsp 檔中。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 3 -->

<table cellpadding="0" cellspacing="0" border="0">
 <tr>
```

```
 <td bgcolor="#ff2d2d" >
  <img src="<c:out value="${sdb.jspStoreDirFilePath}
                          images/male_blueshirt.gif" />" border="0"/>
 </td>
</tr>
    </table>

<h1><fmt:message key="ProgrammerGuide" bundle="${tutorial}" /> </h1>

<h2><fmt:message key="Tutorial" bundle="${tutorial}" /> </h2>

<!-- 區段 3 結束 -->
```

區段 3 會採用在商店特定的影像檔子資料夾 (Stores\Web Content*FashionFlow_name*\images) 中的一個影像檔。這個影像檔也會從內容檔擷取文字。

13. 儲存對 MyNewJSPTemplate.jsp 檔所做的變更 (Ctrl+S)。

為 MyNewView 建立及載入存取控制原則

您必須爲新的檢視畫面指定指令層次的存取控制。在此情況下，指令層次的存取控制原則會指出容許所有的使用者執行檢視畫面。請注意，開發環境可以接受這類存取控制原則，但是其他環境可能不適用。有關進階存取控制的基本需求，請參閱 *WebSphere Commerce Security Guide*。

存取控制原則是由 MyNewViewACPolicy.xml 檔所定義，而您之前在準備步驟時已經將這個檔案放置到下列目錄中：
WCStudio_installdir\Commerce\xml\policies\xml

如果要載入新的原則，請執行下列步驟：

1. 在指令提示下，導覽至下列目錄：
   WCStudio_installdir\Commerce\bin
2. 您必須發出 acpload 指令，其格式如下：

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   其中

   - *db_name* 是您的開發資料庫的名稱。
   - *db_user* 是資料庫使用者的名稱。
   - *db_password* 是資料庫使用者的密碼。
   - *inputXMLFile* 是包含存取控制原則規格的 XML 檔。就本例而言，請指定 `MyNewViewACPolicy.xml`。

   以下是指令的範例（已指定變數）：

```
acpload Demo_Dev db2user db2user MyNewViewACPolicy.xml
```

測試 MyNewView

建立新的檢視畫面時，最後的步驟是在 WebSphere 測試環境中測試新的檢視畫面。請注意，在測試新的檢視畫面時（以及稍後在測試新的指令時），您必須先啓動商店的首頁。您需要啓動商店，這是因爲新的檢視畫面是特別針對您的商店來登錄的。因此，在嘗試存取新的檢視畫面之前，必須先啓動商店首頁，來設定指令環境定義中的商店 ID 值。同樣的，您稍後建立的控制程式指令也是特別針對您的商店來登錄的，因此需要從指令環境定義取得商店 ID。

如果要測試新的檢視畫面，請執行下列步驟：

1. 在 WebSphere Studio Application Developer 中，開啓「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 在 Web 瀏覽器中，輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

   幾秒後，就會顯示新的 JSP 範本（如同下面的螢幕擷取圖所示）：

圖 31.

建立新的控制程式指令

在這個步驟中，您將建立一個叫做 MyNewControllerCmd 的新控制程式指令。一開始，這個指令只會傳回 MyNewView 檢視畫面。

在本節指導教學中，您將學習下列內容：

- 控制程式指令中所含程式碼的最小基本需求
- 如何建立新的控制程式指令介面以及實作類別
- 如何設定一個控制程式指令以傳回檢視畫面
- 如何在指令登錄中登錄控制程式指令
- 如何設定控制程式指令的存取控制

通常，建立新的控制程式指令必須執行下列步驟：

1. 在指令登錄中登錄新的指令。
2. 建立指令的介面。
3. 建立指令的實作類別。
4. 爲指令建立及載入存取控制原則。
5. 測試指令。

登錄 MyNewControllerCmd

在指導教學的這個部分中，您會建立一個叫做 MyNewControllerCmd 的新控制程式指令。這個指令必須登錄到指令登錄中。尤其是，介面必須登錄到 URLREG 表格中，而介面及其實作類別之間的關聯會登錄到 CMDREG 表格中。

DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyNewControllerCmd：

1. 開啓 DB2 指令中心（**開始 > 程式集 >IBM DB2 > 指令行工具 > 指令中心**）。
2. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to developmentDB user dbuser using dbpassword;
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼
- *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

按一下**執行**圖示。

您應該會看到一則訊息，指出 SQL 陳述式已經順利完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyNewControllerCmd：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

   其中

   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

   按 Enter 鍵以執行 SQL 陳述式。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

請注意，CMDREG 表格的第二個 insert 陳述式並不是絕對必要的。在這個實務內容中，介面會使用預設的實作，因此，介面與其實作類別之間的關聯並不需要在指令登錄中指定。它們在這裡只是爲了完整呈現。

建立 MyNewControllerCmd 介面

根據 WebSphere Commerce 程式設計模型，所有新的控制程式指令都必須有一個介面，以及一個實作類別。在這個指導教學中，介面的基礎是在範例程式碼中提供。它可以分成許多不同的區段，而它們目前在程式碼中是當作註解。當您執行指導教學時，必須取消註解許多不同的程式碼區段。

如果要建立 MyNewControllerCmd 介面，請執行下列步驟：

1. 在 WebSphere Studio Application Developer 中，開啓 Java 視景（**視窗 > 開啓視景 > Java**）。

2. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
3. 導覽至 **src** 目錄，然後展開 **com.ibm.commerce.sample.commands** 套件。
4. 按兩下 **MyNewControllerCmd.java** 介面來開啓檔案。
5. 在原始碼中，取消註解區段 1（刪除區段前面的 "/*" 以及區段後面的 "*/"）。這樣會將下列程式碼引入類別中：

```
/// 區段 1 ////////////////////////////////////////////

// 設定預設的指令實作類別

 static final String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewControllerCmdImpl";


/// 區段 1 結束/////////////////////////////////////////
```

   這個程式碼區段會指定在預設的情況下，介面應該使用 MyNewControllerCmdImpl 實作類別。
6. 儲存對介面的變更 (Ctrl+S)。

建立 MyNewControllerCmdImpl 實作類別

一旦建立介面，下一步就是建立指令的實作類別。在這個指導教學中，實作類別的基礎是在範例程式碼中提供。它可以分成許多不同的區段，而它們目前在程式碼中是當作註解。當您執行指導教學時，必須取消註解許多不同的程式碼區段。

如果要建立 MyNewControllerCmdImpl 實作類別，請執行下列步驟：

1. 按兩下 **MyNewControllerCmdImpl.java** 類別來開啓它。
2. 在「大綱」檢視畫面中，選取 **performExecute** 方法來檢視其原始碼。
3. 在 performExecute 方法的原始碼中，取消註解「區段 1」。這樣會將下列程式碼引入方法中：

```
/// 區段 1 ////////////////////////////////////////////////

    /// 為輸出建立新的 TypedProperties。

 TypedProperty rspProp = new TypedProperty();

/// 區段 1 結束 ////////////////////////////////////////
```

   這樣會建立一個新的 TypedProperty 物件，用來保留指令的回應內容。
4. 在 performExecute 方法的原始碼中，取消註解「區段 5」。這樣會將下列程式碼引入方法中：

```
/// 區段 5 ////////////////////////////////////////////////

    /// 察看控制程式指令如何呼叫 JSP
```

```
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyNewView");
setResponseProperties(rspProp);

/// 區段 5 結束 ////////////////////////////////////////
```

此程式碼區段會完成兩個主要作業。首先，WebSphere Commerce 程式設計模型要求所有的程式設計指令傳回一個檢視畫面。在這個區段中，它指定要傳回的檢視畫面是您之前建立的 MyNewView。另外，它會將指令的回應內容設定成新的 rspProp 物件。

5. 儲存您的變更 (Ctrl+S)。
6. 如果要編譯您對程式碼所做的變更，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。

建立及載入指令的存取控制原則

您必須爲新的指令指定指令層次的存取控制。在此情況下，指令層次的存取控制原則會指出容許所有的使用者執行指令。請注意，開發環境可以接受這類存取控制原則，但是其他環境可能不適用。有關進階存取控制的基本需求，請參閱 *WebSphere Commerce Security Guide*。

存取控制原則是由 MyNewControllerCmdACPolicy.xml 檔所定義，而您之前在準備步驟時已經將這個檔案放置到下列目錄中：
WCStudio_installdir\Commerce\xml\policies\xml

如果要載入新的原則，請執行下列步驟：

1. 在指令提示下，導覽至下列目錄：
   WCStudio_installdir\Commerce\bin
2. 您必須發出 acpload 指令，其格式如下：

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   其中

   - *db_name* 是您的開發資料庫的名稱。
   - *db_user* 是資料庫使用者的名稱。
   - *db_password* 是資料庫使用者的密碼。
   - *inputXMLFile* 是包含存取控制原則規格的 XML 檔。就本例而言，請指定 MyNewControllerCmdACPolicy.xml。

   以下是指令的範例（已指定變數）：

   ```
   acpload Demo_Dev db2user db2user MyNewControllerCmdACPolicy.xml
   ```

測試 MyNewControllerCmd

現在介面、實作類別、指令登錄和存取控制資訊都已經建立完成，您可以測試新的控制程式指令。

修改 Java 程式碼之後，必須先重新啓動測試伺服器，才能辨識您的變更。

如果要測試新的程式碼，請執行下列步驟：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 在 Web 瀏覽器中，輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   幾秒後，就會顯示新的 JSP 範本（如同下面的螢幕擷取圖所示）：

圖 32.

從 MyNewControllerCmd 傳送資訊到 MyNewView

在這個步驟中，您會修改 MyNewControllerCmd，使它傳送資訊到 MyNewView。這裡有兩種用來傳送資訊到檢視畫面的方法。第一種，您需要學習如何使用回應內容的 TypedProperties 物件，以及如何將這個物件的資訊擷取到 JSP 範本中。第二種，您需要學習如何建立新的資料 Bean，用來傳送資訊到 JSP 範本。

使用 TypedProperties 物件來傳送資訊

在本節中，您將修改 MyNewControllerCmdImpl，以傳送資訊到 JSP 範本。尤其是您需要修改指令，新增其他的名稱-值配對至用於指令回應內容的現有 rspProp TypedProperties 物件中。在 JSP 範本中，您會使用 JSTL 表示式語言從回應內容擷取資訊。

在本節指導教學中，您將學習下列內容：

- 如何修改控制程式指令，將額外的回應內容併入到 TypedProperty 中
- 如何修改 JSP 範本，以使用 JSTL 表示式語言來從回應內容擷取資訊

如果要讓 TypedProperties 物件的資訊顯示在 JSP 範本中，請執行下列步驟：

1. 第一步是修改 MyNewControllerCmdImpl 類別，方式如下：
   a. 切換到 Java 視景。
   b. 展開下列目錄：**WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands**。
   c. 按兩下 **MyNewControllerCmdImpl.java**，然後在「大綱」檢視畫面中選取其 **performExecute** 方法。
   d. 在 performExecute 方法的原始碼中，取消「區段 2」的註解。這樣會將下列程式碼引入方法中：

```
/// 區段 2 ////////////////////////////////////////////////

   /// 查看控制程式指令如何將變數傳送到 JSP

 /// 將控制程式指令的其他參數新增到 rspProp
  ///  以取得回應
    String message1 = "Hello from IBM!";

 rspProp.put("controllerParm1", message1);
 rspProp.put("controllerParm2", "Have a nice day!");

/// 區段 2 結束 ////////////////////////////////////////
```

   上述的程式碼片段會建立兩個新參數，並放置到回應內容物件中。這個物件最後會傳送到檢視畫面。
   e. 儲存變更。
   f. 編譯指令，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。
2. 接下來您必須執行下列步驟來更新 MyNewJSPTemplate.jsp 檔：
   a. 如果 JSP 範本檔案尚未開啓，請執行下列步驟：
      1) 在 Web 視景中，切換到「J2EE 導覽器」檢視畫面，然後展開 **Stores** Web 專案。
      2) 導覽至 **Web Content*FashionFlow_name*** 目錄。
      3) 以滑鼠右鍵按一下 **MyNewJSPTemplate_All.jsp**，然後選取**開啓方式 > Page Designer**。

4) 以滑鼠右鍵按一下 **MyNewJSPTemplate.jsp**，然後選取**開啓方式 > Page Designer**。

b. 將 MyNewJSPTemplate_All.jsp 檔的區段 4 複製到新的 MyNewJSPTemplate.jsp 檔中。將新的文字放置到 <!-- 區段 4 --> 以及 <!-- 區段 4 結束 --> 標記之間。這樣會將下列文字引入到 MyNewJSPTemplate.jsp 中：

```
<!-- SECTION 4 -->

<h3><fmt:message key="ParametersFromCmd" bundle="${tutorial}" /> </h3>

<fmt:message key="ControllerParm1" bundle="${tutorial}" />
<c:out value="${controllerParm1}"/> <br />

<fmt:message key="ControllerParm2" bundle="${tutorial}" />
<c:out value="${controllerParm2}"/> <br /> <br />

<!-- 區段 4 結束 -->
```

這個區段使用 JSTL 表示式語言來取得並顯示從控制程式指令傳入的值。

c. 儲存變更。

3. 下一步是執行下列步驟來測試對控制程式指令和 JSP 範本的修改：

a. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。

b. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。

c. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
這時會在 Web 瀏覽器中顯示商店首頁。

d. 在 Web 瀏覽器中，輸入下列 URL：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
```

幾秒後，就會顯示新的 JSP 範本（如同下面的螢幕擷取圖所示）：

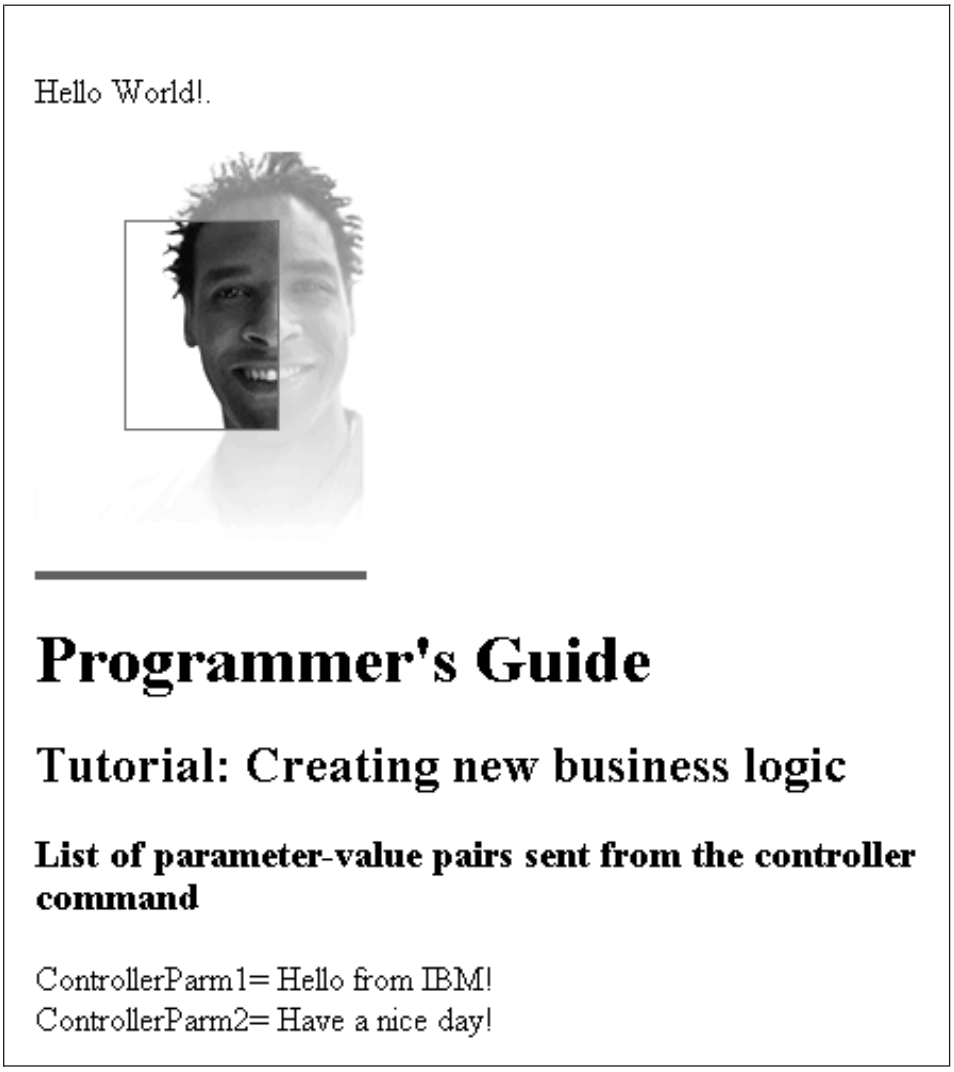

圖 33.

使用資料 Bean 來傳送資訊

在這一節中，您會新增程式碼來判斷是否要直接呼叫檢視畫面，還是要由控制程式指令來呼叫。如果是後面的情況，JSP 範本應該會顯示呼叫範本的指令名稱。

爲了將這項資訊傳送到檢視畫面，因此會建立一個叫做 MyNewDataBean 的新資料 Bean。同時也會修改 MyNewJSPTemplate，使它能顯示新的資訊。

MyNewDataBean 只能用來將控制程式指令的資訊提供給 JSP 範本使用。請將這一點與提供資料庫資訊的功能加以對照。在本指導教學後面，您將會學習如何建立新的資料 Bean，用來將資料庫的資訊提供給 JSP 範本。

您可能會懷疑爲何本指導教學使用資料 Bean 就只是爲了提供控制程式指令的資訊。建立這個 Bean 兩個原因：首先，這是一種良好的程式設計方式，可以容許用邏輯方式將屬性分組，第二，可以讓網頁程式開發人員更容易透過資料 Bean 來新增資訊到網頁中，而不需要使用 TypedProperties 回應內容物件。

在本節指導教學中，您將學習下列內容：

- 如何建立新的資料 Bean
- 如何修改控制程式指令來將資料 Bean 實例化
- 如何使用控制程式指令來設定資料 Bean 中的屬性
- 如果將已實例化的資料 Bean 傳送到 JSP 範本
- 如何修改 JSP 範本來從資料 Bean 擷取資訊
- 請參閱 JSP 範本中關於使用 <if> 標籤的範例

建立 MyNewDataBean

MyNewDataBean 是用來傳送資訊到 MyNewJSPTemplate.jsp 頁面中。與指導教學的其他章節一樣，資料 Bean 的基礎是在範例程式碼中提供。它可以分成許多不同的區段，而它們目前在程式碼中是當作註解。當您執行指導教學時，必須取消註解許多不同的程式碼區段。

如果要建立 MyNewDataBean，請執行下列步驟：

1. 開啓 Java 視景，然後使用 Package Explorer 檢視畫面。
2. 展開下列目錄：**WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.databeans**。
3. 按兩下 **MyNewDataBean.java** 來檢視其原始碼。
4. 在主要類別的原始碼中，取消註解「區段 1」。這樣會將下列程式碼引入類別中：

```
/// 區段 1 ////////////////////////////////////////////
/// 建立欄位和存取元（setter/getter 方法）

 private java.lang.String callingCommandName = null;
 private boolean calledByControllerCmd = false;


 public java.lang.String getCallingCommandName() {
  return callingCommandName;
 }
```

```
public void setCallingCommandName(java.lang.String newCallingCommandName)
 {
 callingCommandName = newCallingCommandName;
}

public boolean getCalledByControllerCmd() {
 return calledByControllerCmd;
}

public void setCalledByControllerCmd(boolean newCalledByControllerCmd)
 {
 calledByControllerCmd = newCalledByControllerCmd;
}

/// 區段 1 結束 ////////////////////////////////////
```

上面的程式碼會引入兩個變數，當控制程式指令傳回檢視畫面時，會使用它們來顯示資訊，而不是直接由檢視畫面的 URL 來呼叫。

5. 儲存變更。

使用 MyNewControllerCmd 將 MyNewDataBean 實例化以及設定其屬性

在這個步驟中，您會修改 MyNewControllerCmdImpl 來將 MyNewDataBean 實例化，以及設定這個 Bean 的屬性。

如果要修改 MyNewControllerCmdImpl，請執行下列步驟：

1. 在 Java 視景中，展開下列目錄：
   WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands。
2. 按兩下 **MyNewControllerCmdImpl.java** 來檢視其原始碼。
3. 在主要類別的原始碼中，取消註解「匯入區段 1」，以便在這個類別中使用新的資料 Bean。這樣會將下列程式碼引入類別中：

   ```
   /// 匯入區段 1 ////////////////////////////////
   import com.ibm.commerce.sample.databeans.*;
   /// 匯入區段 1 結束 ////////////////////////
   ```

4. 在「大綱」檢視畫面中，選取其 **performExecute** 方法。
5. 在 performExecute 方法的原始碼中，取消註解「區段 3A」和「區段 3B」。這樣會將下列程式碼引入方法中：

   ```
   /// 區段 3A ////////////////////////////////////////

    /// 將 MyNewDataBean 資料 Bean 實例化，並設定內容，
     ///  然後將實例新增至回應的 resProp

    MyNewDataBean mndb = new MyNewDataBean();
    mndb.setCallingCommandName(this.getClass().getName());
    mndb.setCalledByControllerCmd(true);
   ```

```
/// 區段 3A 結束 ////////////////////////////////////////

/// 區段 3B ////////////////////////////////////////
 rspProp.put("mndbInstance", mndb);

/// 區段 3B 結束 ////////////////////////////////////////
```

上面的程式碼片段會將 MyNewDataBean 物件實例化，在物件中設定兩個參數（指出是由控制程式指令呼叫，以及用來呼叫的指令），然後將資料 Bean 物件放置到回應內容中，以供 JSP 範本使用。

6. 儲存變更。
7. 編譯程式碼變更，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。

使用 MyNewJSPTemplate 中的 MyNewDataBean

在這一節中，您會修改 MyNewJSPTemplate.jsp，以指出檢視畫面是否由控制程式指令傳回。如果是由控制程式指令傳回，就應該顯示該控制程式指令的名稱。JSP 範本會使用條件邏輯的 JSTL 標籤，並根據 MyNewDataBean 的值來判斷這一點。

如果要修改 JSP 範本，請執行下列步驟：

1. 如果 JSP 範本檔案尚未開啓，請執行下列步驟：
   a. 在 Web 視景中，切換到「J2EE 導覽器」檢視畫面，然後展開 **Stores** Web 專案。
   b. 導覽至 **Web Content*FashionFlow_name*** 目錄。
   c. 同時標示 **MyNewJSPTemplate_All.jsp** 和 **MyNewJSPTemplate.jsp** 檔案，以滑鼠右鍵按一下，然後選取**開啓方式 > Page Designer**。
2. 將 MyNewJSPTemplate_All.jsp 檔的「區段 5」複製到 MyNewJSPTemplate.jsp 檔。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 5 -->

<c:if test="${mndbInstance.calledByControllerCmd}">
   <fmt:message key="Example" bundle="${tutorial}" /> <br />
   <fmt:message key="CalledByControllerCmd" bundle="${tutorial}" />
      <br />
   <fmt:message key="CalledByWhichControllerCmd" bundle="${tutorial}" />
   <b><c:out value="${mndbInstance.callingCommandName}" /></b> <br />
      <br />
</c:if>

<!-- 區段 5 結束 -->
```

這個區段的程式碼使用 JSTL <if> 標籤來判斷是否要顯示關於呼叫控制程式指令的資訊。它也會從指導教學的資源連結擷取可翻譯的文字。

3. 儲存變更。

測試已修改的 JSP 範本

如果要測試已修改的 JSP 範本，請執行下列步驟：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 在 Web 瀏覽器中，輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   幾秒後，就會顯示 JSP 範本（如同下面的螢幕擷取圖所示）：

Hello World!.

Programmer's Guide

Tutorial: Creating new business logic

List of parameter-value pairs sent from the controller command

ControllerParm1= Hello from IBM!
ControllerParm2= Have a nice day!

This is an example of using the tag from JSP Standard Tag Library (JSTL)
MyNewView was called by a controller command
MyNewView was called by the controller command which is -
com.ibm.commerce.sample.commands.MyNewControllerCmdImpl

圖 34.

5. 接下來，輸入 URL 來直接呼叫檢視畫面，並注意顯示的資訊中的差異：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

剖析及驗證 MyNewControllerCmd 中的 URL 參數

在這個步驟中，您會修改控制程式指令，以使用透過 URL（呼叫控制程式指令）來傳入的參數。驗證邏輯也會包含在指令中，以確定其中包含必要的參數，同時確定這些參數使用的是適當的值。

目前您新指令中的 validateParameters 方法，實際上只是一個指令片段（Stub）。它是由下列程式碼組成：

```
public void validateParameters() throws ECApplicationException {
}
```

您現在必須新增自訂參數檢查到指令中，並將 URL 參數傳送到 JSP 範本。在修改 validateParameters 方法時，您可以加入與 URL 參數對應的新欄位。

在本節指導教學中，您將學習下列內容：

- 如何將新欄位加到指令中，以對應至 URL 參數
- 如何使用 getRequestProperties 方法來將 URL 輸入參數移入這些欄位中
- 如何攫取遺漏的參數異常狀況
- 將 URL 參數傳送到檢視畫面的適當方法
- 請參閱遺漏參數值或參數值不正確的各種測試實例的範例

將新的欄位加到 MyNewControllerCmd

在這個步驟中，您會爲 URL 參數建立兩個新欄位。它們會同時加入到指令介面和實作類別中。其中一個 URL 參數是用來保留使用者名稱的字串值，而第二個 URL 參數則是用來接受紅利積點輸入值的整數。

如果要加入這兩個新欄位，請執行下列步驟：

1. 切換到 Java 視景。
2. 展開下列目錄：**WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands**。
3. 按兩下 **MyNewControllerCmd.java** 介面來檢視其原始碼。
4. 取消註解「區段 2」，將新的欄位和對應的 getter 方法加到介面中。這樣會將下列程式碼引入類別中：

```
/// 區段 2 ////////////////////////////////////////////////

// 設定介面方法


 public java.lang.Integer getPoints() ;
```

```
public java.lang.String getUserName() ;

public void setPoints(java.lang.Integer newPoints) ;

public void setUserName(java.lang.String newUserName)  ;


/// 區段 2 結束 ////////////////////////////////////////
```

5. 儲存變更。
6. 按兩下 **MyNewControllerCmdImpl.java** 類別來檢視其原始碼。
7. 取消註解主要類別中的「區段 1」，將新的欄位以及對應的 getter 和 setter 方法加到類別中。這樣會將下列程式碼引入類別中：

```
/// 區段 1 ////////////////////////////////////////////

/// 建立及實作控制程式指令的欄位和存取元
/// （setter/getter 方法）

private java.lang.String userName = null;
private java.lang.Integer points;


public java.lang.Integer getPoints() {
 return points;
}

public java.lang.String getUserName() {
 return userName;
}

public void setPoints(java.lang.Integer newPoints) {
 points = newPoints;
}

public void setUserName(java.lang.String newUserName) {
 userName = newUserName;
}

/// 區段 1 結束 ////////////////////////////////////////
```

8. 儲存變更。

傳送 URL 參數到檢視畫面中

在這個步驟中，您可以併入程式碼，以傳送輸入參數到 JSP 範本中。方法是使用輸入參數的值來設定資料 Bean 中的欄位。

如果要傳送 URL 參數，請執行下列步驟：

1. 按兩下 **MyNewControllerCmdImpl.java**。
2. 在「大綱」檢視畫面中，選取 **performExecute** 方法。

3. 在 performExecute 方法的原始碼中，取消註解「區段 3C」。這樣會將下列程式碼引入方法中：

```
/// 區段 3C ////////////////////////////////////////

// 傳送輸入資訊到資料 Bean
mndb.setUserName(this.getUserName());
mndb.setPoints(this.getPoints());

/// 區段 3C 結束 ////////////////////////////////////
```

這個程式碼會設定資料 Bean 物件中的值，使它們可供 JSP 範本使用。

4. 儲存變更。

攫取遺漏的參數以及驗證該值

在這個步驟中，您會修改 validateParameters 方法來引入錯誤檢查以及參數驗證邏輯。一旦修改之後，程式碼會檢查下列項目：

- 如果沒有提供第一個輸入參數，就會擲出"找不到參數"的異常狀況。這是因爲第一個輸入參數是必要的參數。在此情況下，就會顯示同屬錯誤頁面給客戶。
- 第二個參數是選用的。因此，如果沒有提供第二個輸入參數，就不會擲出"找不到參數異常狀況"。相反的，第二個輸入參數會預設成零，而客戶並不會受到錯誤所影響。程序會繼續進行。

如果要新增這項錯誤檢查，請執行下列步驟：

1. 按兩下 **MyNewControllerCmdImpl.java**。
2. 在「大綱」檢視畫面中，選取其 **validateParameters** 方法。
3. 在 validateParamaters 方法的原始碼中，取消註解「區段 1」。這樣會將下列程式碼引入方法中：

```
/// 區段 1 ////////////////////////////////////////////////

/// 取消註解以檢查參數

    final String strMethodName = "validateParameters";

TypedProperty prop = getRequestProperties();

/// 擷取必要的參數
try {
 setUserName(prop.getString("input1"));

} catch (ParameterNotFoundException e) {
 /// 下一個異常狀況使用 _ERR_CMD_MISSING_PARAM ECMessage 物件
    /// （在 ECMessage 類別中定義）
 throw new ECApplicationException(ECMessage._ERR_CMD_MISSING_PARAM,
       this.getClass().getName(), strMethodName,
  ECMessageHelper.generateMsgParms(e.getParamName()));
```

```
}

/// 擷取選用的 Interger
// 若無輸入值，請將 input2 設為 0
setPoints(prop.getInteger("input2",0));

/// 區段 1 結束 ////////////////////////////////////////////
```

上面的程式碼片段會檢查兩個輸入參數。try 區塊會判斷第一個參數是否存在，若沒有，則會擲出異常狀況。由於第二個參數爲選用的，若該參數遺漏或參數值不正確，此程式碼會將參數值設爲零。

4. 儲存變更。

將新的欄位加到 MyNewDataBean

在這個步驟中，您會將兩個新的欄位及其相關的 getter 方法加到 MyNewDataBean 資料 Bean，使 URL 參數可以提供給 JSP 範本使用。

如果要修改 MyNewDataBean，請執行下列步驟：

1. 按兩下 **MyNewDataBean.java** 來檢視其原始碼。
2. 取消註解「區段 2」來將下列程式碼引入到類別中：

```
/// 區段 2 ////////////////////////////////////////////

private java.lang.String userName = null;
private java.lang.Integer points;


public String getUserName() {
 return userName;
}

public void setUserName(java.lang.String newUserName) {
 userName = newUserName;
}


public Integer getPoints() {
 return points;
}

public void setPoints(java.lang.Integer newPoints) {
 points = newPoints;
}


/// 區段 2 結束 ////////////////////////////////////
```

3. 儲存變更。

4. 編譯程式碼變更，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic**，然後選取**建置專案**。

修改 MyNewJSPTemplate 以顯示 URL 參數

在這個步驟中，您會修改 MyNewJSPTemplate.jsp 檔，加入一個新的區段來顯示 URL 輸入參數，步驟如下：

1. 如果 JSP 範本檔案尚未開啓，請執行下列步驟：
   a. 在 Web 視景中，切換到「J2EE 導覽器」檢視畫面，然後展開 **Stores** Web 專案。
   b. 導覽至 Web Content*FashionFlow_name* 子資料夾。
   c. 同時標示 **MyNewJSPTemplate_All.jsp** 和 **MyNewJSPTemplate.jsp** 檔案，以滑鼠右鍵按一下，然後選取**開啓方式 > Page Designer**。
2. 將 MyNewJSPTemplate_All.jsp 檔的「區段 6」複製到 MyNewJSPTemplate.jsp 檔。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 6 -->

<fmt:message key="UserName" bundle="${tutorial}" />
<c:out value="${mndbInstance.userName}"/> <br>

<fmt:message key="Points" bundle="${tutorial}" />
<c:out value="${mndbInstance.points}"/> <br>

<!-- 區段 6 結束 -->
```

3. 儲存變更。

測試 URL 參數值

下一步是執行下列步驟來測試新的錯誤檢查是否正常運作：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 實例 1：第一個測試實例是將兩個參數排除在 URL 之外。在顯示商店首頁後，輸入下列 URL：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
```

   沒有參數可傳遞給指令，因此會顯示一個通用應用程式錯誤。

圖 35.

WebSphere Studio Application Developer 中的主控台會顯示與下面類似的資訊：
timeStamp 6730e546 CommerceSrvr E com.ibm.commerce.sample.commands.MyNewControllerCmdImpl validateParameters CMN0206E 請檢查所有的欄位。"input1" 是必要的欄位。

5. 實例 2：下一個測試實例是使用有效的第一個參數，但省略第二個參數。在此情況下，應該不會偵測到任何錯誤，因爲在預設的情況下，遺漏的第二個參數會使用零值。輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?input1=abc
   ```

   這個指令的結果是顯示 MyNewJSPTemplate 頁面，並同時在 input2 中顯示零值。

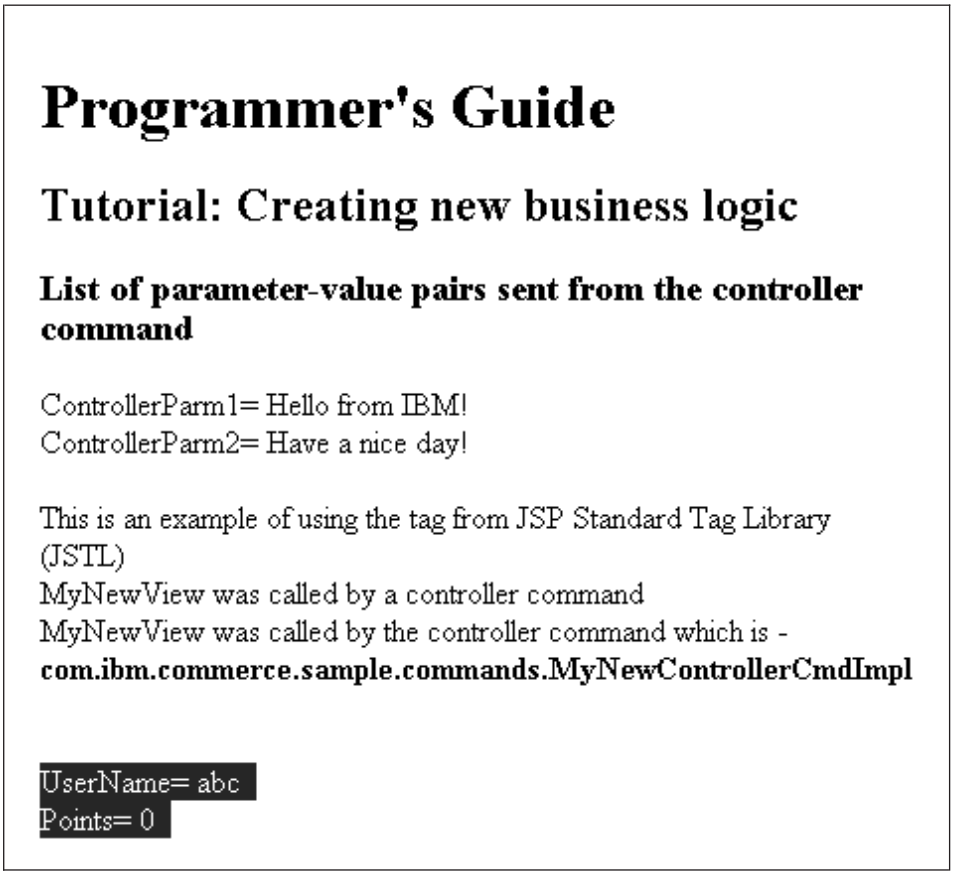

圖 36.

6. 實例 3：在這個測試實例中，爲第一個輸入參數提供一個有效的參數，並且爲第二個參數提供一個無效的參數（使用非整數的字串）。與前一個實例類似，您應該不會看到任何錯誤，因爲錯誤處理會將第二個輸入參數變更爲零。輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc&input2=abc
   ```

   這個指令的結果是顯示 MyNewJSPTemplate 頁面，並同時在 input2 中顯示零值。

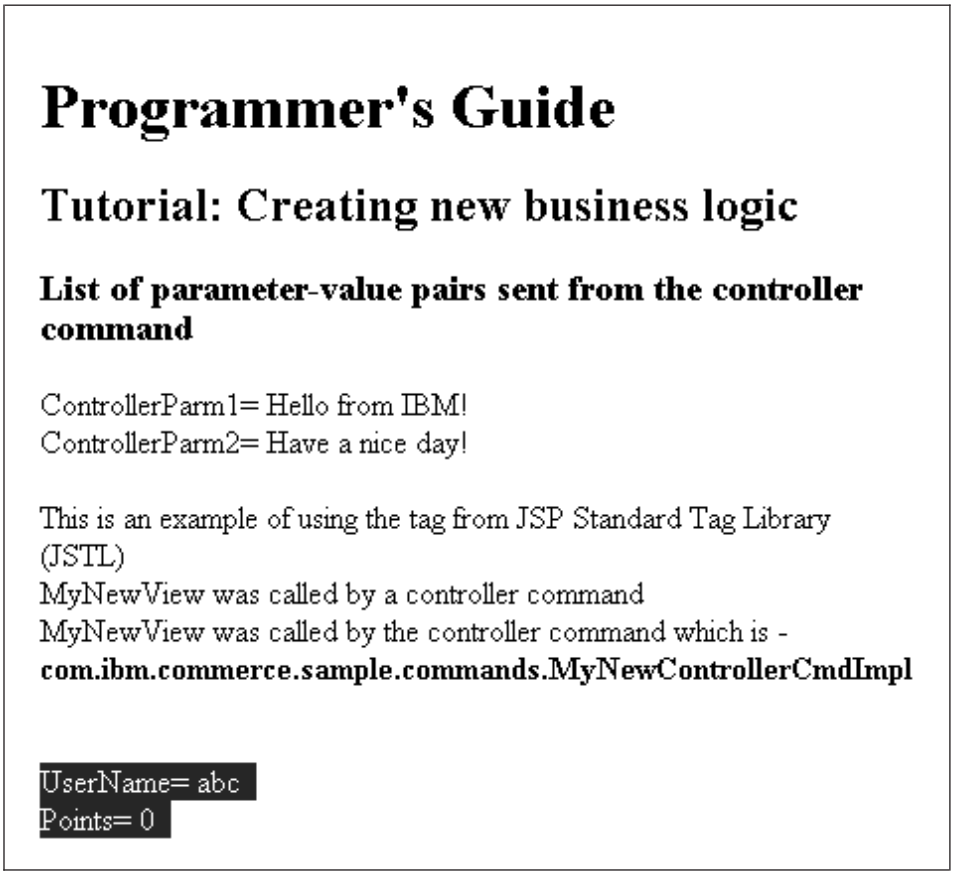

圖 37.

7. 實例 4：在這個實例中，兩個 URL 輸入參數都使用有效的值。輸入下列 URL：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

這個指令的結果是顯示 MyNewJSPTemplate 頁面，並同時爲使用者顯示 1000 點。

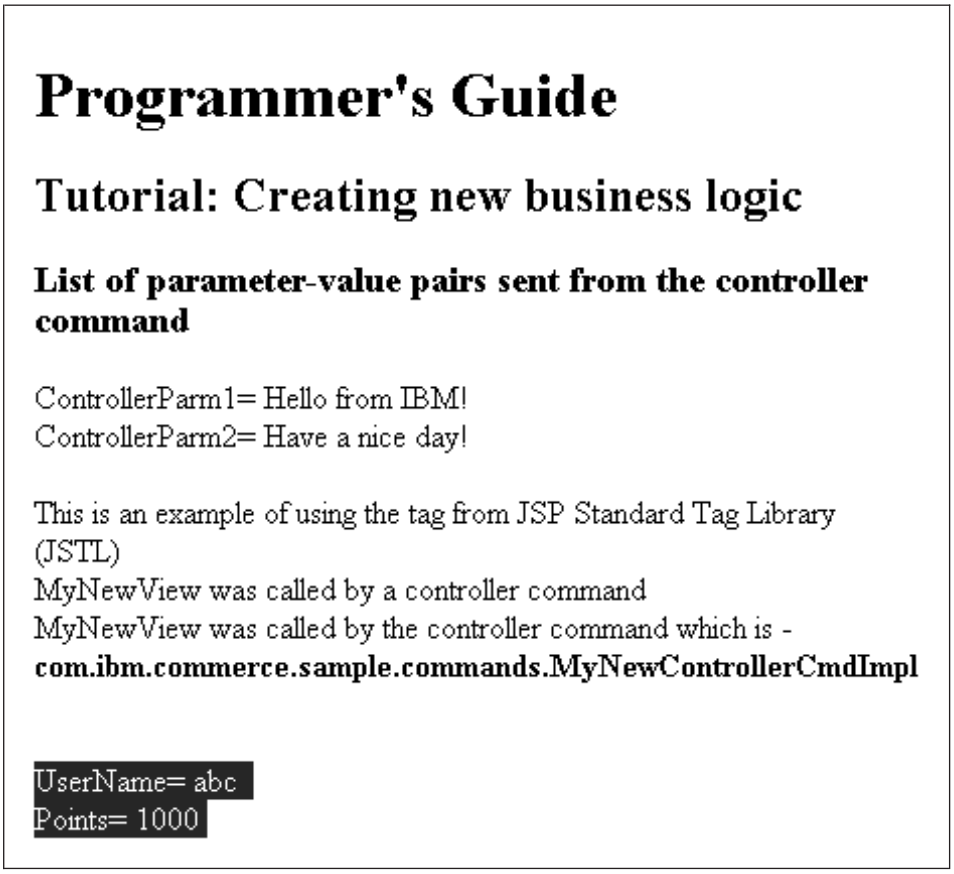

圖 38.

建立新的作業指令

一般而言，控制程式指令代表一個商業程序或複合功能。舉例來說，處理訂單的所有相關商業邏輯全概括於 OrderProcessCmd 控制程式指令中。商業程序通常可分成幾個較小的特定作業。舉例來說，在 OrderProcessCmd 控制程式指令中，會有一些要呼叫的作業指令以執行個別的工作單元。

MyNewControllerCmdImpl 目前尚不會呼叫任何作業指令。這一節可分成兩個步驟。在第一個步驟中，您將建立新的作業指令。在第二個步驟中，您將修改控制程式指令的 performExecute 方法，以呼叫新的作業指令。

在這個步驟中，您將建立一個新的作業指令介面以及它的相關實作類別。剛開始時，新的作業指令只會處理檢視畫面參數。指令中只包含

defaultCommandClassName、URL 參數和現行紅利積點值等欄位。有一個方法可以取得紅利積點的現行值。

在指導教學的這個步驟中，您將學習下列內容：

- 如何建立一個新的作業指令介面以及它的相關實作類別
- 作業指令中所需要的最小程式碼數量
- 如何新增欄位和方法到作業指令中

建立 MyNewTaskCmd

這個步驟教您如何撰寫新的作業指令。建立完整的新作業指令將牽涉到介面與實作類別的建立。在您建立作業指令時，介面應爲 `com.ibm.commerce.commands.TaskCommand` 的延伸。而實作類別應爲 `com.ibm.commerce.command.TaskCommandImpl` 的延伸。

在完成這個練習之後，您會有一個新的作業指令，叫做 MyNewTaskCmd。這個指令是供您根據「流行館」範例所建立的商店使用。

如果要建立 MyNewTaskCmd，請執行下列步驟：

1. 切換到 Java 視景，然後選取 **WebSphereCommerceServerExtensionsLogic** 專案。
2. 展開 **src** 目錄，然後展開 **com.ibm.commerce.sample.commands** 目錄。
3. 按兩下 **MyNewTaskCmd.java** 介面來檢視其原始碼。
4. 在這個介面的原始碼中，取消註解「區段 1」來建立一個欄位，以指定介面所用的預設實作類別。這樣會將下列程式碼引入類別中：

   ```
   /// 區段 1 ////////////////////////////////////////////////

   // 設定預設的指令實作類別

    static final String defaultCommandClassName =
            "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";


   /// 區段 1 結束 ///////////////////////////////////////////
   ```

   由於整個網站採用同一實作類別，且沒有預設內容可傳遞給指令，因此您可以在程式碼中指定此種的預設實作方式。如果您有指令具有多種實作方式或者具有預設內容（儲存在 CMDREG 表格中），您必須將指令登錄在 CMDREG 表格中，以建立介面與實作類別間的對映。
5. 接下來，取消註解「區段 2」來建立將在 MyNewTaskCmdImpl 實作類別中使用的 getter 和 setter 方法。這些方法適用於與下列資訊類型對應的欄位：

- 客戶的使用者 ID
- 紅利積點值
- 問候訊息

取消註解「區段 2」時，會將下列程式碼引入到介面中：

```
/// 區段 2 ////////////////////////////////////////////////
// 設定介面方法

 public void setInputUserName(java.lang.String inputUserName);
 public void setInputPoints(Integer inputPoints);
 public void setGreetings(java.lang.String greeting);

 public java.lang.String getInputUserName();
 public java.lang.Integer getInputPoints();
 public java.lang.String getGreetings();


/// 區段 2 結束 ///////////////////////////////////////////
```

6. 儲存變更。
7. 按兩下 **MyNewTaskCmdImpl.java** 實作類別來檢視其原始碼。
8. 取消註解「區段 1A」和「區段 1B」來建立欄位以及它們在實作類別中的對應 getter 和 setter 方法。 這樣會將下列程式碼引入類別中：

```
//// 區段 1A ////////////////////////////////////////////

 private java.lang.String inputUserName;
 private java.lang.String greetings;
 private java.lang.Integer inputPoints;


//// 區段 1A 結束 ////////////////////////////////////////


//// 區段 1B ////////////////////////////////////////////

 public void setInputUserName(java.lang.String newInputUserName) {
  inputUserName = newInputUserName;
 }

 public void setInputPoints(Integer newInputPoints) {
  inputPoints = newInputPoints;
 }

 public void setGreetings(java.lang.String newGreetings) {
  greetings = newGreetings;
 }

 public java.lang.String getInputUserName() {
  return inputUserName;
 }
```

```
public Integer getInputPoints() {
 return inputPoints;
}

public java.lang.String getGreetings() {
 return greetings;
}


//// 區段 1B 結束 ////////////////////////////////////////
```

儲存您的工作。

9. 在「大綱」檢視畫面中，選取 MyNewTaskCmdImpl 類別的 **performExecute** 方法。
10. 在這個 performExecute 方法的原始碼中，取消註解「區段 1」，以便將下列程式碼引入方法中：

```
/// 區段 1 ////////////////////////////////////////////////

// 修改問候語並在 NVP 清單中查看它

 setGreetings( "Hello ! " + getInputUserName() );

/// 區段 1 結束 ////////////////////////////////////////
```

這樣會更新問候值。之後問候值就可以透過 getGreetings() 方法來提供給其他物件使用。它會新增至 名稱/值配對 (NVP) 清單中。

11. 儲存您的工作。

呼叫作業指令

一旦您建立作業指令後，即得從控制程式指令中呼叫此指令。下列步驟說明如何以這種方式修改您的控制程式指令：

1. 在 Java 視景中，按兩下 **MyNewControllerCmdImpl.java** 類別
2. 在「大綱」檢視畫面中，選取其 **performExecute** 方法。
3. 在 performExecute 方法的原始碼中，導覽至「區域 4」。
4. 取消註解「區段 4A」、「區段 4B」和「區段 4C」，來將下列程式碼引入到方法中：

```
/// 區段 4A
/// 查看控制程式指令如何呼叫作業指令

    MyNewTaskCmd cmd = null;

try {

        cmd = (MyNewTaskCmd) CommandFactory.createCommand(
       "com.ibm.commerce.sample.commands.MyNewTaskCmd", getStoreId());
```

```
        // 這是所有指令的必要項目
               cmd.setCommandContext(getCommandContext());

        /// 設定作業指令的輸入參數
        cmd.setInputUserName(getUserName());
        cmd.setInputPoints(getPoints()); // 變更為整數

      /// 結束區段 4A ////////////////////////////////////

      /// 區段 4B ////////////////////////////////

        /// 呼叫指令的 performExecute 方法
         cmd.execute();

        /// 從作業指令擷取輸出參數，然後置於
           /// 回應內容中
        rspProp.put("taskOutputGreetings", cmd.getGreetings());

      /// 結束區段 4B ////////////////////////////////

      /// 開始區段 4C ////////////////////////////////
       } catch (ECException ex) {
        /// 依現狀擲出異常狀況
               throw (ECException) ex;
       }

      /// 結束區段 4C ////////////////////////////////
```

「區段 4A」會使用指令 factory 來建立新的作業指令物件。接著它會設定指令環境定義，並設定作業指令的輸入參數。「區段 4B」會先呼叫 validateParameters 方法來進行存取控制，然後再呼叫作業指令的 execute 方法。然後它會從作業指令擷取問候值。「區段 4C」是用於異常狀況的簡易攫取區塊。

5. 儲存變更。
6. 編譯程式碼變更，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。

修改 MyNewJSPTemplate 來新增問候訊息

在這個步驟中，您會修改 MyNewJSPTemplate.jsp 檔，加入一個新的區段來顯示問候訊息，步驟如下：

1. 如果 JSP 範本檔案尚未開啓，請執行下列步驟：
   a. 在 Web 視景中，切換到「J2EE 導覽器」檢視畫面，然後展開 **Stores** Web 專案。
   b. 導覽至 Web Content*FashionFlow_name* 子資料夾。

c. 同時標示 **MyNewJSPTemplate_All.jsp** 和 **MyNewJSPTemplate.jsp** 檔案，以滑鼠右鍵按一下，然後選取**開啓方式 > Page Designer**。

2. 將 MyNewJSPTemplate_All.jsp 檔的「區段 7」複製到 MyNewJSPTemplate.jsp 檔中。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 7 -->

<fmt:message key="Greeting" bundle="${tutorial}" />
<c:out value="${taskOutputGreetings}"/> <br /> <br />

<!-- 區段 7 結束 -->
```

3. 儲存變更。

測試 MyNewTaskCmd

下一步是執行下列步驟來測試新的作業指令是否正常運作：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 接下來，輸入下列 URL：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

這時會顯示 MyNewJSPTemplate。其中包含作業指令所建立的問候訊息。

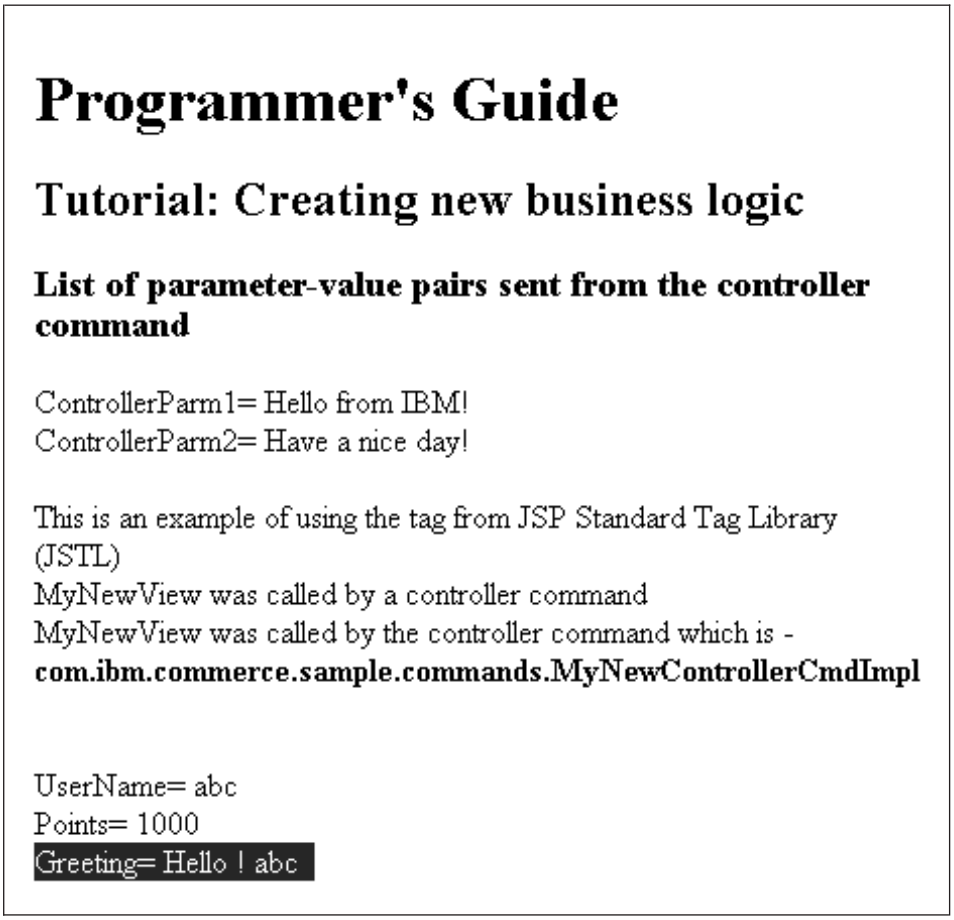

圖 39.

修改 MyNewTaskCmd

在指導教學的這個步驟中，您將會修改 MyNewTaskCmd，來判斷 URL 中的使用者名稱是屬於已登錄的使用者。同時您也會修改 MyNewDataBean 來處理作業指令的欄位（例如，使用者名稱）。同樣的，您也會修改 MyNewJSPTemplate，來顯示使用者是否已經登錄。

在本節指導教學中，您將學習下列內容：

- 如何從您的自訂程式碼中來使用 URL 參數以及存取現有的 WebSphere Commerce 資訊

修改 MyNewControllerCmdImpl 來建立作業指令的物件

爲了提升物件的使用效率，控制程式指令會建立一個 UserRegistryAccessBean 物件實例變數。因此，這個物件可以提供給作業指令來使用。利用這個方法，作業指令就不需要建立個別的物件實例。您在稍後會使用這個 UserRegistryAccessBean 物件（在作業指令中）來判斷購物者是否爲已登錄的使用者。

如果要修改 MyNewControllerCmdImpl，請執行下列步驟：

1. 在 Java 視景中，按兩下 **MyNewControllerCmdImpl.java** 類別來檢視其原始碼。
2. 在這個類別的主要主體中，取消註解「區段 2」，以便將下列程式碼引入到類別中：

```
/// 區段 2 ////////////////////////////////////////////
/// 建立一個使用者登錄 accessbean 資源實例變數

    private UserRegistryAccessBean rrb = null;

/// 區段 2 結束 ////////////////////////////////////////
```

3. 在「大綱」檢視畫面中，選取 **performExecute** 方法。
4. 在 performExecute 方法的原始碼中，取消註解「區段 4D」和「區段 4F」，來將實例變數傳送到作業指令，然後使傳回的使用者 ID 可以在回應內容中使用。這樣會將下列程式碼引入方法中：

```
// 區段 4D ///////////////////////////////////
/// 傳送 rrb 實例變數到作業指令

  cmd.setUserRegistryAccessBean(rrb);

// 區段 4D 結束 /////////////////////////////


// 區段 4F ///////////////////////////////////////
///使用存取 Bean 來取得資料庫的資訊
  if (cmd.getFoundUserId() != null) {
   rspProp.put("taskOutputUserId", cmd.getFoundUserId());
  }
// 區段 4F 結束 /////////////////////////////
```

您會收到一個錯誤，指出某些方法尚未定義，但是這個問題在您進行下一個步驟的修改作業指令時將會解決。

5. 儲存您的工作。

修改新的作業指令以進行使用者名稱驗證

接下來您必須修改新的作業指令，來驗證 URL 中的使用者名稱輸入是否屬於已登錄的使用者，方式如下：

1. 按兩下 **MyNewTaskCmd.java** 介面來檢視其原始碼。
2. 取消註解「區段 3」來將下列程式碼引入到介面中：

```
/// 區段 3 ////////////////////////////////////////////

 public void setFoundUserId(java.lang.String inputUserId);
 public java.lang.String getFoundUserId();

 public void setUserRegistryAccessBean(UserRegistryAccessBean rrb);

/// 區段 3 結束 ///////////////////////////////////////
```

3. 儲存您的工作。
4. 按兩下 **MyNewTaskCmdImpl.java** 類別來檢視其原始碼。取消註解「匯入區段 1」來將下列兩個 import 陳述式引入到程式碼中：

```
/// 匯入區段 1 ////////////////////////////////////
import com.ibm.commerce.user.objects.*;
import com.ibm.commerce.sample.databeans.*;
///  匯入區段 1 結束 ////////////////////////////////
```

5. 取消註解「區段 2A」和「區段 2B」，來建立新的欄位以及和已新增至介面中的方法相對應的 getter 和 setter 方法。這樣會將下列程式碼引入類別中：

```
//// 區段 2A ////////////////////////////////////////////

 private java.lang.String foundUserId = null;

 private UserRegistryAccessBean rrb = null;

//// 區段 2A 結束 ////////////////////////////////////////

//// 區段 2B ////////////////////////////////////////////

 public void setUserRegistryAccessBean(UserRegistryAccessBean newRRB) {
  rrb = newRRB;
 }

 public void setFoundUserId(java.lang.String newFoundUserId) {
  foundUserId = newFoundUserId;
 }

 public java.lang.String getFoundUserId() {
  return foundUserId;
 }


/// 區段 2B 結束 ////////////////////////////////////////
```

6. 在「大綱」檢視畫面中，選取 **validateParameters** 方法並檢視其原始碼。取消註解「區段 1」來將下列程式碼引入到方法中：

```
// 區段 1 ////////////////////////////////////////////////
```

```
// 使用 UserRegistryAccessBean 來檢查使用者 ID

 try {

   if (rrb!=null){
     setFoundUserId(rrb.getUserId());
 } else {
     rrb =new UserRegistryAccessBean();
     rrb=rrb.findByUserLogonId(getInputUserName());
     setFoundUserId(rrb.getUserId());
   }

} catch (javax.ejb.FinderException e) {
      return;

     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(), "validateParameters");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
  }

// 區段 1 結束 ////////////////////////////////////////////
```

7. 儲存您的工作。
8. 編譯程式碼變更，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。

修改 MyNewJSPTemplate 以進行使用者名稱驗證

您必須修改現行的 JSP 範本，才能顯示使用者名稱驗證資訊。如果要修改這個檔案，請執行下列步驟：

1. 切換到「Web」視景。
2. 同時開啓 **MyNewJSPTemplate_All.jsp** 和 **MyNewJSPTemplate.jsp** 檔案。
3. 將 MyNewJSPTemplate_All.jsp 檔的「區段 8」複製到 MyNewJSPTemplate.jsp 檔。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 8 -->

<c:if test="${!empty taskOutputUserId}">
   <fmt:message key="UserId" bundle="${tutorial}" />
   <c:out value="${taskOutputUserId}"/> <br />
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="RegisteredUser" bundle="${tutorial}" /> <br />
   <fmt:message key="ReferenceNumber" bundle="${tutorial}" />
   <b><c:out value="${taskOutputUserId}"/></b> <br /> <br />
```

```
</c:if>

<c:if test="${empty taskOutputUserId}">
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="NotRegisteredUser" bundle="${tutorial}" /> <br />
</c:if>


<!-- 區段 8 結束 -->
```

4. 儲存對 MyNewJSPTemplate.jsp 檔所做的變更。

測試使用者名稱驗證

如果要測試使用者名稱驗證邏輯，請執行以下步驟：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 執行下列步驟來建立一個新的已登錄使用者：
   a. 按一下**登錄**。
   b. 再按一下**登錄**來建立新的客戶。
   c. 在登錄套表中，在所有的必要欄位中輸入適當的值。例如，在電子郵件欄位中，輸入 `tester@mycompany`。請記下電子郵件位址的值：_____。
   d. 一旦輸入這些值之後，請按一下**提交**。
5. 接下來，輸入一個有效的使用者名稱來作爲 input1 的值。輸入下列 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=new_e-mail&input2=1000
   ```

   其中 *new_e-mail* 是您在步驟 4 中所建立的使用者的電子郵件位址。這時會顯示 MyNewJSPTemplate。它現在應該會指出 input1 的值是有效的使用者名稱。

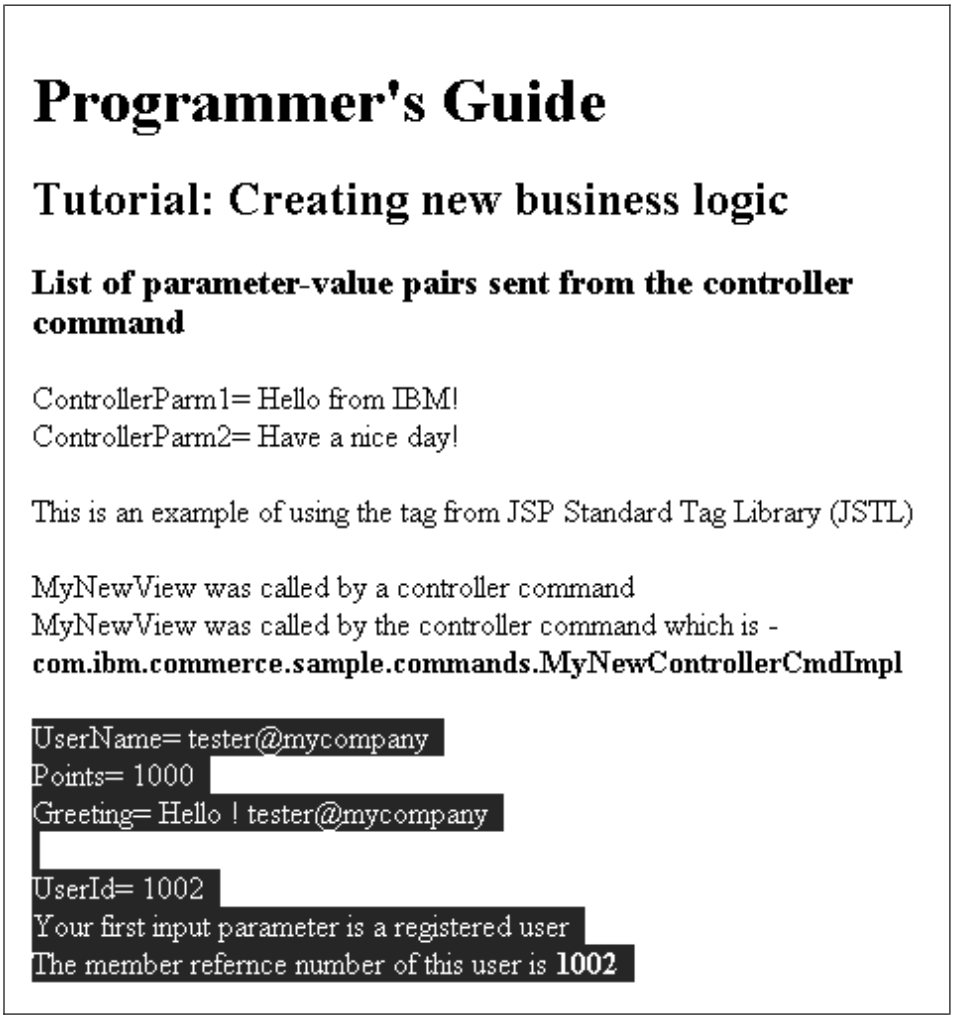

圖 40.

6. 接下來，輸入下列 URL（其中的使用者名稱值是無效的）：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

這時會顯示一個同屬異常狀況。如果您檢視頁面的原始碼，就發現已經發生 _ERR_FINDER_EXCEPTION。這是因爲所提供的使用者名稱並沒有對應到已登錄的使用者。

建立新 Entity Bean

本節說明如何建立新的 Entity Bean。在本範例的實務中，您有一項商業需求，亦即將各使用者的紅利積點納入到商務應用程式中。 WebSphere Commerce 資料庫綱目不含此項資訊，因此您需要另建一個新資料庫表格以存放此資訊。根據 WebSphere Commerce 程式設計模型，一旦建立資料庫表格，就必須建立一個 Entity Bean 來存取資料。

建立 XBONUS 表格

在準備建立 Entity Bean 的過程中，您必須先建立新資料庫表格。要建立的表格稱爲 XBONUS。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令行工具 > 指令中心**），然後按一下 **Script 編寫**標籤。
2. 在「Script」視窗中，輸入下列指令：

```
connect to developmentDB user dbuser using dbpassword;
create table XBONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL,
           constraint p_xbonus primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼

按一下「執行」圖示。
您應該會看到一則訊息，指出 SQL 陳述式已經順利完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
create table XBONUS (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL,
           constraint p_xbonus primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XBOUNU 表格。

6. 輸入下列以確定您的資料庫變更：

```
commit;
```

並按 Enter 鍵，以執行 SQL 陳述式。

建立 BonusBean Entity Bean

一旦建立表格，即可準備開始建立新 Entity Bean。接下來的步驟將使用 WebSphere Studio Application Developer 來建立這個 Bean。

接下來，您可以執行下列步驟來建立新的 Bonus Bean：

1. 在 WebSphere Studio Application Developer 中，切換到「J2EE」視景。
2. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 模組，然後選取**新建 >**（**其他 >**）**Enterprise Bean**。
   這時會開啓「Enterprise Bean 建立」精靈。
4. 從 **EJB 專案**下拉清單中，選取 **WebSphereCommerceServerExtensionsData**，然後按一下**下一步**。
5. 在「建立 Enterprise Bean」視窗中，執行下列步驟：
   a. 選取**具有儲存器所管理的持續 (CMP) 欄位的 Entity Bean**
   b. 在 **Bean 名稱**欄位中，輸入 Bonus。
   c. 在**來源資料夾**欄位中，保留已指定的預設值 (ejbModule)。
   d. 在**預設套件**欄位中，輸入 com.ibm.commerce.extension.objects。
   e. 按**下一步**。
6. 在「Enterprise Bean 明細」視窗中，執行下列步驟：
   a. 按一下**新增**來爲 BONUS 表格中的 MEMBERID 直欄和 BONUSPOINT 直欄加入新的 CMP 屬性。
      這時會開啓「建立 CMP 屬性」視窗。請在此視窗中執行下列步驟：
      1) 在**名稱**欄位中，輸入 memberId。
      2) 在**類型**欄位中，輸入 java.lang.Long。

         註: 您必須使用 *java.lang.Long* 資料類型，而不是使用 *long* 資料類型。

3) 選取**鍵值欄位**勾選框。
4) 按一下**套用**。
5) 在**名稱**欄位中，輸入 bonusPoint。
6) 在**類型**欄位中，輸入 java.lang.Interger。

註: 您必須使用 *java.lang.Interger* 資料類型，而不是使用 *interger* 資料類型。

7) 選取**使用 getter 與 setter 方法來存取**勾選框。
8) 清除**將 getter 與 setter 方法提升至遠端介面**勾選框。**使 getter 成為唯讀**勾選框將會無法使用。
9) 按一下**套用**。
10) 按一下**關閉**來關閉視窗。

b. 清除**在鍵值類別中使用單一鍵值屬性類型**勾選框，然後按一下**下一步**。

7. 在「EJB Java 類別明細」視窗中，執行下列步驟：
   a. 如果要選取 Bean 的超類別，請按一下**瀏覽**。
   這時會開啓「類型選項」視窗。
   b. 在**選取類別的方法：（任何）**欄位中，輸入 ECEntityBean，然後按一下**確定**。這樣就會選取 com.ibm.commerce.base.objects.ECEntityBean 作爲超類別。
   c. 按一下**新增**來指定遠端介面應該延伸的介面。這時會開啓「類型選項」視窗。
   d. 在**選取類別的方法：（任何）**欄位中，輸入 Protectable，然後按一下**確定**。這樣就會選取 com.ibm.commerce.security.Protectable。您需要這個介面才能使新的資源受到存取控制的保護。
   e. 按下**完成**。

執行下列步驟來設定新 Bean 的隔離層次：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 按兩下 **WebSphereCommerceServerExtensionsData** 專案，以使用「部署描述子編輯程式」來開啓它。（在這個編輯程式中開啓這個檔案的另一種方法：
   a. 在「J2EE 導覽器」檢視畫面中，展開 **WebSphereCommerceServerExtensionsData > ejbModule > META-INF**。
   b. 以滑鼠右鍵按一下 **ejb-jar.xml**，然後選取**開啓方式 > 部署描述子編輯程式**）
3. 按一下**存取**標籤。

4. 按一下「隔離層次」文字框旁邊的**新增**。
   這時會開啓「新增隔離層次」視窗。
5. DB2 選取**可重複讀取**，然後按一下**下一步**。
   Oracle 選取**讀取已確定**，然後按一下**下一步**
6. 從**找到的 Bean** 清單中，選取 **Bonus** Bean，然後按一下**下一步**。
7. 從**找到的方法**清單中，選取 **Bonus** 來選取其所有的方法，然後按一下**完成**。
8. 儲存您的工作 (Ctrl + S)，並保持編輯程式開啓。

接下來，設定 Bean 的安全身份，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，確定您已經選取「存取」標籤。
2. 按一下「安全身份」文字框旁邊的**新增**。
   這時會開啓「新增安全身份」視窗。
3. 選取**使用 EJB 伺服器的身份**，然後按一下**下一步**。
4. 從**找到的 Bean** 清單中，選取 **Bonus** Bean，然後按一下**下一步**。
5. 從**找到的方法**清單中，選取 **Bonus** 來選取其所有的方法，然後按一下**完成**。
6. 儲存您的工作 (Ctrl + S)，並保持編輯程式開啓。

接下來，您需要設定 Bean 中的方法的安全職務，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，選取「組譯描述子」標籤。
2. 在「方法許可權」區段中，按一下**新增**。
3. 選取 **WCSecurityRole** 作爲安全職務，然後按一下**下一步**。
4. 從找到的 Bean 清單中，選取 **Bonus**，然後按一下**下一步**。
5. 在「方法元素」頁面中，按一下**全部套用**，然後按一下**完成**。
6. 儲存您的工作 (Ctrl+S)，然後關閉「部署描述子」編輯程式。

下一步是移除 WebSphere Studio Application Developer 產生的某些與實體環境定義相關的欄位和方法。這些欄位需要刪除是因爲 ECEntityBean 基本類別已提供自己對這些方法的實作。如果要刪除產生的實體環境定義欄位與方法，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **WebSphereCommerceServerExtensionsData** 專案。
2. 展開 **Bonus** Bean，然後按兩下 **BonusBean** 類別。
3. 在「大綱」檢視畫面中，執行下列步驟：
   a. 以滑鼠右鍵按一下 **myEntityCtx** 欄位並選取**刪除**。
   b. 以滑鼠右鍵按一下 **getEntityContext()** 方法，並選取**刪除**。

c. 以滑鼠右鍵按一下 **setEntityContext(EntityContext)** 方法，並選取**刪除**。

d. 以滑鼠右鍵按一下 **unsetEntityContext()** 方法並選取**刪除**。

4. 儲存您的工作 (Ctrl+S)。使 BonusBean 類別保持開啓。

接下來，執行下列步驟，將新的 getMemberId 方法加入到 Entrprise Bean 中：

1. 檢視 **BonusBean** 類別的原始碼。
2. 新增下列程式碼到這個類別結尾（仍在類別中）：

```
public java.lang.Long getMemberId() {
   return memberId;
}
```

3. 您必須執行下列步驟，將新的方法加入到遠端介面：
   a. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **getMemberId** 方法，然後選取 **Enterprise Bean > 提升至遠端介面**。一旦完成這個步驟，就會在方法旁邊顯示一個小的 R 圖示，指出該方法已經提升至遠端介面。
4. 儲存您的工作。
5. 關閉 BonusBean 編輯程式。

接下來，執行下列步驟來將新的 FinderHelper 方法加入 BonusHome 介面中：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 按兩下 **WebSphereCommerceServerExtensionsData** 專案來開啓「 EJB 部署描述子編輯程式」。
3. 按一下 **Bean** 標籤。
4. 在 Bean 窗格中，選取 **Bonus** Bean，然後在右邊的窗格中，向下捲動並展開 **WebSphere 延伸**。
5. 按一下**搜尋器**文字框旁邊的**新增**。
   這時會開啓「新增搜尋器描述子」視窗。
6. 選取**新建**，然後在**名稱**欄位中，輸入 `findByMemberId`。
7. 按一下**參數**文字框旁邊的**新增**，然後執行下列步驟
   a. 在**名稱**欄位中，輸入 `memberId`。
   b. 在**類型**欄位中，輸入 `java.lang.Long`。
   c. 按一下**確定**。
8. 從**傳回類型**下拉清單中，選取 **com.ibm.commerce.extension.objects.Bonus**，然後按一下**下一步**。
9. 從**搜尋器類型**下拉清單中，選取 **WhereClauseFinderDescriptor**。
10. 在**搜尋器陳述式**欄位中，輸入 `T1.MEMBERID = ?`，然後按一下**完成**。
11. 儲存您的工作，然後關閉「EJB 部署描述子」編輯程式。

接下來，執行下列步驟，將新的 ejbCreate 方法加到 Bonus Bean 中：

1. 在「J2EE 階層」檢視畫面中，按兩下 **BonusBean** 類別來開啓它並檢視其原始碼。
2. 新增下列程式碼到類別中，以建立一個新的 ejbCreate(Long, Integer) 方法：

```
public com.ibm.commerce.extension.objects.BonusKey ejbCreate(
   java.lang.Long memberId,java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException {
   _initLinks();
      this.memberId=memberId;
      this.bonusPoint=bonusPoint;
return null;
}
```

3. 儲存程式碼變更。
4. 您必須將新的 ejbCreate(Long, Integer) 方法加到發源介面。這樣就能在產生的存取 Bean 中提供該方法。如果要新增方法到發源介面中，請執行下列步驟：
   a. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **ejbCreate(Long, Integer)** 方法，然後選取 **Enterprise Bean > 提升至發源介面**。

接下來，執行下列步驟來建立一個新的 ejbPostCreate(Long, Integer) 方法，使它具備和 ejbCreate(Long, Integer) 方法相同的參數：

1. 按兩下 **BonusBean** 類別來開啓它並檢視其原始碼。
2. 新增下列程式碼到類別中，以建立一個新的 ejbPostCreate(Long, Integer) 方法：

```
public void ejbPostCreate(java.lang.Long memberId,
   java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException
   {
   }
```

3. 儲存程式碼變更。

接下來的步驟是將新的方法加入到 Bonus Bean，使它們能受到 WebSphere Commerce 存取控制系統的保護。您可以執行下列步驟，將 getOwner 和 fulfills 方法新增到 Bean 中：

1. 按兩下 **BonusBean** 類別來開啓它並檢視其原始碼。
2. 新增下列程式碼到類別結尾，來將新的 getOwner 方法加到 BonusBean 類別：

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. 儲存您的工作。
4. 新增下列程式碼到類別結尾來加入新的 fulfills 方法：

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      return false;
   }
```

5. 儲存您的工作並關閉 Bonus Bean 編輯程式。

下一步是將 XBONUS 表格對映到 BonusBean Entity Bean。這時會使用「上下同時進行」對映。如果要建立對映，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**產生 > EJB 至 RDB 對映**。
   這時會開啓「EJB 至 RDB 對映」視窗。
2. 選取**上下同時進行**，然後按一下**下一步**。
3. 在「資料庫連線」視窗中，執行下列步驟：
   a. 在**連線名稱**欄位中，輸入 WebSphereCommerceServerExtensionsData
   b. 在**資料庫欄位**中，輸入您的開發資料庫的名稱。
   c. 在**使用者 ID** 欄位中，輸入資料庫使用者 ID。
   d. 在**密碼**欄位中，輸入資料庫使用者的密碼。
   e. 從**資料庫供應商類型**下拉清單中，選取您的開發資料庫的資料庫供應商類型。
      - DB2 DB2 Universal Database 8.1
      - Oracle Oracle 9i
   f. Oracle 在**主機**欄位中，輸入您的資料庫伺服器的完整主機名稱。例如，輸入 dbserver.yourcompany.com
   g. Oracle 在「類別位置」欄位中，輸入 classes12.zip 檔的位置。例如，輸入 D:\oracle\ora92\jdbc\lib\classes12.zip
   h. 按**下一步**。一旦建立連線，就會顯示資料庫的表格清單。您也可以在稍後察看「資料」視景中的「資料庫伺服器」，來檢視「連線文件」。
4. 選取 **XBONUS** 表格，然後按一下**下一步**。
5. 選取**比對名稱和類型**，然後按一下**完成**。這時會開啓「對映編輯程式」。
6. Oracle 以滑鼠右鍵按一下 XBONUS 表格，然後選取**開啓表格編輯程式**。在表格編輯程式中，執行下列步驟：

a. 選取**直欄**標籤。

b. 選取 BONUSPOINT 直欄，然後將直欄類型從 NUMBER 變更爲 INTEGER

c. 儲存變更。

7. 在 Enterprise Bean 窗格中，展開 **Bonus** Bean。在「表格」窗格中，展開 **XBONUS** 表格。
8. 將 Bonus Bean 中的欄位對映至 XBONUS 表格中的直欄，步驟如下：
   a. 以滑鼠右鍵按一下 Bonus Bean，然後選取**比對名稱**。
9. 按下 Ctrl + S 來儲存 Map.mapxmi 檔。關閉檔案。
10. Oracle 您必須使用文字編輯程式來編輯表格定義，方式如下：
    a. 在文字編輯程式中開啓 XBONUS.xmi 檔。
    b. 將所有出現的 `SQLNumeric6` 換成 `SQLNumeric3`。
    c. 儲存變更。

下一步是修正綱目名稱，使新的 Bean 可以轉移到其他資料庫。如果要進行這項修改，請執行下列步驟：

1. 在「J2EE 視景」中，切換到「J2EE 階層」檢視畫面。
2. 展開**資料庫**，然後展開**WebSphereCommerceServerExtensionsData**。
3. 以滑鼠右鍵按一下 綱目節點（例如，DB2USER），然後選取**更名**。
4. 將值設定爲 `NULLID`。

一旦建立 BonusBean 實體，並且正確對映綱目後，您就可以爲 Entity Bean 建立存取 Bean。此存取 Bean 可方便應用程式存取 Bonus Entity Bean 中所含的資訊。WebSphere Studio Application Developer 中的工具可讓您根據您已經建立的 Entity Bean，來產生這個存取 Bean（特別是已經提升至遠端介面，並且將提供給存取 Bean 使用的方法）。如果要爲 Bonus Entity Bean 建立存取 Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**新建 > 存取 Bean**。
   這時會開啓「新增存取 Bean」視窗。
2. 選取**複製輔助程式**，然後按一下**下一步**。
3. 選取 **Bonus** Bean，然後按一下**下一步**。
4. 從「建構子方法」下拉清單中，選取 **findByPrimaryKey(com.ibm.commerce.extension.objects.BonusKey** 作爲建構子方法。
5. 選取「屬性輔助程式」區段中的所有屬性。

6. 按下**完成**。

您可以切換至「J2EE 導覽器」標籤，然後展開下列項目來檢視新產生的程式碼：**WebSphereCommerceServerExtensionsData > ejbModule > com.ibm.commerce.extension.objects**
此時會建立一個叫做 BonusAccessBean 的新類別，以及一個叫做 BonusAccessBeanData 的新介面，然後在套件中顯示它們。

下一步是產生已部署的程式碼。

程式碼產生公用程式會分析 Bean，以確定是否符合 Sun Microsystem 的 EJB 規格，並確定是否遵循 EJB 伺服器的特定規則。此外，在每一個所選的 Enterprise Bean 方面，程式碼產生工具會爲發源與遠端介面產生發源與 EJBObject（遠端）實作方式與實作類別，以及爲 CMP Bean 產生 JDBC persister 與 finder 類別。它同時會產生透過 IIOP 進行 RMI 存取時所需要的 Java ORB、存根與 Tie 類別，以及發源和遠端介面的存根。

如果要產生已部署的程式碼，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開**EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 並選取**產生 > 部署和 RMIC 程式碼**。
2. 選取 **Bonuss** Bean，然後按一下**完成**。

您可以切換到「J2EE 導覽器」檢視畫面，來檢視新產生的程式碼。您會發現下列內容：

表 11.

程式碼類型	類別名稱
配置區實作產生的程式碼	EJSCMPBonusHomeBean.java
	EJSRemoteCMPBonus.java
	EJSRemoteCMPBonusHome.java
	EJSFinderBonusBean.java
JDBC 存取程式碼	EJSJDBCPersisterCMPBonusBean.java
RMI tie 和存根程式碼	_EJSRemoteCMPBonus_Tie.java
	_Bonus_Stub.java
	_EJSRemoteCMPBonusHome_Tie.java
	_BonusHome_Stub.java

下一步是使用通用測試用戶端來測試新的 Enterprise Bean，步驟如下：

1. 切換到「伺服器」視景。

2. 在「伺服器配置」檢視畫面中，按兩下 **WebSphereCommerceServer** 伺服器，然後按一下**配置**標籤。
3. 選取**啓用通用測試用戶端**。儲存變更。
4. 在「伺服器」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**。
5. 在「J2EE 階層」中，展開 **EJB 模組 > WebSphereCommerceServerExtensionsData**。
6. 以滑鼠右鍵按一下 **Bonus** Bean，然後選取**在伺服器上執行**。這時會開啓 IBM Universal 測試用戶端。
7. 在左窗格中，按一下 **Bonus**，然後按一下 **Bonus 起始**。
8. 按一下 **Bonus create(Long, Integer)** 方法。
9. 在右窗格的 **Long** 欄位中，輸入 -1000，然後在 **Integer** 欄位中，輸入 1000。
10. 按一下**呼叫**，然後結果就會顯示在底端的窗格中。
11. 按一下**處理物件**來將遠端介面新增至「參照」窗格，然後您就會在「EJB 參照」下面看到您所輸入的值。這時 BONUS 表格中已經建立一筆新紀錄。
12. 選取 **getMemberId** 方法，然後按一下**呼叫**，就會在底端窗格中顯示結果是 -1000。
13. 關閉測試用戶端並停止伺服器。

整合 Bonus Entity Bean 與 MyNewControllerCmd

在上一節中，您使用 WebSphere Studio Application Developer 中產生的測試用戶端來測試新 Bonus Entity Bean。因爲這麼做，您已經判斷出您可以順利更新資料庫資訊。現在，您將整合 Bonus Entity Bean 與 MyNewControllerCmd 邏輯。一旦更新 Java 程式碼，就會更新 MyNewJSPTemplate.jsp 檔來建立一個介面，以容許更新客戶的紅利積點結餘。

整合 Bonus Entity Bean 將牽涉到如下的高階步驟：

1. 修改 MyNewTaskCmd 作業指令以併入紅利積點的欄位和方法，更新 validateParameters 方法，以及新增邏輯來更新使用者的紅利積點結餘。
2. 新增 getResources 方法至 MyNewControllerCmdImpl 類別，以傳回指令所使用的資源的清單。包含此方法其目的在於存取控制。
3. 建立一個新的 BonusDataBean，以便在 JSP 範本中顯示紅利積點。
4. 爲新資源建立新的存取控制原則。
5. 修改 MyNewJSPTemplate.jsp 範本，以讓您輸入使用者的紅利積點，然後顯示該使用者的新紅利積點結餘。

修改 MyNewTaskCmd 介面以併入紅利積點

在這個步驟中，您會修改 MyNewTaskCmd 介面來指定紅利積點的必要欄位和方法，步驟如下：

1. 切換到 Java 視景，然後展開 **WebSphereCommerceServerExtensionsLogic** 專案。
2. 展開 **com.ibm.commerce.sample.commands\src** 目錄。
3. 按兩下 **MyNewTaskCmd** 介面來檢視其原始碼。
4. 取消註解「匯入區段 2」來併入下列套件：

```
/// 匯入區段 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// 匯入區段 2 結束 ////////////////////////////////
```

5. 取消註解「區段 4」來將下列程式碼引入到方法中：

```
/// 區段 4 ////////////////////////////////////////////////

 public java.lang.Integer getOldBonusPoints();
 public Integer getTotalBonusPoints();

 public void setBonusAccessBean(BonusAccessBean bb);
 public BonusAccessBean getBonusAccessBean();


/// 區段 4 結束 ///////////////////////////////////////////
```

6. 儲存變更。

修改 MyNewTaskCmdImpl 來計算紅利積點

MyNewTaskCmdImpl 是用來作爲 Bonus Entity Bean 與 MyNewControllerCmd 之間的整合點（因爲 MyNewControllerCmd 會呼叫 MyNewTaskCmd）。

如果要修改 MyNewTaskCmdImpl 來計算紅利積點，請執行下列步驟：

1. 選取 **MyNewTaskCmdImpl** 類別來檢視其原始碼。
2. 取消註解「匯入區段 2」來引入下列套件：

```
/// 匯入區段 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
///  匯入區段 2 結束 ////////////////////////////////
```

3. 取消註解「區段 3A」和「區段 3B」來將下列程式碼引入到類別中：

```
//// 區段 3A ////////////////////////////////////////////

 private java.lang.Integer oldBonusPoints;
 private java.lang.Integer totalBonusPoints;


 private BonusAccessBean bb = null;

//// 區段 3A 結束 //////////////////////////////////////
```

```
//// 區段 3B ////////////////////////////////////////////

 public void setBonusAccessBean(BonusAccessBean newBB) {
  bb = newBB;
 }

 public BonusAccessBean getBonusAccessBean(){
  return bb;
 }

    public java.lang.Integer getOldBonusPoints() {
  return oldBonusPoints;
 }

 public Integer getTotalBonusPoints(){
  return totalBonusPoints;
 }


/// 區段 3B 結束 ///////////////////////////////////////
```

4. 在「大綱」檢視畫面中，選取 **validateParameters** 方法，然後取消註解「區段 2」，以便將下列程式碼引入到方法中：

```
// 區段 2 ////////////////////////////////////////////////////

 try {
    oldBonusPoints = bb.getBonusPoint();
} catch (javax.ejb.FinderException e) {
  try {
   // 若 bb 是空值，則建立新的實例
    bb = new BonusAccessBean(new Long(foundUserId), new Integer(0));
    oldBonusPoints = new Integer(0);
     } catch (javax.ejb.CreateException ec) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     this.getClass().getName(), "validateParameters");
     } catch (javax.naming.NamingException ec) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
     this.getClass().getName(), "validateParameters");
     } catch (java.rmi.RemoteException ec) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
     this.getClass().getName(), "validateParameters");
   }
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(), "validateParameters");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
```

```
    }

// 區段 2 結束 ////////////////////////////////////////////
```

5. 在「大綱」檢視畫面中，選取 **performExecute** 方法。
6. 在 performExecute 方法的原始碼中，取消註解「區段 2」。這樣會將下列程式碼引入方法中：

```
/// 使用 BonusAccessBean 來更新新的紅利積點
/// 區段 2 ////////////////////////////////////////////////

 int newBP =  oldBonusPoints.intValue() + getInputPoints().intValue();
 totalBonusPoints = new Integer (newBP);
 bb.setBonusPoint(totalBonusPoints)   ;

 try {
   bb.commitCopyHelper();
} catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
               this.getClass().getName(), "performExecute");
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
               this.getClass().getName(), "performExecute");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
               this.getClass().getName(), "performExecute");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
               this.getClass().getName(), "performExecute");
 }

/// 區段 2 結束 ////////////////////////////////////////////
```

7. 儲存變更。

新增 getResources 方法到 MyNewControllerCmdImpl 類別中

在本節中，您會將新的 getResources 方法加入 MyNewControllerCmdImpl 中。這個方法會傳回指令在處理程序期間所使用的資源的清單。在資源層次的存取控制方面，需用到此方法。

如果要新增 getResources 方法，請執行下列步驟：

1. 按兩下 **MyNewControllerCmdImpl** 類別來開啓並檢視其原始碼。
2. 在原始碼中，取消註解「匯入區段 2」，以便將下列套件引入到類別中：

```
/// 匯入區段 2 ////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// 匯入區段 2 結束 ///////////////////////
```

3. 取消註解「區段 3」來將下列程式碼引入到類別中：

```
/// 區段 3 ////////////////////////////////////////////
/// 建立一個類型為 AccessVector 的實例變數來保留
/// 資源，並建立一個 BonusAccessBean 實例變數來進行
/// 存取控制。

private AccessVector resources = null;
    private BonusAccessBean bb = null;

/// 區段 3 結束 ///////////////////////////////////////
```

4. 在原始碼中，取消註解「存取控制區段」。此區段如下列的程式碼片段所示：

```
/// AccessControl 區段 ////////////////////////////////

public AccessVector getResources() throws ECException{

   if (resources == null) {

    /// 使用 UserRegistryAccessBean 來檢查使用者參考號碼

 String refNum = null;
 String methodName = "getResources";

  rrb = new UserRegistryAccessBean();

 try {
   rrb = rrb.findByUserLogonId(getUserName());
          refNum = rrb.getUserId();
} catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
        this.getClass().getName(),methodName,e);
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
        this.getClass().getName(), methodName,e);
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
        this.getClass().getName(), methodName,e);
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        this.getClass().getName(), methodName,e);
  }


   /// 尋找這個已登錄使用者的 Bonus Bean

    bb = new com.ibm.commerce.extension.objects.BonusAccessBean();
 try {
     if (refNum != null) {
    bb.setInitKey_memberId(new Long(refNum));
    bb.refreshCopyHelper();
    resources = new AccessVector(bb);
   }
} catch (javax.ejb.FinderException e) {

  /// 沒有 Bonus 物件，所以傳回在建立時
     /// 將用來保留 Bonus 物件的配置區
```

```
  resources = new AccessVector(rrb);
   return resources;

     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
      this.getClass().getName(), methodName);
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
      this.getClass().getName(), methodName);
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
      this.getClass().getName(), methodName);
  }

 }
   return resources;
}

/// AccessControl 區段結束 ////////////////////////////////
```

5. 儲存變更。

修改 MyNewControllerCmdImpl 類別的 performExecute 方法

在這個步驟中，您會修改 MyNewControllerCmdImpl 的 performExecute 方法中的程式碼，來併入與新的 Bonus Bean 相關的程式碼，方式如下：

1. 按兩下 **MyNewControllerCmdImpl** 類別。
2. 在「大綱」檢視畫面中，選取 **performExecute** 方法。
3. 在原始碼中，取消註解「區段 4E」、「區段 4G」和「區段 4H」，以便將下列程式碼引入到方法中：

```
// 區段 4E ////////////////////////////////////
/// 傳送 bb 實例變數到作業指令
    cmd.setBonusAccessBean(bb);
// 區段 4E 結束 ////////////////////////////////

// 區段 4G ////////////////////////////////////
if (cmd.getOldBonusPoints() != null) {
      rspProp.put("oldBonusPoints", cmd.getOldBonusPoints());
}
// 區段 4G 結束 ////////////////////////////////

// 區段 4H ////////////////////////////////////
///將 Bonus 資料 Bean 實例化，然後放置到回應內容中
   BonusDataBean bdb  =
      new com.ibm.commerce.sample.databeans.BonusDataBean(
         cmd.getBonusAccessBean());
  rspProp.put("bdbInstance", bdb );
// 區段 4H 結束 ////////////////////////////////
```

4. 儲存變更。

註: 您會看到錯誤，因爲 BonusDataBean 尚未定義。下一節將修正這個問題。

建立 BonusDataBean 資料 Bean

爲了符合程式設計模型，您應該建立一個新的資料 Bean 來對應到新的 Bonus Entity Bean。並非所有的 Entity Bean 都需要有對應的資料 Bean，如果您要能夠從 JSP 範本中的 Entity Bean 來顯示資訊，就應該針對這個原因來建立一個新的資料 Bean。

在這個實務內容中，您需要建立一個新的 BonusDataBean 資料 Bean 來延伸 BonusAccessBean。如同指導教學的其他部分一樣，此處提供的是基本程式碼，您需要取消註解許多不同的程式碼區段。

如果要將 BonusDataBean 引入到程式碼中，請執行下列步驟：

1. 第一步是匯入新資料 Bean 的基本程式碼，方式如下：
   a. 切換到 Java 視景中的「J2EE 導覽器」檢視畫面。
   b. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
   c. 以滑鼠右鍵按一下 **src** 資料夾，然後選取**匯入**。
      這時會開啓「匯入」精靈。
   d. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
   e. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
      yourDirectory\WC_SAMPLE_55.zip
      其中 *yourDirectory* 是您下載套件的目錄。
   f. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
      - com\ibm\commerce\sample\databeans\BonusDataBean.java
   g. 在**資料夾**欄位中，已經指定 `WebSphereCommerceServerExtensionsLogic/src` 資料夾。請保留這個值。
   h. 按下**完成**。
2. 按兩下 **BonusDataBean** 類別來檢視其原始碼。
3. 在原始碼中，取消註解「區段 1」，以便將下列程式碼引入到 Bean 中：

```
/// 區段 1 ////////////////////////////////////////////////

// 建立欄位和存取元（setter/getter 方法）

 private java.lang.String userId;
 private java.lang.Integer totalBonusPoints;


 public java.lang.String getUserId() {
  return userId;
 }

 public void setUserId(java.lang.String newUserId) {
```

```
 userId = newUserId;

 ////////////////////////////////////
 /// 區段 A：將 BonusAccessbean 實例化

 if (userId != null)
  this.setInitKey_memberId(new Long(newUserId));

 ////////////////////////////////////
}


   public java.lang.Integer getTotalBonusPoints() {
 return totalBonusPoints;
}
public void setTotalBonusPoints(java.lang.Integer newTotalBonusPoints) {
 totalBonusPoints= newTotalBonusPoints;
}


//// 區段 1 結束 ////////////////////////////////////////////
```

4. 接下來，取消註解「區段 2」，以便將下列程式碼區段引入到 Bean 中：

```
/// 區段 2 ////////////////////////////////////////////////

// 建立一個新的建構子來將存取 Bean 傳送到 databean 中
// 讓 JSP 可以使用存取 Bean

public BonusDataBean(BonusAccessBean bb)
   throws com.ibm.commerce.exception.ECException {
 try {
  super.setEJBRef(bb.getEJBRef());
} catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "BonusDataBean(bb)");
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
  }
}

//// 區段 2 結束 ////////////////////////////////////////////
```

5. 接下來，取消註解「區段 3」，以便將下列程式碼引入到 Bean 中：

```
//// 區段 3 ////////////////////////////////////////////////

// 設定用來將 BonusAccessbean 實例化的其他資料欄位

 try {
```

```
  setUserId(getRequestProperties().getString("taskOutputUserId"));

  try {
   super.refreshCopyHelper();
} catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "populate");
     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
           "BonusDataBean", "populate");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
           "BonusDataBean", "populate");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           "BonusDataBean", "populate");
  }


 }
catch (ParameterNotFoundException e){}


///// 區段 3 結束 ////////////////////////////////////////
}
```

6. 接下來，取消註解「區段 4」，以便將下列程式碼引入到 Bean 中：

```
/// 區段 4 ////////////////////////////////////////

   // 將輸入 TypedProperteis 複製到本端

 requestProperties = aParam;

/// 區段 4 結束 ////////////////////////////////////
```

7. 儲存變更。
8. 用滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**重新建置專案**，來編譯已變更的程式碼。

建立新 Entity Bean 的存取控制原則

在此提供一個範例存取控制原則。此原則建立了下列的存取控制物件：

動作　所建的動作爲 com.ibm.commerce.sample.commands.MyNewControllerCmd

動作群組

所建的動作群組爲 MyNewControllerCmdActionGroup。此動作群組只含有一個動作：com.ibm.commerce.sample.commands.MyNewControllerCmd

資源種類

所建的資源種類爲 `com.ibm.commerce.sample.objects.BonusResourceCategory`。此資源種類爲 Bonus Entity Bean 所用。

資源群組

所建的資源群組爲 `BonusResourceGroup`。此資源群組只含有上述的資源種類。

原則 所建的原則爲 `AllUsersUpdateBonusResourceGroup`。只有在使用者爲 Bonus 物件的「擁有者」時，此原則才會容許使用者對 Bonus Bean 執行 MyNewControllerCmd 動作。比方說，如果使用者登入成 `tester@mycompany` 使用者，則使用者只能修改自己的紅利積點。

設置 `AllUsersUpdateBonusResourceGroup` 原則涉及下列步驟：

1. 修改存取控制原則以反映您的環境。
2. 使用 `acpload` 指令來載入 SampleACPolicy.xml 檔。
3. 使用 `acpnlsload` 指令來載入 SampleACPolicy_en_US.xml 說明。

如果要爲您的環境自訂存取控制原則，請執行下列步驟：

1. 判斷您的「流行館」商店的成員 ID 值，方式如下：
   - DB2 如果您使用 DB2 資料庫，請執行下列步驟：
     a. 連接您的開發資料庫。
     b. 發出下列 SQL 陳述式：

        ```
        select member_id from storeent where storeent_id=FF_storeent_ID
        ```

        其中 *FF_storeent_ID* 是您的「流行館」商店的商店實體 ID。例如，您可以輸入：

        ```
        select member_id from storeent where storeent_id=10001
        ```

        請記下成員 ID 值：_____
   - Oracle 如果您使用 Oracle 資料庫，請執行下列步驟：
     a. 開啓 Oracle SQL Plus 指令視窗，然後連線到您的開發資料庫。
     b. 輸入下列指令：
        `column member_id format 99999999999999999999999999999999999`（共有 35 個 9）
        這樣會設定足夠的 member_id 直欄顯示寬度，以檢視整個值。
     c. 發出下列 SQL 陳述式：

```
select member_id from storeent where storeent_id=FF_storeent_ID
```

其中 *FF_storeent_ID* 是您的「流行館」商店的商店實體 ID。例如，您可以輸入：

```
select member_id from storeent where storeent_id=10001
```

請記下成員 ID 值：_____

d. 輸入下列指令來重設 member_id 直欄的顯示寬度：

```
column member_id clear
```

2. 使用文字編輯程式，開啓 SampleACPolicy_template.xml 檔（在 *WCStudio_installdir*\Commerce\xml\policies\xml 目錄中），然後將 *FashionFlowMemberId* 取代爲您的「流行館」商店的成員 ID 值（根據步驟 1 中的判斷）。將您修改過的檔案儲存成 SampleACPolicy.xml（在同一個目錄中）。
3. 使用文字編輯程式，開啓 SampleACPolicy_template_en_US.xml 檔（在 *WCStudio_installdir*\Commerce\xml\policies\xml 目錄中），然後將 *FashionFlowMemberId* 取代爲您的「流行館」商店的成員 ID 值（根據步驟 1 中的判斷）。將您修改過的檔案儲存成 SampleACPolicy_en_US.xml（在同一個目錄中）。
4. 在指令提示下，切換至下列目錄：
   WCStudio_installdir\commerce\bin
5. 如果要載入 SampleACPolicy.xml 檔，您必須發出採用下列格式的 acpload 指令：

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   其中

   - *db_name* 爲您資料庫名稱
   - *db_user* 爲您資料庫使用者名稱
   - *db_password* 爲您資料庫密碼
   - *inputXMLFile* 爲內含原則的 XML 檔名稱。就本例而言，請輸入 SampleACPolicy.xml

   舉例來說，您可發出下列指令：

   ```
   acpload Demo_dev db2user db2user SampleACPolicy.xml
   ```

6. 如果要載入原則說明，您必須發出採用下列格式的 acpnlsload 指令：

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

   舉例來說，您可發出下列指令：

```
acpnlsload Demo_dev db2user db2user SampleACPolicy_en_US.xml
```

7. 如果測試環境的伺服器目前正在執行中，您可以使用 WebSphere Commerce「管理主控台」中的登錄重新整理選項來更新存取控制登錄，方式如下：
   a. 開啓 Web 瀏覽器，然後輸入以下的 URL：

      ```
      https://localhost/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=
      adminconsole.AdminConsoleLogon
      ```

   b. 在提示時，請使用網站管理者 ID 登入。
   c. 選取要處理**網站**，然後按一下**確定**。
   d. 從**配置**功能表中選取**登錄**。
   e. 按一下**全部更新**，稍後按一下**重新整理**來驗證更新已經完成。
   f. 登出並關閉「管理主控台」視窗。

修改 MyNewJSPTemplate.jsp 範本來併入紅利積點

如果要修改顯示頁面，請執行下列步驟：

1. 在 WebSphere Studio Application Developer 中，切換到 Web 視景。
2. 同時開啓 **MyNewJSPTemplate_All.jsp** 和 **MyNewJSPTemplate.jsp** 檔案。
3. 將 MyNewJSPTemplate_All.jsp 檔的「區段 9」複製到 MyNewJSPTemplate.jsp 檔中。這樣會將下列文字引入到 JSP 範本中：

```
<!-- 區段 9 -->

<h2><fmt:message key="BonusAdmin" bundle="${tutorial}" /> </h2>

<c:if test="${!empty taskOutputUserId}">
 <ul>
  <li>
   <b>
   <fmt:message key="PointBeforeUpdate" bundle="${tutorial}" />
   <c:out value="${oldBonusPoints}"/>
   </b>
  </li>
  <li>
   <b>
   <fmt:message key="PointAfterUpdate" bundle="${tutorial}" />
      <c:out value="${bdbInstance.bonusPoint}" />
      </b>
     </li>
 </ul>
</c:if>


<br />
<b><fmt:message key="EnterPoint" bundle="${tutorial}" /></b><p />


<form name="Bonus" action="MyNewControllerCmd">
```

```
<table>
 <tr>
  <td>
   <b>Logon ID </b>
  </td>
  <td>
   <input type="text" name="input1" value="<c:out
    value="${userName}"/>" />
  </td>
 </tr>
 <tr>
  <td>
   <b>Bonus Point</b>
  </td>
  <td>
   <input type="text" name="input2" />
  </td>
 </tr>
 <tr>
  <td colspan="2">
   <input type="submit" />
  </td>
 </tr>
     </table>
</form>

<!-- 區段 9 結束 -->
```

4. 儲存 MyNewJSPTemplate.jsp 檔。

測試整合的 Bonus Bean

由於新 Bonus Bean 受存取控制保護，且使用者只能對本身擁有的 Bean 執行 MyNewControllerCmd 動作，因此使用者必須登入。因而您將使用範例商店中的登入特性，以容許使用者登入。

如果要測試新的邏輯，請執行下列步驟：

1. 切換至「伺服器」檢視畫面。
2. 以滑鼠右鍵按一下 **WebSpherCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 您的商店的 **index.jsp** 檔，然後選取**在伺服器上執行**。這時會顯示商店首頁。
4. 執行下列步驟，以已登錄的使用者身份登入：
   a. 按一下**登錄**鏈結。
      這時會顯示「登錄」頁面。
   b. 在**電子郵件位址**欄位中，輸入您在 第 244 頁的『測試使用者名稱驗證』中建立的使用者的電子郵件位址。
   c. 在**密碼**欄位中，輸入這個使用者的密碼，然後按一下**登入**。

d. 在登入完成後，請在同一個瀏覽器中輸入下列 URL：

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
   input1=user_e-mail&input2=1000
```

其中 *user_e-mail* 是您在第 244 頁的『測試使用者名稱驗證』中建立的使用者的電子郵件位址。這時會顯示一個頁面，其中含有先前所有的輸出參數，以及一份新套表，可讓您更新使用者的紅利積點結餘。

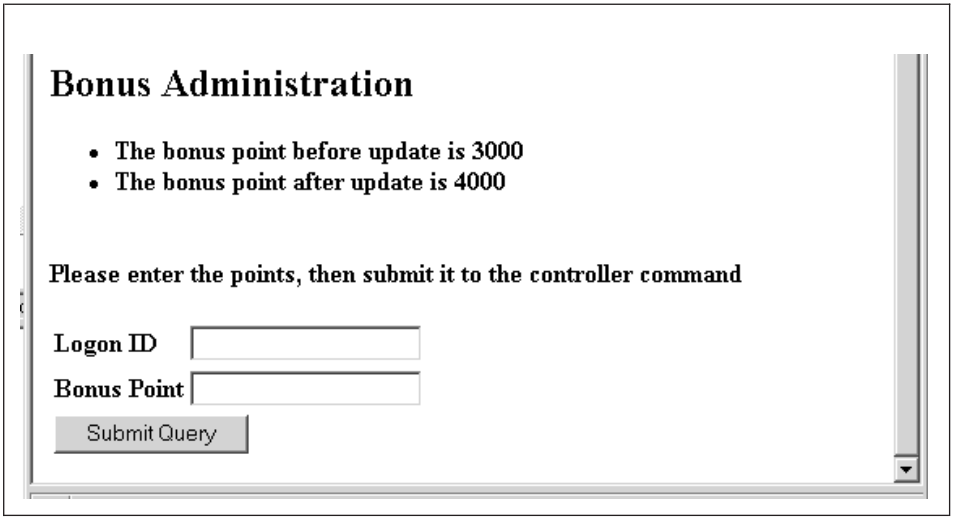

圖 41.

e. 現在，在**登入 ID** 欄位中，輸入使用者的電子郵件位址，然後在**紅利積點**欄位中，輸入 500。按一下**提交**。這時會顯示一個類似以下的頁面，其中顯示紅利積點的結餘已經更新。

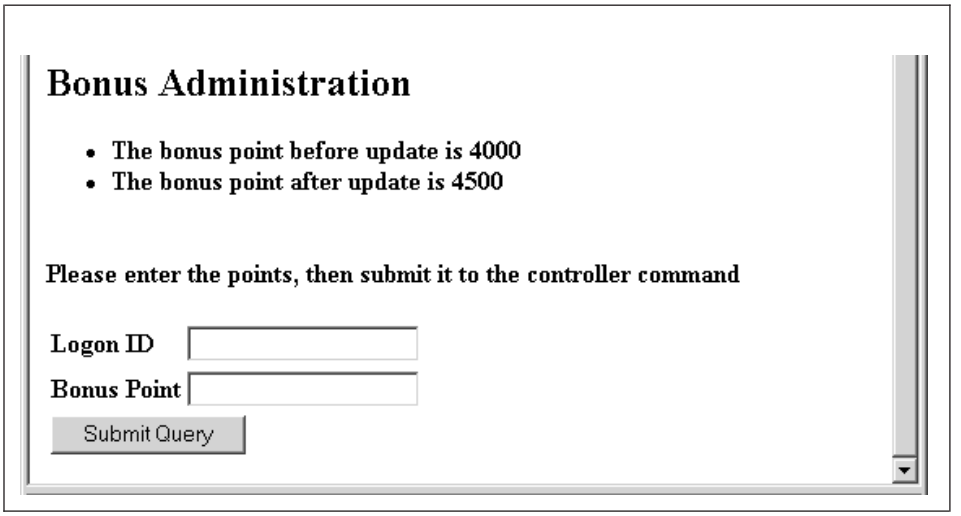

圖 42.

部署紅利積點邏輯

本節說明如何將新商業邏輯部署到執行於遠端 WebSphere Commerce Server 的商店中。在您開始進行這些部署步驟前，您必須已經在遠端 WebSphere Commerce Server 上建立一家商店（以「流行館」範例商店爲基礎）。

部署程序包括：在開發機器上執行的步驟以及在目標 WebSphere Commerce Server 上執行的步驟。

有許多不同類型的資產必須部署到目標 WebSphere Commerce Server 上。這些作業包括：

- 控制程式指令、作業指令和資料 Bean 邏輯
- Enterprise Bean 邏輯
- JSP 範本和影像檔
- 內容檔和資源連結
- 資料庫更新包括綱目更新（新表格）以及指令登錄更新
- 存取控制更新

本節說明如何將所有的資產以*遞增的*方式部署到目標 WebSphere Commerce Server 上。這是以遞增式部署的方式執行，而不是部署整個 EAR 檔。

建立指令及資料 Bean JAR 檔

本節說明如何建立包含控制程式指令、作業指令和資料 Bean 邏輯的 JAR 檔。

如果要建立這個 JAR 檔，請在開發機器上執行下列步驟：

1. 在您的本端檔案系統上建立一個叫做 *drive*:\ExportTemp 的目錄。
2. 在 WebSphere Studio Application Developer 中，切換到「J2EE 導覽器」檢視畫面。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**匯出**。
   這時會開啓「匯出」精靈。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **JAR 檔**，然後按一下**下一步**。
   b. **選取要匯出的資源**下面的左窗格中，會預先移入專案的名稱。請保留這個值。
   c. 在右窗格中，確定只有選取下列資源：
      - .classpath
      - .project
      - .serverPreference
   d. 確定**匯出產生的類別檔和資源**已經選取。
   e. *請勿*選取**匯出 Java 原始檔和資源**。
   f. 在**選取匯出目的地**欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar。
      請注意，JAR 檔名必須是 WebSphereCommerceServerExtensionsLogic.jar。
   g. 按下**完成**。

建立 EJB JAR 檔

如果要建立 EJB JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
2. 展開 **WebSphereCommerceServerExtensionsData** 專案。
3. 按兩下 **EJB 部署描述子**。
4. 在選取「概觀」標籤之後，捲動至窗格底端，尋找 **WebSphere 連結**區段。
5. 在**資料來源 JNDI 名稱**欄位中，輸入目標 WebSphere Commerce Server 的資料來源 JNDI 名稱。以下是範例值：

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`

其中目標 WebSphere Commerce Server 是使用 DB2 資料庫，而 WebSphere Commerce 實例名稱是 “demo”

Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
其中目標 WebSphere Commerce Server 是使用 Oracle 資料庫，而 WebSphere Commerce 實例名稱是 “demo”。

「資料來源 JNDI」名稱的值是藉由新增 “jdbc/” 到目標 WebSphere Commerce Server 的資料來源名稱所建立的。您可以開啓目標 WebSphere Commerce Server 上的 *instanceName*.xml 檔，並搜尋檔案中的 `DatasourceName=`，來驗證資料來源名稱。

6. 儲存您的部署描述子變更 (Ctrl + S)。
7. 在「J2EE 導覽器」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 專案，然後選取**匯出**。這時會開啓「匯出」精靈。
8. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **EJB JAR 檔**，然後按一下**下一步**。
   b. **您要匯出的資源**？的值會預先移入 EJB 專案的名稱。請保留這個值。
   c. 在**您要將資源匯出至**？欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*`:\ExportTemp\WebSphereCommerceServerExtensionsData.jar`。
   d. 按下**完成**。
9. 在建立 JAR 檔之後，請還原在步驟 5 中對本端部署描述子所作的變更，來還原您的本端測試伺服器所需要的設定。

註: 本指導教學假設您的開發資料庫與目標 WebSphere Commerce Server 所使用的資料庫是相同的類型。如果您是部署到不同的資料庫類型，就應該遵循第 188 頁的『使用轉換來建立 EJB JAR 檔』中的指示。

匯出商店資產

如果要匯出商店資產，請執行下列步驟：

1. 切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Stores** 資料夾。
3. 以滑鼠右鍵按一下 **Web Content** 資料夾，然後選取**匯出**。這時會開啓「匯出精靈」。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取**檔案系統**，然後按一下**下一步**。
   b. 按一下**取消全選**。

c. 選取要匯出下列資源：
   - Web Content*FashionFlow_name*\MyNewJSPTemplate.jsp
   - Web Content*FashionFlow_name*\images\male_blueshirt.gif
   - Web Content\WEB-INF\classes*FashionFlow_name*\Tutorial_NLS_en_US.properties
   - Web Content\WEB-INF\lib\jstl.jar
   - Web Content\WEB-INF\lib\standard.jar

d. 選取**建立檔案的目錄結構**。

e. 在「目錄」欄位中，輸入要放置這些資源的暫時目錄。例如，輸入 `C:\ExportTemp`

f. 按下**完成**。

包裝存取控制原則

在這一節中，您會將爲新資源所建立的存取控制原則複製到 *drive*:\ExportTemp 目錄中，方式如下：

1. 導覽至 *WCStudio_installdir*\Commerce\xml\policies\xml 目錄。
2. 將下面的檔案複製到 *drive*:\ExportTemp\ACPolicies 目錄中：
   - MyNewViewACPolicy.xml
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy_template.xml
   - SampleACPolicy_template_en_US.xml

將資產轉送到您的目標 WebSphere Commerce Server

在這個步驟中，您會在目標 WebSphere Commerce Server 上建立一個暫時目錄，然後將您的紅利積點資產複製到這個目錄中。在後續的步驟中，您會將不同類型的程式碼放置到 WebSphere Commerce 應用程式內的適當位置中。

如果要將檔案從您的開發機器複製到目標 WebSphere Commerce Server，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 上，建立一個叫做 *drive*:\ImportTemp 的暫時目錄。
2. 決定您要如何將檔案從某一部電腦複製到另一個電腦上。您可以將目標 WebSphere Commerce Server 上的磁碟機對映到開發機器上，或者使用 FTP 應用程式（如果已經配置）來進行複製。
3. 從開發機器上，將 *drive*:\ExportTemp 的內容複製到目標 WebSphere Commerce Server 上的 *drive*:\ImportTemp。

停止您的目標 WebSphere Commerce Server

在開始部署步驟前，您應該停止目標 WebSphere Commerce Server。有關停止 WebSphere Commerce Server 的明細，請參閱 *WebSphere Commerce Studio 安裝手冊*。

更新您的目標 WebSphere Commerce Server 上的資料庫

在更新目標資料庫前，請驗證您要部署自訂邏輯的商店的商店實體 ID。您可以使用下列 SQL 陳述式來判斷這個值：

```
select STOREENT_ID from STOREENT where IDENTITY='FashionFlow_name'
```

其中 *FashionFlow_name* 是您要部署程式碼的商店名稱。

登錄檢視畫面

DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyNewView：

1. 開啓 DB2 指令中心（**開始 > 程式集 >IBM DB2 > 指令行工具 > 指令中心**）。
2. 從**工具**功能表中選取**工具設定**。
3. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
4. 關閉工具設定。
5. 選取「Script」標籤後，在 Script 視窗中輸入下列資訊，以便在 VIEWREG 表格中建立必要的項目：

   ```
   connect to targetDB user dbuser using dbpassword;
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
   values ('MyNewView',-1, FF_storeent_ID,
      'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
      'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
      0, null);
   ```

   其中

   - *targetDB* 是目標資料庫的名稱
   - *dbuser* 是資料庫使用者
   - *dbpassword* 是資料庫使用者的密碼
   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼

   按一下**執行**圖示。使「指令中心」保持開啓。

Oracle 如果您所用的是 Oracle 資料庫，請執行下列步驟，以便將檢視畫面登錄在資料庫中：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, FF_storeent_ID,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   其中

   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

   按 Enter 鍵以執行 SQL 陳述式。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

這時 MyNewView 已經登錄完成。

登錄新的控制程式指令

如果要登錄 MyNewControllerCmd，請執行下列步驟：

1. DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyNewControllerCmd：
   a. 在「指令中心」中，選取「Script」標籤，然後在「Script」視窗中輸入下列資訊，以便在 URLREG 表格中建立必要的項目：

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

   其中

- *targetDB* 是目標資料庫的名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是資料庫使用者的密碼
- *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

按一下**執行**圖示。使「指令中心」保持開啓。

2. Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyNewControllerCmd：
   a. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
   b. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
   c. 在**密碼**欄位中，輸入您的 Oracle 密碼。
   d. 在**主機字串**欄位中，輸入您的連接字串。
   e. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FF_storeent_ID,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

   其中

   - *FF_storeent_ID* 是您根據「流行館」範例商店所建立的商店的唯一識別碼。

   按 Enter 鍵以執行 SQL 陳述式。
   f. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

建立 XBONUS 表格

在這個步驟中，您將在目標 WebSphere Commerce Server 所使用的資料庫中建立 XBONUS 表格。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 在「Script」視窗中，輸入下列指令：

```
create table XBONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

   其中

   - *targetDB* 是目標資料庫的名稱
   - *dbuser* 是資料庫使用者
   - *dbpassword* 是資料庫使用者的密碼

   按一下「執行」圖示。
   現在 XBonus 表格已建立完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
create table XBONUS (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XBOUNU 表格。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

在您的目標 WebSphere Commerce Server 上載入存取控制原則

在這個步驟中，您會將新資源的存取控制原則載入到目標 WebSphere Commerce Server 上。

如果要載入新的原則，請執行下列步驟：

1. 更新存取控制原則，以反映出您的目標 WebSphere Commerce Server 的特定值，方式如下：

a. 判斷您的「流行館」商店的成員 ID 值，方式如下：

- DB2 如果您使用 DB2 資料庫，請執行下列步驟：
  1) 連接您的開發資料庫。
  2) 發出下列 SQL 陳述式：

     ```
     select member_id from storeent where storeent_id=FF_storeent_ID
     ```

     其中 *FF_storeent_ID* 是您的「流行館」商店的商店實體 ID。例如，您可以輸入：

     ```
     select member_id from storeent where storeent_id=10001
     ```

     請記下成員 ID 值：_____

- Oracle 如果您使用 Oracle 資料庫，請執行下列步驟：
  1) 開啓 Oracle SQL Plus 指令視窗，然後連線到您的開發資料庫。
  2) 輸入下列指令：
     `column member_id format 999999999999999999999999999999999999`（共有 35 個 9）
     這樣會設定足夠的 member_id 直欄顯示寬度，以檢視整個值。
  3) 發出下列 SQL 陳述式：

     ```
     select member_id from storeent where storeent_id=FF_storeent_ID
     ```

     其中 *FF_storeent_ID* 是您的「流行館」商店的商店實體 ID。例如，您可以輸入：

     ```
     select member_id from storeent where storeent_id=10001
     ```

     請記下成員 ID 值：_____
  4) 輸入下列指令來重設 member_id 直欄的顯示寬度：

     ```
     column member_id clear
     ```

b. 導覽至下列目錄：
drive:\ImportTemp\ACPolicies

c. 使用文字編輯程式，開啓 SampleACPolicy_template.xml 檔並將 *FashionFlowMemberId* 換成您的「流行館」商店的成員 ID 值（根據步驟 1a 中的判斷）。將您修改的檔案儲存成 SampleACPolicy.xml。

d. 使用文字編輯程式，開啓 SampleACPolicy_template_en_US.xml 檔，並將 *FashionFlowMemberId* 取代爲您的「流行館」商店的成員 ID 值（根據步驟 1a 中的判斷）。將您修改的檔案儲存成 SampleACPolicy_en_US.xml。

2. 將下列存取控制原則複製到 *WC_installdir*\xml\policies\xml 目錄中：

- MyNewViewACPolicy.xml
- MyNewControllerCmdACPolicy.xml
- SampleACPolicy.xml
- SampleACPolicy_en_US.xml

3. 在指令提示下，導覽至下列目錄：
   WC_installdir\bin
4. 您必須發出 acpload 指令，其格式如下：

   ```
   acpload targetDB dbuser dbpassword inputXMLFile
   ```

   其中

   - *targetDB* 是您的開發資料庫的名稱。
   - *dbuser* 是資料庫使用者的名稱。
   - *dbpassword* 是資料庫使用者的密碼。
   - *inputXMLFile* 是包含存取控制原則規格的 XML 檔。就本例而言，請指定 `MyNewViewACPolicy.xml`。

   以下是指令的範例（已指定變數）：

   ```
   acpload Demo_Dev db2admin db2admin MyNewViewACPolicy.xml
   ```

5. 爲下列每一個存取控制原則重複步驟 4。
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy.xml
6. 如果要載入原則說明（包含在 SampleACPolicy_en_US.xml 檔中），您必須發出採用下列格式的 acpnlsload 指令：

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

   舉例來說，您可發出下列指令：

   ```
   acpnlsload Demo_dev user password SampleACPolicy_en_US.xml
   ```

7. 檢查 *WC_installdir*\xml\policies\xml 目錄，看看在載入存取控制原則時是否產生任何錯誤日誌。

更新您的目標 WebSphere Commerce Server 上的商店資產

在這個步驟中，您會使用已修改的商店資產來更新商店，方式如下：

1. 備份您的 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war 目錄（其中 *cellName* 通常視您的機器的主機名稱，而 *instanceName* 是您的 WebSphere Commerce 實例的名稱）。
2. 導覽至 *drive*:\ImportTemp\Stores\Web Content 目錄。

3. 將 *FashionFlow_name* 和 WEB-INF 資料夾複製到下列目錄中：
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\Stores.war

更新您的目標 WebSphere Commerce Server 上的指令和資料 Bean JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的指令和資料 Bean JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 WebSphereCommerceServerExtensionsLogic.jar 檔的副本，並將它儲存在備份位置中。
2. 將新的 WebSphereCommerceServerExtensionsLogic.jar 檔從 *drive*:\ImportTemp 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中。

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新您的目標 WebSphere Commerce Server 上的 EJB JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的 EJB JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 WebSphereCommerceServerExtensionsData.jar 檔的副本，並將它儲存在備份位置中。
2. 將新的 WebSphereCommerceServerExtensionsData.jar 檔從 *drive*:\ImportTemp 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中。
3. 接下來，您必須修改 EJB 部署描述子資訊，方式如下：
   a. 尋找這個 WebSphere Application Server 資料格的部署儲存庫（META-INF 目錄）。其格式通常如下：
      WAS_installdir\config\cells*cellName*\applications\WC_*instance_name*.ear\deployments\WC_*instance_name**EJBModuleName*.jar\META-INF。
      以下是這項資訊的特定範例：
      D:\WebSphere\AppServer\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\WebSphereCommerceServerExtensionsData.jar\META-INF
   b. 目錄包含下列檔案：
      - ejb-jar.xml

- ibm-ejb-access-bean.xmi
- ibm-ejb-jar-bnd.xmi
- ibm-ejb-jar-ext.xmi
- MANIFEST.MF

備份所有的檔案。

c. 使用工具來開啓新的 WebSphereCommerceServerExtensionsData.jar 檔並檢視其內容。

d. 將 meta-inf 目錄的內容（前面列出的檔案）從這個 WebSphereCommerceServerExtensionsData.jar 檔擷取到步驟 3a 中的目錄裡。在擷取檔案之後，確定目錄結構仍然正確。

4. 在指令行中，使用 WebSphere Application Server startServer 指令來重新啓動您的 WebSphere Commerce 實例。請參閱您的平台和資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於啓動和停止這個實例的其他資訊。

驗證目標 WebSphere Commerce Server 上的紅利積點邏輯

在這個步驟中，您將執行下列步驟來驗證已經順利部署到目標 WebSphere Commerce Server 的紅利積點邏輯：

1. 開啓 Web 瀏覽器，然後輸入 URL 來啓動您根據「流行館」範例商店所建立的商店。
2. 執行下列步驟來建立一個新的已登錄使用者：
   a. 按一下**登錄**。
   b. 再按一下**登錄**來建立新的客戶。
   c. 在登錄套表中，在所有的必要欄位中輸入適當的值。例如，在電子郵件欄位中，輸入 tester@mycompany。請記下電子郵件位址的值：_____。
   d. 一旦輸入這些值之後，請按一下**提交**。
3. 在登入完成後，請在同一個瀏覽器中輸入下列 URL：

```
http://hostname/webapp/wcs/stores/servlet/MyNewControllerCmd?
   input1=user_e-mail&input2=1000
```

其中 *hostname* 是主機名稱，而 *user_e-mail* 是您在步驟 2 中建立的使用者的電子郵件位址。這時會顯示一個頁面，其中含有先前所有的輸出參數，以及一份新套表，可讓您更新使用者的紅利積點結餘。

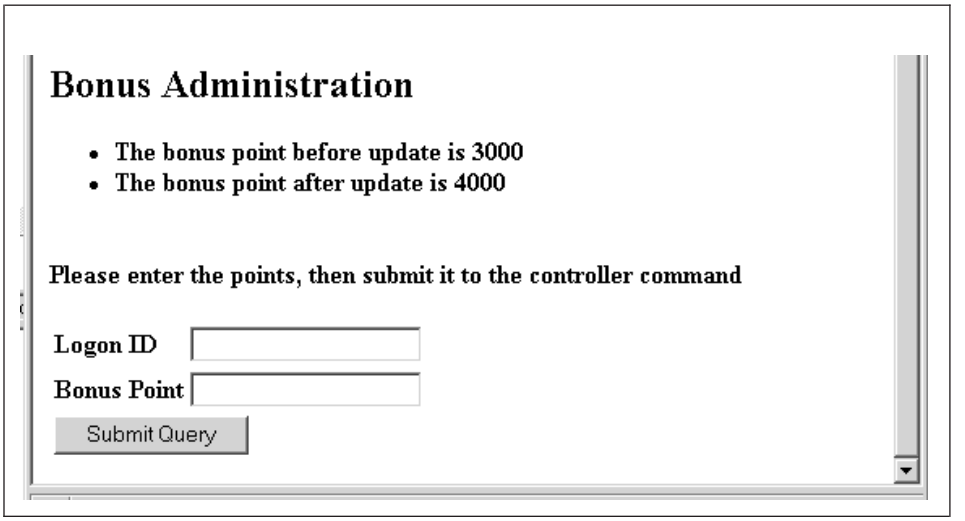

圖 43.

4. 現在，在**登入 ID** 欄位中，輸入使用者的電子郵件位址，然後在**紅利積點**欄位中，輸入 500。按一下**提交**。這時會顯示一個類似下面的頁面，其中顯示紅利積點的結餘已經更新。

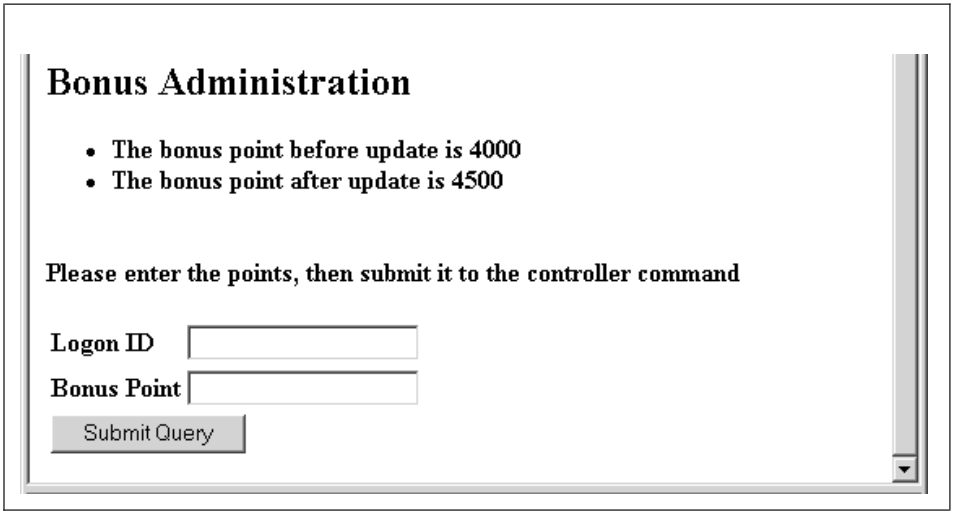

圖 44.

第 11 章 指導教學：修改現有的控制程式指令

本指導教學的目的是示範修改現有控制程式指令的程序。

在這個指導教學中，您可以將客戶購物車中的項目數目限制爲五個或以下。如果要實作這個解決方案，您必須使用自己的實作（其中包含用來檢查購物車中的項目數目的邏輯）來改寫 OrderItemAddCmdImpl。如果客戶嘗試新增第六個項目到購物車中，就會擲出異常狀況。這個異常狀況使用新的錯誤訊息。

請注意，此指導教學的目的是示範用來修改現有指令邏輯的開發程序。這個範例並不適合用來作爲限制購物車中的項目的範例。此指導教學中使用的邏輯已針對指導教學加以簡化。

在這個指導教學中，您將學習下列內容：

- 如何爲現有的控制程式指令建立新的實作
- 如何更新指令登錄，以便在應用程式中使用新的實作
- 如何將已修改的控制程式指令部署到現有的 WebSphere Commerce 應用程式中

必備需求

本指導教學並不要求您事先完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』。如果您已經完成該指導教學，您仍然可以將程式碼保留在您的工作區中，也不會與本指導教學有任何衝突。

在開始本指導教學之前，您必須已經根據「流行館」範例商店公佈了一家商店。在這家商店中，您必須能夠完成購買（例如，瀏覽型錄、新增項目至購物車、結帳以及察看訂單資訊）。

爲了完成部署步驟，商店必須也在目標 WebSphere Commerce Server 上。

建立新的 MyOrderItemAddCmdImpl 類別

在指導教學的這個步驟中，您將建立新的 MyOrderItemAddCmdImpl 類別。如果要建立這個類別，請執行下列步驟：

1. 在 WebSphere Studio Application Developer 中，開啓 Java 視景（**視窗 > 開啓視景 > Java**）。
2. 導覽至 **WebSphereCommerceServerExtensionsLogic** 專案。

3. 導覽至 **src** 目錄。
4. 如果您尚未完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，以滑鼠右鍵按一下 **src** 目錄，然後選取**新建 > 套件**。
   這時會開啓「新建 Java 套件」精靈。在這個精靈中，請執行下列步驟：
   a. 在**名稱**欄位中，輸入 com.ibm.commerce.sample.commands。
   b. 按一下**完成**。
5. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.commands** 套件。選取**新建 > 類別**。
   這時會開啓「新建 Java 類別」精靈。
6. 在「新建 Java 類別」精靈中，執行下列步驟：
   a. 在**名稱**欄位中，輸入 MyOrderItemAddCmdImpl。
   b. 如果要指定超類別，請按一下「超類別」欄位旁邊的**瀏覽**按鈕，然後輸入 OrderItemAddCmdImpl。按一下**確定**。
   c. 如果要指定要實作的介面，請按一下**新增**，然後輸入 OrderItemAddCmd，再按一下**確定**。
   d. 按一下**完成**。

這時會顯示 MyOrderItemAddCmdImpl 類別的原始碼。

下一步是更新這個類別中的重要陳述式，方式如下：

1. 在「大綱」檢視畫面中，選取並開啓**匯入宣告**。您會在此找到兩個已經建立好的重要陳述式。
2. 在 import 陳述式的原始碼中，新增下面的 import 陳述式：

```
import com.ibm.commerce.exception.ECApplicationException;
import com.ibm.commerce.exception.ECException;
import com.ibm.commerce.exception.ECSystemException;
import com.ibm.commerce.order.objects.OrderAccessBean;
import com.ibm.commerce.ras.ECMessage;import com.ibm.commerce.sample.messages.MyNewMes
```

3. 儲存變更。

下一步是新增商業邏輯和異常狀況處理，以便在客戶新增更多訂單項目前，先判斷客戶的購物車中是否有五個以上的項目。請依照下列方式來更新程式碼：

1. 在「大綱」檢視畫面中，選取 MyOrderItemAddCmdImpl 類別並檢視其原始碼。
   原始碼目前只有下列項目：

```
public class MyOrderItemAddCmdImpl
 extends OrderItemAddCmdImpl
 implements OrderItemAddCmd {

}
```

2. 您必須新增一個 performExecute 方法到這個類別中。這個方法包含用來檢查購物車中項目數目的邏輯，如果數目小於五，則會依照一般狀況，呼叫超類別的一般 performExecute 方法 (OrderItemAddCmdImpl)。如果有五個以上的項目，就會擲出異常狀況，而使用者也無法新增更多個項目到購物車中。如果要新增這個方法，請將下列原始碼複製到類別中（確定程式碼是在類別結尾的最後一個結束大括弧 "}" 前面）：

```
public void performExecute() throws ECException {
  // 取得訂單 ID 的清單
  String[] orderIds = getOrderId();

  // 檢查以確定 ID 已經存在
  // 如果訂單 ID 存在，則取得訂單中的項目數目
  // 如果訂單 ID 不存在，則執行一般程式碼
  if (orderIds != null && orderIds.length > 0) {
   // 在嘗試新增第六個項目到購物車時，應該會擲出異常狀況。
   // 由於這個程式碼是在新增任何項目之前就已經執行
   // 如果購物車中有五個或更多項目，就會擲出異常狀況
   if (itemsInOrder(orderIds[0]) >= 5) {
          throw new ECApplicationException(
     MyNewMessages._ERR_TOO_MANY_ITEMS,
    this.getClass().getName(),
     "performExecute");
   }
   // 否則就執行一般流程
  }
  super.performExecute();
 }
 // 取得訂單中的項目數目
 protected int itemsInOrder(String orderId) throws ECException {
  try {
   OrderAccessBean order = new OrderAccessBean();
   order.setInitKey_orderId(orderId);
   order.refreshCopyHelper();
   return order.getOrderItems().length;
} catch (javax.ejb.FinderException e) {
   throw new ECSystemException(
    ECMessage._ERR_FINDER_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
     } catch (javax.naming.NamingException e) {
   throw new ECSystemException(
    ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
     } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(
    ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
} catch (javax.ejb.CreateException e) {
   throw new ECSystemException(
    ECMessage._ERR_CREATE_EXCEPTION,
```

```
        this.getClass().getName(),
        "itemsInOrder");
    }
}
```

將新的程式碼貼到類別中以後，請用滑鼠右鍵按一下原始碼，然後選取**格式**來設定程式碼的格式。

3. 儲存您的工作。

   註： 這時會出現警告訊息，指出遺漏訊息資訊。後續的步驟將會更正這個問題。

建立訊息資訊

新的指令實作使用新的錯誤訊息：_ERR_TOO_MANY_ITEMS。在這一節中，您將爲新的訊息及其相關內容檔建立程式碼。如果要匯入這個程式碼，請執行下列步驟：

1. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
2. 以滑鼠右鍵按一下 **src** 目錄，然後選取**新建 > 套件**。
   這時會開啓「新建 Java 套件」精靈。在這個精靈中，請執行下列步驟：
   a. 在**名稱**欄位中，輸入 com.ibm.commerce.sample.messages。
   b. 按一下**完成**。
3. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.messages** 套件。選取**新建 > 類別**。
   這時會開啓「新建 Java 類別」精靈。
4. 在「新建 Java 類別」精靈中，執行下列步驟：
   a. 在**名稱**欄位中，輸入 MyNewMessages。
   b. 按一下**完成**。
5. 按兩下 **MyNewMessages** 類別來檢視其原始碼。
6. 在程式碼的 public class MyNewMessages 這一行行前面，新增下列 import 陳述式：

```
import com.ibm.commerce.ras.ECMessage;import com.ibm.commerce.ras.ECMessageSeverity;
import com.ibm.commerce.ras.ECMessageType;
```

7. 在這個類別中，新增下列程式碼：

```
// 用來擷取異常狀況的文字的資源連結
 static final String errorBundle = "MyNewErrorMessages";

 // ECMessage 是用來說明 ECException，而在擲出時會傳送
 // 到 ECException
```

```
public static final ECMessage _ERR_TOO_MANY_ITEMS =
     new ECMessage(ECMessageSeverity.ERROR, ECMessageType.USER,
     MyNewMessageKeys._ERR_TOO_MANY_ITEMS, errorBundle);
```

8. 儲存變更。
9. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.messages** 套件。選取**新建 > 類別**。
   這時會開啓「新建 Java 類別」精靈。
10. 在「新建 Java 類別」精靈中，執行下列步驟：
    a. 在**名稱**欄位中，輸入 MyNewMessageKeys。
    b. 按一下**完成**。
11. 將類別中的程式碼定義成下列內容：

```
public class MyNewMessageKeys {
// 這個類別會定義用來建立自訂程式碼所擲出的新異常狀況的鍵值
// 。
 public static final String _ERR_TOO_MANY_ITEMS = "_ERR_TOO_MANY_ITEMS";
}
```

12. 儲存變更。
13. 編譯您的程式碼，方法是以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。
14. 建立新的內容檔以包含訊息資訊，方式如下：
    a. 開啓「Web」視景（**視窗 > 開啓視景 > Web**）。
    b. 在 **Stores** Web 專案中，展開 **Web Content > WEB-INF > classes** 資料夾。
    c. 以滑鼠右鍵按一下 **classes** 資料夾，然後選取**新建 > 其他 > 簡易 > 檔案 > 下一步**，來建立新的內容檔。
       這時會開啓「新建檔案」視窗。
    d. 在**檔名**欄位中，輸入 MyNewErrorMessages.properties，然後按一下**完成**。
       這時會開啓新的空白檔案。
    e. 將下列文字複製到新的檔案中：

```
_ERR_TOO_MANY_ITEMS=您嘗試在一個購物車中放置
太多不同的項目。
```

       註: 在前面的程式碼中，換行僅是爲了方便顯示。請將您的文字輸入到同一行中。

    f. 儲存變更。

修改指令登錄

在這個步驟中，您會修改指令登錄，以使用新的 MyOrderItemAddCmdImpl 實作類別來替代原始的 OrderItemAddCmdImpl 實作類別。在指令登錄中，唯一需要修改的表格是 CMDREG 表格。在此情況下，所有的商店都會使用新的實作類別。

如果要修改指令登錄，請執行下列步驟：

1. DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：
   a. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
   b. 從**工具**功能表中選取**工具設定**。
   c. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
   d. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to developmentDB user dbuser using dbpassword;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME=
'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

   其中

   - *developmentDB* 是您的開發資料庫名稱
   - *dbuser* 是資料庫使用者
   - *dbpassword* 是您的資料庫使用者的密碼

   按一下**執行**圖示。

2. Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：
   a. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
   b. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
   c. 在**密碼**欄位中，輸入您的 Oracle 密碼。
   d. 在**主機字串**欄位中，輸入您的連接字串。
   e. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME=
'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

按 Enter 鍵以執行 SQL 陳述式。

f. 輸入下列以確定您的資料庫變更：

```
commit;
```

並按 Enter 鍵，以執行 SQL 陳述式。

測試 MyOrderItemAddCmdImpl 指令

下一步是進行測試，以確定新的邏輯可以正常運作。如果要進行這個測試，您必須能夠順利新增五個項目到購物車中，並且在嘗試新增第六個項目到購物車時會擲出錯誤。

如果要測試新的商業邏輯，請執行下列步驟：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下的 **index.jsp**，然後選取**在伺服器上執行**。
   這時會在 Web 瀏覽器中顯示商店首頁。
4. 在商店中購物，並新增五個項目到購物車中。在新增第五個項目之後，您的購物車會與下面的購物車類似：

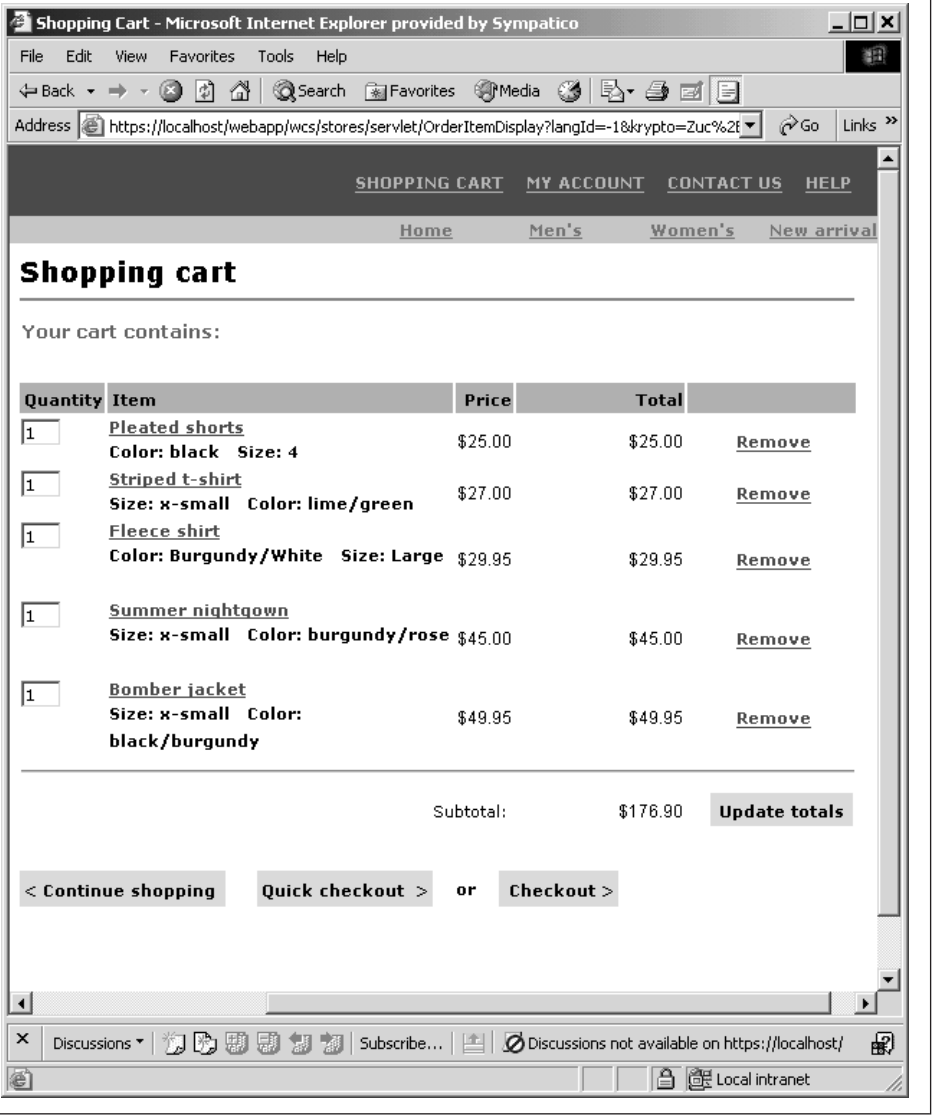

圖 45.

5. 按一下**繼續購物**並選取另一個項目。針對這個選取的項目，按一下**新增至購物車**。您會看到下面的錯誤頁面：

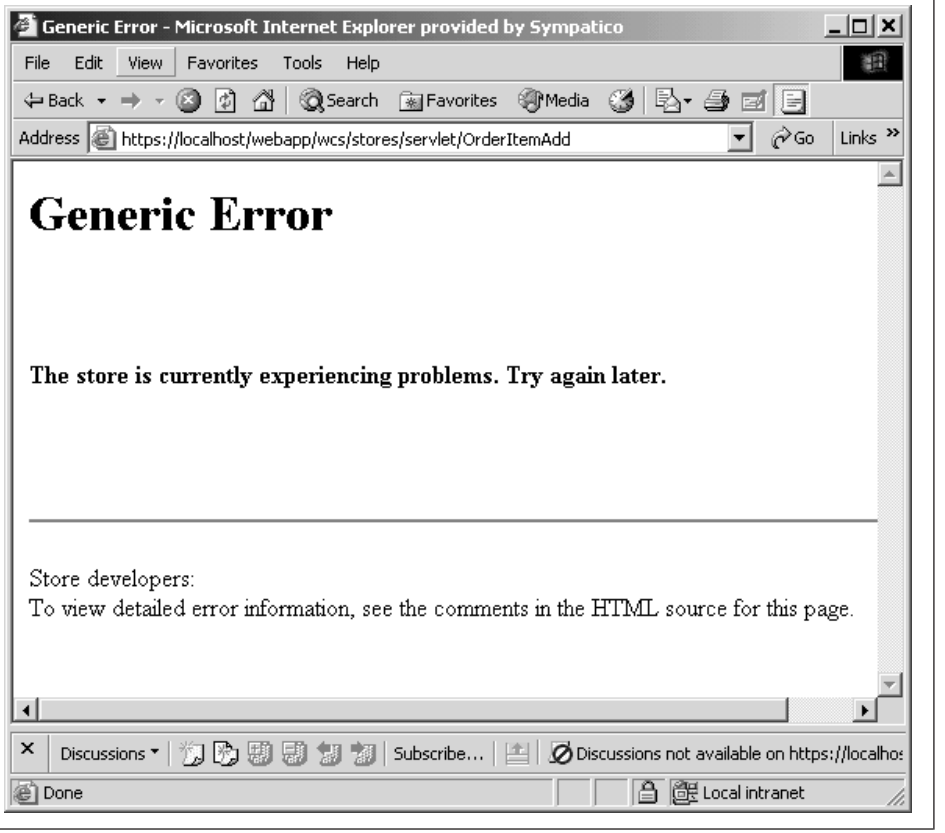

圖 46.

檢視錯誤頁面的原始碼。在原始碼中，向下捲動以尋找下列錯誤資訊：

```
訊息鍵值： _ERR_TOO_MANY_ITEMS
訊息：您嘗試在一個購物車中放置太多不同的
項目
```

部署 MyOrderItemAddCmdImpl

在這個步驟中，您會將已修改的商業邏輯部署到目標 WebSphere Commerce Server。在此情況下，部署是由下面的高階步驟所組成：

1. 建立包含指令邏輯和錯誤類別的 JAR 檔。
2. 匯出錯誤訊息內容檔。
3. 將資產轉送到目標 WebSphere Commerce Server。
4. 更新目標 WebSphere Commerce Server 上的指令登錄。

5. 驗證目標 WebSphere Commerce Server 上的新邏輯。

建立指令 JAR 檔

根據 WebSphere Commerce 程式碼自訂策略，自訂指令和資料 Bean 都是放置在 WebSphereCommerceServerExtensionsLogic 專案中。因此，在部署自訂程式碼時，您會注意到您已經將先前自訂的程式碼併入到 JAR 檔中。比方說，如果您已經完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，就會發現在您建立 JAR 檔來部署 MyOrderItemAddCmdImpl 類別時，檔案中會包含先前建立的 MyNewControllerCmd 類別和其他類別。

如果要建立包含 MyOrderItemAddCmdImpl 類別的 JAR 檔，請在開發機器上執行下列步驟：

1. 在您的本端檔案系統上建立一個 *drive*:\ExportTemp2 目錄。
2. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**匯出**。
   這時會開啓「匯出」精靈。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **JAR 檔**，然後按一下**下一步**。
   b. **選取要匯出的資源**下面的左窗格中，會預先移入專案的名稱。請保留這個值。
   c. 在右窗格中，確定只有選取下列資源：
      - .classpath
      - .project
      - .serverPreference
   d. 確定**匯出產生的類別檔和資源**已經選取。
   e. *請勿*選取**匯出 Java 原始檔和資源**。
   f. 在**選取匯出目的地**欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp2\WebSphereCommerceServerExtensionsLogic.jar。
      請注意，JAR 檔名必須是 WebSphereCommerceServerExtensionsLogic.jar。
   g. 按一下**完成**。

匯出訊息內容檔

在這一節中，您將匯出包含新訊息的文字的內容檔，方式如下：

1. 切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Stores** 資料夾。
3. 以滑鼠右鍵按一下 **Web Content** 資料夾，然後選取**匯出**。這時會開啓「匯出精靈」。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取**檔案系統**，然後按一下**下一步**。
   b. 按一下**取消全選**。
   c. 選取要匯出下列資源：
      - Web Content\WEB-INF\classes\MyNewErrorMessages.properties
   d. 選取**建立檔案的目錄結構**。
   e. 在「目錄」欄位中，輸入要放置這些資源的暫時目錄。例如，輸入 `C:\ExportTemp2`
   f. 按一下**完成**。

將資產轉送到目標 WebSphere Commerce Server

在這個步驟中，您會在目標 WebSphere Commerce Server 上建立一個暫時目錄，然後將您的 MyOrderItemAddCmdImpl 資產複製到這個目錄中。

如果要將檔案從您的開發機器複製到目標 WebSphere Commerce Server，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 上，建立一個叫做 *drive*:\ImportTemp2 的暫時目錄。
2. 決定您要如何將檔案從某一部電腦複製到另一個電腦上。您可以將目標 WebSphere Commerce Server 上的磁碟機對映到開發機器上，或者使用 FTP 應用程式（如果已經配置）來進行複製。
3. 從開發機器上，將 *drive*:\ExportTemp2 的內容複製到目標 WebSphere Commerce Server 上的 *drive*:\ImportTemp2。

停止您的目標 WebSphere Commerce Server

在開始部署步驟前，您應該停止目標 WebSphere Commerce Server。有關停止目標 WebSphere Commerce Server 的明細，請參閱*WebSphere Commerce Studio 安裝手冊*。

更新目標 WebSphere Commerce Server 上的資料庫

在這個步驟中，您會修改指令登錄，以使用新的 MyOrderItemAddCmdImpl 實作類別。

DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
2. 從**工具**功能表中選取**工具設定**。
3. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
4. 關閉工具設定。
5. 選取「Script」標籤後，在 Script 視窗中輸入下列資訊，以便在 CMDREG 表格中建立必要的項目：

   ```
   connect to targetDB user dbuser using dbpassword;
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
   WHERE INTERFACENAME=
   'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
   and storeent_Id=0;
   ```

   其中

   - *targetDB* 是您的目標 WebSphere Commerce Server 所使用的資料庫的名稱
   - *dbuser* 是資料庫使用者
   - *dbpassword* 是資料庫密碼

   按一下**執行**圖示。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

   ```
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
   WHERE INTERFACENAME=
   'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
   and storeent_Id=0;
   ```

按 Enter 鍵以執行 SQL 陳述式。

6. 輸入下列以確定您的資料庫變更：

   ```
   commit;
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。

更新目標 WebSphere Commerce Server 上的指令 JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用包含新的 MyOrderItemAddCmdImpl 的 JAR 檔，方式如下：

1. 在指令行中，使用 WebSphere Application Server stopServer 指令來停止您的 WebSphere Commerce 實例。必要時，請參閱您的平台與資料庫的 *WebSphere Commerce 安裝手冊*，以取得關於啓動和停止這個實例的資訊。
2. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄，其中 *cellName* 通常是機器的主機名稱。
   b. 製作 WebSphereCommerceServerExtensionsLogic.jar 檔的副本，並將它儲存在備份位置中。
3. 將新的 WebSphereCommerceServerExtensionsLogic.jar 檔從 *drive*:\ImportTemp2 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中。

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新目標 WebSphere Commerce Server 上的訊息內容

在這個步驟中，您會將新的訊息內容檔加入到應用程式中，方式如下：

1. 備份您的 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\Stores.war 目錄（其中 *cellName* 通常是您的機器的主機名稱，而 *instanceName* 是您的 WebSphere Commerce 實例的名稱）。
2. 導覽至 *drive*:\ImportTemp\Stores\Web Content 目錄。
3. 將 WEB-INF 資料夾複製到下列目錄中：
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\Stores.war
4. 在指令行中，使用 WebSphere Application Server startServer 指令來重新啓動您的 WebSphere Commerce 實例。

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

驗證目標 WebSphere Commerce Server 上的 MyOrderItemAddCmdImpl 邏輯

在這個步驟中，您會執行一項快速檢查，來驗證程式碼在部署之後的運作正常。

如果要驗證程式碼，請執行下列步驟：

1. 開啓一個 Web 瀏覽器，然後啓動您的「流行館」商店。例如，輸入下列 URL 來啓動商店：

   ```
   http://hostname/webapp/wcs/stores/servlet/FashionFlow/index.jsp
   ```

   其中 *hostname* 是您的實例的主機名稱。
2. 在商店中購物，並新增五個項目到購物車中。在新增第五個項目之後，您的購物車會與下面的購物車類似：

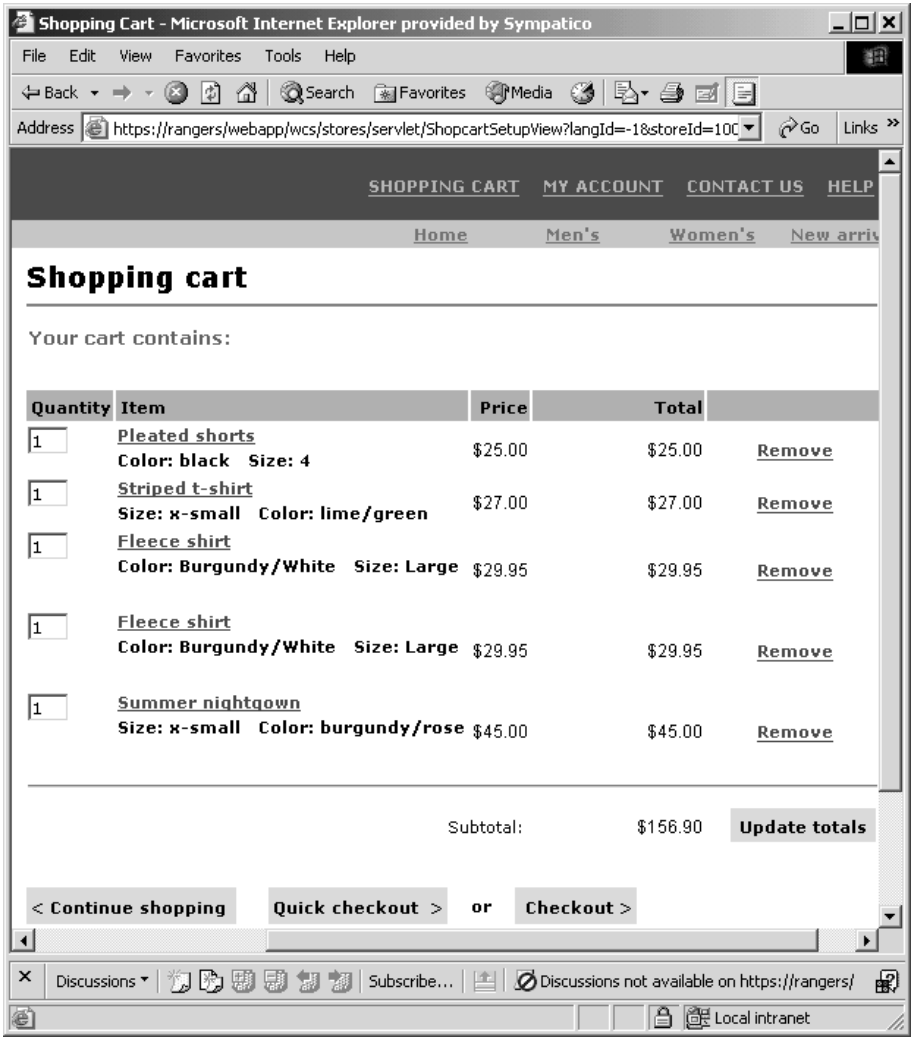

圖 47.

3. 按一下**繼續購物**並選取另一個項目。針對這個選取的項目，按一下**新增至購物車**。您會看到下面的錯誤頁面：

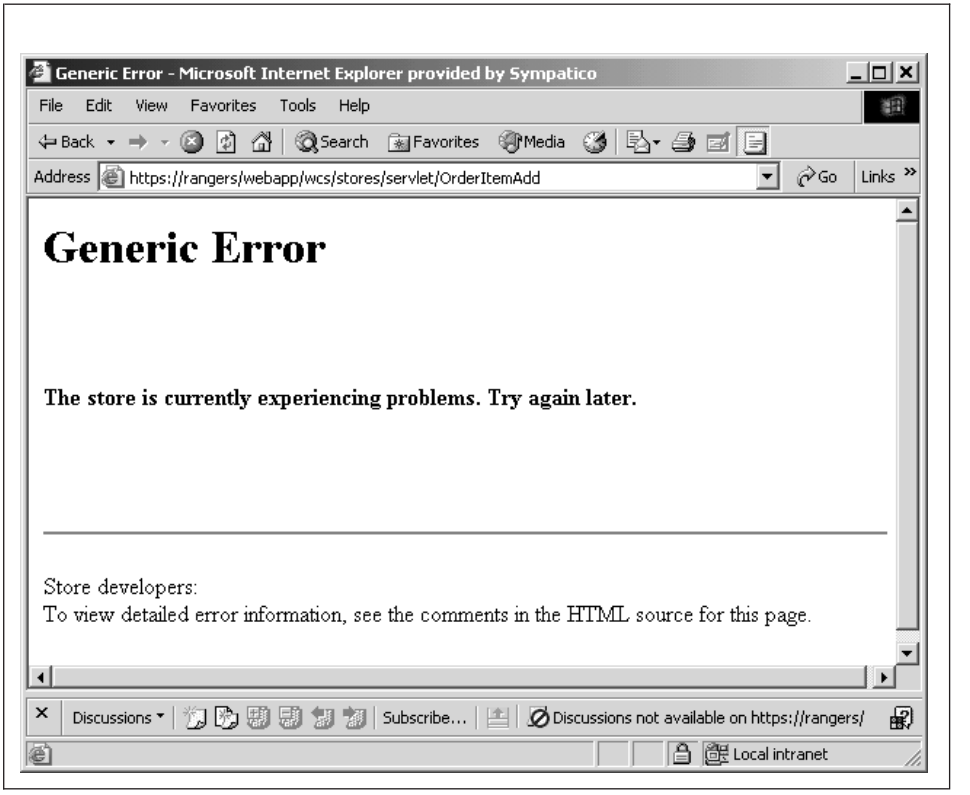

圖 48.

檢視錯誤頁面的原始碼。向下捲動原始碼，就會發現錯誤的訊息鍵值是 _ERR_TOO_MANY_ITEMS。

第 12 章 指導教學：延伸物件模型以及修改現有的作業指令

在本指導教學中，您必須滿足一個需求：爲訂單收集禮品卡資訊。您必須收集的資訊包括收件人的名稱、寄件人的名稱以及兩則訊息。資訊在客戶提交訂單時收集。

由於贈品卡的資訊必須儲存在資料庫中，因此需要一個新的資料庫表格。由於這是訂單程序的延伸，有兩種可能的方法可以修改物件模型。通常，您可以修改現有的 WebSphere Commerce 公用 Entity Bean，並且將已修改的 Bean 中的新欄位對映到新的表格，或者您可以建立一個新的 Entity Bean 來直接對映到新表格。由於這個方法會要求您延伸 Order Entity Bean，而這個 Bean 使用「尋找以更新」這類的 SQL 查詢，所以您無法以這種方式來延伸 Bean。因此，在此情況下，您必須建立新的 Entity Bean 來對映到新表格中。

除了新的 Entity Bean 之外，現有的 ExtOrderProcessCmdImpl 作業指令也會被延伸。延伸是用來將對應至新表格的新資料 Bean 實例化，而該資料 Bean 是用來更新資料庫中的禮品資訊。

本指導教學包含下列高階作業：

1. 建立新的 XORDGIFT 表格並移入資料
2. 建立新的 OrderGift Entity Bean
3. 建立 XORDGIFT 綱目
4. 建立表格定義並將 OrderGift Entity Bean 中的欄位對映到 XORDGIFT 表格中的直欄
5. 爲 OrderGift Bean 產生已部署的程式碼和存取 Bean。
6. 建立新的 OrderGiftDataBean
7. 建立新的 MyExtOrderProcessCmdImpl 作業指令實作
8. 修改 OrderSubmitForm.jsp 來收集訊息資訊以及修改 OrderDetailDisplayForm.jsp 來顯示訊息資訊。
9. 測試修改的程式碼

必備需求

本指導教學並不要求您事先完成指導教學。如果您已經完成這些指導教學，您仍然可以將程式碼保留在您的工作區中，也不會與本指導教學有任何衝突。

在開始本指導教學之前，您必須已經根據「流行館」範例商店公佈了一家商店。在這家商店中，您必須能夠完成購買（例如，瀏覽型錄、新增項目至購物車、結帳以及察看訂單資訊）。

建立 XORDGIFT 表格並移入資料

在這個步驟中，您將建立 XORDGIFT 表格。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令行工具 > 指令中心**），然後按一下 **Script** 標籤。
2. 在 Script 視窗中，輸入下列指令：

```
connect to developmentDB user dbuser using dbpassword;
create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼

按一下「執行」圖示。
現在 XORDGIFT 表格已建立完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在「SQL Plus」視窗中，輸入下列 SQL 陳述式：

```
create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XORDGIFT 表格。

註: 若先前已經有人使用這個資料庫來執行本範例，您必須在建立 XORDGIFT 表格前先發出下列指令：

```
drop table XORDGIFT;
```

6. 輸入下列以確定您的資料庫變更：

```
commit;
```

並按 Enter 鍵，以執行 SQL 陳述式。

建立 OrderGift Entity Bean

一旦建立資料庫表格後，就可以準備開始建立新的 Entity Bean。接下來的步驟將使用 WebSphere Studio Application Developer 來建立這個 Bean。

接下來，您可以執行下列步驟來建立新的 OrderGift Bean：

1. 啓動 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**）。
2. 開啓 J2EE 視景。
3. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
4. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 模組並選取**新建 > Enterprise Bean**。
   這時會開啓「Enterprise Bean 建立」精靈。
5. 從 **EJB 專案**下拉清單中，選取 **WebSphereCommerceServerExtensionsData**，然後按一下**下一步**。
6. 在「建立 Enterprise Bean」視窗中，執行下列步驟：
   a. 選取**具有儲存器所管理的持續 (CMP) 欄位的 Entity Bean**
   b. 在 **Bean 名稱**欄位中，輸入 `OrderGift`。
   c. 在**來源資料夾**欄位中，保留已指定的預設值 (ejbModule)。
   d. 在**預設套件**欄位中，輸入 `com.ibm.commerce.extension.objects`。
   e. 按一下**下一步**。
7. 在「CMP 屬性」視窗中，執行下列步驟：
   a. 按一下**新增**來將下列直欄的新欄位加到表格中：
      - ORDERSID
      - RECEIPTNAME
      - SENDERNAME

- MSGFIELD1
- MSGFIELD2

這時會開啓「建立 CMP 屬性」視窗。請在此視窗中執行下列步驟：

1) 建立 ordersId 欄位，方式如下：

表 12.

參數名稱	參數值
名稱	ordersId
類型	java.lang.Long **註:** 您必須使用 *java.lang.Long* 資料類型，而不是使用 *long* 資料類型。
關鍵字欄位	Select

按一下**套用**。

2) 建立 receiptName 欄位，方式如下：

表 13.

參數名稱	參數值
名稱	receiptName
類型	java.lang.String
使用 getter 與 setter 方法存取	Select
將 getter 與 setter 方法引介到遠端介面中	Clear

按一下**套用**。

3) 建立 senderName 欄位，方式如下：

表 14.

參數名稱	參數值
名稱	senderName
類型	java.lang.String
使用 getter 與 setter 方法存取	Select
將 getter 與 setter 方法引介到遠端介面中	Clear

按一下**套用**。

4) 建立 msgField1 欄位，方式如下：

表 15.

參數名稱	參數值
名稱	msgField1

表 15. (繼續)

參數名稱	參數值
類型	java.lang.String
使用 getter 與 setter 方法存取	Select
將 getter 與 setter 方法引介到遠端介面中	Clear

按一下**套用**。

5) 建立 msgField2 欄位，方式如下：

表 16.

參數名稱	參數值
名稱	msgField2
類型	java.lang.String
使用 getter 與 setter 方法存取	Select
將 getter 與 setter 方法引介到遠端介面中	Clear

按一下**套用**。

6) 按一下**關閉**來關閉視窗。

b. 清除**在鍵值類別中使用單一鍵值屬性類型**勾選框，然後按一下**下一步**。

8. 在「EJB Java 類別明細」視窗中，執行下列步驟：
   a. 如果要選取 Bean 的超類別，請按一下**瀏覽**。
   這時會開啓「類型選項」視窗。
   b. 在**選取類別的方法：（任何）**欄位中，輸入 ECEntityBean，然後按一下**確定**。這樣就會選取 com.ibm.commerce.base.objects.ECEntityBean 作爲超類別。
   c. 按下**完成**。

執行下列步驟來設定新 Bean 的隔離層次：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 按兩下 **WebSphereCommerceServerExtensionsData** 專案，以使用「部署描述子編輯程式」來開啓它。
3. 按一下**存取**標籤。
4. 按一下「隔離層次」文字框旁邊的**新增**。
   這時會開啓「新增隔離層次」視窗。
5. DB2 選取**可重複讀取**，然後按一下**下一步**。
   Oracle 選取**讀取已確定**，然後按一下**下一步**
6. 選取 **OrderGift** Bean，然後按一下**下一步**。

7. 選取 **OrderGift** 來選取其所有的方法，然後按一下**完成**。
8. 儲存您的工作 (Ctrl+S)。

接下來您需要設定 Bean 的安全身份，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，選取「存取」標籤。
2. 按一下「安全身份」（「方法」層次）文字框旁邊的**新增**。
   這時會開啓「新增安全身份」視窗。
3. 選取**使用 EJB 伺服器的身份**，然後按一下**下一步**。
4. 選取 **OrderGift** Bean，然後按一下**下一步**。
5. 選取 **OrderGift** 來選取其所有的方法，然後按一下**完成**。
6. 儲存您的工作 (Ctrl + S)，並保持編輯程式開啓。

接下來，您需要設定 Bean 中的方法的安全職務，方法是執行下列步驟：

1. 在「部署描述子」編輯程式中，選取「組譯描述子」標籤。
2. 在「方法許可權」區段中，按一下**新增**。
3. 選取 **WCSecurityRole** 作爲安全職務，然後按一下**下一步**。
4. 從找到的 Bean 清單中，選取 **OrderGift**，然後按一下**下一步**。
5. 在「方法元素」頁面中，按一下**全部套用**，然後按一下**完成**。
6. 儲存您的工作 (Ctrl+S)，然後關閉「部署描述子」編輯程式。

下一步是移除 WebSphere Studio Application Developer 產生的某些與實體環境定義相關的欄位和方法。這些欄位需要刪除是因爲 ECEntityBean 基本類別已提供自己對這些方法的實作。如果要刪除產生的實體環境定義欄位與方法，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **WebSphereCommerceServerExtensionsData** 專案。
2. 展開 **OrderGift** EJB 模組，然後按兩下 **OrderGiftBean**。
3. 在「大綱」檢視畫面中，執行下列步驟：
   a. 以滑鼠右鍵按一下 **myEntityCtx** 欄位並選取**刪除**。
   b. 以滑鼠右鍵按一下 **getEntityContext()** 方法，並選取**刪除**。
   c. 以滑鼠右鍵按一下 **setEntityContext(EntityContext)** 方法，並選取**刪除**。
   d. 以滑鼠右鍵按一下 **unsetEntityContext()** 方法並選取**刪除**。
4. 儲存您的工作 (Ctrl+S)。

您應該修改產生的 ebjCreate 方法，以便在建立新的 OrderGift Bean 時可以明確設定所有的參數，方式如下：

1. 在「J2EE 階層」檢視畫面中，按兩下 **OrderGiftBean** 類別來開啓它並檢視其原始碼。
2. 在「大綱」檢視畫面中，選取 ejbCreate(Long) 方法。其原始碼如下：

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
return null;
}
```

3. 修改程式碼，使它變成下面的樣子：

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
      this.senderName = null;
      this.receiptName = null;
      this.msgField1 = null;
      this.msgField2 = null;
      return null;
}
```

4. 儲存程式碼變更。

接下來，執行下列步驟來將新的 ejbCreate 方法加入到 OrderGift Bean 中：

1. 在「J2EE 階層」檢視畫面中，按兩下 **OrderGiftBean** 類別來開啓它並檢視其原始碼。
2. 將下列程式碼新增到類別中，以建立一個新的 ejbCreate(Long, String, String, String, String) 方法：

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId,
             java.lang.String receiptName,
             java.lang.String senderName,
             java.lang.String msgField1,
             java.lang.String msgField2)
   throws javax.ejb.CreateException {
   _initLinks();
      this.ordersId = ordersId;
      this.senderName = senderName;
      this.receiptName = receiptName;
      this.msgField1 = msgField1;
      this.msgField2 = msgField2;
return null;
 }
```

3. 儲存程式碼變更。

4. 您必須將這個新的 ejbCreate 方法加入到發源介面。這樣就能在產生的存取 Bean 中提供該方法。如果要新增方法到發源介面中，請執行下列步驟：
   a. 在「大綱」檢視畫面中，以滑鼠右鍵按一下 **ejbCreate(Long String, String, String, String)** 方法，然後選取 **Enterprise Bean > 提升至發源介面**。

接下來，執行下列步驟來建立一個新的 ejbPostCreate(Long, String, String, String, String) 方法，使它具備與 ejbCreate(Long, String, String, String, String) 方法相同的參數：

1. 按兩下 **OrderGiftBean** 類別來開啓它，並檢視其原始碼。
2. 新增下列程式碼到類別中，以建立一個新的 ejbPostCreate(Long ordersId, String receiptName, String senderName, String msgField1, String msgField2) 方法：

```
public void ejbPostCreate(
   java.lang.Long ordersId,
   java.lang.String receiptName,
   java.lang.String senderName,
   java.lang.String msgField1,
   java.lang.String msgField2)
   throws javax.ejb.CreateException {
   }
```

3. 儲存程式碼變更。

接下來，您必須建立 XORDGIFT 表格的定義。如果您*尚未*完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，請執行下列步驟來建立 XORDGIFT 表格定義：

1. 開啓「資料」視景並切換到「資料定義」檢視畫面。
2. 導覽至下列目錄：
   WebSphereCommerceServerExtensionsData > ejbModule > META-INF
3. 以滑鼠右鍵按一下 **META-INF**，然後選取**新建資料庫定義**。
   這時會開啓「新建資料庫定義」精靈。
4. 在**資料庫名稱**欄位中，輸入您的開發資料庫的名稱。例如，輸入 `Demo_Dev`。
5. 從「資料庫供應商類型」下拉清單中，選取您的開發環境所適用的資料庫類型：

   DB2 **DB2 Universal Database V8.1**

   Oracle **Oracle 9i**

6. 按一下**完成**。現在已經建立好新的資料庫定義。
7. 導覽至下列目錄：
   WebSphereCommerceServerExtensionsData > ejbModule > META-INF > Schema > *developmentDB*。

8. 以滑鼠右鍵按一下 *developmentDB*，然後選取**新建 > 新建綱目定義**。
這時會開啓「新建綱目定義」精靈。
9. 在**綱目名稱**欄位中，輸入 NULLID，然後按一下**完成**。
10. 導覽至 **WebSphereCommerceServerExtensionsData > ejbModule > META-INF > 綱目 > *developmentDB* > NULLID > 表格**。
11. 以滑鼠右鍵按一下**表格**，然後選取**新建表格定義**。
這時會開啓「新建表格定義」精靈。
12. 在**表格名稱**欄位中，輸入 XORDGIFT，然後按一下**下一步**。
13. 接下來，您必須新增鍵值直欄到表格定義中，方式如下：
    a. 按一下**新增另一個**。
    b. 在**直欄名稱**欄位中，輸入 ORDERSID。
    c. 從**直欄類型**下拉清單中，選取下列項目：
       DB2 BIGINT
       Oracle NUMBER
    d. 選取**鍵值直欄**。
    e. Oracle 在**數字精準度**欄位中，輸入 38。
    f. Oracle 將**數字表**的值保持爲 0。
14. 接下來，新增另一個直欄到表格定義中，方式如下：
    a. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 17.

內容	值
直欄名稱	RECEIPTNAME
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

    b. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 18.

內容	值
直欄名稱	SENDERNAME

表 18. (繼續)

內容	值
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

c. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 19.

內容	值
直欄名稱	MSGFIELD1
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

d. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 20.

內容	值
直欄名稱	MSGFIELD2
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

e. 按下**完成**。

f. Oracle 您必須使用文字編輯程式來編輯表格定義，方式如下：

1) 切換到「J2EE 導覽器」檢視畫面。

2) 展開 **WebSphereCommerceServerExtensionsData** 專案。

3) 展開下列項目：**ejbModule > META-INF >Schema**。

4) 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT.xmi** 檔，然後選取**開啓方式 > 文字編輯程式**。

5) 將所有出現的 SQLNumeric_6 取代爲 SQLNumeric_3。

6) 儲存您的變更並關閉文字編輯程式。

如果您已經完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，請執行下列步驟來建立 XORDGIFT 表格定義：

1. 開啓「資料」視景並切換到「資料定義」檢視畫面。
2. 導覽至下列目錄：
   WebSphereCommerceServerExtensionsData > ejbModule > META-INF > Schema > *developmentDB* **> NULLID > Tables**
3. 以滑鼠右鍵按一下 **Tables** 目錄，然後選取**新建 > 新建表格定義**。
   這時會開啓「新建表格定義」精靈。
4. 在**表格名稱**欄位中，輸入 XORDGIFT，然後按一下**下一步**。
5. 接下來，您必須新增鍵值直欄到表格定義中，方式如下：
   a. 按一下**新增另一個**。
   b. 在**直欄名稱**欄位中，輸入 ORDERSID。
   c. 從**直欄類型**下拉清單中，選取下列項目：
      DB2 BIGINT
      Oracle NUMBER
   d. 選取**鍵值直欄**。
   e. Oracle 在**數字精準度**欄位中，輸入 38。
   f. Oracle 將**數字表**的值保持爲 0。
6. 接下來，新增另一個直欄到表格定義中，方式如下：
   a. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 21.

內容	值
直欄名稱	RECEIPTNAME
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select

表 21. (繼續)

內容	值
字串長度	50
若為位元資料	Clear

b. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 22.

內容	值
直欄名稱	SENDERNAME
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

c. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 23.

內容	值
直欄名稱	MSGFIELD1
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select
字串長度	50
若為位元資料	Clear

d. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 24.

內容	值
直欄名稱	MSGFIELD2
直欄類型	DB2 VARCHAR Oracle VARCHAR2
可為空值	Select

表 24. (繼續)

內容	值
字串長度	50
若為位元資料	Clear

e. 按下**完成**。

7. Oracle 您必須使用文字編輯程式來編輯表格定義，方式如下：
   a. 切換到「J2EE 導覽器」檢視畫面。
   b. 展開 **WebSphereCommerceServerExtensionsData** 專案。
   c. 展開下列項目：**ejbModule > META-INF >Schema**。
   d. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT.xmi** 檔，然後選取**開啓方式 > 文字編輯程式**。
   e. 將所有出現的 SQLNumeric_6 取代爲 SQLNumeric_3。
   f. 儲存您的變更並關閉文字編輯程式。

下一步是將 XORDGIFT 表格對映到 OrderGiftBean Entity Bean。由於您的開發資料庫和 XORDGIFT 表格已經存在，因此使用「上下同時進行」對映。

如果您已經完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，請執行下列步驟來建立對映：

1. 在「J2EE 階層」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 專案，然後選取**產生 > EJB 至 RDB 對映**。
   這時會開啓「建立新的 EJB/RDB 對映」精靈。
2. 選取**上下同時進行**，然後按一下**下一步**。
3. 選取**比對名稱及類型**，然後按一下**完成**。
   這時會開啓「Map.mapxmi」編輯程式。
4. 在 Enterprise Bean 窗格中，展開 **OrderGift** Bean。在「表格」窗格中，展開 **XORDGIFT** 表格。
5. 將 Bonus Bean 中的欄位對映至 XORDGIFT 表格中的直欄，步驟如下：
   a. 以滑鼠右鍵按一下 **OrderGift** Bean 並選取**比對名稱**。
6. 按下 Ctrl + S 來儲存 Map.mapxmi 檔。關閉檔案。

如果您已經完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，請執行下列步驟來建立對映：

1. 在「J2EE 階層」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 專案，然後選取**產生 > EJB 至 RDB 對映**。
   這時會開啓「Map.mapxmi」編輯程式。
2. 在 Enterprise Bean 窗格中，展開 **OrderGift** Bean。在「表格」窗格中，展開 **XORDGIFT** 表格。
3. 將 Bonus Bean 中的欄位對映至 XORDGIFT 表格中的直欄，步驟如下：
   a. 以滑鼠右鍵按一下 **OrderGift** Bean 並選取**比對名稱**。
4. 按下 Ctrl + S 來儲存 Map.mapxmi 檔。關閉檔案。

一旦建立 OrderGiftBean 實體，且正確對映綱目以後，您必須爲 Entity Bean 建立存取 Bean。此存取 Bean 可方便應用程式存取 OrderGift Entity Bean 中所含的資訊。 WebSphere Studio Application Developer 工具可讓您根據您所建立的實體，來產生此存取 Bean （尤其是存取 Bean 只能使用已引介到遠端介面中的方法）。如果要爲您的 OrderGift Entity Bean 建立存取 Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，然後選取**新建 > 存取 Bean**。
   這時會開啓「新增存取 Bean」視窗。
2. 選取**複製輔助程式**，然後按一下**下一步**。
3. 選取 **OrderGift** Bean，然後按一下**下一步**。
4. 從**建構子方法**下拉清單中，選取 **findByPrimaryKey(com.ibm.commerce.extension.objects.OrderGiftKey)**。
5. 選取「屬性輔助程式」區段中的所有屬性。
6. 按下**完成**。

您可以切換至「J2EE 導覽器」檢視畫面，展開 **WebSphereCommerceServerExtensionsData** 專案，然後展開 **ejbModule** 子資料夾，再展開 **com.ibm.commerce.extension.objects**，來檢視新產生的程式碼。套件中會建立並顯示一個叫做 OrderGiftAccessBean 的新類別以及一個叫做 OrderGiftAccessBeanData 的新介面。

下一步是產生已部署的程式碼。

程式碼產生公用程式會分析 Bean，以確定是否符合 Sun Microsystem 的 EJB 規格，並確定是否遵循 EJB 伺服器的特定規則。此外，在每一個所選的 Enterprise Bean 方面，程式碼產生工具會爲發源與遠端介面產生發源與 EJBObject（遠端）實

作方式與實作類別，以及爲 CMP Bean 產生 JDBC persister 與 finder 類別。它同時會產生透過 IIOP 進行 RMI 存取時所需要的 Java ORB、存根與 Tie 類別，以及發源和遠端介面的存根。

如果要產生部署的程式碼，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**，然後以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData**，再選取**產生 > 部署和 RMIC 程式碼**。
   這時會開啓「部署和 RMIC 程式碼」視窗。
2. 選取 **OrderGift**，然後按一下**完成**。

您可以切換到「J2EE 導覽器」檢視畫面，來檢視新產生的程式碼。您會發現下列內容：

表 25.

程式碼類型	類別名稱
配置區實作產生的程式碼	EJSCMPOrderGiftHomeBean.java
	EJSRemoteCMPOrderGift.java
	EJSRemoteCMPOrderGiftHome.java
	EJSFinderOrderGiftBean.java
JDBC 存取程式碼	EJSJDBCPersisterCMPOrderGiftBean.java
RMI tie 和存根程式碼	_EJSRemoteCMPOrderGift_Tie.java
	_OrderGift_Stub.java
	_EJSRemoteCMPOrderGiftHome_Tie.java
	_OrderGiftHome_Stub.java

將 OrderGift Entity Bean 整合到購物流程中

在本節中，您執行下列步驟來將 OrderGift Entity Bean 整合到範例商店的一般購物流程中：

1. 建立新的 OrderGiftDataBean 資料 Bean
2. 建立新的 MyExtOrderProcessCmdImpl 作業指令
3. 修改 OrderSubmitForm.jsp 顯示頁面

上述每一個步驟都將在後續各節中詳細說明。

建立 OrderGiftDataBean

您必須建立 OrderGiftDataBean，使 OrderGift Entity Bean 的屬性可以顯示在 JSP 顯示頁面上。如同指導教學的其他部分一樣，此處提供的是基本程式碼，您需要取消註解許多不同的程式碼區段。

如果要建立 OrderGiftDataBean，請執行下列步驟：

1. 第一步是匯入新資料 Bean 的基本程式碼，方式如下：
   a. 確定您已經完成第 200 頁的『找出範例程式碼』中的步驟。
   b. 切換到「J2EE 視景」，然後選取「J2EE 導覽器」檢視畫面。
   c. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
   d. 以滑鼠右鍵按一下 **src** 資料夾，然後選取**匯入**。
      這時會開啓「匯入」精靈。
   e. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
   f. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
      yourDirectory\WC_SAMPLE_55.zip
      其中 *yourDirectory* 是您下載套件的目錄。
   g. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
      - com\ibm\commerce\sample\databeans\OrderGiftDataBean.java
   h. 在**資料夾**欄位中，已經指定 `WebSphereCommerceServerExtensionsLogic/src` 資料夾。請保留這個值。
   i. 按下**完成**。

一旦匯入 Bean 的程式碼之後，請檢查程式碼。

建立 MyExtOrderProcessCmdImpl 類別

在本節中，您會在 OrderProcess 商業程序的結尾加入新的邏輯，以更新 XORDGIFT 資料庫表格中與贈品訂單相關的資訊。ExtOrderProcessCmdImpl 指令是這個商業程序的延伸點。在程式設計模型之後，如果要延伸這個邏輯，您可以建立作業指令的新實作類別，並且在這個類別中併入新的邏輯。然後您必須更新指令登錄，將新的實作類別關聯到 ExtOrderProcessCmd 介面。

如果要建立 MyExtOrderProcessCmdImpl 類別，請執行下列步驟：

1. 第一步是匯入新指令的基本程式碼，方式如下：
   a. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
   b. 以滑鼠右鍵按一下 **src** 資料夾，然後選取**匯入**。
      這時會開啓「匯入」精靈。

c. 在匯入來源方面，選取 **Zip 檔**並按一下**下一步**。

d. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。

e. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
yourDirectory\WC_SAMPLE_55.zip
其中 *yourDirectory* 是您下載套件的目錄。

f. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。

- com\ibm\commerce\sample\commands\ MyExtOrderProcessCmdImpl.java

g. 在**資料夾**欄位中，已經指定 `WebSphereCommerceServerExtensionsLogic/src` 資料夾。請保留這個值。

h. 按下**完成**。

一旦匯入這個新指令的程式碼之後，您就可以檢查原始碼來察看指令的作用。請注意，指令會呼叫其超類別的 performExecute() 方法，以確定來自該指令的任何處理都會被執行。然後指令會將用來設定禮品訂單資訊的邏輯併入到新的資料 Bean 中。

在這個步驟中，您會修改指令登錄，以使用新的 MyExtOrderProcessCmdImpl 實作類別來替代原始的 ExtOrderProcessCmdImpl 實作類別。在指令登錄中，唯一需要修改的表格是 CMDREG 表格。在此情況下，所有的商店都會使用新的實作類別。

如果要修改指令登錄，請執行下列步驟：

DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyExtOrderProcessCmdImpl：

1. 開啓 DB2 指令中心（**開始 > 程式集 >IBM DB2 > 指令行工具 > 指令中心**）。
2. 從**工具**功能表中選取**工具設定**。
3. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (**;**)。
4. 關閉工具設定。
5. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to developmentDB user dbuser using dbpassword;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼
- *FashionFlow_storeent_ID* 是您的範例商店的商店 ID

按一下**執行**圖示。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

   按 Enter 鍵以執行 SQL 陳述式。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

編譯變更

重要事項：請確定您沒有重新建置 Stores Web 專案。

在本節中，您會編譯對程式碼所做的變更，方式如下：

1. 在「J2EE 導覽器」檢視畫面中，選取下列專案：
   - WebSphereCommerceServerExtensionsData
   - WebSphereCommerceServerExtensionsLogic

2. 在標示前面的專案之後，以滑鼠右鍵按一下並選取**建置專案**。

修改贈品訊息的顯示頁面

在這個步驟中，您需要修改 OrderSubmitForm 和 OrderDetailDisplay 範本，使客戶可以輸入贈品訂單的訊息資訊，並且檢視摘要頁面中的資訊。修改這些頁面的策略是併入其他的 JSP 範本以指定頁面的新資訊。這些新頁面（OrderSubmitFormInclude.jsp 和 OrderDetailDisplayInclude.jsp）使用 JSTL 來顯示新的資訊。

如果要修改顯示頁面，請執行下列步驟：

1. 切換到「Web」視景，然後使用「J2EE 導覽器」檢視畫面。
2. 如果您尚未完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，就必須依照下列方式修改 Stores Web 專案的內容：
   a. 以滑鼠右鍵按一下 **Stores** Web 專案，然後選取**內容**。
   b. 在左窗格中選取 **Web**，然後從「可用的 Web 專案特性」清單中，選取**包含 JSP 標準標籤程式庫**。按一下**套用**。當更新完成時，按一下**確定**來關閉內容編輯程式。
3. 展開下列目錄：
   Stores\Web Content*FashionFlow_name*.
4. 執行下列步驟來建立 OrderSubmitForm.jsp 檔的備份：
   a. 展開 **ShoppingArea>CheckoutSection>StandardCheckoutSubsection** 目錄。
   b. 以滑鼠右鍵按一下 **OrderSubmitForm.jsp** 檔，然後選取**更名**。
   c. 在「更名」視窗中，輸入 `OrderSubmitForm_bak.jsp`，然後按一下**確定**。
   d. 在提示您是否要更新此檔案的鏈結時，請按一下**否**。
5. 執行下列步驟來建立 OrderDetailDisplay.jsp 檔的備份：
   a. 展開 **UserArea>ServiceSection>TrackOrderStatusSubsection** 目錄。
   b. 以滑鼠右鍵按一下 **OrderDetailDisplay.jsp** 檔，然後選取**更名**。
   c. 在「更名」視窗中，輸入 `OrderDetailDisplay_bak.jsp`，然後按一下**確定**。
   d. 在提示您是否要更新此檔案的鏈結時，請按一下**否**。
6. 以滑鼠右鍵按一下 *FashionFlow_name* 目錄，然後選取**匯入**。
   這時會開啓「匯入」精靈。
7. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
8. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：

yourDirectory\WC_SAMPLE_55.zip
其中 *yourDirectory* 是您下載套件的目錄。

9. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\ OrderSubmitForm.jsp
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\ OrderSubmitFormInclude.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\ OrderDetailDisplay.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\ OrderDetailDisplayInclude.jsp
10. 在**資料夾**欄位中，已經指定 `Stores/Web Content/`*`FashionFlow_name`* 資料夾。請保留這個值。
11. 按下**完成**。

如果您檢查 OrderSubmitForm.jsp 檔，您會發現其中包含下列項目：

```
<!-- 指導教學開始-->
   <jsp:include page="OrderSubmitFormInclude.jsp" flush="true" />
<!-- 指導教學完成 -->
```

這是將新的 OrderSubmitFormInclude.jsp 檔納入現有 JSP 範本的方法。請檢查 OrderSubmitFormInclude.jsp，以取得如何在您的範本中使用 JSTL 的範例。同樣的，您可以檢查 OrderDetailDisplay.jsp 和 OrderDetailDisplayInclude.jsp 檔案。

您也必須匯入含有已修改 JSP 範本中使用的字串值的內容檔。這個檔案稱爲 ordergift.properties。如果要匯入這個檔案，請執行下列步驟：

1. 在「J2EE 導覽器」檢視畫面中，展開下列目錄：
   Stores > Web Content> WEB-INF> classes> *FashionFlow_name* 目錄。
2. 以滑鼠右鍵按一下 *FashionFlow_name* 目錄，然後選取**匯入**。
   這時會開啓「匯入」精靈。
3. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
4. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
   yourDirectory\WC_SAMPLE_55.zip
   其中 *yourDirectory* 是您下載套件的目錄。
5. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
   - ordergift.properties
6. 在**資料夾**欄位中，已經指定 `Stores/Web Content/WEB-INF/classes/`*`FashionFlow_name`* 資料夾。請保留這個值。
7. 按下**完成**。

測試新贈品頁面的功能

在本節中，您將測試贈品訊息功能，方式如下：

1. 切換到「伺服器」視景（**視窗 > 開啓視景 > 伺服器**）。
2. 啓動您的 Payment Server。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 伺服器，然後選取**啓動**（或**重新啓動**）。
4. 開啓 Web 瀏覽器，然後輸入以下的 URL：

   ```
   http://localhost/webapp/wcs/stores/servlet/FashionFlow/index.jsp
   ```

5. 以新使用者的身份登入。例如，按一下**登錄**，然後建立一個新使用者，或者登入爲現有的使用者。
6. 使用新的已登錄使用者的身份，瀏覽商店，將項目新增至購物車中，然後完成購買。您可以新增贈品訊息到訂單中（如下面的螢幕擷取圖所示）：

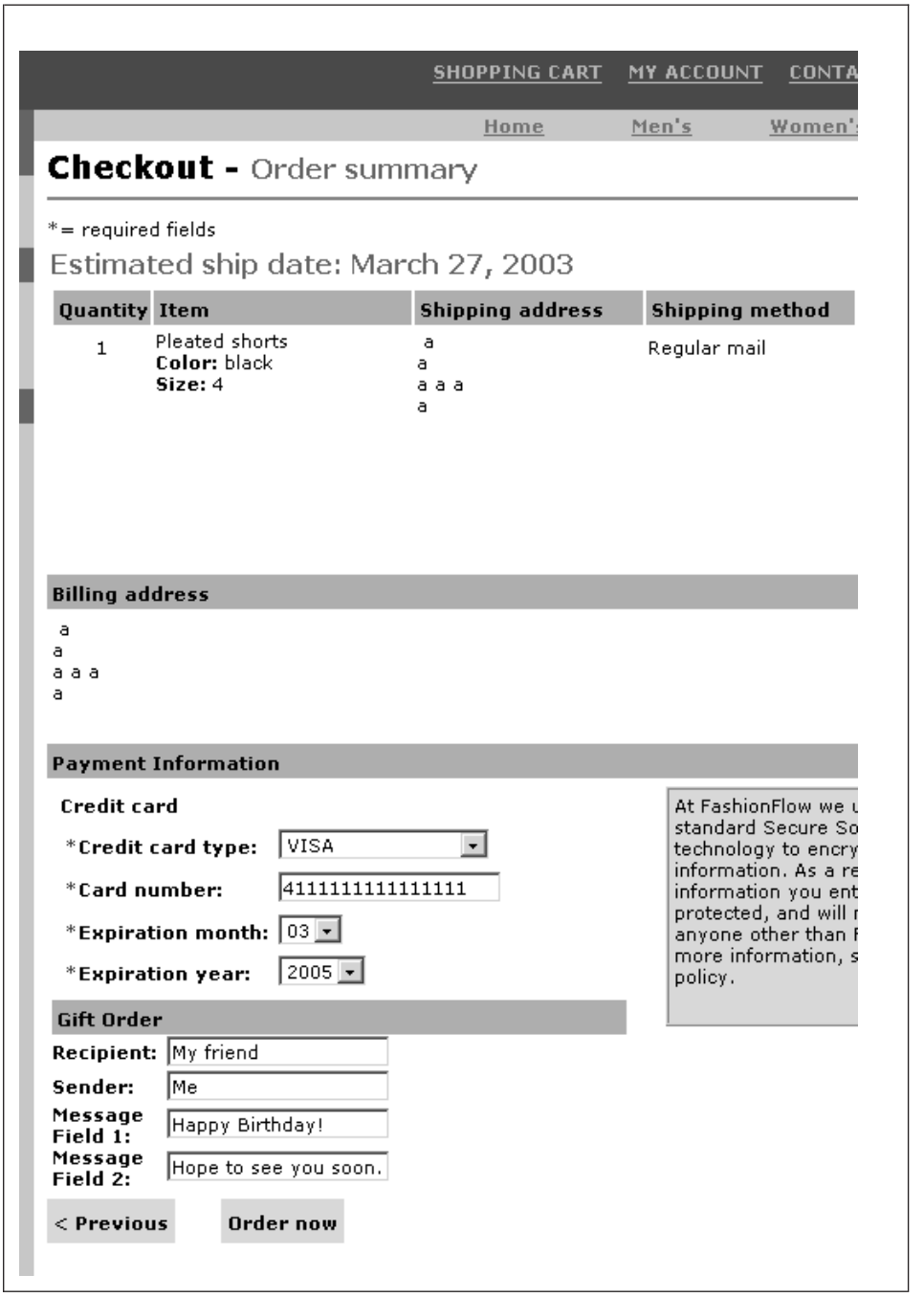

圖 49.

記下與這份訂單相關的訂單號碼。

7. 按一下**我的帳戶**。
8. 按一下**檢視訂單**。
9. 選取您在步驟 6 中建立的訂單。您會看到類似以下的畫面：

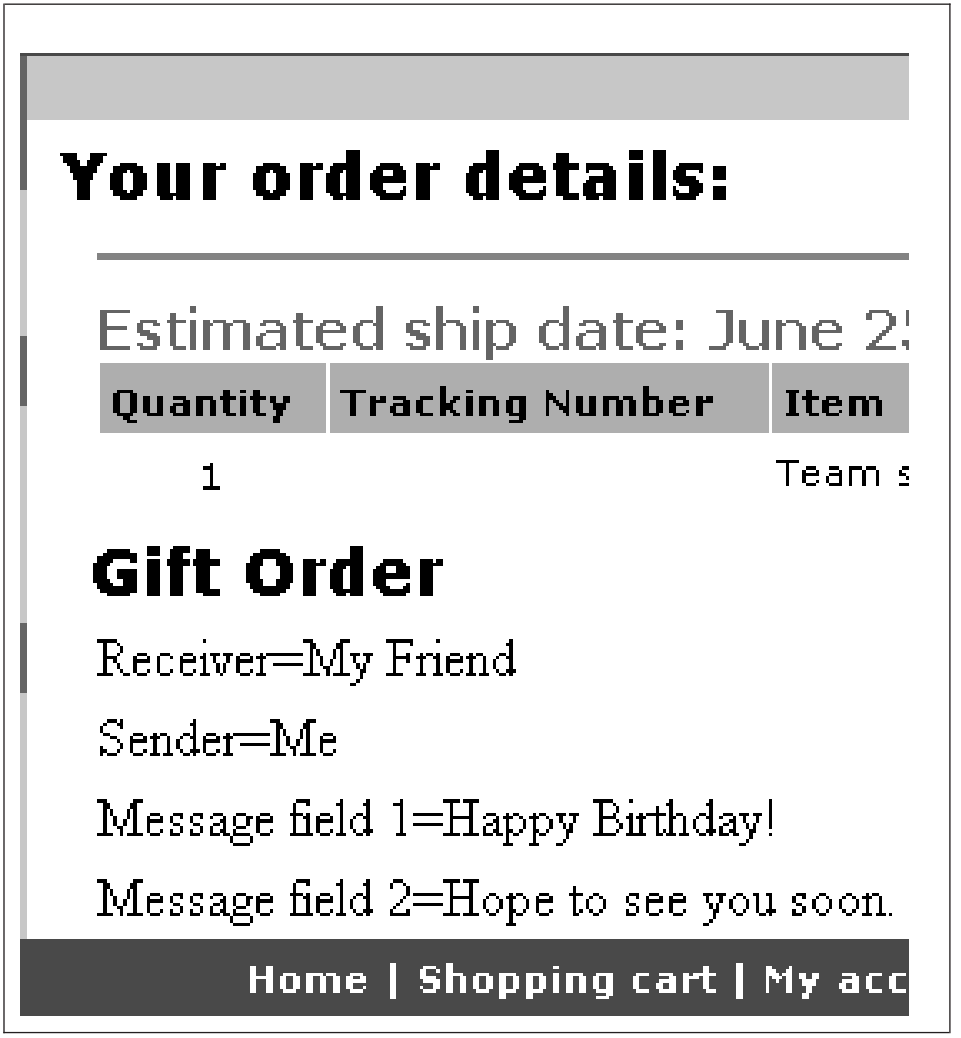

圖 50.

部署贈品訊息功能

本節說明如何將新商業邏輯部署到執行於遠端 WebSphere Commerce Server 的商店中。在您開始進行這些部署步驟前，您必須已經在遠端 WebSphere Commerce Server 上建立一家商店（以「流行館」範例商店爲基礎）。在該商店中，您必須能夠完成一項交易。

部署程序包括：在開發機器上執行的步驟以及在目標 WebSphere Commerce Server 上執行的步驟。

有許多不同類型的資產必須部署到目標 WebSphere Commerce Server 上。這些作業包括：

- 作業指令和資料 Bean 邏輯
- Enterprise Bean 邏輯
- 已更新的 JSP 範本
- 資料庫更新包括綱目更新（新表格）以及指令登錄更新

本節說明如何將所有的資產以*遞增的*方式部署到目標 WebSphere Commerce Server 上。這是以遞增式部署的方式執行，而不是部署整個 EAR 檔。

建立指令及資料 Bean JAR 檔

本節說明如何建立包含控制程式指令、作業指令和資料 Bean 邏輯的 JAR 檔。

如果要建立這個 JAR 檔，請在開發機器上執行下列步驟：

1. 在您的本端檔案系統上建立一個叫做 *drive*:\ExportTemp3 的目錄。
2. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**匯出**。
   這時會開啓「匯出」精靈。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **JAR 檔**，然後按一下**下一步**。
   b. **選取要匯出的資源**下面的左窗格中，會預先移入專案的名稱。請保留這個值。
   c. 在右窗格中，確定只有選取下列資源：
      - .classpath
      - .project
      - .serverPreference

d. 確定**匯出產生的類別檔和資源**已經選取。

e. *請勿*選取**匯出 Java 原始檔和資源**。

f. 在**選取匯出目的地**欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp3\WebSphereCommerceServerExtensionsLogic.jar。請注意，JAR 檔名必須是 WebSphereCommerceServerExtensionsLogic.jar。

g. 按下**完成**。

建立 EJB JAR 檔

如果要建立 EJB JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
2. 展開 **WebSphereCommerceServerExtensionsData** 專案。
3. 按兩下 **EJB 部署描述子**。
4. 在選取「概觀」標籤之後，捲動至窗格底端，尋找 **WebSphere 連結**區段。
5. 在**資料來源 JNDI 名稱**欄位中，輸入目標 WebSphere Commerce Server 的資料來源 JNDI 名稱。以下是範例值：

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 DB2 資料庫，而 WebSphere Commerce 實例名稱是 “demo”

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 Oracle 資料庫，而 WebSphere Commerce 實例名稱是 “demo”。

「資料來源 JNDI」名稱的值是藉由新增 “jdbc/” 到目標 WebSphere Commerce Server 的資料來源名稱所建立的。您可以開啓目標 WebSphere Commerce Server 上的 *instanceName*.xml 檔，並搜尋檔案中的 DatasourceName=，來驗證資料來源名稱。

6. 儲存您的部署描述子變更 (Ctrl + S)。
7. 在「J2EE 導覽器」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsData** 專案，然後選取**匯出**。這時會開啓「匯出」精靈。
8. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **EJB JAR 檔**，然後按一下**下一步**。
   b. **您要匯出的資源**？的值會預先移入 EJB 專案的名稱。請保留這個值。

c. 在**您要將資源匯出至**？欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp3\WebSphereCommerceServerExtensionsData.jar。

d. 按下**完成**。

9. 在建立 JAR 檔之後，請還原在步驟 5 中對本端部署描述子所作的變更，來還原您的本端測試伺服器所需要的設定。

匯出商店資產

如果要從 WebSphere Studio Application Developer 匯出 OrderSubmitForm 以及 OrderDetailDisplay 顯示範本，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Stores** 資料夾。
3. 以滑鼠右鍵按一下 **Web Content** 資料夾，然後選取**匯出**。
這時會開啓「匯出精靈」。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取**檔案系統**，然後按一下**下一步**。
   b. 選取要部署下列資源：
      - Web Content*FashionFlow_name*\ShoppingArea\CheckoutSection\StandardCheckoutSubsection\ OrderSubmitForm.jsp
      - Web Content*FashionFlow_name*\ShoppingArea\CheckoutSection\StandardCheckoutSubsection\ OrderSubmitFormInclude.jsp
      - Web Content*FashionFlow_name*\UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplay.jsp
      - Web Content*FashionFlow_name*\UserArea\ServiceSection\TrackOrderStatusSubsection\ OrderDetailDisplayInclude.jsp
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
      - Web Content\WEB-INF\classes*FashionFlow_name*\ordergift.properties
   c. 選取**建立檔案的目錄結構**。
   d. 在「目錄」欄位中，輸入要放置這些資源的暫時目錄。例如，輸入 C:\ExportTemp3
   e. 按下**完成**。

將資產轉送到您的目標 WebSphere Commerce Server

在這個步驟中，您會在目標 WebSphere Commerce Server 上建立一個暫時目錄，然後將您的贈品訂單資產複製到這個目錄中。在後續的步驟中，您會將不同類型的程式碼放置到 WebSphere Commerce 應用程式內的適當位置中。

如果要將檔案從您的開發機器複製到目標 WebSphere Commerce Server，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 上，建立一個叫做 *drive*:\ImportTemp3 的暫時目錄。
2. 決定您要如何將檔案從某一部電腦複製到另一個電腦上。您可以將目標 WebSphere Commerce Server 上的磁碟機對映到開發機器上，或者使用 FTP 應用程式（如果已經配置）來進行複製。
3. 從開發機器上，將 *drive*:\ExportTemp3 的內容複製到目標 WebSphere Commerce Server 上的 *drive*:\ImportTemp3。

停止您的目標 WebSphere Commerce Server

在開始部署步驟前，您應該停止目標 WebSphere Commerce Server，方法是在指令行發出 stopServer 指令。有關這個指令的明細，請參閱*WebSphere Commerce Studio 安裝手冊*。

更新您的目標 WebSphere Commerce Server 上的資料庫

登錄作業指令實作

如果要修改指令登錄，請執行下列步驟：

1. DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyExtOrderProcessCmdImpl：
   a. 開啓 DB2 指令中心（**開始 > 程式集 >IBM DB2 > 指令行工具 > 指令中心**）。
   b. 從**工具**功能表中選取**工具設定**。
   c. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
   d. 關閉工具設定。
   e. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to targetDB user dbuser using dbpassword;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
```

```
'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
'This is a new task command for tutorial two.',
'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
'local');
```

其中

- *targetDB* 是您的目標資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是資料庫使用者的密碼
- *FashionFlow_storeent_ID* 是您的範例商店的商店 ID

按一下**執行**圖示。使「指令中心」保持開啓。

2. Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyOrderItemAddCmdImpl：
   a. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
   b. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
   c. 在**密碼**欄位中，輸入您的 Oracle 密碼。
   d. 在**主機字串**欄位中，輸入您的連接字串。
   e. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (FashionFlow_storeent_ID,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

   按 Enter 鍵以執行 SQL 陳述式。
   f. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

建立 XORDGIFT 表格

在這個步驟中，您將建立 XORDGIFT 表格。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 在「Script」視窗中，輸入下列指令：

```
create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
constraint p_xordgift primary key (ORDERSID),
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

其中

- *targetDB* 是您的目標資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是資料庫使用者的密碼

按一下「執行」圖示。
現在 XORDGIFT 表格已建立完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在「SQL Plus」視窗中，輸入下列 SQL 陳述式：

   ```
   create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
   SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
   constraint p_xordgift primary key (ORDERSID),
   constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
   on delete cascade);
   insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XORDGIFT 表格。
6. 輸入下列以確定您的資料庫變更：

   ```
   commit;
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。

更新您的目標 WebSphere Commerce Server 上的商店資產

在這個步驟中，您會使用已修改的商店資產來更新商店，方式如下：

1. 備份您的 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war 目錄。
2. 導覽至 *drive*:\ImportTemp3\Stores\Web Content 目錄。

3. 將 *FashionFlow_name* 資料夾複製到下列目錄中：
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新您的目標 WebSphere Commerce Server 上的指令和資料 Bean JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的指令和資料 Bean JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 WebSphereCommerceServerExtensionsLogic.jar 檔的副本，並將它儲存在備份位置中。
2. 將新的 WebSphereCommerceServerExtensionsLogic.jar 檔從 *drive*:\ImportTemp3 目錄複製到 *WAS_installdir*\nstalledApps*cellName*\WC_*instanceName*.ear 目錄中

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新您的目標 WebSphere Commerce Server 上的 EJB JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的 EJB JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 WebSphereCommerceServerExtensionsData.jar 檔的副本，並將它儲存在備份位置中。

   其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。
2. 將新的 WebSphereCommerceServerExtensionsData.jar 檔從 *drive*:\ImportTemp3 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中
3. 接下來，您必須修改 EJB 部署描述子資訊，方式如下：
   a. 尋找這個 WebSphere Application Server 資料格的部署儲存庫（META-INF 目錄）。其格式通常如下：
      WAS_installdir\config\cells*cellName*
      \applications\WC_*instance_name*.ear\deployments\
      WC_*instance_name**EJBModuleName*.jar\META-INF
      以下是這項資訊的特定範例：
      D:\WebSphere\AppServer\config\cells\myCell\applications\

WC_demo.ear\deployments\WC_demo\
WebSphereCommerceServerExtensionsData.jar\META-INF
其中

- mycell 是 WebSphere Application Server 資料格的名稱
- demo 是 WebSphere Commerce 實例的名稱

b. 目錄包含下列檔案：

- ejb-jar.xml
- ibm-ejb-access-bean.xmi
- ibm-ejb-jar-bnd.xmi
- ibm-ejb-jar-ext.xmi
- MANIFEST.MF

備份所有的檔案。

c. 使用工具來開啓新的 WebSphereCommerceServerExtensionsData.jar 檔並檢視其內容。

d. 將 meta-inf 目錄的內容從這個 WebSphereCommerceServerExtensionsData.jar 檔擷取到步驟 3a 中的目錄裡。

4. 在指令行中，使用 WebSphere Application Server startServer 指令來重新啓動您的 WebSphere Commerce 實例。

驗證目標 WebSphere Commerce Server 上的禮品訊息功能

在這一節中，您將執行下列步驟來驗證目標 WebSphere Commerce Server 上的禮品訊息邏輯可以正常運作：

1. 開啓 Web 瀏覽器，然後輸入您根據「流行館」範例商店所建立的商店的 URL。
2. 以新使用者的身份登入。例如，按一下「登錄」，然後建立使用者 ″shopper″。
3. 使用新的已登錄使用者的身份，瀏覽商店，將項目新增至購物車中，然後完成購買。您可以新增贈品訊息到訂單中（如下面的螢幕擷取圖所示）：

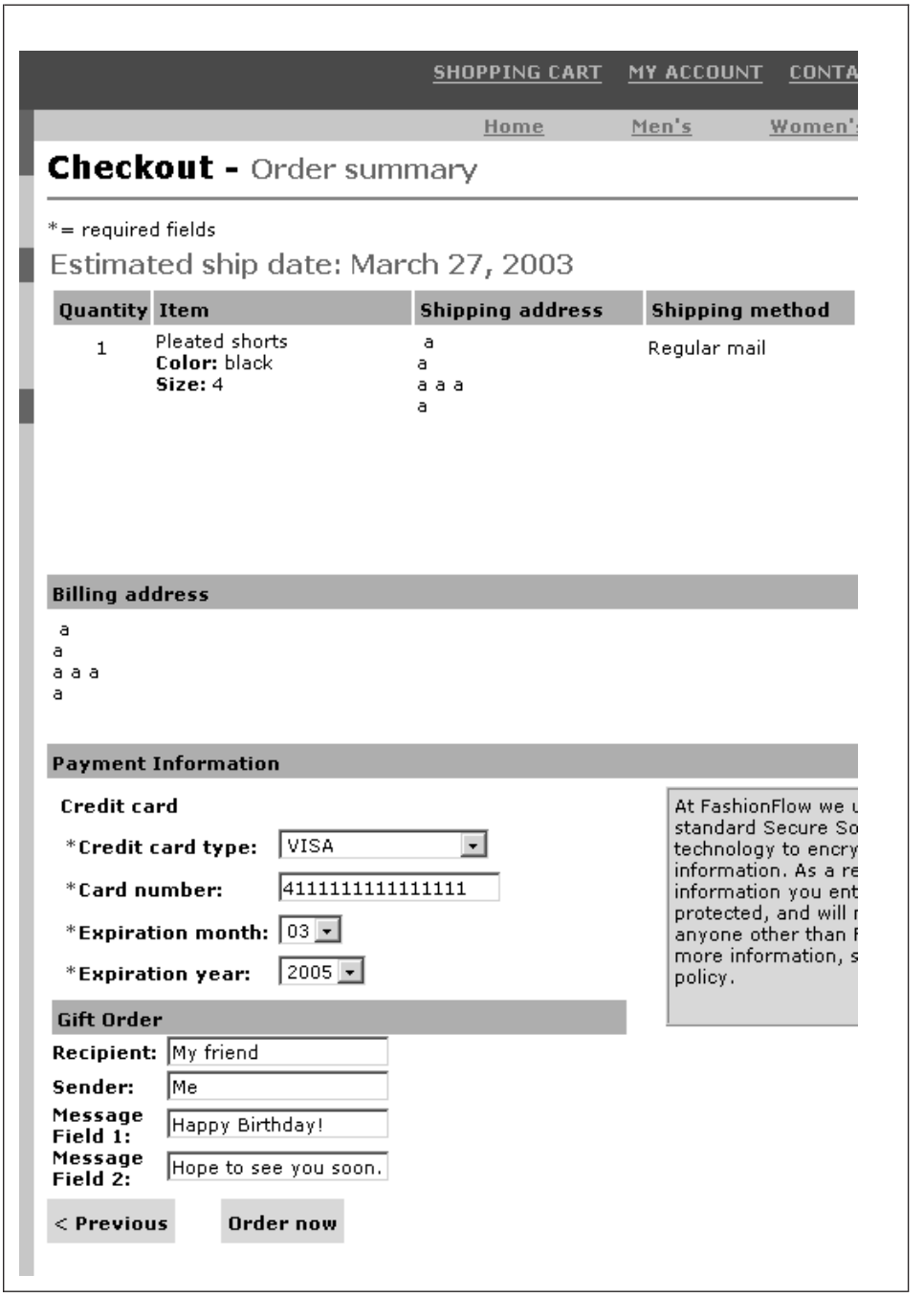

圖 51.

記下與這份訂單相關的訂單號碼。

4. 按一下**我的帳戶**。
5. 按一下**檢視訂單**。
6. 選取您在步驟 3 中建立的訂單。您會看到類似以下的畫面：

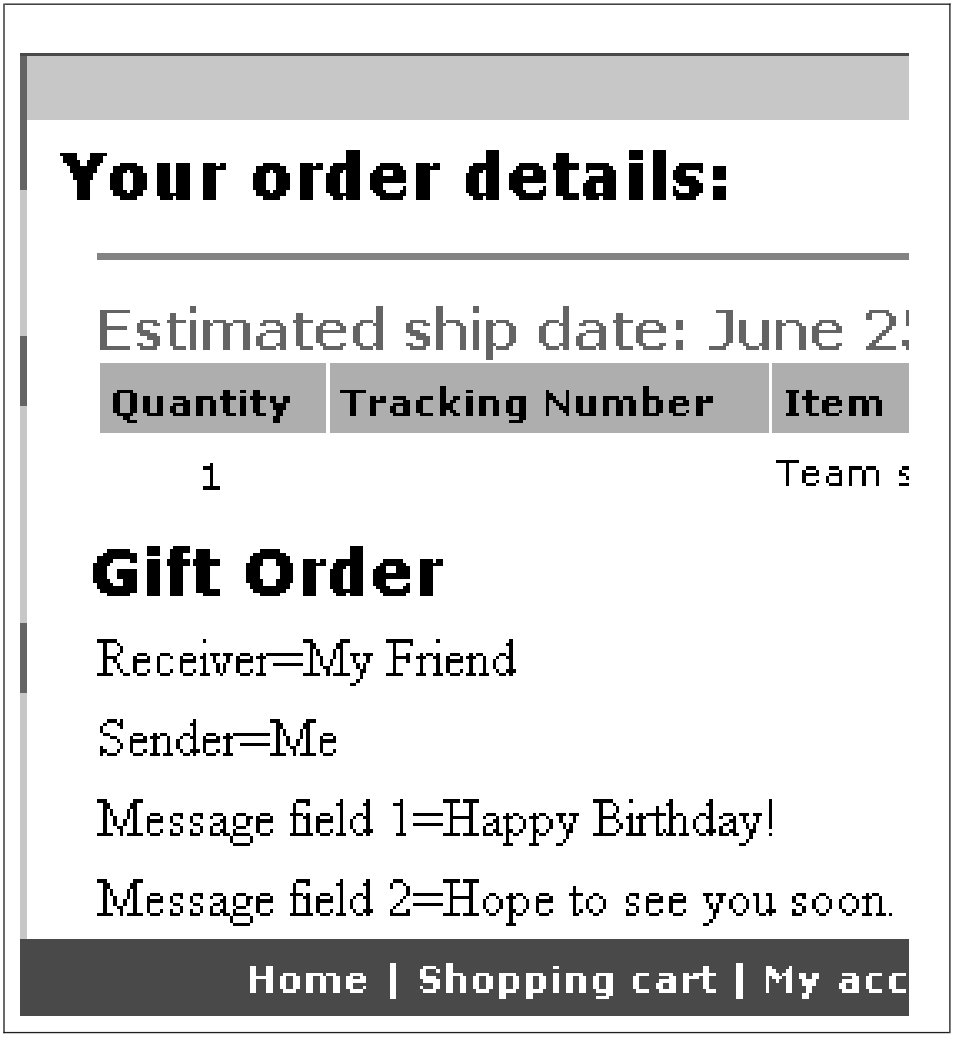

圖 52.

第 13 章 指導教學：延伸現有的 WebSphere Commerce Entity Bean

在本指導教學中，您將學習用來修改現有 WebSphere Commerce Entity Bean 的程序。其中所使用的實務內容會將額外的住家資訊調查新增至使用者登錄處理中。在此情況下，會修改 User Entity Bean 以併入用來儲存使用者住家類型值以及位置值的其他欄位。同時建立一個新的資料庫（稱爲 XHOUSING），然後爲 User Bean 修改現有的對映資訊，以併入這個新表格。

除了新表格和 Entity Bean 的變更外，也會建立一個新的 MyPostUserRegistrationAddCmdImpl 實作類別。其中負責包含處理住家調查資訊以及使用新資訊來更新資料庫的邏輯。

爲了能夠收集住家資訊，並且在稍後顯示結果，所以會修改 UserRegistrationAddForm.jsp 和 UserRegistrationUpdateForm.jsp 檔。

必備需求

本指導教學並不要求您事先完成指導教學。如果您已經完成這些指導教學，您仍然可以將程式碼保留在您的工作區中，也不會與本指導教學有任何衝突。

在開始本指導教學之前，您必須已經根據「流行館」範例商店公佈了一家商店。在這家商店中，您必須能夠完成購買（例如，瀏覽型錄、新增項目至購物車、結帳以及察看訂單資訊）。

建立 XHOUSING 表格並移入資料

在準備建立 Entity Bean 的過程中，您必須先建立新資料庫表格並移入資料。要建立的表格稱爲 XHOUSING。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令行工具 > 指令中心**），然後按一下 **Script 編寫**標籤。
2. 在「Script」視窗中，輸入下列指令：

```
connect to developmentDB user dbuser using dbpassword;
create table XHOUSING (MEMBERID bigint not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
```

```
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼

按一下「執行」圖示。
您應該會看到一則訊息，指出 SQL 陳述式已經順利完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
create table XHOUSING (MEMBERID number not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

   並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XHOUSING 表格。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

將新的欄位加到 User Entity Bean 中

在這一節中，您會加入兩個新欄位到用來擷取住家資訊的 User Entity Bean 中。這兩個新欄位稱爲 housingType 與 location。這兩個欄位最後會對映到 XHOUSING 表格的 HOUSINGTYPE 和 LOCATION 直欄。

如果要加入這兩個新欄位，請執行下列步驟：

1. 若尚未開啓的話，請啓動 WebSphere Commerce Studio（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**）。

2. 切換到「伺服器」視景，並確定 WebSphereCommerceServer 測試伺服器已經停止。
3. 切換到「J2EE 視景」並選取「J2EE 階層」檢視畫面。
4. 展開 **EJB 模組**，然後按兩下 **Member-MemberManagementData** EJB 專案。這樣就會開啓「EJB 部署描述子」編輯程式。
5. 按一下 **Bean** 標籤，然後從顯示的 Bean 清單中選取 **User** Bean。
6. 按一下 CMP 欄位文字框旁的**新增**。
   這時會開啓「建立 CMP 屬性」視窗。
7. 使用下列內容來建立 CMP 欄位：
   a. 在**名稱**欄位中，輸入 `housingType`。
   b. 在「類型」欄位中，輸入 `java.lang.Integer`。
   c. 啓用**使用 getter 與 setter 方法來存取**。
   d. 清除**將 getter 與 setter 方法提升至遠端介面**勾選框。（這樣就會自動清除使 getter 變成唯讀的選項。）
   e. 按一下**確定**。
8. 再按一下**新增**，利用下列內容來建立另一個 CMP 欄位：
   a. 在**名稱**欄位中，輸入 `location`。
   b. 在「類型」欄位中，輸入 `java.lang.Integer`。
   c. 啓用**使用 getter 與 setter 方法來存取**。
   d. 清除**將 getter 與 setter 方法提升至遠端介面**勾選框。（這樣就會自動清除使 getter 變成唯讀的選項。）
   e. 按一下**確定**。

   新的欄位會顯示在 CMP 欄位清單中。
9. 儲存您的變更，然後關閉「EJB 部署描述子」編輯程式。

更新綱目與表格對映資訊

在下列各節中，您將以新 XHOUSING 表格來更新 User 綱目、建立新表格的外來鍵關係，以及在 User Entity Bean 的欄位與 XHOUSING 表格的直欄之間建立表格對映。利用這個方法來延伸物件模型時，就像是將新的欄位直接加入到 USERS 表格中一樣來編寫程式碼。

建立 XHOUSING 表格的表格定義

如果要建立 XHOUSING 表格定義以及在 USERS 和 XHOUSING 表格之間建立外來鍵關係，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開**資料庫**。
2. 展開 **Member-MemberManagementData** 資料庫，然後展開 **NULLID** 綱目。
3. 以滑鼠右鍵按一下 **表格**，然後選取**新建 > 新表格定義**。
   這時會開啓「表格定義」視窗。
4. 在**表格名稱**欄位中，輸入 XHOUSING，然後按一下**下一步**。
5. 接下來，您必須新增鍵值直欄到表格定義中，方式如下：
   a. 按一下**新增另一個**。
   b. 在**直欄名稱**欄位中，輸入 MEMBERID。
   c. DB2 從**直欄類型**下拉清單中，選取 BIGINT。
      Oracle 從**直欄類型**下拉清單中，選取 NUMBER。
   d. 選取**鍵值直欄**。
   e. Oracle 在**數字精準度**欄位中，輸入 38。
   f. Oracle 將**數字表**的值保持爲 0。
6. 接下來，新增另一個直欄到表格定義中，方式如下：
   a. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 26.

內容	值
直欄名稱	HOUSINGTYPE
直欄類型	INTEGER
可為空值	Select

   b. 按一下**新增另一個**，然後使用下列內容來建立一個直欄：

表 27.

內容	值
直欄名稱	LOCATION
直欄類型	INTEGER
可為空值	Select

   c. 按**下一步**。
7. 在**主要鍵名稱**欄位中，輸入 p_xhousing，然後按一下**下一步**。
8. 按一下**新增另一個**來新增外來鍵。指定下列的值：
   a. 在**外來鍵名稱**欄位中，輸入 f_xhousing。
   b. 從**刪除時**下拉清單中，選取 **CASCADE**。

c. 從**目標表格**下拉清單中，選取 **NULLID.USERS**。

d. 在「來源直欄」窗格中，按一下**MEMBERID**，然後按一下**>**來新增外來鍵。

9. 按一下**完成**。

10. Oracle 您必須使用文字編輯程式來編輯表格定義，方式如下：

a. 切換到「J2EE 導覽器」檢視畫面。

b. 展開 **Member-MemberManagementData** 專案。

c. 展開下列項目：**ejbModule > META-INF >Schema**。

d. 以滑鼠右鍵按一下 **Member-MemberManagementData_NULL_XHOUSING.xmi** 檔，然後選取**開啓方式 > 文字編輯程式**。

e. 將所有出現的 `SQLNumeric_6` 取代爲 `SQLNumeric_3`。

f. 儲存您的變更並關閉文字編輯程式。

您可以展開 Member-MemberManagementData/NULLID 下面的「表格」資料夾，來檢視新的 Member-MemberManagementData_NULL_XHOUSING 表格定義。

建立 XHOUSING 表格對映

在本節中，您可以在 XHOUSING 表格中的兩個直欄（HOUSINGTYPE 和 LOCATION）以及 User Entity Bean 中的兩個欄位（housingType 和 location）之間建立對映。

如果要建立這個對映，請執行下列步驟：

1. 切換到「J2EE 導覽器」檢視畫面。
2. 展開下列資料夾：**Member-MemberManagementData > ejbModule > META-INF**。
3. 按兩下 **Map.mapxmi** 檔。
4. 在「表格」窗格中，表示下列表格：
   - **MEMBER**
   - **USERS**
   - **XHOUSING**

   （按住 Ctrl 鍵來同時選取多個表格。）
5. 在 Enterprise Beans 窗格中（位於編輯程式頂端），展開**成員**群組，然後以滑鼠右鍵按一下 **User** Entity Bean，然後選取**建立對映**。
6. 展開 **User** Entity Bean。

7. 在「表格」窗格中，展開 **XHOUSING** 表格，以便檢視其直欄。
8. 標示 **housingType** Bean 屬性，然後將它拖曳到 **HOUSINGTYPE** 直欄來建立對映。
9. 標示 **location** Bean 屬性，然後將它拖曳到 **LOCATION** 直欄來建立對映。
10. 儲存您的變更，然後關閉 Map.mapxmi 檔。

更新對映檔

1. 展開下列資料夾：**Member-MemberManagementData > ejbModule > META-INF**。
2. 以滑鼠右鍵按一下 **Map.mapxmi** 檔，然後選取**開啓方式 > 文字編輯程式**。
3. 尋找下列字串：

   ```
   User_EJB
   ```

   在這個字串下面的兩行中，您會發現：

   ```
   <discriminatorValues>User</discriminatorValues>
   ```

   如果您找到前面這一行，則您必須將它變更爲

   ```
   <discriminatorValues>'U'</discriminatorValues>
   ```
4. 儲存您的變更。

產生存取 Bean 以及已部署的程式碼

由於您已經修改了 User Entity Bean，您必須重新產生其存取 Bean 以及已部署的程式碼。

如果要重新產生存取 Bean，請執行下列步驟：

1. 在「J2EE 階層」檢視畫面中，展開 **EJB 模組**。
2. 以滑鼠右鍵按一下 **Member-MemberManagementData** EJB 模組，然後選取**存取 Bean> 編輯存取 Bean**。
3. 選取 **CopyHelper**，然後按一下**下一步**。
4. 在「選取 EJB 專案」視窗中，按一下**下一步** 。
5. 在「複製輔助程式存取 Bean」視窗中，執行下列步驟：
   a. 從 Enterprise Bea 下拉清單中，選取 **User**。
   b. 從**建構子方法**下拉清單中，選取 **findByPrimaryKey(com.ibm.commerce.user.objects.MemberKey)**。
   c. 從「屬性輔助程式」清單中，確定 **housingType** 和 **location** 屬性已經與其他所有的屬性一併選取。

6. 按一下**完成**。
7. 以滑鼠右鍵按一下 **Member-MemberManagementData** EJB 模組，然後選取 **存取 Bean> 重新產生存取 Bean**。
8. 按一下**全選**，然後按一下**完成**。

如果要重新產生已部署的程式碼，請執行下列步驟：

1. 以滑鼠右鍵按一下 **Member-MemberManagementData** EJB 模組，然後選取 **產生 > 部署和 RMIC 程式碼**。
2. 按一下**全選**，然後按一下**完成**。

建立 MyPostUserRegistrationAddCmdImpl 實作

在這個步驟中，您將會延伸 UserRegistrationAddCmd 控制程式指令，以併入用來剖析在登錄套表中所收集的新住家資訊的新邏輯。這個延伸的執行方式是藉由建立一個新的 MyPostUserRegistrationAddCmdImpl 實作類別來實作 PostUserRegistrationCmd 介面。在建立這個新實作類別之後，您必須更新指令登錄，以反映這項變更。如同其他指導教學一樣，也會提供這個新指令的程式碼。

如果要建立新的 MyPostUserRegistrationAddCmdImpl 實作類別，請執行下列步驟：

1. 確定您已經完成第 200 頁的『找出範例程式碼』中的步驟。
2. 在 WebSphere Studio Application Developer 中，開啓 Java 視景（**視窗 > 開啓視景 > Java**）。
3. 展開 **WebSphereCommerceServerExtensionsLogic** 專案。
4. 以滑鼠右鍵按一下 **src** 資料夾，然後選取**匯入**。
   這時會開啓「匯入」精靈。
5. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
6. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
   yourDirectory\WC_SAMPLE_55.zip
   其中 *yourDirectory* 是您下載套件的目錄。
7. 按一下**取消全選**，然後展開目錄並選取要匯入下列檔案：
   - com\ibm\commerce\sample\commands\ MyPostUserRegistrationAddCmdImpl.java
8. 在**資料夾**欄位中，已經指定 `WebSphereCommerceServerExtensionsLogic/src` 資料夾。請保留這個值。
9. 按一下**完成**。
10. 展開 **com.ibm.commerce.sample.commands** 套件。
11. 按兩下新的 **MyPostUserRegistrationAddCmdImpl** 類別。

12. 取消註解「區段 1」。這個程式碼會設定變數並且爲這些變數建立 getters 和 setters：

```
/// 區段 1 ////////////////
 Integer housingType = null;
 Integer location = null;

 public Integer getHousingType() {
  return housingType;
 }

 public Integer getLocation() {
  return location;
 }

 public void setHousingType(Integer newHousingType) {
  housingType = newHousingType;
 }

 public void setLocation(Integer newLocation) {
  location = newLocation;
 }
/// 區段 1 結束 ////////////////
```

13. 取消註解「區段 2」。這樣會將下列程式碼引入類別中：

```
/// 區段 2 ////////////////
public void performExecute() throws ECException {
  super.performExecute();

  // 在處理調查之前設定必要的欄位
  setHousingType(requestProperties.getInteger("housingType", 0));
  setLocation(requestProperties.getInteger("location", 0));

  processSurvey();
 }

/// 區段 2 結束 ////////////////
```

前面的程式碼會呼叫超類別的 performExecute 方法，以便執行 PostUserRegistrationAddCmdImpl 的一般邏輯。一旦執行完成之後，就會呼叫新的 processSurvey 方法。

14. 取消註解「區段 3」來將下列程式碼引入到類別中：

```
/// 區段 3 ////////////////

 private void processSurvey() throws ECException {

  try {
   // 載入使用者資料
   UserAccessBean abUser = new UserAccessBean();
   abUser.setInitKey_MemberId(commandContext.getUserId().toString());
   abUser.refreshCopyHelper();
```

```
        // 儲存新的屬性
        abUser.setHousingType(getHousingType());
        abUser.setLocation(getLocation());

        abUser.commitCopyHelper();
          } catch (javax.ejb.FinderException e) {
        throw new ECSystemException(
         ECMessage._ERR_FINDER_EXCEPTION,
         this.getClass().getName(),
         "processSurvey");
          } catch (javax.naming.NamingException e) {
        throw new ECSystemException(
         ECMessage._ERR_NAMING_EXCEPTION,
         this.getClass().getName(),
         "processSurvey");
          } catch (java.rmi.RemoteException e) {
        throw new ECSystemException(
         ECMessage._ERR_REMOTE_EXCEPTION,
         this.getClass().getName(),
         "processSurvey");
      } catch (javax.ejb.CreateException e) {
        throw new ECSystemException(
         ECMessage._ERR_CREATE_EXCEPTION,
         this.getClass().getName(),
         "processSurvey");
      }

     }
    /// 區段 3 結束 ////////////////
```

15. 儲存變更。
16. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**建置專案**。

修改指令登錄

您必須修改指令登錄，以便在購物流程中使用新的實作類別。

如果要修改指令登錄，請執行下列步驟：

1. DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyPostUserRegistrationAddCmdImpl：
   a. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
   b. 從**工具**功能表中選取**工具設定**。
   c. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (**;**)。
   d. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to developmentDB user dbuser using dbpassword;
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Command for modified user bean tutorial',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼
- *FashionFlow_storeent_Id* 是您的商店的唯一商店實體 ID。

按一下**執行**圖示。

2. Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyPostUserRegistrationAddCmdImpl：
   a. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
   b. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
   c. 在**密碼**欄位中，輸入您的 Oracle 密碼。
   d. 在**主機字串**欄位中，輸入您的連接字串。
   e. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

      ```
      insert into CMDREG values (FashionFlow_storeent_Id,
      'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
      'Command for modified user bean tutorial',
      'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
      null, null, 'Local');
      ```

      按 Enter 鍵以執行 SQL 陳述式。
   f. 輸入下列以確定您的資料庫變更：

      ```
      commit;
      ```

      並按 Enter 鍵，以執行 SQL 陳述式。

修改 JSP 範本來收集和部署住家資訊

在這個步驟中，您需要修改 UserRegistrationAddForm 和 UserRegistrationUpdateForm 範本，使客戶可以在登入時輸入住家資訊，並且檢視摘要頁面中的住家資訊。修改這些頁面的策略是併入其他的 JSP 範本以指定頁面的新資訊。這些新頁面（UserRegistrationAddFormInclude.jsp 和 UserRegistrationUpdateFormInclude.jsp）使用 JSTL 來顯示新的資訊。

如果要修改這些頁面，請執行下列步驟：

1. 切換到「Web」視景。
2. 如果您尚未完成第 199 頁的第 10 章, 『指導教學：建立新商業邏輯』，就必須依照下列方式修改 Stores Web 專案的內容：
   a. 以滑鼠右鍵按一下 **Stores** Web 專案，然後選取**內容**。
   b. 在左窗格中選取 **Web**，然後從「可用的 Web 專案特性」清單中，選取**包含 JSP 標準標籤程式庫**。按一下**套用**。當更新完成時，按一下**確定**來關閉內容編輯程式。
3. 展開下列目錄：
   Stores\Web Content*FashionFlow_name*。
4. 執行下列步驟來建立 UserRegistrationAddForm.jsp 檔的備份：
   a. 展開 **UserArea>AccountSection>RegistrationSubsection** 目錄。
   b. 以滑鼠右鍵按一下 **UserRegistrationAddForm.jsp** 檔，然後選取**更名**。
   c. 在「更名」視窗中，輸入 `UserRegistrationAddForm_bak.jsp`，然後按一下**確定**。
   d. 在提示您是否要更新此檔案的鏈結時，請按一下**否**。
5. 執行下列步驟來建立 UserRegistrationUpdateForm.jsp 檔的備份：
   a. 展開 **UserArea>AccountSection>RegistrationSubsection** 目錄。
   b. 以滑鼠右鍵按一下 **UserRegistrationUpdateForm.jsp** 檔，然後選取**更名**。
   c. 在「更名」視窗中，輸入 `UserRegistrationUpdateForm_bak.jsp`，然後按一下**確定**。
   d. 在提示您是否要更新此檔案的鏈結時，請按一下**否**。
6. 以滑鼠右鍵按一下 *FashionFlow_name* 目錄，然後選取**匯入**。
   這時會開啓「匯入」精靈。
7. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
8. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
   yourDirectory\WC_SAMPLE_55.zip
   其中 *yourDirectory* 是您下載套件的目錄。
9. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
   - UserArea\AccountSection\RegistrationSubsection\ UserRegistrationAddForm.jsp
   - UserArea\AccountSection\RegistrationSubsection\ UserRegistrationAddFormInclude.jsp
   - UserArea\AccountSection\RegistrationSubsection\ UserRegistrationUpdateForm.jsp

- UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp

10. 在**資料夾**欄位中，已經指定 Stores/Web Content/*FashionFlow_name* 資料夾。請保留這個值。
11. 按一下**完成**。

如果您檢查 UserRegistrationAddForm.jsp 檔，您會發現此指導教學已經加入下面的區段：

```
<%-- 指導教學的新增項目 --%>
<tr>
      <td colspan="3">
         <jsp:include page="UserRegistrationAddFormInclude.jsp" flush="true" />
      </td>
</tr>
<%-- 指導教學結束 --%>
```

您也可以檢查 UserRegistrationAddFormInclude.jsp 檔來察看如何使用 JSTL 來收集新的資訊。同樣的，您可以檢查 UserRegistrationAddForm.jsp 和 UserRegistrationAddFormInclude.jsp 檔案。

您也必須匯入含有已修改 JSP 範本中使用的字串值的內容檔。這個檔案稱爲 Housing.properties。如果要匯入這個檔案，請執行下列步驟：

1. 在「J2EE 導覽器」檢視畫面中，展開下列目錄：
   Stores > Web Content> WEB-INF> classes> *FashionFlow_name* 目錄。
2. 以滑鼠右鍵按一下 *FashionFlow_name* 目錄，然後選取**匯入**。
   這時會開啓「匯入」精靈。
3. 從**選取匯入來源**清單中，選取 **Zip 檔**，然後按一下**下一步**。
4. 按一下**瀏覽**（在 **Zip 檔**欄位旁），並導覽至範例程式碼。這個檔案位於下列位置：
   yourDirectory\WC_SAMPLE_55.zip
   其中 *yourDirectory* 是您下載套件的目錄。
5. 按一下**取消全選**，然後展開目錄，選取下面的檔案來匯入。
   - Housing.properties
6. 在**資料夾**欄位中，已經指定
   Stores/Web Content/WEB-INF/classes/*FashionFlow_name* 資料夾。請保留這個值。
7. 按一下**完成**。

測試修改的程式碼

接下來，您必須測試修改後的登錄資訊，步驟如下：

1. 切換到「伺服器」視景。
2. 以滑鼠右鍵按一下 **WebSphereCommerceServer** 測試伺服器，然後選取**啓動**。
3. 以滑鼠右鍵按一下 Stores\Web Content*FashionFlow_name* 目錄下面的 **index.jsp**，然後選取**在伺服器上執行**。
4. 當商店的首頁開啓時，按一下**登錄**。
5. 再按一下**登錄**來建立新的使用者。
6. 這時會顯示一個修改後的登錄頁面，其中包含住家調查（如下面的螢幕擷取圖所示）：

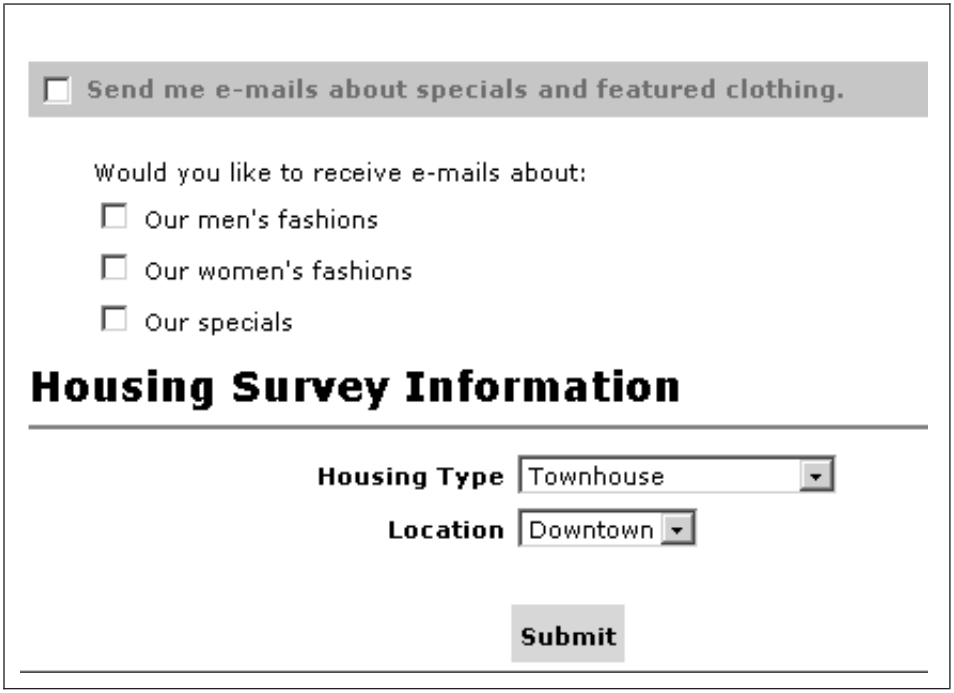

圖 53.

7. 輸入適當的新使用者資訊，然後按一下**提交**。
8. 在提交資訊之後，按一下**變更個人資訊**來驗證住家資訊已經擷取完成。這時會出現一個畫面，顯示您的調查回應摘要，內容如下：

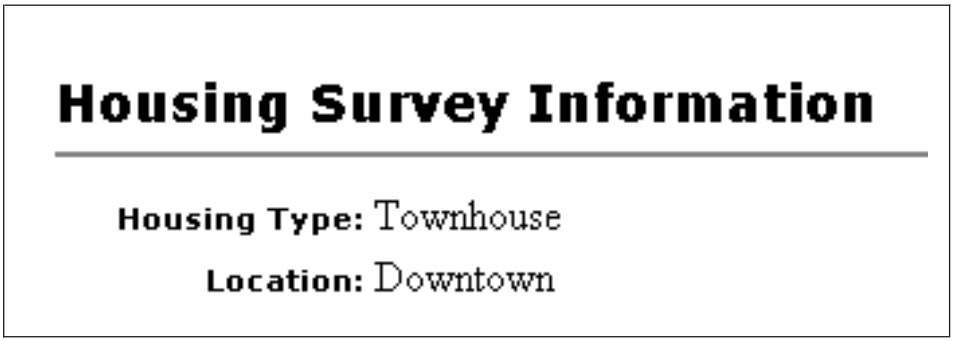

圖 54.

部署住家調查邏輯

本節說明如何將新商業邏輯部署到執行於遠端 WebSphere Commerce Server 的商店中。在您開始進行這些部署步驟前，您必須已經在遠端 WebSphere Commerce Server 上建立一家商店（以「流行館」範例商店爲基礎）。

部署程序包括：在開發機器上執行的步驟以及在目標 WebSphere Commerce Server 上執行的步驟。

有許多不同類型的資產必須部署到目標 WebSphere Commerce Server 上。這些作業包括：

- 指令邏輯
- 已修改的 Enterprise Bean 邏輯
- JSP 範本
- 內容檔
- 資料庫更新包括綱目更新（新表格）以及指令登錄更新

本節說明如何將所有的資產以*遞增的*方式部署到目標 WebSphere Commerce Server 上。

建立指令 JAR 檔

本節說明如何建立包含新的 MyPostUserRegistrationAddCmdImpl 邏輯的 JAR 檔。

如果要建立這個 JAR 檔，請在開發機器上執行下列步驟：

1. 在您的本端檔案系統上建立一個 *drive*:\ExportTemp4 目錄。

2. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
3. 以滑鼠右鍵按一下 **WebSphereCommerceServerExtensionsLogic** 專案，然後選取**匯出**。
   這時會開啓「匯出」精靈。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **JAR 檔**，然後按一下**下一步**。
   b. **選取要匯出的資源**下面的左窗格中，會預先移入專案的名稱。請保留這個值。
   c. 在右窗格中，確定只有選取下列資源：
      - .classpath
      - .project
      - .serverPreference
   d. 確定**匯出產生的類別檔和資源**已經選取。
   e. *請勿*選取**匯出 Java 原始檔和資源**。
   f. 在**選取匯出目的地**欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp4\WebSphereCommerceServerExtensionsLogic.jar。
      請注意，JAR 檔名必須是 WebSphereCommerceServerExtensionsLogic.jar。
   g. 按一下**完成**。

建立 EJB JAR 檔

如果要建立 EJB JAR 檔，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Member-MemberManagementData** 專案。
3. 按兩下 **EJB 部署描述子**。
4. 在選取「概觀」標籤之後，捲動至窗格底端，尋找 **WebSphere 連結**區段。
5. 在**資料來源 JNDI 名稱**欄位中，輸入目標 WebSphere Commerce Server 的資料來源 JNDI 名稱。以下是範例值：

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 DB2 資料庫，而 WebSphere Commerce 實例名稱是 “demo”

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   其中目標 WebSphere Commerce Server 是使用 Oracle 資料庫，而 WebSphere Commerce 實例名稱是 “demo”。

6. 儲存您的部署描述子變更 (Ctrl + S)。
7. 在「J2EE 導覽器」檢視畫面中，以滑鼠右鍵按一下 **Member-MemberManagementData** 專案，然後選取**匯出**。這時會開啓「匯出」精靈。
8. 在「匯出」精靈中，執行下列步驟：
   a. 選取 **EJB JAR 檔**，然後按一下**下一步**。
   b. **您要匯出的資源**？的值會預先移入 EJB 專案的名稱。請保留這個值。
   c. 在**您要將資源匯出至**？欄位中，輸入要使用的完整 JAR 檔名。就本例而言，請輸入 *drive*:\ExportTemp4\Member-MemberManagementData.jar。
   d. 按一下**完成**。
9. 在建立 JAR 檔之後，請還原在步驟 5 中對本端部署描述子所作的變更，來還原您的本端測試伺服器所需要的設定。

匯出商店資產

如果要匯出已修改的 JSP 範本和新的內容檔，請執行下列步驟：

1. 開啓 WebSphere Studio Application Developer 並切換到「J2EE 導覽器」檢視畫面。
2. 展開 **Stores** 資料夾。
3. 以滑鼠右鍵按一下 **Web Content** 資料夾，然後選取**匯出**。這時會開啓「匯出精靈」。
4. 在「匯出」精靈中，執行下列步驟：
   a. 選取**檔案系統**，然後按一下**下一步**。
   b. 選取要部署下列資源：
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddForm.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddFormInclude.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateForm.jsp
      - Web Content*FashionFlow_name*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
      - Web Content\WEB-INF\classes*FashionFlow_name*\Housing.properties
   c. 選取**為選取的檔案建立目錄結構**。

d. 在「目錄」欄位中，輸入要放置這些資源的暫時目錄。例如，輸入 C:\ExportTemp4

e. 按一下**完成**。

將資產轉送到您的目標 WebSphere Commerce Server

在這個步驟中，您會在目標 WebSphere Commerce Server 上建立一個暫時目錄，然後將您的住屋調查資產複製到這個目錄中。在後續的步驟中，您會將不同類型的程式碼放置到 WebSphere Commerce 應用程式內的適當位置中。

如果要將檔案從您的開發機器複製到目標 WebSphere Commerce Server，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 上，建立一個叫做 *drive*:\ImportTemp4 的暫時目錄。
2. 決定您要如何將檔案從某一部電腦複製到另一個電腦上。您可以將目標 WebSphere Commerce Server 上的磁碟機對映到開發機器上，或者使用 FTP 應用程式（如果已經配置）來進行複製。
3. 從開發機器上，將 *drive*:\ExportTemp4 的內容複製到目標 WebSphere Commerce Server 上的 *drive*:\ImportTemp4。

停止您的目標 WebSphere Commerce Server

在開始部署步驟前，您應該停止目標 WebSphere Commerce Server，方法是在指令行發出 stopServer 指令。有關這個指令的明細，請參閱*WebSphere Commerce Studio 安裝手冊*。

更新您的目標 WebSphere Commerce Server 上的資料庫

建立 XHOUSING 表格

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以建立表格：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令行工具 > 指令中心**），然後按一下 **Script 編寫**標籤。
2. 在「Script」視窗中，輸入下列指令：

```
connect to developmentDB user dbuser using dbpassword;
create table XHOUSING (MEMBERID bigint not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼

按一下「執行」圖示。
您應該會看到一則訊息，指出 SQL 陳述式已經順利完成。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以建立表格：

1. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
2. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
3. 在**密碼**欄位中，輸入您的 Oracle 密碼。
4. 在**主機字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
create table XHOUSING (MEMBERID number not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

   並按 Enter 鍵，以執行 SQL 陳述式。現在已經建立 XHOUSING 表格。
6. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

登錄作業指令

如果要修改指令登錄，請執行下列步驟：

1. DB2 如果您使用 DB2 資料庫，請執行下列步驟來登錄 MyPostUserRegistrationAddCmdImpl：
   a. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
   b. 從**工具**功能表中選取**工具設定**。
   c. 選取**使用陳述式終端字元**勾選框，並確定所指定的字元爲分號 (;)。
   d. 在選取 Script 標籤下，於 Script 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
connect to developmentDB user dbuser using dbpassword;
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

其中

- *developmentDB* 是您的開發資料庫名稱
- *dbuser* 是資料庫使用者
- *dbpassword* 是您的資料庫使用者的密碼
- *FashionFlow_storeent_Id* 是您的商店的唯一商店實體 ID。

按一下**執行**圖示。

2. Oracle 如果您使用 Oracle 資料庫，請執行下列步驟來登錄 MyPostUserRegistrationAddCmdImpl：
   a. 開啓 Oracle SQL Plus 指令視窗（**開始 > 程式集 > Oracle > 應用程式開發 > SQL Plus**）。
   b. 在**使用者名稱**欄位中，輸入您的 Oracle 使用者名稱。
   c. 在**密碼**欄位中，輸入您的 Oracle 密碼。
   d. 在**主機字串**欄位中，輸入您的連接字串。
   e. 在 SQL Plus 視窗中輸入下列 SQL 陳述式：

```
insert into CMDREG values (FashionFlow_storeent_Id,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

   按 Enter 鍵以執行 SQL 陳述式。
   f. 輸入下列以確定您的資料庫變更：

```
commit;
```

   並按 Enter 鍵，以執行 SQL 陳述式。

更新您的目標 WebSphere Commerce Server 上的商店資產

在這個步驟中，您會使用已修改的商店資產來更新商店，方式如下：

1. 備份您的 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear\ Stores.war 目錄（其中 *cellName* 通常是您的機器的主機名稱）。
2. 導覽至 *drive*:\ImportTemp4\Stores\Web Content 目錄。

3. 將 *FashionFlow_name* 和 WEB-INF 資料夾複製到下列目錄中：
   WAS_installdir\installedApps*cellName*\WC_*instanceName*.ear\Stores.war

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新您的目標 WebSphere Commerce Server 上的指令 JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的指令 JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 WebSphereCommerceServerExtensionsLogic.jar 檔的副本，並將它儲存在備份位置中。
2. 將新的 WebSphereCommerceServerExtensionsLogic.jar 檔從 *drive*:\ImportTemp4 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中

其中 *instanceName* 是您的 WebSphere Commerce 實例的名稱。

更新您的目標 WebSphere Commerce Server 上的 EJB JAR 檔

在這個步驟中，您會更新目標 WebSphere Commerce Server，以使用新的 EJB JAR 檔，方式如下：

1. 您應該爲現有的 JAR 檔製作備份副本，方式如下：
   a. 導覽至 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄。
   b. 製作 Member-MemberManagementData.jar 檔的副本，並將它儲存在備份位置中。
2. 將新的 Member-MemberManagementData.jar 檔從 *drive*:\ImportTemp4 目錄複製到 *WAS_installdir*\installedApps*cellName*\WC_*instanceName*.ear 目錄中
3. 接下來，您必須修改 EJB 部署描述子資訊，方式如下：
   a. 尋找這個 WebSphere Application Server 資料格的部署儲存庫（META-INF 目錄）。其格式通常如下：
      WAS_installdir\config\cells*cellName*
      \applications\WC_*instance_name*.ear\deployments\
      WC_*instance_name**EJBModuleName*.jar\META-INF。
      以下是這項資訊的特定範例：
      D:\WebSphere\AppServer\config\cells\myCell\applications\
      WC_demo.ear\deployments\WC_demo\
      Member-MemberManagementData.jar\META-INF
      其中
      - myCell 是 WebSphere Application Server 資料格的名稱

- demo 是 WebSphere Commerce 實例的名稱
- Member-MemberManagementData 是自訂 EJB 模組的名稱

b. 目錄包含下列檔案：

- ejb-jar.xml
- ibm-ejb-access-bean.xmi
- ibm-ejb-jar-bnd.xmi
- ibm-ejb-jar-ext.xmi
- MANIFEST.MF

備份所有的檔案。

c. 使用工具來開啓新的 Member-MemberManagementData.jar 檔並檢視其內容。

d. 將 meta-inf 目錄的內容從這個 Member-MemberManagementData.jar 檔擷取到步驟 3a 中的目錄裡。

4. 在指令行中，使用 WebSphere Application Server startServer 指令來重新啓動您的 WebSphere Commerce 實例。

驗證目標 WebSphere Commerce Server 上的住家調查邏輯

在這個步驟中，您將執行下列步驟來驗證已經順利部署到目標 WebSphere Commerce Server 的住家調查邏輯：

1. 開啓 Web 瀏覽器，然後輸入 URL 來啓動您根據「流行館」範例商店所建立的商店。
2. 當商店的首頁開啓時，按一下**登錄**。
3. 再按一下**登錄**來建立新的使用者。
4. 這時會顯示一個修改後的登錄頁面，其中包含住家調查（如下面的螢幕擷取圖所示）：

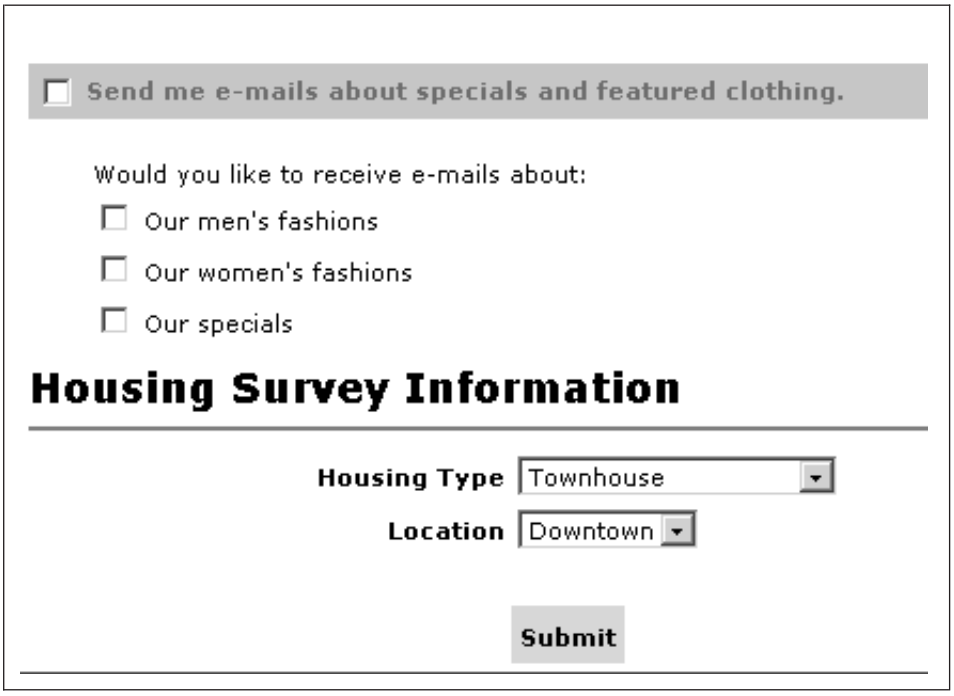

圖 55.

5. 輸入適當的新使用者資訊，然後按一下**提交**。
6. 在提交資訊之後，按一下**變更個人資訊**來驗證住家資訊已經擷取完成。這時會出現一個畫面，顯示您的調查回應摘要，內容如下：

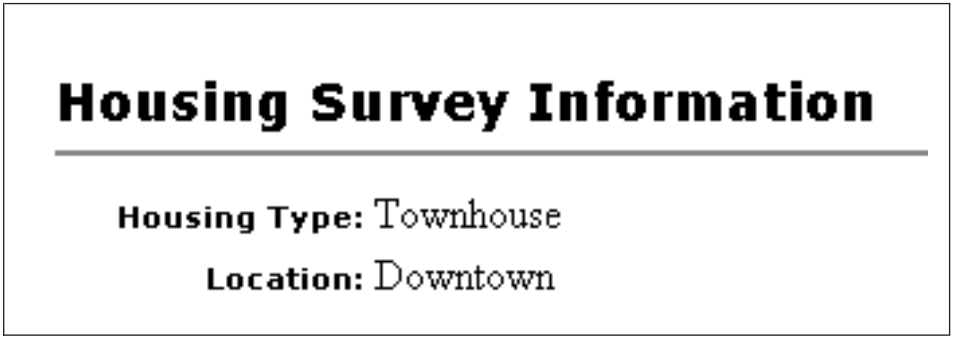

圖 56.

第 5 篇 附錄與後記

附錄 A. 在 WebSphere Commerce Studio 中配置 WebSphere Commerce 元件追蹤

本附錄說明如何在 WebSphere Commerce 開發環境中執行各種 WebSphere Commerce 元件時啓用它們的追蹤。如果要啓用元件追蹤，請執行下列步驟：

1. 必要時，請開啓 WebSphere Commerce 開發環境（**開始 > 程式集 > IBM WebSphere Commerce Studio > WebSphere Commerce 開發環境**）。
2. 切換至「伺服器」視景。
3. 在「伺服器」檢視畫面中，以滑鼠右鍵按一下 **WebSphereCommerceServer**，然後選取**停止**（如果伺服器目前正在執行中）。
4. 在「伺服器配置」檢視畫面中，展開**伺服器配置**資料夾。
5. 按兩下 **WebSphereCommerceServer**。
   這時會開啓「WebSphereCommerceServer」編輯程式。
6. 選取「追蹤」標籤。
7. 選取**啓用追蹤**。
8. 在**追蹤字串**文字框中，指定應該啓用追蹤的元件。使用 WebSphere JRas 延伸追蹤日誌程式 ID 值，後面接著 "=all=enabled"。如需這些值的完整清單，請參閱 *WebSphere Commerce 管理手冊*的「配置」主題。您應該使用冒號 (:) 將多個元件隔開。舉例來說，如果要同時啓用 SERVER 和 RAS 元件的追蹤，請依照下列方式指定追蹤字串：

   ```
   com.ibm.websphere.commerce.WC_SERVER=all=enabled:
      com.ibm.websphere.commerce.WC_RAS=all=enabled
   ```

   請注意，換行只是爲了方便顯示。
9. 儲存您的變更 (Ctrl+S)。

輸出檔

在預設的情況下，輸出日誌檔是稱爲 activity.log。這個檔案是位於下列目錄中：
workspace_dir\.metadata\.plugin\com.ibm.etools.server.core\tmp0\logs

由於 activity.log 檔是二進位檔，因此會使用「日誌分析程式」來加以讀取。一旦啓用元件追蹤，WebSphere JRas 也會用純文字格式將日誌項目隨著追蹤項目一併寫入到追蹤輸出檔中。

有關在 WebSphere Commerce Studio 中配置「日誌分析程式」的資訊，請參閱 *WebSphere Commerce Studio 安裝手冊*。

另外，訊息也會顯示在 WebSphere Studio Application Developer 的「主控台」檢視畫面中。

附錄 B. 其他相關資訊的位置

您可以從多個來源取得不同格式的 WebSphere Commerce Studio 系統和其元件的其他相關資訊。下面提供您有哪些可用的資訊以及存取方式。

WebSphere Commerce Studio 資訊

以下是 WebSphere Commerce Studio 資訊的來源：

- 『WebSphere Commerce Studio 線上說明』
- 第 360 頁的『WebSphere Commerce 網站』
- 第 360 頁的『WebSphere Developer Domain』
- 第 360 頁的『IBM 紅皮書』

WebSphere Commerce Studio 線上說明

WebSphere Commerce Studio 線上資訊是您在 WebSphere Commerce Studio 中建立及公佈商店時的主要資訊來源。

如果要檢視 WebSphere Commerce Studio 線上說明，請執行下列步驟：

1. 啓動 WebSphere Commerce Studio，方法是選取**開始 → 程式集 → IBM WebSphere Commerce Studio → WebSphere Commerce 開發環境**。
2. 從**說明**功能表，選取**說明內容**。

註: 如果您參閱 『WebSphere Commerce Studio 線上說明』中的多重平台的指示，請確定您遵循 WebSphere Commerce Studio 的指示。當說明頁面包含多重平台的資訊時， WebSphere Commerce Studio 特定的資訊會以下列圖示來指示：

如果沒有 WebSphere Commerce Studio 特定的資訊，請確定您遵循的是 Windows 特定的指示。當說明頁面包含多重平台的指示資訊時， Windows 的指示會以下列圖示來指示：

WebSphere Commerce 網站

WebSphere Commerce Studio 產品資訊可以在 WebSphere Commerce 網站中取得。請造訪下列 URL，以取得其他產品資訊：

`http://www.ibm.com/software/webservers/commerce/library/`

WebSphere Developer Domain

有關 WebSphere Commerce Studio 以及 WebSphere Commerce 的其他資訊，也可以在 WebSphere Developer Domain 的 WebSphere Commerce Zone 取得：

`http://www.ibm.com/websphere/developer/zones/commerce/`

IBM 紅皮書

WebSphere Commerce Studio 和 WebSphere Commerce 資訊可以在 IBM Redbooks™ 網站中取得：

`http://www.ibm.com/redbooks`

WebSphere Studio Application Developer 資訊

以下是 WebSphere Studio Application Developer 的資訊來源：

- 『WebSphere Studio Application Developer 線上說明』
- 『WebSphere Studio Application Developer 網站』
- 第 361 頁的『WebSphere Developer Domain』
- 第 361 頁的『IBM 紅皮書』

WebSphere Studio Application Developer 線上說明

WebSphere Studio Application Developer 線上說明是您在 WebSphere Studio Application Developer 中執行作業時的主要資訊來源。

如果要檢視 WebSphere Studio Application Developer 線上說明，請執行下列步驟：

1. 啓動 WebSphere Commerce Studio，方法是選取**開始 → 程式集 → IBM WebSphere Studio → Application Developer 5.0**。
2. 從**說明**功能表，選取**說明內容**。

WebSphere Studio Application Developer 網站

WebSphere Studio Application Developer 產品資訊可以在 WebSphere Studio Application Developer 網站中取得：

`http://www.ibm.com/software/ad/studioappdev/library/`

WebSphere Developer Domain

有關 WebSphere Studio Application Developer 的其他資訊，可以在 WebSphere Developer Domain 的 WebSphere Studio Zone 的 WebSphere Studio Application Developer 頁面中取得：

`http://www.ibm.com/websphere/developer/zones/studio/appdev/`

IBM 紅皮書

WebSphere Studio Application Developer 資訊可以在 IBM Redbooks 網站中取得：

`http://www.ibm.com/redbooks`

注意事項

本資訊是針對 IBM 在美國所提供之產品與服務開發出來的。而在其他國家中，IBM 不見得有提供本書中所提的各項產品、服務、或功能。 要知道在您所在之區是否可用到這些產品與服務時，請向當地的 IBM 服務代表查詢。亦不表示該產品、程式或服務爲唯一的選擇。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。不過，其他非 IBM 產品、程式、或服務在運作上的評價與驗證，其責任屬於使用者。

IBM 可能已經申請與本書有關（包括本書的主題內容）的各項專利權，或者具有正在審理中的專利權申請書。本書使用者並不享有前述專利之任何授權。您可以用書面方式來查詢授權，來函請寄到：

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

如果要查詢有關二位元組（DBCS）資訊的授權事宜，請聯絡您國家的 IBM 智慧財產部門，或者用書面方式寄到：

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

下列段落若與該國之法律條款抵觸，即視為不適用：

IBM 就本書僅提供「交付時之現況」保證，而並不提供任何明示或默示之保證，如默示保證書籍之適售性或符合客戶之特殊使用目的；有些地區在某些固定的交易上並不接受明示或默示保證的放棄聲明，因此此項聲明不見得適用於您。

本資訊中可能會有技術上或排版印刷上的訛誤。因此，IBM 會定期修訂；並將修訂後的內容納入新版中。同時，IBM 會隨時改進並（或）變動本書中所提及的產品及（或）程式。

本資訊中所提及的任何非 IBM 網站只是供您參考，並無爲這些網站背書之意。這些網站中的教材不屬於此 IBM 產品的相關教材，若使用這些網站則風險自行負責。

IBM 對您以任何方式提供的資訊隨時享有使用或公開權，且不須負法律責任。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊者可洽詢 IBM。其下列資訊指的是： (1) 獨立建立的程式與其他程式 (包括此程式) 之間更換資訊的方式 (2) 相互使用已交換之資訊方法若有任何問題請聯絡：

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue, Markham, Ontario L6G 1C7
Canada

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於雙方之「IBM 客戶合約」、「國際程式授權合約」或任何同等合約之條款，提供本文件中所述之授權程式與其所有適用的授權資料。

任何此處涵蓋的執行效能資料都是在一個受控制的環境下決定出來的。因此，若在其他作業環境下，所得的結果可能會大大不同。有些測定已在開發階段系統上做過，不過這並不保證在一般系統上會出現相同結果。再者，有些測定可能已透過推測方式評估過。但實際結果可能並非如此。本書的使用者應依自己的特定環境，查證適用的資料。

本書所提及之非 IBM 產品資訊，取自產品的供應商，或其公佈的聲明或其他公開管道。IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性、或任何對產品的其他主張是否完全無誤。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

有關 IBM 未來動向的任何陳述，僅代表 IBM 的目標而已，並可能於未事先聲明的情況下有所變動或撤回。

所有顯示的 IBM 售價都是 IBM 的建議零售價，這些都是目前的售價且隨時會不經過通知即變更。經銷商提供的價格可會有不同。

此資訊僅供規劃用。因此在產品尚未上市前此資訊仍有變更的可能。

本資訊中的範例包含了用於日常商業活動的資料及報告。爲了儘可能使說明更加完整，這些範例包括個人、公司、廠牌和產品的名稱。所有名稱純屬虛構，如有雷同純屬巧合。

著作權授權：

本書包含範例應用程式的來源語言，用來說明在各種作業平台上的程式設計技術。客戶不需付費給 IBM 即可以任何格式複製、修改或散布這些範例程式，作爲開發、使用、行銷或散布符合範例程式針對的目標作業平台應用程式設計介面的應用程式。這些範例並未在所有條件下進行完整測試。 IBM 不保證或暗示這些程式的可靠性、有用性或功能。客戶不需付費給 IBM 即可以任何格式複製、修改或散布這些範例程式，作爲開發、使用、行銷或散布符合 IBM 應用程式設計介面的應用程式。

這些範例程式或任何衍生程式的全部或部分拷貝，都必須包括下列版權聲明：

如果您看到的是本資訊的軟本。其圖片和顏色圖例可能不會出現。

商標及服務標示

IBM 標誌與下列詞彙爲 International Business Machines Corporation 在美國及（或）其他國家或地區的商標或註冊商標：

400
AIX
AS/400
DB2
DB2 Universal Database
@server
IBM
iSeries
WebSphere

Windows 是 Microsoft Corporation 在美國與（或）其他國家或地區的商標或註冊商標。

Java 和所有 Java 相關的註冊商標和標示是 Sun Microsystems, Inc. 在美國與（或）其他國家或地區的商標或註冊商標。

其他公司、產品及服務名稱可能是其他者的商標或服務標記。

索引

索引順序以中文字，英文字，及特殊符號之次序排列。

讀者意見表

爲使本書盡善盡美，本公司極需您寶貴意見；懇請您閱讀後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號（ V ）；我們會在下一版中，作適當修訂，謝謝您的合作！

評估項目	評 估 意 見		備 註
正 確 性	內容說明與實際程序是否符合	□是□否	
	參考書目是否正確	□是□否	
一 致 性	文句用語及風格，前後是否一致	□是□否	
	實際產品介面訊息與本書中所提是否一致	□是□否	
完 整 性	是否遺漏您想知道的項目	□是□否	
	字句、章節是否有遺漏	□是□否	
術語使用	術語之使用是否恰當	□是□否	
	術語之使用，前後是否一致	□是□否	
可 讀 性	文句用語是否通順	□是□否	
	有否不知所云之處	□是□否	
內容說明	內容說明是否詳盡	□是□否	
	例題說明是否詳盡	□是□否	
排版方式	本書的形狀大小，版面安排是否方便閱讀	□是□否	
	字體大小，顏色編排，是否有助於閱讀	□是□否	
目錄索引	目錄內容之編排，是否便於查找	□是□否	
	索引語錄之排定，是否便於查找	□是□否	
	※評估意見爲"否"者，請於備註欄提供建議。		

其他：（篇幅不夠時，請另外附紙說明。）

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。

註： 您也可將寶貴的意見以電子郵件寄至 NLSC01@tw.ibm.com，謝謝。

IBM WebSphere Commerce

程式設計手冊與指導教學

5.5 版

折疊線

台北市105敦化南路一段2號4樓

大臺灣中華研發中心

國際商業機器股份有限公司

軟體國際部 啟

寄件人 姓名：

地址：

（免貼郵票）

廣告回信

台灣北區郵政管理局登記證

北台字第00176號

寄

折疊線

讀者意見表