IBM WebSphere Commerce

程式設計手冊

第 5.4 版

GC40-0809-00

IBM WebSphere Commerce

程式設計手冊

第 5.4 版

GC40-0809-00

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（2002 年 3 月），版次 2

本版適用於下列產品：

- IBM® WebSphere® Commerce Business Edition for Windows NT® 與 Windows® 2000 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Business Edition for AIX® 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Business Edition for Solaris Operating Environment Software 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Business Edition for Linux 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Studio, Business Developer Edition for Windows NT and Windows 2000 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Professional Edition for Windows NT and Windows 2000 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Professional Edition for AIX 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Professional Edition for Solaris Operating Environment Software 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Professional Edition for Linux 5.4 版（程式編號 5724-A18）
- IBM WebSphere Commerce Studio, Professional Developer Edition for Windows NT and Windows 2000 5.4 版（程式編號 5724-A18）

此外，亦適用於上述產品之所有後續版次與修正層次，除非新版中另有提及。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。下列地點恕不供應。

IBM 歡迎您提供意見。您可採下列一種方法傳送您對本書的意見：

1. 傳送到下列的電子郵件位址：

   torrcf@ca.ibm.com

2. 郵寄到下列地址：

   IBM Canada Ltd. Laboratory
   B3/KB7/8200/MKM
   8200 Warden Avenue
   Markham, Ontario, L6G 1C7
   Canada

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

*IBM WebSphere Commerce 程式設計手冊*提供 WebSphere Commerce 架構與程式設計模型相關資訊。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
- 指令的施行
- 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：

建立新商業邏輯

此指導教學會示範如何按照 WebSphere Commerce 程式設計模型，來建立新指令、資料 Bean 與 Enterprise Bean。此外，亦會示範如何將邏輯整合到現有的商店中，並將程式碼部署到目標 WebSphere Commerce Server 中。

修改與延伸現有的商業邏輯

本指導教學分成兩節討論。第一節示範如何新增新邏輯到現有的控制程式指令中。第二節示範如何修改現有的作業指令與 WebSphere Commerce 實體 Bean。此外，亦會示範如何將這些修改整合到現有的商店中，並將程式碼部署到目標 WebSphere Commerce Server 中。

本書使用的慣例

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。

等寬字型表示您必須輸入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」-- 亦即提供其它資訊以協助您完成作業。

Windows 表示 WebSphere Commerce for Windows NT 與 Windows 2000 專用的資訊。

AIX 表示 WebSphere Commerce for AIX 專用的資訊。

Solaris 表示 WebSphere Commerce for Solaris™ 作業環境軟體專用的資訊。

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

Linux 表示 WebSphere Commerce for Linux 專用的資訊。

DB2 表示 DB2® Universal Database 專用的資訊

Oracle 表示 Oracle® 專用的資訊

Professional 表示 WebSphere Commerce Professional Edition 專用的資訊。

Business 表示 WebSphere Commerce Business Edition 專用的資訊。

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™
- Enterprise JavaBeans (EJB) 元件架構
- JavaServer Pages 技術
- HTML
- 資料庫技術
- VisualAge® for Java, Enterprise Edition 3.5 版

其它相關資訊的位置

本書日後可能有所更新。請檢查下列 WebSphere Commerce 網站取得更新資訊：

Business

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

Professional

http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html

更新版中可能含有新資訊、新增或更新過的指導教學、以及指導教學的相關範例程式碼。

目錄

第 1 篇 概念與結構

第 1 章 概觀

WebSphere Commerce 的軟體元件

在瞭解 WebSphere Commerce Server 如何運作之前，查看與自訂程序有關的軟體元件圖相當有助益。下圖顯示這些軟體產品的概要檢視：

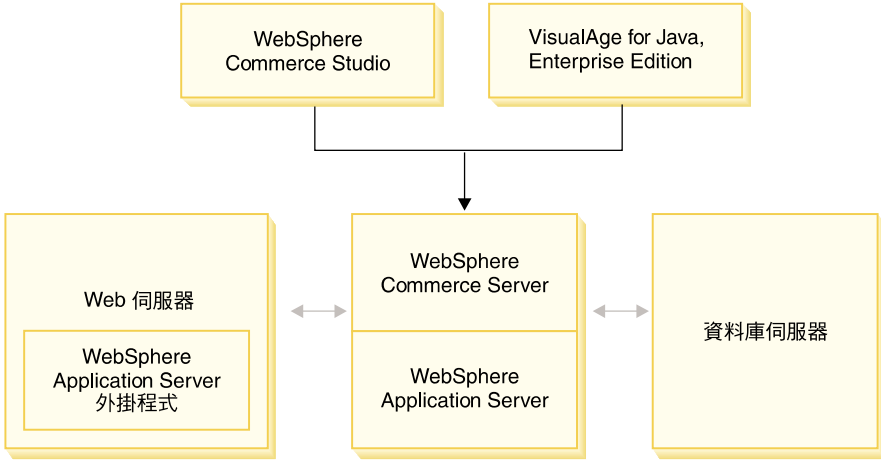


圖 1.

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。爲了有效與 WebSphere Application Server 通信，它採用了 WebSphere Application Server 外掛程式。

WebSphere Commerce Server 是在 WebSphere Application Server 中執行，可讓它善用了應用程式伺服器中的眾多特性。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。

程式開發人員可利用下列兩種主要工具來建立自訂的商業邏輯：WebSphere Commerce Studio 與 VisualAge for Java Enterprise Edition。WebSphere Commerce Studio 用來建立與管理商店門面資產（例如：JSP 範本）。VisualAge for Java Enterprise Edition 是讓您使用 Java 來建立新商業邏輯，以延伸現有的功能或建立全新功能。如果您的應用程式需要延伸資料庫綱目，則資料庫開發人員應使用慣用的資料庫開發工具來建立新表格。

WebSphere Commerce 應用程式結構

到目前爲止，您已看完各種和自訂有關的軟體元件如何相互配合，這對瞭解應用程式的結構來說是很重要的。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：

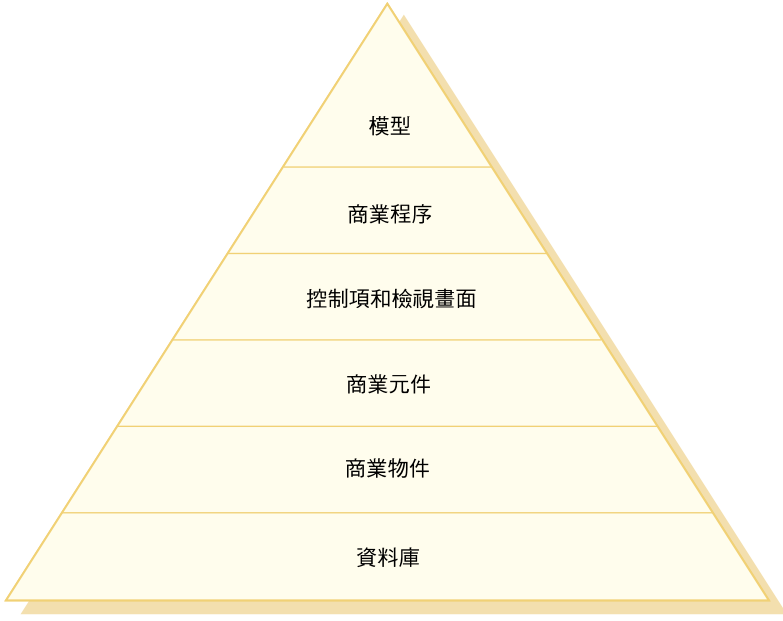


圖 2.

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：

- 使用者
- 訂單
- 產品

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。

這些實體 Bean 扮演著商業應用程式與資料庫間的介面。此外，實體 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施行。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施行以及要使用的檢視畫面。施行方式可視商店而定。

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 範本來施行。舉例來說，像 ProductDisplay（會傳回產品頁面，其中顯示購物者所選產品的相關資訊）與 OrderPrepare（在購物者面前顯示一份套表，以提交適當的訂單資訊）即爲檢視畫面。

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：

使用者登錄

此商業程序包含一些商業元件（例如，會建立新使用者之登錄記錄的 UserRegistrationAdd 指令）以及在登錄使用者過程中所有涉及步驟的相關檢視畫面。

型錄導覽

此商業程序包含一些商業元件（例如，分別會顯示商店型錄與型錄中之種類的 StoreCatalogDisplay 與 CategoryDisplay 指令）以及在導覽型錄過程中所有涉及步驟的相關檢視畫面。

模型 整體而言，圖中的低階層構成了電子商務商業模型。電子商務企業運作模式的其中一個範例就是「時尚館」範例商店所用的大眾消費型（business-to-consumer，B2C）商務模式。另一個範例是「工具屋」範例商店所用的企業消費型商務（business-to-business，B2B）模式。

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施行的。

以下是 WebSphere Commerce 執行期間結構的主要元件：

- Servlet 引擎

- 通信協定接收程式
- 配接器管理程式
- 配接器
- Web 控制程式
- 指令
- 實體 Bean
- 資料 Bean
- 資料 Bean 管理程式
- 顯示頁面
- XML 檔

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

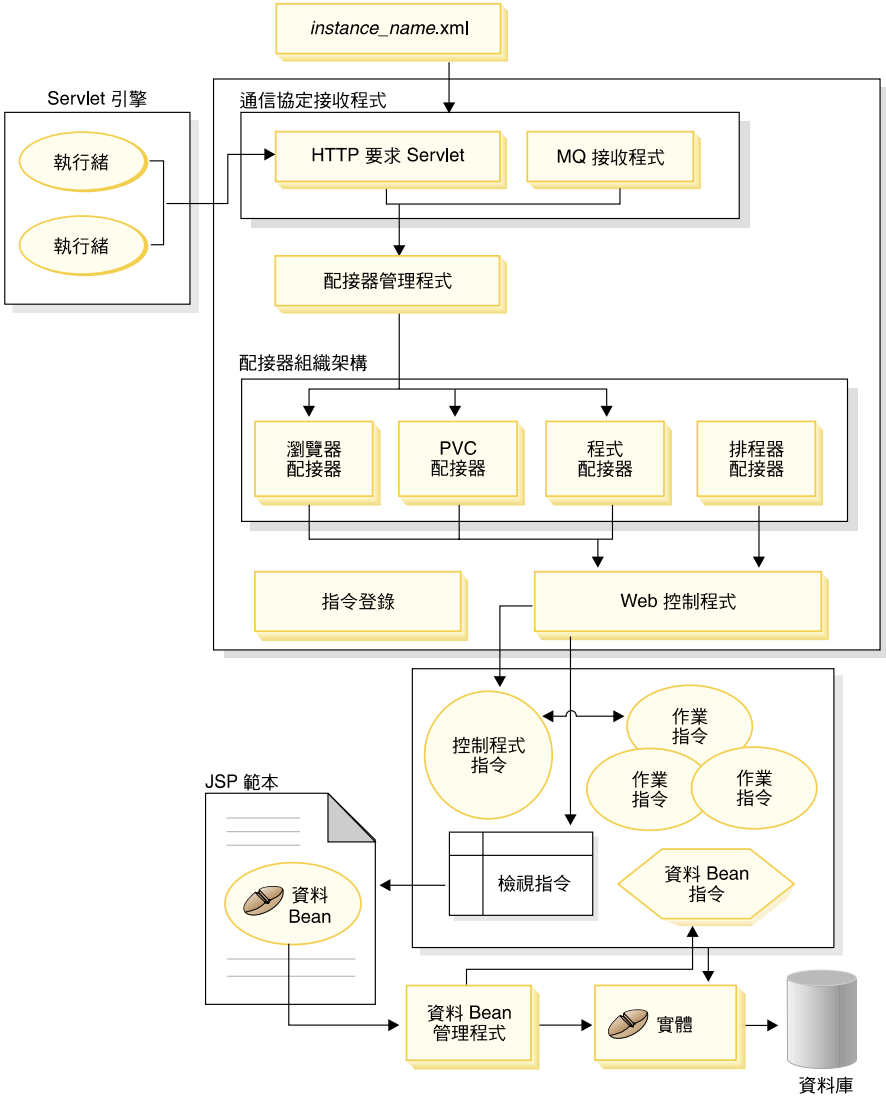


圖 3.

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。

舊版的 WebSphere Commerce 採用 C++ 應用程式伺服器，亦即藉由維護一組預先配置的系統程序，而在 URL 要求方面施行本身的作業分派程式。此種採用系統程序的舊模型所用到的資源會比採用 Java 執行緒的新模型來得密集。在新版 WebSphere Commerce 執行期間結構中，則藉由利用 Servlet 引擎，而提供一種更具伸縮性的商務解決方案。

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。

通信協定接收程式

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信協定。通信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，並根據所用的通信協定將這些要求發送到適當的配接器。通信協定接收程式包括：

- 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。

當起始設定「要求 Servlet」時，它會讀取 *instance_name*.xml 架構檔。XML 檔其中一個架構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信協定接收程式。

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 配接器組織架構的施行類別階層。

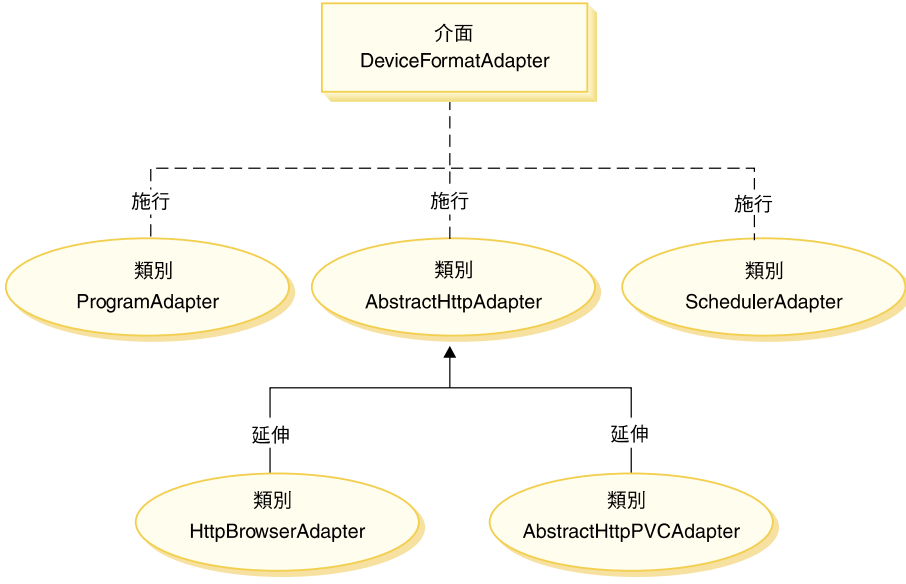


圖 4.

如上圖所示，所有配接器皆施行 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，並使用訊息映射器將要求轉換成 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。
- 建立連接新通信協定接收程式的新配接器。這種新配接器必須施行 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施行，而檢視邏輯必須用檢視指令來施行。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從屬站，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。舉例來說，控制程式指令包括用來處理訂單的 *OrderProcessCmd* 指令，以及用來建立新登錄使用者的 *UserRegistrationAddCmd* 指令。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施行類別，並執行該檢視畫面指令。

作業指令

作業指令是施行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，共同施行某個 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 指令是在某個資料 Bean 案例化時，由 JSP 範本所呼叫。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，作爲從屬站要求的回應。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信協定（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信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。

當使用重新導向通信協定時，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從屬站。

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。
- 從屬站直接要求檢視畫面。
- 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從屬站。

WebSphere Commerce 實體 Bean

實體 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，實體 Bean 直覺代表 WebSphere Commerce 資料。亦即，您不用全盤瞭解資料庫綱目，便可以從實體 Bean（密切仿造商務領域中的概念與物件）中存取資料。您可以延伸現有的實體 Bean。此外，您可以根據應用程式特有的商業需求，來完整部署新實體 Bean。

實體 Bean 是根據 Enterprise JavaBeans (EJB) 元件模型來施行。

有關實體 Bean 的詳細資訊，請參閱第 43 頁的『施行 WebSphere Commerce 實體 Bean』。

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

當透過 WebSphere Studio Page Designer 在 JSP 範本中插入一個 WebSphere Commerce 資料 Bean 時，則在執行期間將會呼叫資料 Bean 管理程式，以產生一行程式碼以移入到資料 Bean 中。

以下是 Page Designer 中的程式碼範例：

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

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。從屬站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 JSP 範本的保護與其他資源的相關資訊，請參閱第 79 頁的第 4 章, 『存取控制』。

Instance_name.xml 架構檔

Instance_name.xml 架構檔負責設定案例的架構資訊。在起始設定「要求 Servlet」時會讀取此架構檔。

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。

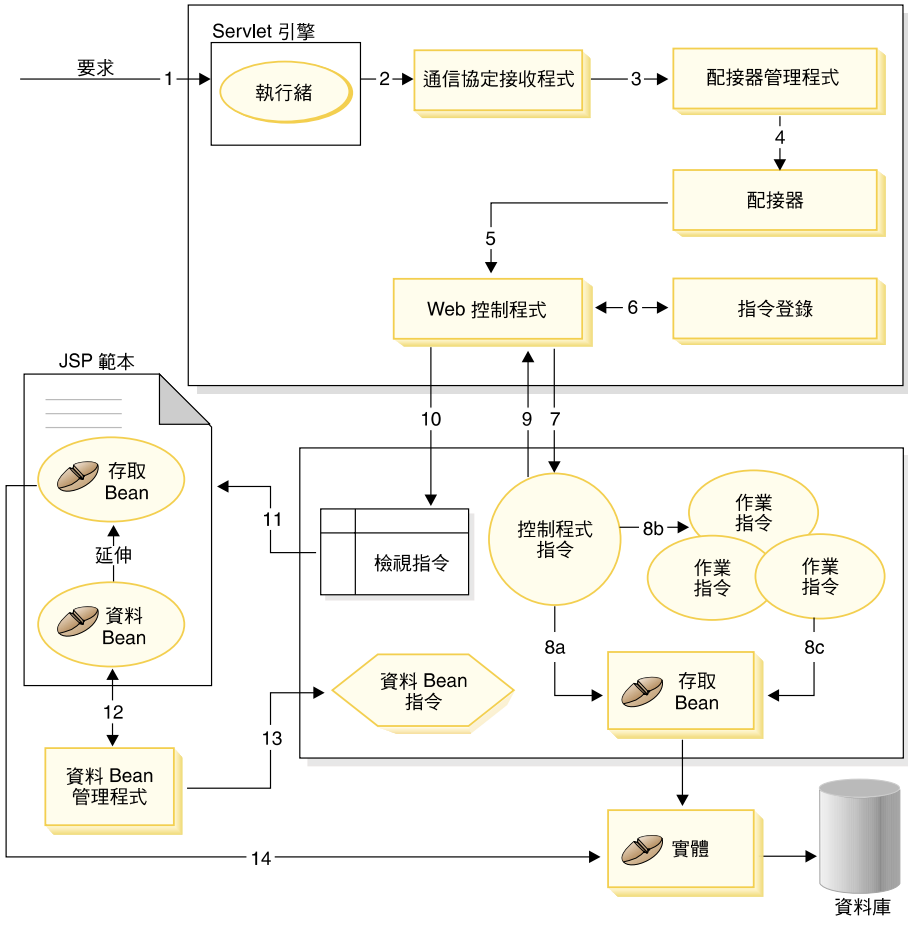


圖 5.

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信協定接收程式。通信協定接收程式可爲 HTTP 要求 Servlet 或 MQ 接收程式。
3. 通信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，其執行途徑可能有：
   a. 控制程式指令可使用存取 Bean 與對應的實體 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 與其對應的實體 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施行方式。
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 會使用其對應的實體 Bean 來存取資料庫。

舊版中之自訂間的主要差異

從 WebSphere Commerce Suite 5.1 版起，WebSphere Commerce Server 完全以 Java 寫成。因此，您必須使用 Java 來自訂功能。這和 WebSphere Commerce Suite 4.1 版（以及舊版 Net.Commerce™）中所用的模型相當不同，在這些版本中自訂時是採用 C++ 與 Net.Data® 巨集。

下表摘要說明主要的變更。

	採用 C++ 的 4.1 版（與其他舊版）	**採用 Java 的 5.1 版（以及後續版本）**
應用程式模型	指令	控制程式指令
	作業	作業指令
	可改寫函數	作業指令
	檢視作業	檢視畫面指令
	錯誤作業	檢視畫面指令
顯示模型	Net.Data 巨集	JSP 範本、資料 Bean、資料 Bean 指令以及檢視畫面指令
持續性模型	Net.Data 與 ODBC	實體 Bean
URL 分派程式	WebSphere Commerce C++ URL 分派程式	WebSphere Servlet 引擎
作業模型	系統程序	Java 執行緒

	採用 C++ 的 4.1 版（與其他舊版）	採用 Java 的 5.1 版（以及後續版本）
指令配接器	無	Web 控制程式與配接器

本書不說明移轉程序。有關移轉資訊，請參閱 *WebSphere Commerce 移轉手冊*。

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

「模式-檢視畫面-控制程式」設計型樣

「模型-檢視畫面-控制程式（MVC）」設計型樣用以指定一個由資料模型、呈現資訊與控制資訊組成的應用程式。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。

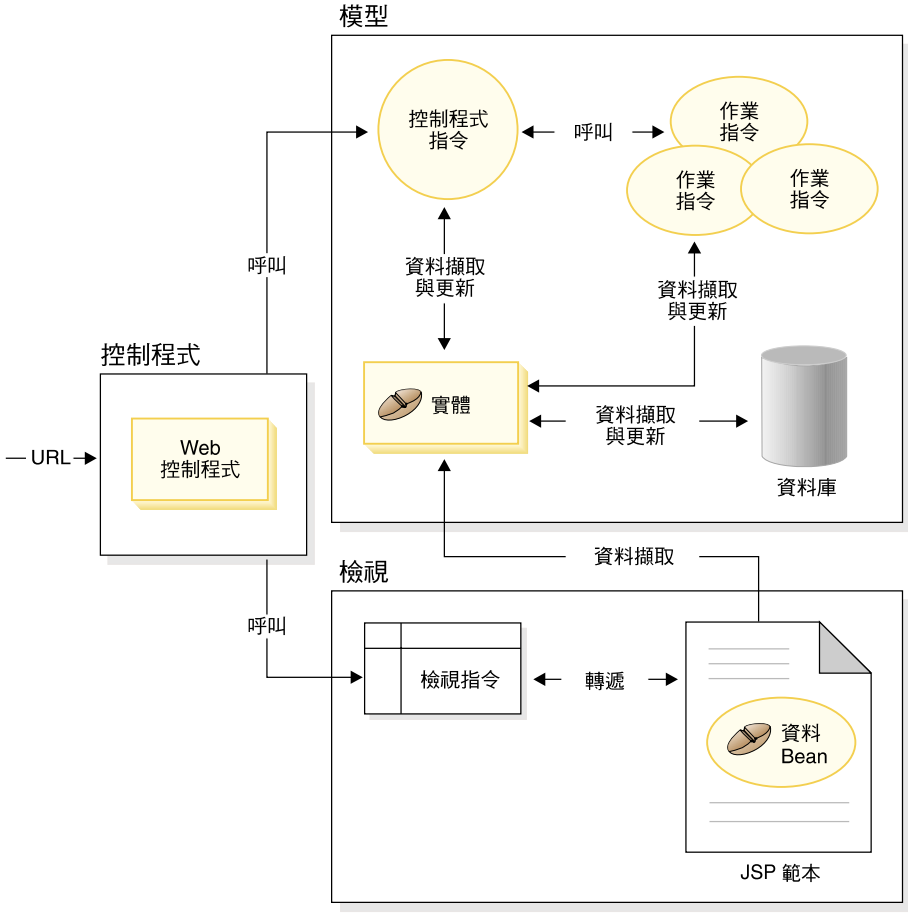


圖 6.

指令設計型樣

WebSphere Commerce Server 接受瀏覽器型小型從屬站（Thin Client）應用程式以及其它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`）與一個施行類別（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。

而從指令施行端的角度來看，指令會遵循 WebSphere 指令組織架構，以施行標準指令設計型樣，並容許呼叫端與施行間存在一個間接層。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，以針對不同商店施行不同的指令。
3. 能根據要求端的裝置類型，執行不同的檢視畫面施行。

下圖顯示四種主要指令類型的介面概觀：

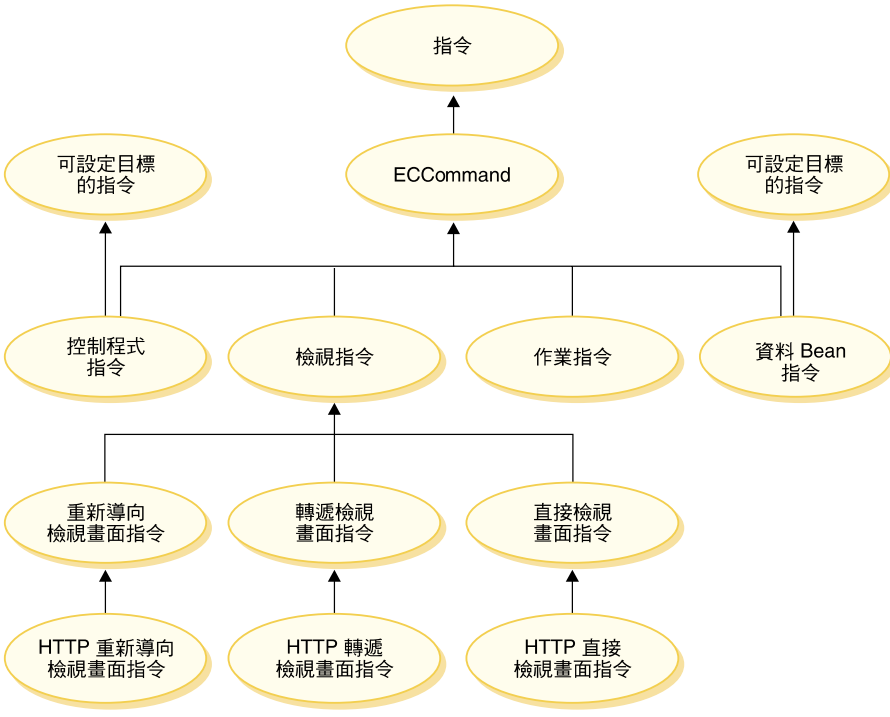


圖 7.

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。舉例來說，控制程式指令包括用來處理訂單的 *OrderProcessCmd* 指令，以及用來建立新登錄使用者的 *UserRegistrationAddCmd* 指令。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作業的適當施行類別，並執行檢視作業。

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

作業指令是施行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，共同施行某個 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 案例化時，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，作爲從屬站要求的回應。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。
- 從屬站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從屬站。

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信協定（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信協定，應會傳回此種檢視畫面類型的檢視畫面指令。當使用重新導向通信協定時，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從屬站。

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，用來案例化指令。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（瞭解所要案例化的施行類別）之物件的案例化。

Factory 會提供一種智慧型方法來案例化新物件。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施行類別。一旦案例化，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。

指定指令之施行類別的方法有兩種。預設的施行類別可藉由 `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"
```

指定施行類別的第二種方法是使用 WebSphere Commerce 指令登錄。當施行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 26 頁找到。

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

使用指令 Factory 的語法如下：

```
cmd = CommandFactory.createCommand(interfaceName, commandContext.getStoreId())
```

其中 *interfaceName* 爲新指令 Bean 的介面名稱， getStoreId 方法則用以決定要使用該指令的商店。

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 159 頁的『呼叫新商業原則』。

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。

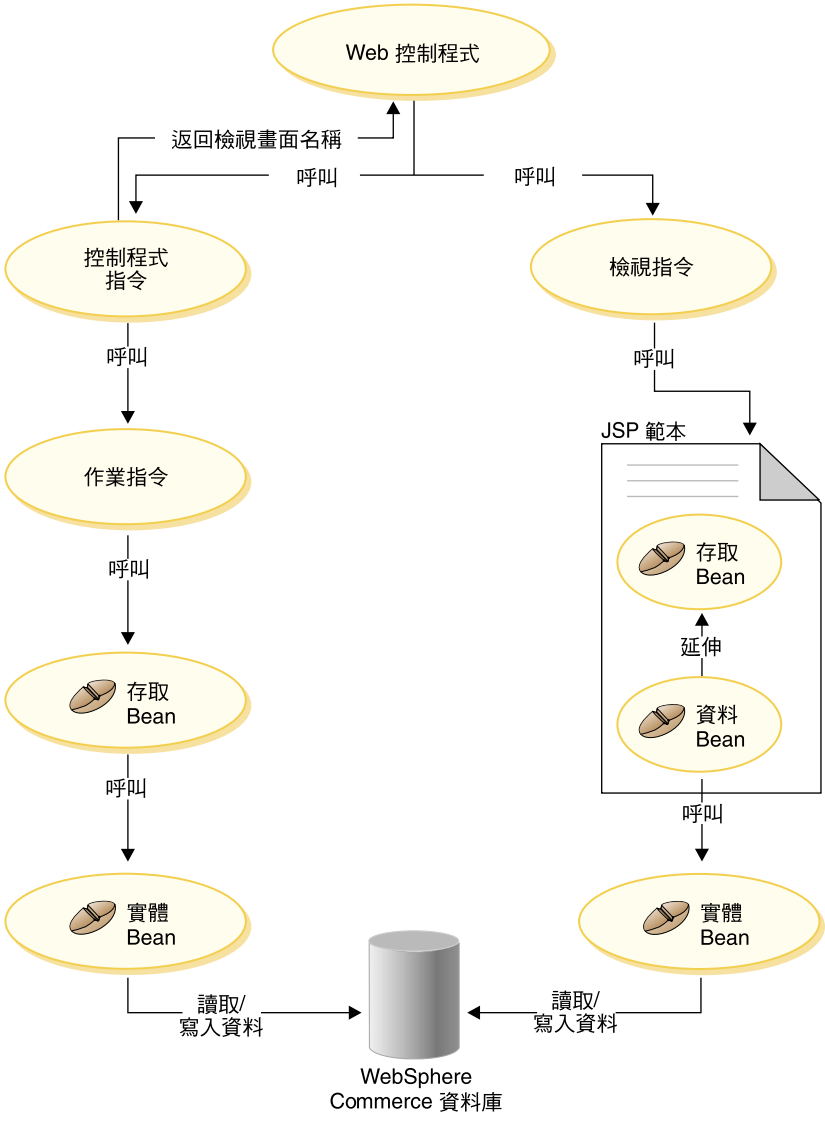


圖 8.

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施行類別並呼叫之。

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 移往實體 Bean，而這類實體 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 範本以顯示回應給從屬站。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）會呼叫其對應的實體 Bean。當從 JSP 範本間接存取時，實體 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 144 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）映射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&catalogId=catalog_Id&langId=-1
```

當 URI 與介面名稱間採一對一映射時，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 表格中的正確施行類別。

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 ProductDisplay
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.commands.GetProductDisplay.TemplateCmd
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 表格中查看施行類別名稱。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映射至其施行類別的機制。介面提供多種施行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	指令施行類別名稱	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。舉例來說，下列 SQL 陳述式是建立 MyNewCommand 的項目，並供特定商店使用（其商店識別碼爲 5）：

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

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。

如果您所撰寫的指令一律使用同樣的施行類別，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 屬性來指定施行類別。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施行類別，您便無法將預設內容傳遞給施行類別，且必須將同一施行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施行方式。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：

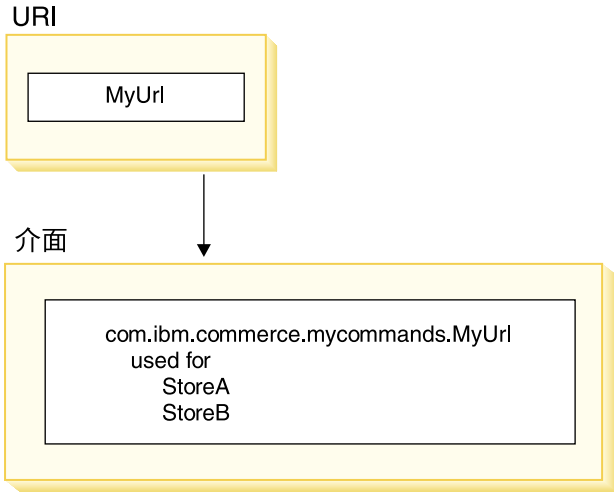


圖 9.

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 介面的施行類別爲 com.ibm.commerce.mycommands.MyUrlStoreAImpl。而在商店 B 方面，同一介面的施行類別爲 com.ibm.commerce.mycommands.MyUrlStoreBImpl。下圖顯示此項流程：

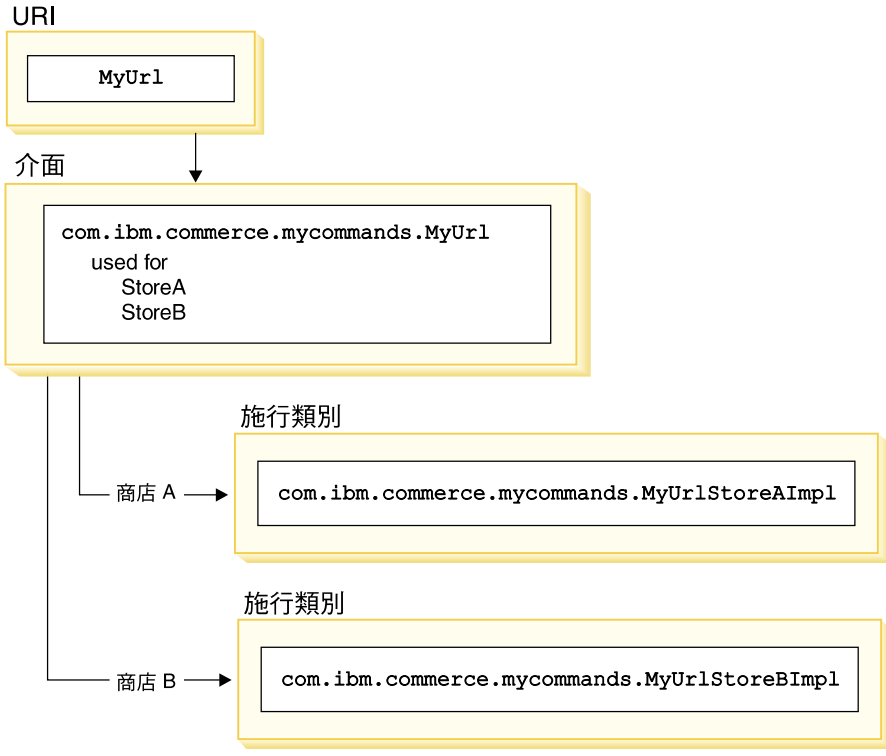


圖 10.

VIEWREG 表格

VIEWREG 表格容許登錄裝置特有之檢視畫面指令的施行方式。透過此表格，可爲檢視畫面指令登錄多種施行方式。如此一來，指令組織架構便能傳回不同的檢視畫面給各種從屬站。

當控制程式指令傳回或在異常狀況中指定檢視畫面指令名稱時， Web 控制程式會由 VIEWREG 表格來判斷檢視畫面指令類別。可有多個檢視畫面指令名稱映射至同一施行類別。

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	檢視畫面指令的施行類別名稱	可使用預設施行方式。
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指令時，您可能得在 VIEWREG 表格中建立對應的項目。只要符合下列情況之一，即必須將檢視畫面指令登錄在 VIEWREG 表格中：

- 檢視畫面指令是在存取控制下執行
- 該檢視畫面指令有多種施行方式
- 在 PROPERTIES 直欄中設定內容

已登錄的檢視畫面指令可透過指令登錄以檢視畫面名稱來存取，或直接使用實際的顯示檔名稱來存取。而未登錄在 VIEWREG 表格中的檢視畫面，則只能在從屬站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，從屬站可使用指令名稱，或將指令名稱換成實際的顯示檔名稱，以存取檢視畫面。 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
```

如果從屬站有直接呼叫已登錄檢視畫面的可能（採用顯示檔名稱），則您必須如同本範例所示（MyView 與 MyView.jsp）使用與實際顯示檔名稱同名的檢視畫面名稱來登錄指令。

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 SQL 陳述式範例會建立 *MyNewViewCommand* 的項目，並供特定商店使用：

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

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

下表提供另一個範例 VIEWREG 表格與主要資訊：

COMMAND - NAME	DEVICE - FMT_ID	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplayView	BROWSER	轉遞檢視畫面指令	HttpForwardViewCommandImpl	
InterestItemAddView	BROWSER	重新導向檢視畫面指令	HttpRedirectViewCommandImpl	docname =item.jsp
InterestItemDeleteView	BROWSER	重新導向檢視畫面指令	HttpRedirectViewCommandImpl	docname =item.jsp
GenericApplication Error	BROWSER	重新導向檢視畫面指令	HttpRedirectViewCommandImpl	docname =usererr.jsp
GenericSystemError	BROWSER	重新導向檢視畫面指令	HttpRedirectViewCommandImpl	docname =syserr.jsp
Logon	BROWSER	轉遞檢視畫面指令	HttpForwardViewCommandImpl	docname= logon.jsp & storeDir=no

註: COMMANDNAME、INTERFACENAME、 CLASSNAME 與 PROPERTIES 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。 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，使用不同的施行類別。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。
- InterestItemAdd 與 InterestItemDelete 指令分別傳回 *InterestItemAddView* 與 *InterestItemDeleteView* 檢視畫面名稱給 Web 控制程式。這兩種指令皆要求使用重新導向檢視畫面，因此，這兩個檢視畫面的檢視畫面指令介面名稱皆爲 RedirectViewCommand。另外已替這兩個檢視畫面登錄一個共通的 JSP 範本。Web 控制程式會提取內容 (docname=item.jsp)，並將之傳遞給檢視畫面指令 (HttpRedirectViewCommandImpl)。

- 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。以上表爲例，這會導致 `usererr.jsp` 範本的顯示。
- 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。以上表爲例，這會導致 `syserr.jsp` 範本的顯示。
- 瀏覽器從屬站可藉由輸入登入 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從屬站。一般而言，顯示頁面採 JSP 範本方式施行（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 實體 Bean。資料 Bean 概括了一些可從實體 Bean 中擷取或設於實體 Bean 中的內容。因此，資料 Bean 等於簡化了將動態資料納入 JSP 範本中的工作。

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(DataBean, HTTPServletRequest)
```

當將 WebSphere Commerce 資料 Bean 插入 JSP 範本時， WebSphere Studio Page Designer 會自動在其前產生一行程式碼。

商店程式開發人員在開發 JSP 範本時，應將商店內容與啓用多種文化的問題納入考慮。有關啓用多種文化的詳細資訊，請參閱 WebSphere Commerce 線上說明。

JSP 範本與資料安全考量

藉由特定的編碼練習以熟悉 JSP 範本以及資料 Bean 的用法，可將某些蓄意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。您可使用參數插入方式來收集執行期間輸入資訊。

以下是使用參數插入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 應是從對應實體 Bean 的存取 Bean 延伸而來，且會施行 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」實體 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，且會施行 com.ibm.commerce.CommandDataBean 介面。

資料 Bean 介面

資料 Bean 會施行下列之一或所有的 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。藉由施行多種介面，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: 施行 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 通常是從對應實體 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 是從實體 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，會自動將對應實體 Bean 中的所有屬性移入到智慧型資料 Bean 中。不過，假設實體 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：

- 施行方式較爲簡單，且不需撰寫資料 Bean 指令。
- 當實體 Bean 中新增新欄位時，並不需要在資料 Bean 中進行變更。在實體 Bean 被修改後，必須重新產生存取 Bean（採用 VisualAge for Java 中的工具）。一旦重新產生存取 Bean，即會自動提供所有新屬性給智慧型資料 Bean 使用。
- 實體 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：

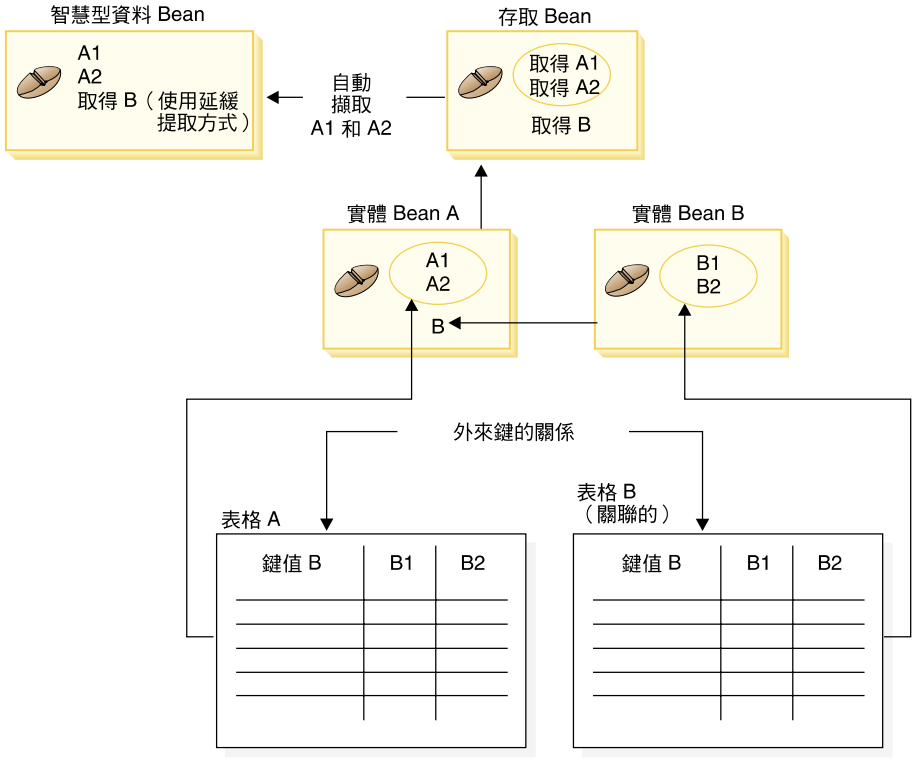


圖 11.

有關施行「延緩提取」擷取方式的詳細說明，請參閱第 39 頁的『延緩提取資料擷取方式』。

CommandDataBean 介面: 施行 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 必須施行 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: 施行 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它資料的主要鍵欄位。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，WebSphere Studio 會針對要透過 Page Designer 插入頁面中的每一個資料 Bean 產生下列程式碼：

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean, HTTPServletRequest);
```

BeanInfo 類別

資料 Bean 若少了會施行 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 施行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。

有關施行 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 範本呼叫控制程式指令，和將邏輯與顯示區隔開來並不一致，不過，您仍有可能會遇到需要採行此方式的情況。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。

一旦執行控制程式指令，您可以執行檢視畫面。

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 的對應實體 Bean。實體 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。它們會施行將傳回給從屬站的商業邏輯與產品資料，而不含有從屬站的相關資訊。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.xxx.NewView		docName=NewView.jsp

VPd: docName=NewView.jsp

URL: http://hostname.com/NewCommand?storeID=1&....

CCPu: storeID=1&...

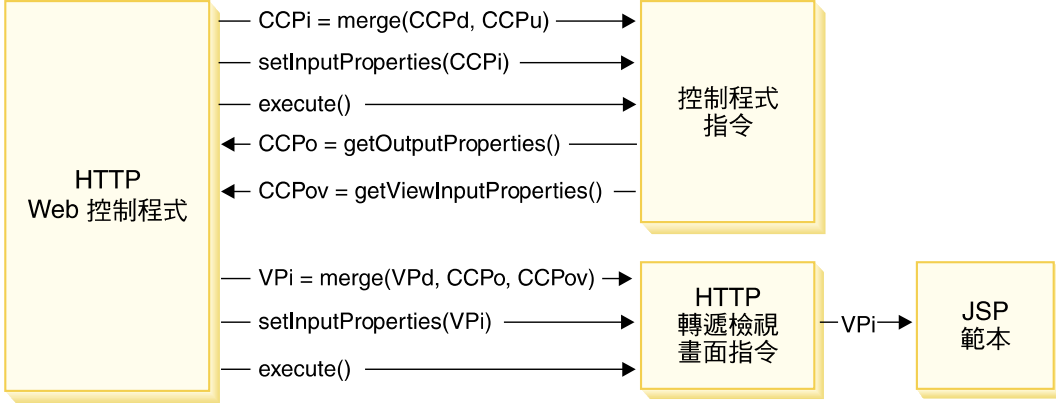


圖 12.

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。有關此方法的詳細資訊，請參閱 WebSphere Commerce 線上說明中的「參照」區段。

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 = productDisplay.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。

  當使用 DirectView 指令時，您可選擇性地設定 `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 520 個以上的表格。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) 元件架構爲基礎的實體 Bean 作爲持續性物件層。這些實體 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。

VisualAge for Java 企業版提供可支援此組織架構之開發的更先進 EJB 工具與單元測試環境。

施行 WebSphere Commerce 實體 Bean

WebSphere Commerce 實體 Bean - 概觀

一如上述，WebSphere Commerce 結構中的持續層是根據 EJB 元件架構來施行的。EJB 架構定義兩種 Enterprise Bean 類型：實體 Bean 與階段作業 Bean。實體 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 實體 Bean 皆屬於 CMP 實體 Bean。有少數無狀態式階段作業 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 實體 Bean 的優點之一是開發人員可利用 VisualAge for Java 企業版中所提供的 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映射。這些工具會自動爲實體 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。

VisualAge for Java 提供兩項現行 EJB 規格的延伸特性：EJB 繼承特性與連結特性。 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 實體 Bean 間的關係。

有些 WebSphere Commerce 實體 Bean 會利用到 EJB 繼承特性。WebSphere Commerce 實體 Bean 不會用到 VisualAge for Java 所提供的連結特性。在您開發自己的實體 Bean 時，建議您不要使用 VisualAge for Java 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 VisualAge for Java 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 分成下列 EJB 群組：

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

上述某些 EJB 群組中含有階段作業 Bean。爲了簡化日後的移轉工作，您最好不要修改階段作業 Bean 類別。必要時您可以在新 EJB 群組中建立新階段作業 Bean。有關建立新階段作業 Bean 的詳細資訊，請參閱第 67 頁的『撰寫新階段作業 Bean』。

WebSphere Commerce Enterprise Bean 的部署描述子

部署描述子是一種已序列化的特殊類別，內含 Enterprise Bean 的執行期間設定。WebSphere Commerce Enterprise Bean 的部署描述子是以特定方式設定，而不應修改。

在您建立新的 Enterprise Bean（實體或階段作業 Bean）時，請在 VisualAge for Java 的 EJB 部署環境中設定部署描述子；方法是以滑鼠右鍵按一下 Bean，並選取**內容**。新 Enterprise Bean 的部署描述子應遵循與 WebSphere Commerce Enterprise Bean 之部署描述子相同的使用慣例。尤其請確定您是按如下來設定屬性：

屬性	**值**
交易屬性	TX_REQUIRED
隔離層次	TRANSACTION_READ_COMMITTED
執行模型	SYSTEM_IDENTITY 或 CLIENT_IDENTITY
重入	請確定未選取此項。

Enterprise Bean 中通常含有一些只會讀取資料庫中資訊但永不會執行資料庫更新的方法。這些方法即所謂的*唯讀*方法。因此您應明確標示所有的唯讀方法（以滑鼠右鍵按一下該方法，並選取 **EJB 方法屬性 > 唯讀方法**）。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。

隔離層次

WebSphere Commerce Enterprise Bean（而非 VisualAge for Java）所用的交易隔離層次爲 TRANSACTION_READ_COMMITTED。請注意， DB2 JDBC 驅動程式與 Oracle JDBC 驅動程式在此種隔離層次的施行上有所差異。因此，當部署到 WebSphere Application Server 環境時，所用的交易隔離層次會因所用的資料庫而有所不同。

當在 WebSphere Test Environment 外運作時，DB2 資料庫所用的隔離層次爲 TRANSACTION_REPEATABLE_READ。當在 WebSphere Test Environment 外運作時，Oracle 資料庫所用的隔離層次爲 TRANSACTION_READ_COMMITTED。

如果您要部署到 DB2 資料庫，則不需用手動變更交易隔離層次；在部署步驟期間如果您發出 modifyIsolationLevel 指令時即會加以變更。

下表顯示 DB2 與 Oracle 交易隔離層次和其對應 JDBC 交易隔離層次間的映射。

JDBC	**DB2**	**Oracle**
讀取尚未確定	讀取尚未確定	讀取尚未確定
讀取已確定	游標穩定性	（不適用）

JDBC	DB2	Oracle
可重複讀取	讀取穩定性	讀取已確定
可序列化	可重複讀取	可序列化
無	（不適用）	（不適用）

就 DB2 中的「游標穩定性」交易隔離層次而言，唯有在給定交易期間有所更新之列才會被專門鎖定。如果給定列中沒有直欄更新，即使它是 SQL 陳述式所傳回之結果集中的一部份，只要游標移離到另一列上，即會釋放該特定列的鎖定。在某些情況下（如：更新庫存），並不希望發生此行爲。因此，藉由在 DB2 中將交易隔離層次改爲「讀取穩定性」，同時採取些微的取捨，將可大大提昇資料的完整性。

假設 Oracle 中只有「讀取已確定」交易隔離層次或 JDBC「可重複讀取」同義的交易隔離層次可用，則會採用相當類似於 DB2 中之「可重複讀取」交易隔離層次的行爲，來執行實際的施行。

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
- 撰寫新實體 Bean
- 撰寫新無狀態式階段作業 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它屬性。這可藉由下列方法之一來完成：

不修改現有的 WebSphere Commerce 公用實體 Bean

建立新資料庫表格，然後爲該表格建立一個新實體 Bean。視需要在該實體 Bean 中新增欄位與方法，以操作新屬性。針對新實體 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，它會案例化存取 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 公用實體 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 公用實體 Bean 中建立新欄位，並使用次要表格映射，將欄位映射至新表格中的對應直欄。新增任何必要的方法。針對現有實體 Bean 重新產生部署程式碼與存取 Bean。當應用程式案例化存取 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。在您檢查 WebSphere Commerce 程式碼儲存庫時，您會發現 WCSUser EJB 群組中有一個 ″Demographics″（個人背景資訊）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 表格互動的新實體 Bean，或者在 Demographics Bean 中新增新欄位（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 實體 Bean，並將其欄位映射至 USERRES 表格中的直欄。當應用程式需要客戶的住家類型時，它必須案例化一個 Userres 存取 Bean 物件並擷取資料。如果應用程式同時需要其它的個人背景資訊時，亦必須案例化 Demographics 存取 Bean 物件，並擷取其它任何必要的屬性。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，以案例化新存取 Bean 以及原始的存取 Bean。下圖顯示此種延伸物件模型方法：

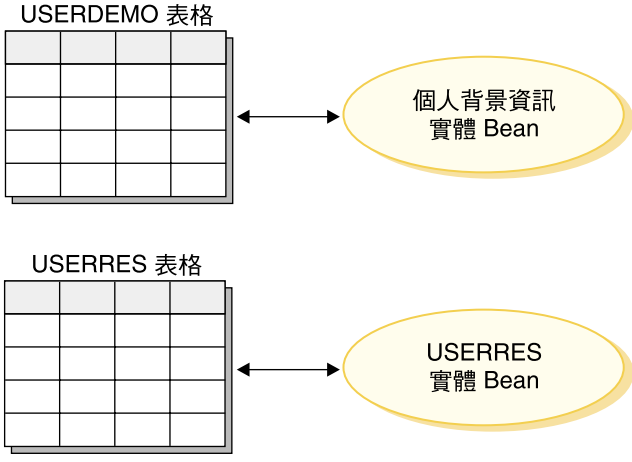


圖 13.

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，您應爲原來的現有實體 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 施行實務：

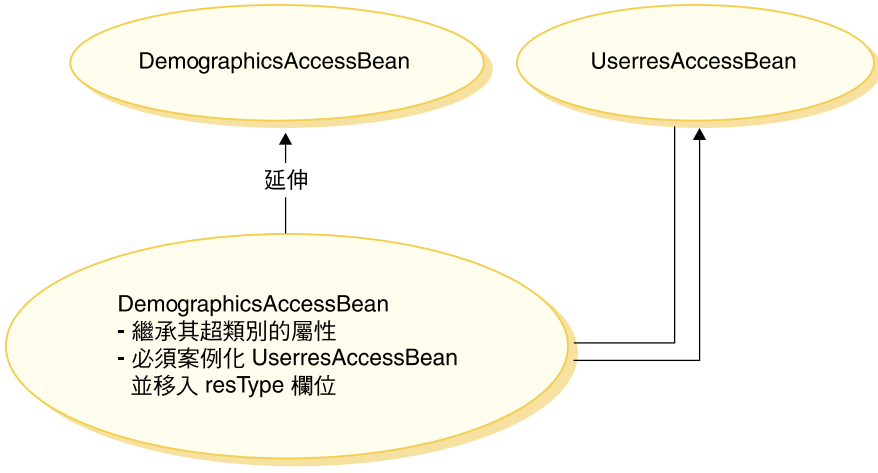


圖 14.

如果您使用第二種方法（修改現有程式碼），則您將新增新欄位到 Demographics 實體 Bean 中，並在新欄位與 USERRES 表格中的適當直欄間，建立次要的表格映射。當應用程式需要客戶的住家類型時，它會案例化 Demographics 存取 Bean 物件並擷取住家類型。如果應用程式需要客戶其它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：

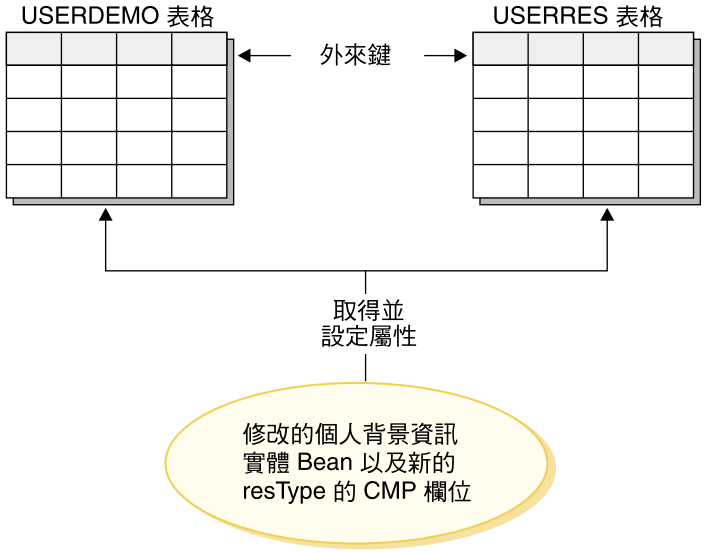


圖 15.

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，只需案例化單一實體 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，當發行 WebSphere Commerce 程式碼儲存庫新版次時，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，但當您匯入新的程式碼儲存庫時，將不會保留您新增到 Enterprise Bean 之欄位與新表格之間的映射資訊。因此，當您要移轉至 WebSphere Commerce 程式碼儲存庫新版次時，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 VisualAge for Java 中的工具，比較程式碼自訂版本和 WebSphere Commerce 程式碼新版次。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，用手動方式重新映射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。

階段作業 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) 實體 Bean。CMP 實體 Bean 是一種分散、持續和交易式的伺服器端 Java 元件，且可使用 VisualAge for Java 中提供的工具產生。在眾多情況中，就物件持續性而言，CMP 實體 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它「物件對關聯性」映射選項。基於這些原因，WebSphere Commerce 已使用 CMP 實體 Bean 來施行核心商務物件。

不過，在某些情況下，建議您使用階段作業 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 陳述式執行資料庫密集作業時。這可稱爲*任意 SQL* 情況。

下列各節將進一步詳細說明。

請注意，如果階段作業 Bean 是做爲 JDBC 外層用，以擷取資料庫中的資訊，則較難施行資源層次的存取控制。當在此種方式下使用階段作業 Bean 時，階段作業 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，建議您使用無狀態階段作業 Bean，並在該階段作業 Bean 中，建立一個 Finder 方法，以執行和實體 Bean 中之 Finder 方法相同的功能。亦即，無狀態階段作業 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，案例化存取 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 是由階段作業 Bean 中的 Finder 方法所傳回，或是由實體 Bean 中的 Finder 方法傳回。因此，在階段作業 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階段作業 Bean 或實體 Bean 中的 Finder 方法。它對於程式設計模型中的其它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 join，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，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)
```

VisualAge for Java 中的 EJB 工具不支援這種深度提取概念。相反地，它採用延緩提取方式，因而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，建議您使用階段作業 Bean。在該階段作業 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：

- 執行 SQL select 陳述式以進行深度提取
- 針對主要表格中的每一列以及每一個相關物件，各案例化一個存取 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，階段作業 Bean Finder 方法會針對 CUSTOMER 表格中的每一列各案例化一個 CustomerAccessBean，以及針對 ADDRESS 表格中的每一個對應列各案例化一個 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（名稱、年齡與地址）。請見下圖：

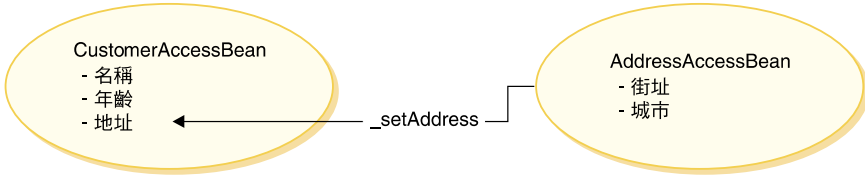


圖 16.

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實體 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，可建立一個階段作業 Bean helper 方法。此階段作業 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。

延伸公用實體 Bean

本節說明 WebSphere Commerce 公用實體 Bean 的設計型樣。此設計型樣可讓您進行延伸，像是新增新持續性欄位、新商業方法或新 Finder 方法。

下圖顯示 Catalog（型錄）實體 Bean 的施行類別。

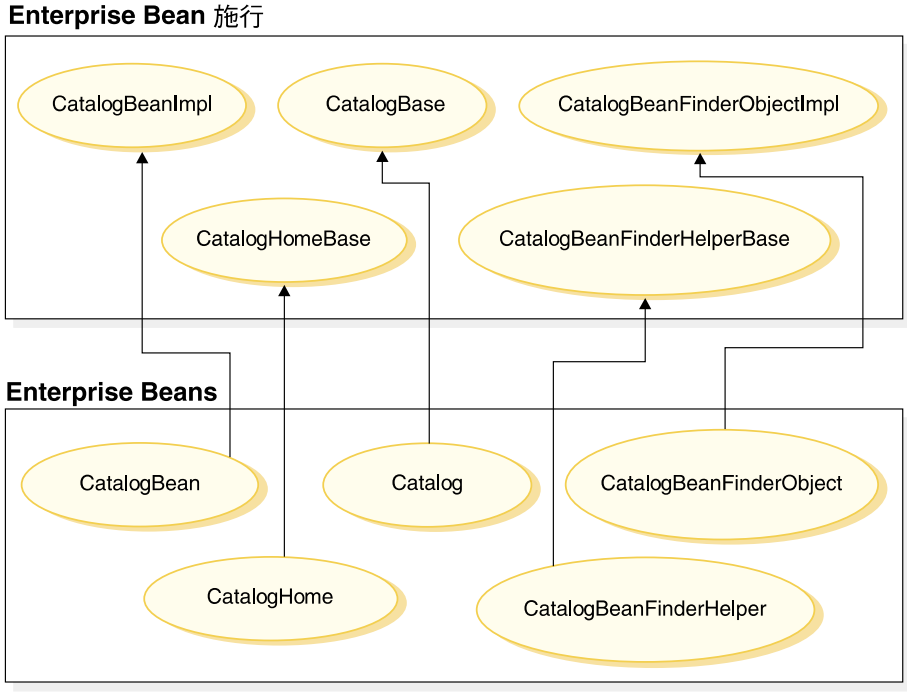


圖 17.

上圖亦適用於其它實體 Bean，因爲這些實體 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它實體 Bean 上，請更換實體 Bean 名稱 “Catalog”。舉例來說， InterestItemBean 類別是 InterestItemBeanImpl 類別的延伸，而 InterestItem 介面則是 InterestItemBase 介面的延伸。

在本圖中，使用 Java 繼承特性將公用 Enterprise Bean 的施行類別或介面分成兩部份。超類別或介面中含有 WebSphere Commerce 施行程式碼。這些超類別與介面皆定義於有別於子類別與介面的套件與專案內。

除了 *bean_name*BeanFinderHelperBase 與 *bean_name*BeanFinderObjectImpl 類別外，WebSphere Commerce 程式碼儲存庫中含有這些所有超類別與介面的二進位程式碼。由於含有 *bean_name*BeanFinderHelperBase 與 *bean_name*BeanFinderObjectImpl 類別的原始碼，您可以看到各 finder 之 SQL 陳述式的定義方式。您不應以任何方式來修改這些類別。

如果要檢查 CatalogBeanFinderHelperBase 與 CatalogBeanFinderObjectImpl 的原始碼，請開啓 com.ibm.commerce.catalog.objsrc 套件。其它套件所用的命名慣例類似。

子類別與介面則可供修改。

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。

修改現有 WebSphere Commerce 公用實體 Bean 的一個方法是新增其它欄位。在此情況下，在您新增新欄位後，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。

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，您將建立一個對應至新資料庫表格中之屬性的欄位。

如果要建立新 CMP Enterprise Bean，您必須在 VisualAge for Java 中執行下列步驟：

1. 確定您的工作區擁有者是設爲 WCS 程式開發人員。
2. 爲 Bean 建立一個新 EJB 群組。
3. 使用「建立 Enterprise Bean 引導精靈」工具，來建立新 CMP Enterprise Bean。
4. 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。
5. 必要時可針對每一個所建的 CMP 欄位，各建立一對 getter 與 setter 方法。
6. 必要時可在 FinderHelper 介面中定義 FinderHelper 欄位，並新增新 FinderHelper 方法。
7. 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。
8. 將 Enterprise Bean 中的欄位映射至資料庫表格中的直欄。
9. 爲 Enterprise Bean 產生存取 Bean 與部署的程式碼。

有關上述每一個步驟的詳述，請參閱下列章節。

註: 如果您的新 Enterprise Bean 受 WebSphere Commerce 存取控制系統保護，請參閱第 79 頁的第 4 章, 『存取控制』取得進一步的資訊。存取控制可在您建立 Bean 後新增。

假設您有一個新表格 USERRES，其中指定使用者住家類型的部份相關資訊。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 EJB 群組: 在您建立新實體 Bean 時，您必須將之建於有別於 WebSphere Commerce EJB 群組的 EJB 群組中。當您在 VisualAge for Java 中使用 Enterprise Bean 時，您必須切換至工作區中的 EJB 標籤處。然後從 **EJB** 功能表中，您可以選取**新增 > EJB 群組**，以啓動「新增 EJB 群組引導精靈」。

在您建立 EJB 群組時必須指定下列兩要項：將儲存 EJB 程式碼的專案以及 EJB 群組的名稱。

EJB 專案可在工作區的「專案」標籤中檢視。建立新 EJB 群組時，您必須指定一個有別於 WebSphere Commerce 專案的專案。舉例來說，您可以讓 EJB 群組使用 MyCustomEJB 專案。在您建立群組前，這個專案不一定得存在，這是因爲 VisualAge for Java 可自動爲您建立。這個專案應專門用於 EJB 程式碼上；而不應用於任何指令或資料 Bean 程式碼上。基於部署目的，有必要對程式碼類型採取此種區隔方式。藉由將您自訂程式碼與 WebSphere Commerce 程式碼加以區隔，可在您要移轉至新版時將影響降至最少。

確定專案名稱與 EJB 群組名稱，有遵循適合您應用程式的命名慣例。

建立新 CMP Enterprise Bean: 如果要建立新 CMP Enterprise Bean，您可以使用「建立 Enterprise Bean 引導精靈」工具。如果要啓動此工具，請以滑鼠右鍵按一下將新增新 Bean 的 EJB 群組，並選取**新增 > Enterprise Bean**。

選擇要建立新 Enterprise Bean 並指定 Bean 的名稱。就 WebSphere Commerce 的 Enterprise Bean 命名慣例而言，會使用 Bean 之對應表格的名稱來命名 Bean。舉例來說，如果您的新資料庫表格名稱爲 USERRES，則 Enterprise Bean 名稱會是 UserRes。

「專案」欄位中會自動移入您專案的名稱。您必須在專案中，指定一個應儲存程式碼的套件名稱。例如，像您爲 Enterprise Bean 所建之存取 Bean 的程式碼，即會儲存在套件中。同樣地，在您命名套件時，請確定您是遵循適合您應用程式的命名慣例來命名。舉例來說，套件名稱可爲 `com.mycompany.mycustombeans`。

在 Bean 類別中，VisualAge for Java 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，而您的新實體 Bean 應使用該欄位，而非使用您類別中所產生的欄位。因此，您應移除新實體 Bean 中所產生的 EntityContext。

您的新 Enterprise Bean 必須含有一個 serialVersionUID 欄位。如果您使用「引導精靈」來建立新 Bean，VisualAge for Java 會爲您產生此欄位。如果您不是使用此工具來建立 Bean，則必須自行新增此欄位。

在 Superclass 欄位中您必須指定 com.ibm.commerce.base.objects.ECEntityBean 類別。以下的程式碼範例是示範超類別所提供的函數：

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

您也應移除任何不會採用引數的 ejbCreate() 與 ejbPostCreate() 方法。如果實體 Bean 中仍留有這些方法，可能會造成執行期間錯誤。

您必須針對資料庫表格中的每一個直欄，分別在 Bean 中建立一個新 CMP 欄位（您必須按一下「引導精靈」中的「下一步」，才能看到「新增 CMP 欄位到 Bean 中」選項）。請分別指定各欄位的欄位名稱與欄位的資料類型。請針對屬於主要鍵或主要鍵之一部份的直欄，勾選「關鍵欄位」勾選框。並針對其它所有直欄，勾選「將 getter 與 setter 方法引介至遠端介面」勾選框。

當您填妥所有欄位後，請按一下「完成」，VisualAge for Java 將會建立新 CMP Enterprise Bean。

在 *Bean_Name*FinderHelper 介面中建立新 FinderHelper 欄位：
*Bean_Name*FinderHelper 介面含有對應至 findByPrimaryKey 方法以外之所有 FinderHelper 方法的 SQL 搜尋子句。

新 Enterprise Bean 會使用定義於 Bean 之 FinderHelper 介面中的 findXBy*ArgName*（其中 *ArgName* 爲引數的名稱）方法以及 FinderHelper 方法來撰寫 SQL 查詢。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。

如果要在 Bean 中建立新 FinderHelper 欄位，您必須選取 *Bean_Name*_FinderHelper 介面，並修改原始碼，以制定 select 陳述式應如何形成。例如：

```
public interface UserResFinderHelper {
public static final String findXByHomeAndRoomsWhereClause = "(T1.HOME = ?
      and T1.ROOMS = ?)";
}
```

發源介面需要 findXByHomeAndRooms 方法，以取得每一個 HOME 與 ROOMS 的輸入參數，以便在 ? 字元所代表的值中移入資料。此種查詢建構類型稱爲*參數插入*。

如果輸入參數爲 ″detached″ 與 ″3″，則產生的 SQL 陳述式會是：

```
select * from USERRES where HOME=detached and ROOMS=3
```

基於安全理由，在您爲新實體 Bean 建立 FinderHelper 方法時，您應使用參數插入方式。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：

```
public static final String
     findXByOwnerIdWhereClause = " (T1.OWNERID = 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，建議您使用參數插入方式。

如果您無法使用參數插入方式，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。

在 *Bean_NameHome* 介面中建立新 FinderHelper 方法: 您必須針對您在 *Bean_Name*FinderHelper 介面中指定的每一個 FinderHelper 欄位，分別在 Bean 的發源介面中建立一個 FinderHelper 方法。除了除去 ″WhereClause″ 外， FinderHelper 方法的名稱必須完全和 FinderHelper 欄位名稱相符。亦即，以欄位名稱 findXByHomeAndRoomsWhereClause 爲例，其對應的方法名稱爲 findXByHomeAndRooms。

如果要建立新 FinderHelper 方法，請執行下列步驟：

1. 以滑鼠右鍵按一下 *Bean_Name***Home** 介面，並選取**新增 > 方法**。
   會開啓「建立方法引導精靈」。
2. 選取**建立新方法**，然後按**下一步**。

3. 在**方法名稱**欄位中，輸入 FinderHelper 方法的名稱。除了除去 ″WhereClause″ 外，此 FinderHelper 方法的名稱必須完全和 FinderHelper 欄位名稱相符。舉例來說，請輸入 `findXByHomeAndRooms`。
4. 在**傳回類型**欄位中，輸入下列之一：
   - 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`。
5. 按一下**此方法應有哪些參數**？旁的**新增**按鈕，以新增適當參數。舉例來說，您可以新增**字串**類型的 `argHome`，以包含住家類型，以及新增**位元組**類型的 `argRooms`，以包含房間數。
6. 當您新增完所有參數後，請按**下一步**。
7. 按一下**此方法可能擲出哪些異常狀況**？旁的**新增**按鈕，並新增下列的異常狀況：
   - `java.rmi.RemoteException`
   - `javax.ejb.FinderException`

   註: 上述所列的異常狀況是您方法應會擲出的一組最基本的異常狀況。視您的程式碼而定，您可能得指定其它的異常狀況。
8. 按一下**完成**。

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，在案例化期間只有這些值才會案例化。

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，以便明確案例化這些欄位。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「類型」窗格中展開 ***Bean_Name*Bean** 類別。例如，選取 **UserResBean**。
2. 按一下現有的 ejbCreate 方法，以檢視原始碼。（請注意，視您 Enterprise Bean 的主要鍵而定，這可能是 ejbCreate(String) 或 ejbCreate(String, int)，或者可能會採用其它某些輸入參數。）
3. 您必須修改原始碼，以便將每一個 CMP 欄位當作輸入參數納入到方法中，並以適當值案例化每一個 CMP 欄位。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, java.rmi.RemoteExeption {
   _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, java.rmi.RemoteExeption {
   _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 71 頁的『主要鍵』，取得詳細資訊。

4. 儲存修改過的方法。當您儲存程式碼時，VisualAge for Java 會建立一個採用新參數的新 ejbCreate 方法。原始的 ejbCreate 方法還在。
5. 刪除原始的 ejbCreate 方法（其中不含任何引數）。
6. 以滑鼠右鍵按一下新 ejbCreate 方法，並選取**新增 > EJB 發源介面**。
7. 建立對應的 ejbPostCreate 方法。（此方法不需新增到發源介面中。）

將資料庫表格映射至新 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映射。此映射稱爲綱目。VisualAge for Java 中提供工具可讓您簡化此作業。

如果要建立綱目，請執行下列步驟：

1. 從 **EJB** 功能表中選取**開啓到 > 資料庫綱目**。會開啓「綱目瀏覽器」。
2. 從**綱目**功能表中選取**匯入 / 匯出綱目 > 從資料庫匯入綱目**。
3. 輸入新綱目的名稱，並按一下**確定**。
4. 在「資料庫連線」視窗中，依下列所示輸入資訊：

屬性	DB2 值	Oracle 值
連線類型	COM.ibm.db2.jdbc.app. DB2Driver	Oracle.jdbc.driver. OracleDriver
資料來源	jdbc:db2:wc_database_name	jdbc:oracle:thin@*hostname*: *port*:*Oracle_SID*
使用者名稱	*wc_db_user_name*	*wc_db_user_name*
密碼	*wc_db_password*	*wc_db_password*

按如下所述置換各值：

- DB2 *wc_database_name* 爲您 WebSphere Commerce 資料庫的名稱
- Oracle *hostname* 爲 Oracle 伺服器主電腦名稱。
- Oracle *port* 爲 Oracle 資料庫的埠號。
- Oracle *Oracle_SID* 爲 Oracle 案例 ID。
- *wc_db_user_name* 爲資料庫的使用者名稱。
- *wc_db_password* 爲資料庫密碼。

5. 從**限定元**清單中，選取您資料庫的限定元（可以是您資料庫的使用者名稱）。
6. 按一下**建置表格清單**。
7. 從產生的清單中選取 *your_new_table*，並按一下「確定」，以產生綱目。
8. 在產生綱目後，按一下「綱目」窗格中的 *your_new_table*。
9. 在「表格」窗格中標示並按兩下 *your_new_table*。
會開啓「表格編輯器」視窗。
10. 移除**限定元**欄位中的任何資訊。如此您的 Enterprise Bean 便可攜到其它機器上。
11. 從**綱目**功能表中選取**儲存綱目**。輸入適當專案、套件與類別名稱。

接下來，您必須建立綱目映射。綱目映射是指資料庫直欄和 Enterprise Bean 中之欄位間的映射。

如果要建立綱目映射，請執行下列步驟：

1. 從 **EJB** 功能表中選取**開啓到 > 綱目映射**。
會開啓「資料儲存庫映射」視窗。
2. 輸入映射的名稱。有關命名新映射方面的建議，請參閱第 73 頁的『命名資料庫綱目物件時的注意事項』。
3. 選取 EJB 群組與綱目。有關命名新綱目方面的建議，請參閱第 73 頁的『命名資料庫綱目物件時的注意事項』。
4. 在「資料儲存庫映射」窗格中，選取您的映射。
5. 在「持續性類別」窗格中選取您的類別。
6. 從**表格映射**功能表中，選取**新表格映射 > 新增無繼承的表格映射**。
7. 從**表格**下拉清單中選取您的表格，並按一下**確定**。

8. 在「表格映射」畫面中標示並以滑鼠右鍵按一下您的表格映射，然後選取**編輯內容映射**。
   會開啓「內容映射編輯器」。
9. 您必須針對每一個類別屬性（Bean 中的 CMP 欄位），分別指定映射類型以及應映射至哪個資料庫直欄。舉例來說，您可使用映射類型 ″Simple″ 將 UserId 類別屬性映射至資料庫表格中的 USERID 直欄。
10. 在所有欄位皆已映射至對應的資料庫直欄後，您必須儲存綱目映射。請從**資料庫映射**功能表中，選取**儲存資料儲存庫映射**。然後輸入適當的專案、套件與類別名稱，以儲存映射。按一下**完成**。

建立存取 Bean 並產生部署的程式碼: 存取 Bean 形同 Enterprise Bean 的外層，可簡化其它元件和 Enterprise Bean 間的互動方式。您必須爲您的新 Enterprise Bean 建立一個存取 Bean。

在您產生部署的程式碼時，VisualAge for Java 中的工具會分析 Bean，以確定符合 Sun Microsystems EJB 規格中指定的規則以及 EJB 伺服器的特定規則。

如果要建立新 Enterprise Bean 的存取 Bean，請執行下列步驟：

1. 在「Enterprise Bean」窗格中，以滑鼠右鍵按一下新 Enterprise Bean，並選取**新增 > 存取 Bean**。（您可能得先展開內含新 Bean 的 EJB 群組，才能看到該 Bean。）
   會開啓「建立存取 Bean 引導精靈」。
2. 在 **EJB 群組**、**Enterprise Bean** 與**存取 Bean 名稱**欄位中，指定適當的 EJB 群組、Bean 名稱與存取 Bean 名稱。
3. 選取**存取 Bean 類型之實體 Bean 的 Copy Helper**，並按**下一步**。
4. 從**選取零引數建構子的發源方法**下拉清單中選取 **findByPrimaryKey**。
5. 從**轉換器**下拉清單中選取起始內容的 **WCSStringConverter**，並按**下一步**。
6. 在「選取和自訂 Copy Helper 的 Bean 內容」視窗中，爲每一個欄位選取 **WCSStringConverter**。
7. 按一下**完成**。

如果要產生部署的程式碼，請執行下列步驟：

1. 在「Enterprise Bean」窗格中，以滑鼠右鍵按一下新 Enterprise Bean，並選取**產生部署的程式碼**。

請注意，以這項工具產生的部署程式碼必須符合 EJB 1.0 規格，且唯有在您的 Enterprise Bean 執行於 VisualAge for Java 中時才能使用。在稍後的階段中，當您將 Enterprise Bean 部署至執行於 WebSphere Application Server V4.0 中的 WebSphere Commerce 應用程式期間，會要求您產生一個 JAR 檔，且其中含有符

合 EJB 1.1 規格的部署程式碼。有關建立此 EJB 1.1 匯出 JAR 檔的進一步資訊，請參閱第 171 頁的『EJB 部署程式碼相關資訊』與第 326 頁的『產生部署程式碼』。

使用測試從屬站來測試 Enterprise Bean: VisualAge for Java 提供一個測試從屬站，可用來測試 Enterprise Bean。如果要使用測試從屬站來測試您的新 Bean，請執行下列步驟：

1. 啓動內含您要測試之 Enterprise Bean 的 EJB 伺服器。
2. 以滑鼠右鍵按一下 Enterprise Bean，並選取**執行測試從屬站**。會開啓「EJB 測試從屬站與 EJB 查閱」視窗。
3. 在 **JNDI 名稱**欄位中輸入 Enterprise Bean 的 JNDI 名稱，並按一下**查閱**。
4. 以滑鼠右鍵按一下內含引數的 **findByPrimaryKey** 方法，並選取**呼叫**。

編碼慣例: 請注意如下的 Enterprise Bean 編碼慣例：

- 請勿使用 BLOB 或 CLOB 資料類型。
- LONG（即所謂的 LONG VARCHAR）資料類型的 CMP 欄位不能是 Enterprise Bean 之 CMP 欄位中的第一個或最後一個成員。如果要驗證清單，請檢查 `EJSJDBCPersister._hydrate()` 方法，並查看清單中第一個或最後一個元素的類型是否爲 LONG VARCHAR。如果第一個欄位爲此種類型，請執行下列步驟：
  1. 取消設定第一個欄位。假設其名稱爲 `fieldA`。
  2. 取消設定*不*是 LONG VARCHAR 類型的另一個欄位。假設其名稱爲 `fieldB`。
  3. 重設 `fieldA`。
  4. 重設 `fieldB`。
  5. 開啓「綱目映射瀏覽器」，並編輯表格映射以反映這些變更。儲存映射。
  6. 重新產生部署的程式碼。
- Enterprise Bean 程式碼不應參照 Enterprise Bean 套件外的任何項目。舉例來說，您不應參照 Enterprise Bean 程式碼中的指令或資料 Bean。
- 如果要對您的 Enterprise Bean 啓用存取控制，請在 Enterprise Bean 的遠端介面中新增 com.ibm.commerce.security.Protectable 介面與（或）com.ibm.commerce.security.Groupable 介面。新增這些介面後，請重新產生 Bean 的部署程式碼與存取 Bean。您也必須在 objsrc 套件中建立一個存取 Helper 類別。

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，並施行 SmartDataBean 介面。VisualAge for Java 會爲資料 Bean 自動產生大部份的程式碼。

如果要建立簡單的資料 Bean，您必須執行下列步驟：

1. 建立一個專案與套件，以儲存資料 Bean 程式碼。
2. 建立一個延伸了對應的存取 Bean 並施行適當資料 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。

為資料 Bean 程式碼建立專案與套件: 建立專案與套件，主要用意是儲存您的資料 Bean 程式碼。

如果要建立新專案，請執行下列步驟：

1. 選取**專案**標籤。
2. 從**已選取**功能表中選取**新增 > 專案**。
   會開啓「新增專案」引導精靈。
3. 確定您已選出**建立指名的新專案**，並輸入您新專案的名稱。舉例來說，輸入 `My Data Beans`。
4. 按一下**完成**。

如果要建立新套件，請執行下列步驟：

1. 以滑鼠右鍵按一下您建給資料 Bean 程式碼的專案，並選取**新增 > 套件**。舉例來說，以滑鼠右鍵按一下 **My Data Beans**，並選取**新增 > 套件**。
   會開啓「新增套件」引導精靈。
2. 確定您已選出**建立指名的新套件**，並爲您的資料 Bean 套件輸入一個適當名稱。舉例來說，輸入 `com.mycompany.mydatabeans`。
3. 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 提供實體 Bean 的簡單呈現（間接）。資料 Bean 概括了一些可從實體 Bean 中擷取或設於實體 Bean 中的內容。

如果要建立資料 Bean，請執行下列步驟：

1. 以滑鼠右鍵按一下用以儲存資料 Bean 的套件，並選取**新增 > 類別**。
   會開啓「建立類別」引導精靈。
2. 專案與套件名稱欄位皆已移入資料。

3. 請確定您已選出**建立新類別**，並按**下一步**。
4. 在**類別名稱**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`。
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`，並按一下**確定**。
6. 按**下一步**。
7. 如果要指定資料 Bean 應施行的介面，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**關閉**。
8. 按一下**完成**。

新增必要欄位到資料 Bean 中: 本節說明如何新增必要欄位到您的新資料 Bean 中。

如果要新增 iCommandContext 欄位，請執行下列步驟：

1. 以滑鼠右鍵按一下新資料 Bean（例如 UserResDataBean），並選取「新增 > 欄位」。
   會開啓「建立欄位」引導精靈。
2. 在**欄位名稱**欄位中，輸入 `iCommandContext`。
3. 按一下**瀏覽**，以新增欄位類型，並輸入 `com.ibm.commerce.command.CommandContext`。按一下**確定**。
4. 針對存取修飾元，選取**保護**。
5. 按一下**完成**。

如果要新增 iRequestProperties 欄位，請執行下列步驟：

1. 以滑鼠右鍵按一下新資料 Bean（例如 UserResDataBean），並選取「新增 > 欄位」。
   會開啓「建立欄位」引導精靈。
2. 在**欄位名稱**欄位中，輸入 `iRequestProperties`。
3. 按一下**瀏覽**，以新增欄位類型，並輸入 `com.ibm.commerce.datatype.TypedProperty`。按一下**確定**。
4. 針對存取修飾元，選取**保護**。
5. 按一下**完成**。

修改資料 Bean 的 set 方法: 在您建立資料 Bean 後，您必須以產生的部份 set 方法來修改程式碼。

如果要更新 set 方法，請執行下列步驟：

1. 展開您的新資料 Bean，以檢視其欄位與方法。
2. 選取 **setCommandContext(CommandContext)** 方法以檢視其原始碼。
   「原始碼」窗格中會顯示如下的原始碼：

   ```
   public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
    {
    }
   ```

3. 修改原始碼，讓方法如下所示：

   ```
   public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
   {
      iCommandContext = arg1;
   }
   ```

   儲存您的工作 (Ctrl+S)。

4. 選取 **setRequestProperties(TypedProperty)** 方法以檢視其原始碼。
   「原始碼」窗格中會顯示如下的原始碼：

   ```
   public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
      throws Exception {}
   ```

5. 您或許想修改原始碼，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編碼。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，在設定主要鍵的程序上，建議您按步驟 5 所示採用間接方法。

修改資料 Bean 的 get 方法: 在您建立資料 Bean 後，您必須以產生的部份 get 方法來修改程式碼。

如果要更新 get 方法，請執行下列步驟：

1. 展開您的新資料 Bean，以檢視其欄位與方法。
2. 選取 **getCommandContext()** 方法以檢視其原始碼。
   「原始碼」窗格中會顯示如下的原始碼：

   ```
   public com.ibm.commerce.comand.CommandContext getCommandContext () {
      return null;
   }
   ```

3. 修改原始碼，讓方法如下所示：

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

   儲存您的工作 (Ctrl+S)。

4. 選取 **getRequestProperties()** 方法以檢視其原始碼。
   「原始碼」窗格中會顯示如下的原始碼：

   ```
   public com.ibm.commerce.datatype.TypedProperty setRequestProperties() {
      return null;
   }
   ```

5. 依下列所示修改原始碼：

   ```
   public com.ibm.commerce.datatype.TypedProperty setRequestProperties() {
      return iRequestProperties;
   }
   ```

   儲存您的工作 (Ctrl+S)。

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. 選取 **populate()** 方法以檢視其原始碼。
   「原始碼」窗格中會顯示如下的原始碼：

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

3. 修改原始碼，讓方法如下所示：

   ```
   public void populate () throws Exception {
      super.refreshCopyHelper();
   }
   ```

   儲存您的工作 (Ctrl+S)。

撰寫新階段作業 Bean

在您建立新階段作業 Bean 時，您必須將階段作業 Bean 建於有別於 WebSphere Commerce EJB 群組的 EJB 群組中。請將這個新 EJB 群組儲存在有別於 WebSphere Commerce 專案的新專案中。舉例來說，您可在 com.mycompany.mycustomcode 套件中建立 MyCustomBeans EJB 群組。藉由將您自訂程式碼與 WebSphere Commerce 程式碼加以區隔，可在您要移轉至新版時將影響降至最少。

您的新階段作業 Bean 應爲 com.ibm.commerce.base.helpers.BaseJDBCHelper 類別的延伸。超類別會提供一些方法讓您從 Commerce Server 所用的資料來源物件中取得 JDBC 連線物件，以便讓階段作業 Bean 能參與和其它實體 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,
      java.rmi.RemoteException, 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 V1.0 結構在案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 V1.0 規格文件。

雖然 WebSphere Application Server 會施行選項 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" 案例。一旦交易完成，即會將案例傳回到可用案例儲存池中。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。在此種施行方式下，在每次交易開始時，儲存器即會載入實體 Bean 的持續性資料，而在交易期間只會快取實體 Bean。儲存器會啓動實體 Bean 的多個案例，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。

實體 Bean 的其它注意事項

尋找以更新（Find for Update）

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 並不會將您對實體 Bean 所作的變更寫到資料庫中，因此資料庫可能會與實體 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 USER 表格中的 REGISTEREDSTAMP。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，您必須使用「WebSphere Application Server 管理主控台」將任何新 Enterprise Bean 的方法指定到 WCSMethodGroup 安全方法群組中。請在部署新 Enterprise Bean 時執行此步驟。此外，如果您修改現有 WebSphere Commerce 實體 Bean，您必須將受影響之 EJB 群組中的所有實體 Bean 的方法指定給 WCSMethodGroup 安全方法群組。有關自訂程式碼的部署程序說明，請參閱第 171 頁的『程式碼部署』。

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實體 Bean。從而當案例化某個實體 Bean 時，可能會建立資料庫記錄。因此，案例化實體 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，會案例化實體 Bean 的對應存取 Bean，並取得或設定各欄位，從而間接使用實體 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）案例化存取 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 方法用以案例化實體 Bean 的新案例。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, java.rmi.RemoteException {
      //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 表格中其它直欄的預設值。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整數 (long) 資料類型（若爲 DB2 則爲 BIGINT，若爲 Oracle，則爲 NUMBER），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案例化一個新的 UserProfileAccessBean，稱爲 ″abUserProfile″。第二行則是在存取 Bean 中設定主要鍵。 VisualAge for Java 採用 `setInitKey_`*`xxx`* （其中 *xxx* 爲主要鍵的欄位名稱）命名慣例來命名主要鍵的 set 方法。當案例化存取 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。

使用實體 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 VisualAge for Java 中的工具來產生此種存取 Bean。

WebSphere Commerce 指令是與存取 Bean 互動，而非直接與實體 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，而不讓從屬站看到。
- 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 會施行一種 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、實體 Bean 與資料庫之間的互動。

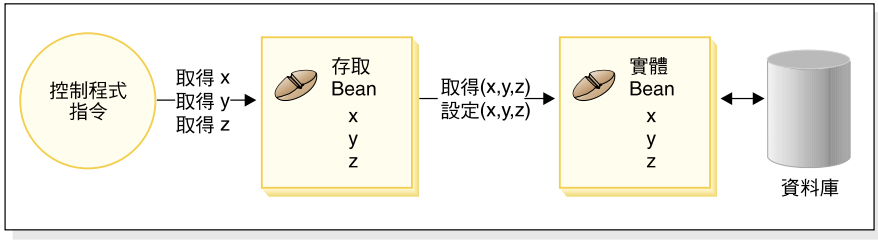


圖 18.

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

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。如果您所用的是其它資料庫，第 77 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映射至非物件式 Java 資料類型，且這些非物件式資料類型會在某些 Enterprise Bean 物件案例化時造成問題。

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，且不需修改資料庫架構。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。而在其它浮點十進位數方面，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，並顯示不同資料庫施行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	BIGINT
Double	FLOAT	NUMBER	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映射至實體 Bean 中的 CMP 欄位。相反地，請使用階段作業 Bean 來執行此直欄的擷取與更新。

第 4 章 存取控制

瞭解存取控制

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：

- 實體 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，可供遠端從屬站存取。
- 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實體 Bean 中擷取資料的資料 Bean。從屬站可藉由撰寫 URL 要求來要求 JSP 頁面。
- 控制程式與檢視畫面指令
  從屬站可藉由撰寫 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 架構成使用如下的 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 路徑架構，顯示要求在存取 WebSphere Commerce 資源時可能的遵循途徑。

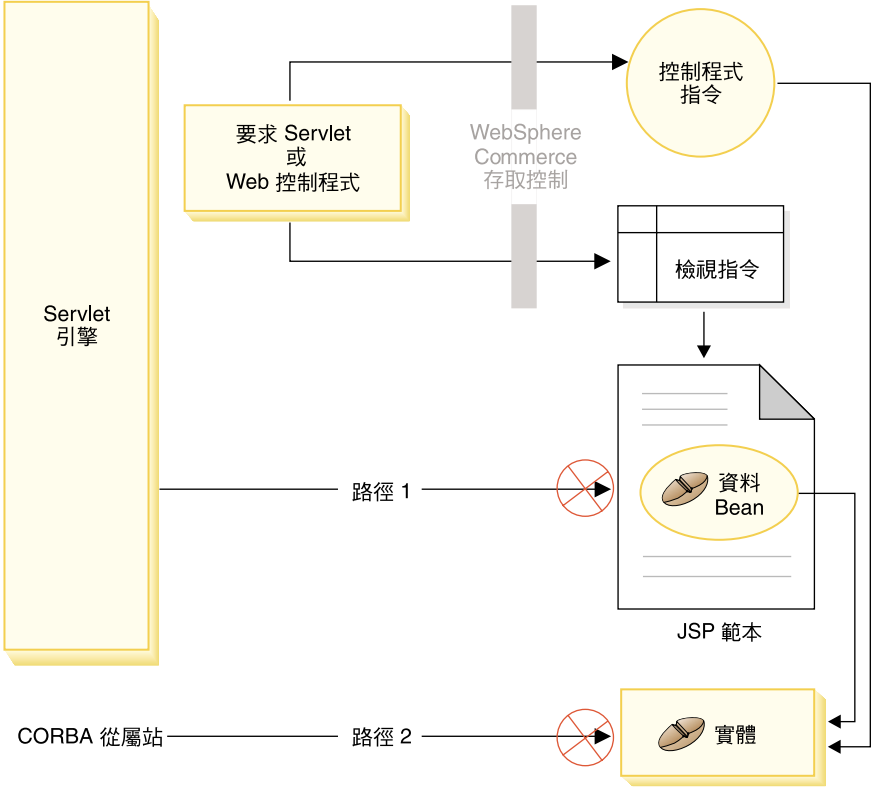


圖 19.

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施行存取控制。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）與實體 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 範本與實體 Bean：

WebSphere Application Server 安全特性

WebSphere Application Server 提供一種安全特性。當使用此方法時，會將所有的 Enterprise Bean 方法與 JSP 範本架構成只能由系統身份呼叫之。如果要存取這些 WebSphere Commerce 資源，則必須先將 URL 要求引導至將系統身份設爲現行執行緒的「要求 Servlet」處，然後再將之傳遞給 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 範本與實體 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。

有關將 WebSphere Application Server 架構爲保護 WebSphere Commerce 資源的說明，請參閱 *WebSphere Commerce 安裝手冊*。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。

有關爲自訂 Enterprise Bean 中之方法架構 WebSphere Application Server 安全特性的說明，請參閱第 336 頁的『將新 Enterprise Bean 組譯到企業應用程式中』與第 341 頁的『將已修改的 Enterprise Bean 組譯到企業應用程式中』。

防火牆保護

當 WebSphere Commerce Server 在防火牆後面執行時，網際網路從屬站將無法直接存取實體 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。

WebSphere Commerce 存取控制原則的簡介

WebSphere Commerce 存取控制模型是以施行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。舉例來說，某存取控制原則可指出容許擔任「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進一步資訊，請參閱 WebSphere Commerce 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，則資源亦必須施行「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」遠端介面的案例，其狀態爲 "Z"（代表共用的需求項目清單）。

註: 有關 ACRESGRP 表格之 Conditions 直欄的 XML 資訊，請參閱 *WebSphere Commerce 存取控制手冊*。

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.ContractCreateCmd
10248	com.ibm.commerce.contract.commands.ContractCreateCmd
10249	com.ibm.commerce.contract.commands.ContractCreateCmd
10250	com.ibm.commerce.contract.commands.ContractCreateCmd

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。

存取控制原則另一項重要的概念是存取控制原則*擁有者*的概念。存取控制原則擁有者是擁有該存取控制原則的組織實體。知道存取控制原則擁有者很重要，這是因爲存取控制原則僅適用於存取控制原則擁有者所擁有的資源。

存取控制原則管理程式會針對考量中的每一項資源，套用擁有組織實體（或在成員階層中其上層組織實體）所擁有的存取控制原則，直到找到授與許可權的原則爲止，或檢查完所有原則而沒有授與許可權爲止。

請注意下圖所示的成員階層。

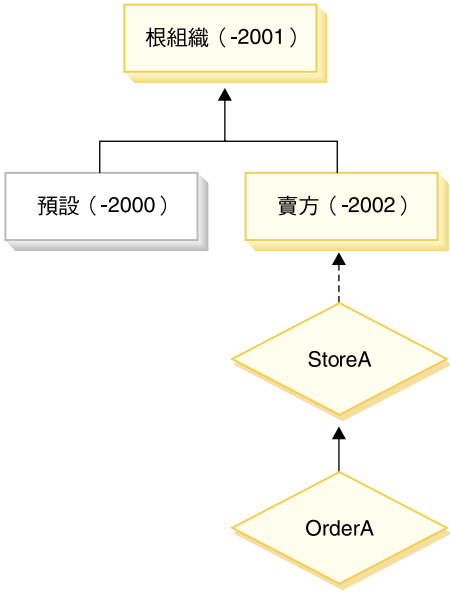


圖 21.

在資源 “OrderA” 方面，可套用「賣方」或「根組織」所擁有的任何存取控制原則。只要存取控制原則管理程式找到這些組織所擁有且會授與使用者許可權的一項原則（根據存取控制原則中的四項元素），即會立即停止搜尋存取控制原則。不過，如果找不到這些組織所擁有，且會授與使用者對受保護資源執行動作的存取控制原則，則會拒絕存取。

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。凡提及其它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。您可以修改 `defaultAccessControlPolicies.xml` 檔，或建立自己的 XML 檔。有關建立這些 XML 型資訊的進一步資訊，請參閱 *WebSphere Commerce 存取控制手冊*。

下列各節顯示各種不同關係群組類型的範例。

由單一關係鏈組成的關係群組: Business 您可能需在存取控制原則中指出使用者所隸屬的組織實體必須是資源的 BuyingOrganizationalEntity。因此，您必須建立一個由單一關係鏈（長度 2）組成的關係群組。關係鏈的長度之所以爲 2 是因爲其由兩個個別的關係組成。第一個關係位於使用者與其上層組織實體間。使用者在該關係中屬於 ″下層″。在第二個關係方面，存取控制原則管理程式會檢查上層組織實體和資源間有否執行 BuyingOrganizationalEntity 關係。換句話說，如果它是資源的買方組織實體，則會傳回 ″true″。

下列 XML 片段取自 `defaultAccessControlPolicies.xml` 檔，顯示如何定義此種關係群組類型：

```
<RelationGroup Name="MemberOf->BuyerOrganizationalEntity"
     OwnerID="RootOrganization">
 <RelationCondition><![CDATA[
  <profile>
```

```
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="HIERARCHY" value="child"/>
     <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
    </openCondition>
  </profile>
 ]]></RelationCondition>
</RelationGroup>
```

Business 另舉一例，使用者在組織實體（爲該資源的買方組織實體）中必須擔任「帳戶代表」職務。同樣地，這會用到一個由單一關係鏈（長度 2）組成的關係群組。關係鏈中的第一個部份將找出使用者在其中擔任「帳戶代表」職務的所有組織實體。接著存取控制原則管理程式會針對此組組織實體，檢查其中是否至少有一個有和資源間執行 BuyingOrganizationalEntity 關係。換句話說，如果其中有一個是資源的買方組織實體，則會傳回 ″true″。

下列 XML 片段取自 defaultAccessControlPolicies.xml 檔，顯示如何定義此種關係群組類型：

```
<RelationGroup Name="AccountRep->BuyerOrganizationalEntity"
     OwnerID="RootOrganization">
 <RelationCondition><![CDATA[
  <profile>
    <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="ROLE" value="Account Representative"/>
     <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
    </openCondition>
  </profile>
 ]]></RelationCondition>
</RelationGroup>
```

由多個關係鏈組成的關係群組： 您可以撰寫一個由多個關係鏈組成的關係群組。如果要如此做，您必須指出使用者必須符合所有的關係鏈（亦即，*AND* 情況），或者至少必須符合其中一個關係鏈（亦即，*OR* 情況）。

Business 爲了示範此種關係類型，下列 XML 片段指出使用者必須是資源的建立者，且該使用者亦必須隸屬於資源中指定的 BuyingOrganizationalEntity。第一個關係鏈指出使用者必須是資源的建立者，且長度爲 1。第二個關係鏈指出使用者必須隸屬於資源中指定的 BuyingOrganizationalEntity，且長度爲 2。

```
<RelationGroup Name="Creator_And_MemberOf->BuyerOrganizationalEntity"
     OwnerID="RootOrganization">
 <RelationCondition><![CDATA[
  <profile>
   <andListCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="RELATIONSHIP" value="creator" />
    </openCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="HIERARCHY" value="child"/>
```

```
    <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
   </openCondition>
  </andListCondition>
 </profile>
]]></RelationCondition>
</RelationGroup>
```

如果不是 *AND* 情況，亦即您要求使用者必須符合這兩個關係鏈中的一個，則 <andListCondition> 標籤應改爲 <orListCondition> 標籤。

Professional Business 爲了示範一個可用於 WebSphere Commerce Professional Edition（以及 WebSphere Commerce Business Edition）中的關係群組，在此假設有一個關係群組指出使用者是資源的建立者或提交者。請見下列 XML 片段。

```
<RelationGroup Name="Creator_Or_Submitter"
     OwnerID="RootOrganization">
  <RelationCondition><![CDATA [
 <profile>
     <orListCondition>
   <openCondition name="RELATIONSHIP_CHAIN">
          <parameter name="RELATIONSHIP"value="creator"/>
   </openCondition>
   <openCondition name="RELATIONSHIP_CHAIN">
          <parameter name="RELATIONSHIP"value="submitter"/>
   </openCondition>
     </orListCondition>
 </profile>
]]></RelationCondition>
</RelationGroup>
```

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置存取群組的詳細資訊，請參閱 WebSphere Commerce 線上說明。

ACPOLICY

PolicyName	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdmin 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	StoreAdminCmdResourceGroup

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，案例化指令物件）。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在 WebSphere Commerce 存取控制原則組織架構下存取控制如何運作。

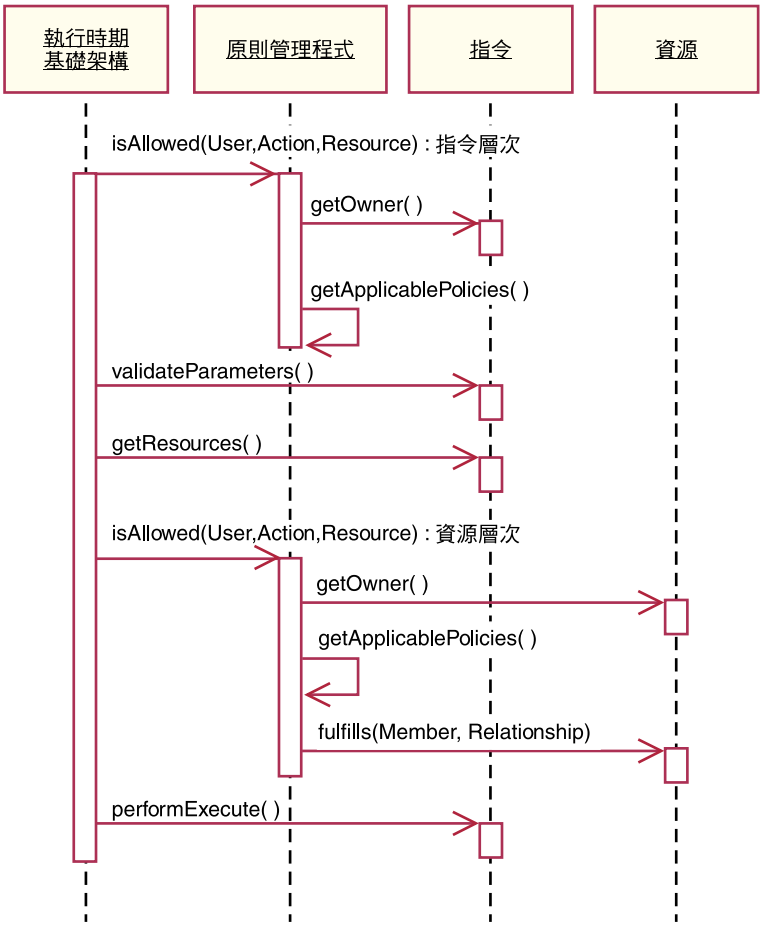


圖 23.

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與其上層組織所擁有的原則來作出判斷。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()`
   執行期間元件判斷使用者對於控制程式指令或檢視畫面是否具備指令層次的存取權。
2. `getOwner()`
   存取控制原則管理程式判斷指令層次資源的擁有者。就預設施行而言，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. getApplicablePolicies()
   存取控制原則管理程式根據指定的使用者、動作與資源，來尋找與處理適用的原則。
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（即施行 com.ibm.commerce.security.Protectable 介面的物件）的案例。在眾多情況下，資源爲一種存取 Bean。

   存取 Bean 不見得施行 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 96 頁的『在 Enterprise Bean 中施行存取控制』）。

   *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()
   執行期間元件判斷使用者對於 getResources() 指定的所有「資源/動作」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。只會套用資源擁有者與其上層組織所擁有的原則。
8. getApplicablePolicies()
   存取控制原則管理程式搜尋適用的原則，並加以套用。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()
   如果適用的原則有指定關係群組，則會檢查資源，看看該成員是否符合指定的資源關係。
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施行 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 才需施行此介面。

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，則其必須施行 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，假設您要施行一項僅套用於處於擱置狀態（status = P（擱置））的訂單之原則， Order 實體 Bean 的遠端介面會施行「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 存取控制模型的詳細資訊，請參閱 *WebSphere Commerce 存取控制手冊*。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。

施行存取控制

本節說明如何以自訂程式碼來施行存取控制。

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，需要保護的資源已施行可保護介面。不過當您建立新 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 施行可保護介面（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 out-of-the-box WebSphere Commerce 應用程式碼中， Order 實體 Bean 爲可保護的資源，OrderItem 實體 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 實體 Bean 爲可保護的資源，但 Address 實體 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 中施行存取控制

如果您建立的新 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 的遠端介面亦必須是 com.ibm.commerce.grouping.Groupable 介面的延伸。
4. Enterprise Bean 類別中含有下列方法的預設施行：
   - getOwner
   - fulfills
   - getGroupingAttributeValue

   請改寫任何您要的方法。您至少需改寫 getOwner 方法。
   下列程式碼片段顯示這些方法的預設施行。

```
*************************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException {
   return null;
}
*************************************************************************
*************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException
{
      return false;
}
*************************************************************************
*************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception, java.rmi.RemoteException {
   return null;
}
*************************************************************************
```

   以下是這些方法的範例施行，其以 OrderBean Bean 爲基礎：

```
*************************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException {
   com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB = new
   com.ibm.commerce.common.objects.StoreEntityAccessBean();
   storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
   return storeEntAB.getMemberIdInEJBType();
}
*************************************************************************
*************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException
```

```
   {
if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      else if (relationship.equalsIgnoreCase (
         com.ibm.commerce.base.helpers.EJBConstants.
         SAME_ORGANIZATIONAL_ENTITY_AS_CREATOR_RELATION)) {
            com.ibm.commerce.user.objects.UserAccessBean creator = new
               com.ibm.commerce.user.objects.UserAccessBean();
            creator.setInitKey_MemberId(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;
}
*****************************************************************************
*****************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception {
      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。

在資料 Bean 中施行存取控制

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. 施行 `com.ibm.commerce.security.Protectable` 介面。在此情況下， Bean 必須提供 `getOwner()` 與 `fulfills(Long member, String relationship)` 方法的施行。這些應施行於 Bean 的遠端介面上。

   當資料 Bean 施行「可保護」介面時，資料 Bean 管理程式會呼叫 `isAllowed` 方法，以根據目前的存取控制原則，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);
   ```

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 必須施行 `com.ibm.commerce.grouping.Groupable` 介面。

3. 施行 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. 施行 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 必須施行「可保護」介面。

如果資料 Bean 未施行「委任者」介面，則會在沒有存取控制原則保護下移入資料。

在控制程式指令中施行存取控制

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。

對於控制程式指令的指令層次原則，指令的介面名稱是指定爲資源。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，方式是讓指令的介面繼承自 com.ibm.commerce.command.ControllerCommand 介面，而且指令的實作是繼承自 com.ibm.commerce.commands.ControllerCommandImpl。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 類型的專用案例變數，以放置資源。然後改寫 getResources 方法，這是因爲方法的預設施行是傳回空值，因而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。

此外，建議您由方法來判斷是否必須案例化資源，或者可否使用內含資源參照的現有案例變數。檢查資源物件是否存在，有助提昇系統效能。必要時，您可在新控制程式指令的 performExecute 方法中使用相同的 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。此指令的 getResources 方法會傳回「訂單」與「使用者」可保護的物件。如果未指定動作，將預設爲 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，則動作不見得能繼續進行（三個全需要）。

如果您需要檢查控制程式指令中的其它參數或解析其中的參數，您可以使用 validateParameters() 方法。此爲選用方法。

其它的資源層次檢查

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，作業指令也可施行 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。此方法適用於任何繼承自 com.ibm.commerce.command.AbstractECTargetableCommand 類別的類別。例如，下列類別繼承自 AbstractECTargetableCommand 類別：

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

checkIsAllowed 方法亦可在繼承 com.ibm.commerce.command.AbstractECCommand 類別的類別中使用。例如，下列類別繼承自 AbstractECCommand 類別：

- com.ibm.commerce.command.TaskCommandImpl

下列顯示 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 方法應會傳回資源的建立者。舉例來說，指令由指令 factory 所建立的，訂單建於商店中，使用者則建於組織中。

指令層次之存取控制的預設施行

在指令層次的存取控制方面，getOwner() 方法的預設施行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() 方法的預設施行而言，是傳回 null。

就 validateParameters() 的預設施行而言，則不執行任何動作。

在檢視畫面中施行存取控制原則

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）皆必須施行「委任者」介面。凡受直接保護的資料 Bean 將委任給自己，因而亦必須施行「可保護」介面。而受間接保護的資料 Bean 則應委任給施行「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存取控制檢查。

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，指出從屬站所用的語言。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。

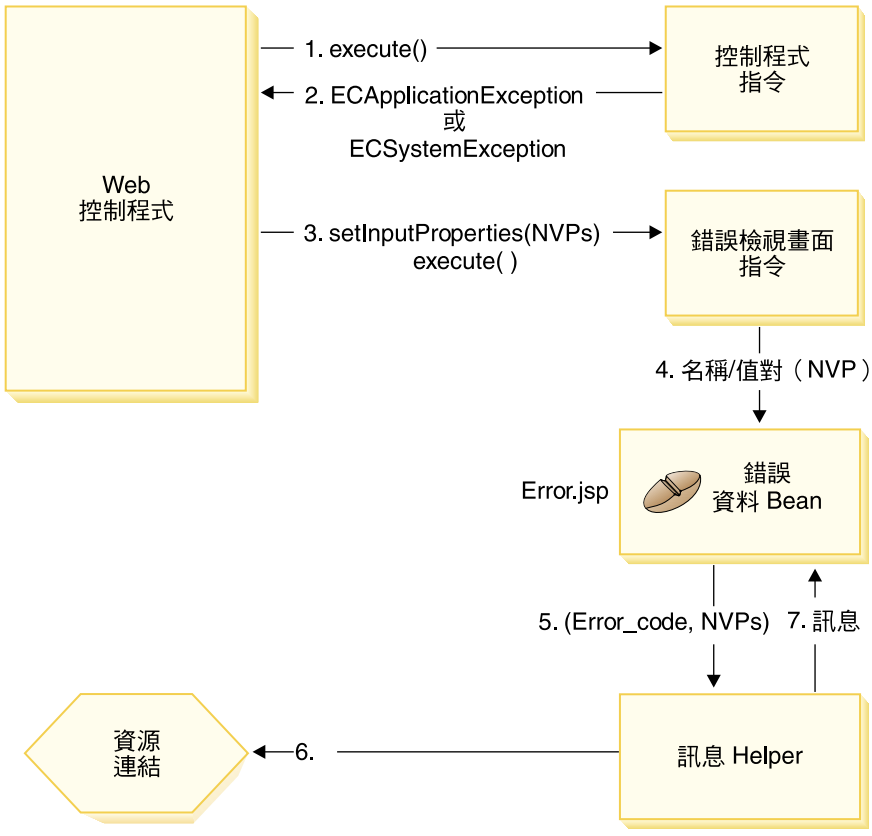


圖 24.

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。

訊息物件	說明
_ERR_REMOTE_EXCEPTION	當 EJB remote 方法呼叫傳回錯誤時擲出。
_ERR_CREATE_EXCEPTION	當建立 EJB 案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。
4. 單位測試該訊息。

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。此新類別必須建立在您自己的套件中，且儲存在有別於 WebSphere Commerce 專案的專案中。

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 物件或內容檔來施行。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。如果訊息是供單一商店 Servlet 使用，請將此檔案置於下列目錄中：

```
drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_instanceName.ear\
   wcstores.war\WEB-INF\classes
```

如果訊息是供單一工具 Servlet 使用，請將此檔案置於下列目錄中：

```
drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_instanceName.ear\
   wctools.war\WEB-INF\classes
```

如果訊息可供企業應用程式中的任何 Servlet 使用，請將檔案置於下列目錄中：

```
drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_instanceName.ear\
   properties
```

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}".
```

單元測試訊息

在您建立內含訊息關鍵字的類別、內含 ECMessage 物件的類別以及資源連結後，您應單元測試您的新訊息。

如果要測試前幾節中所描述的新訊息，請執行下列步驟：

1. 建立新類別作爲測試用。在本範例中，則是建立 MyTestingClass。
2. 在類別中新增下列 import 陳述式：

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

3. 在類別中新增 main() 方法。
4. 檢視下列的程式碼片段。並根據自己的需求加以修改（例如，確定目錄路徑是指向您系統中的有效資訊），並將之加到 main() 方法中

```
// fileName 字串應指向
// 有效的 WebSphere Commerce 架構檔：
  String fileName = "E:\\WebSphere\\CommerceServer\\instances
      \\demo\\xml\\demo.xml";

   LogConfiguration config = LogConfiguration.getUniqueInstance ();
   config.initialize (fileName, "testClone");

   ECMessageLog.out (MyMessage._ERR_CUSTOMER,
        "MyTestingClass", "main", "Hello");
```

程式碼中的前三行會起始設定 WebSphere Commerce 日誌記載服務程式。而最後一行則指示 ECMessageLog 印出 MyMessage._ERR_CUSTOMER 訊息。MyTestingClass 與 main 爲日誌追蹤格式的一部份。Hello 字串則放在定義於 ecCustomerMessages.properties 檔中之訊息的 {0} 位置保留元中。

5. 執行類別。在日誌檔中，訊息追蹤將類似如下：

```
==============
時間戳記：       2000-11-29 16:41:42.5
執行緒 ID：      <main>
類別：           MyTestingClass
方法：           main
嚴重程度：       1
訊息文字：       客戶訊息 "Hello"。
```

您可以執行類似測試，以看看第二種訊息。

執行流程追蹤

WebSphere Commerce 含有一個 ECTrace 類別，用以追蹤在 WebSphere Commerce Server 中執行之元件的執行流程。ECTrace 類別爲 com.ibm.commerce.ras 套件的一部份。

爲了除錯，在您建立新商業邏輯時，可在程式碼中插入一項追蹤，以追蹤方法。追蹤產生的資訊會擷取到追蹤日誌中。您可以指定追蹤的進出點。此外，您可以指定要追蹤這兩個點間的特定資料。

如果要使用追蹤特性，則必須針對您要進行追蹤的元件啓用追蹤特性。如果要針對特定元件啓用追蹤特性，您可使用「管理主控台」或「架構管理程式」。

在追蹤自訂程式碼期間，您*必須*使用 EXTERN 元件。 在「架構管理程式」中，這稱爲*外部*。

如果要在您的程式碼中設定追蹤的進入點，請使用下列語法：

```
ECTrace.entry (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName, myMethodName);
```

其中 *myClassName* 字串代表內含追蹤方法的類別。由於此字串可用來追蹤檔的剖析，因此應含有完整的類別名稱。 如果所追蹤的方法爲靜態，以 myClassName 爲例其宣告爲：

```
String myClassName = "com.mycompany.agrouping.MyTracedClass";
```

如果所追蹤的方法並非靜態，以 myClassName 爲例，其宣告爲：

```
String myClassName = this.getClass().getName();
```

如果要在方法中設定追蹤資料的追蹤點，請使用下列語法：

```
ECTrace.trace (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, myText);
```

其中 *myText* 爲將出現在追蹤日誌中的文字。

如果要在您的程式碼中設定追蹤的出口點，請使用下列語法：

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName);
```

如果您要追蹤所追蹤方法所傳回的物件，請依照下列的方法設定出口點：

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, returnedObject);
```

其中 returnedObject 代表方法所傳回的 Java 物件。

舉例來說，假設您想追蹤新控制程式指令 MyNewControllerCmd 的 performExecute 方法。下列的程式碼片段顯示如何在您的 performExecute 方法中使用 ECTrace 方法。

```
public void performExecute() throws ECException {
     ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN,
               this.getClass().getName(), "performExecute");

     super.performExecute();


//////////////////////////////////////////
//   您的一些商業邏輯                   //
//////////////////////////////////////////


     ECTrace.trace(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute",
          "My code is great!");

//////////////////////////////////////////
//   其它一些商業邏輯                   //
```

```
//////////////////////////////////////////


    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN,
              this.getClass().getName(), "performExecute");

}
```

當呼叫前述的 performExecute 方法時，追蹤日誌檔所擷取到的資訊如下：

```
==============
時間戳記：       2000-12-05 17:32:00.257
執行緒 ID：        <P=502832:O=0:CT>
元件：          EXTERN
類別：          com.mycompany.agrouping.MyNewControllerCmd
方法：          performExecute
追蹤：           ENTRY POINT
==============
時間戳記：       2000-12-05 17:32:00.257
執行緒 ID：        <P=502832:O=0:CT>
元件：          EXTERN
類別：          com.mycompany.agrouping.MyNewControllerCmd
方法：          performExecute
追蹤：           My code is great!
==============
時間戳記：      2000-12-05 17:32:00.258
執行緒 ID：        <P=502832:O=0:CT>
元件：          EXTERN
類別：          com.mycompany.agrouping.MyNewControllerCmd
方法：          performExecute
追蹤：          EXIT POINT
```

建議您僅對主要功能使用追蹤特性。追蹤特性不會針對多種語言啓用，這是因爲追蹤特性主要是供商店程式開發人員使用。此點與會針對多種語言啓用的訊息有別，這是因爲系統訊息是作爲管理用，而使用者訊息則會顯示在客戶面前。

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 指令的施行

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 135 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。

新指令必須施行其對應的介面（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施行類別。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施行類別延伸而來，以及其應施行哪一種介面：

指令類型	範例指令名稱	延伸自	施行範例介面
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: 施行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（採用現有的抽象施行類別與介面）之介面與施行類別間的關係。抽象類別與介面皆可在 com.ibm.commerce.command 套件中找到。

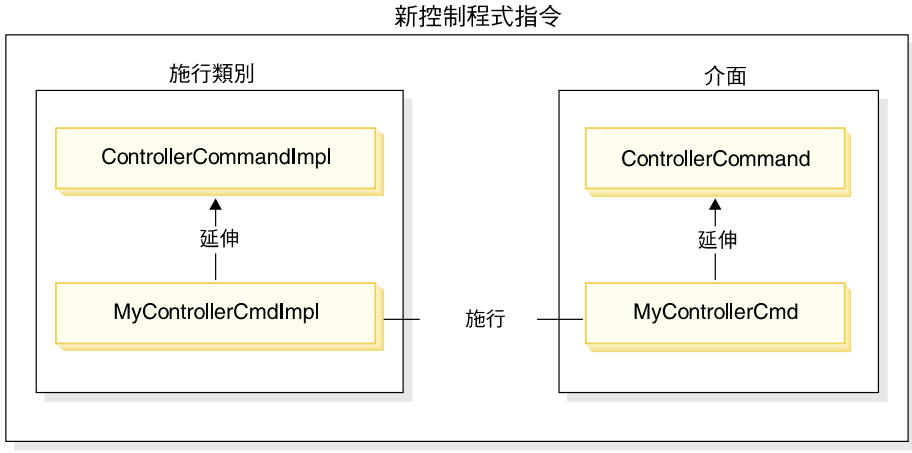


圖 25.

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

圖 26.

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

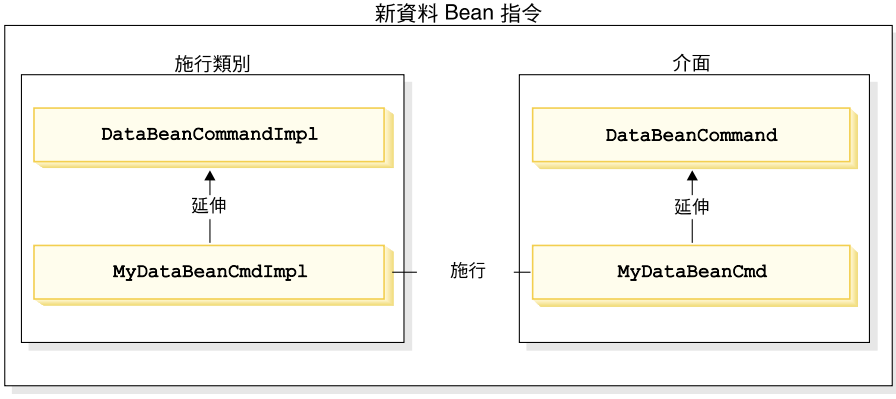


圖 27.

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從屬站。產品中會提供許多的通用檢視畫面指令，供您以不同的通信協定將回應給從屬站。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 檢視畫面指令會將從屬站導向到某個 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，並將頁面顯示給從屬站。

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*（傳給從屬站的回應）。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，自訂程式碼是放在套件與專案（有別於 WebSphere Commerce 中所提供的）中維護。

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。此外，這些套件應儲存在一個有別於 WebSphere Commerce 專案的專案中。舉例來說，上述套件可放在 `BigBusinessCustomCode` 專案中。

在您建立新資料 Bean 時，這些資料 Bean 必須置於有別於指令邏輯的套件中，不過此套件可置於儲存指令套件的專案中。以上例來說，是將 com.bigbusiness.databeans 套件置於 BigBusinessCustomCode 專案中。

在您建立新實體 Bean 時，這些實體 Bean 必須儲存在唯一的專案中。因此，您可能會有一個內含 com.bigbusiness.objects 套件的 BigBusinessCustomEntityBeans 專案。

爲了部署程式碼，此套件策略爲必要的。

指令環境定義

指令環境定義爲 Web 控制程式的控點。指令可使用指令環境定義取得 Web 控制程式中的資訊。舉例來說，可用的資訊包括：使用者 ID、使用者物件、語言識別碼以及商店識別碼。

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 控制程式會施行「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」的一部份，此方法背後的邏輯和 getLanguageId() 方法類似。

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）的完整詳述，請參閱 WebSphere Commerce 線上說明的「參照」區段。

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 施行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 91 頁的『存取控制的互動』。

getResources 方法

此方法用來施行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 79 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：

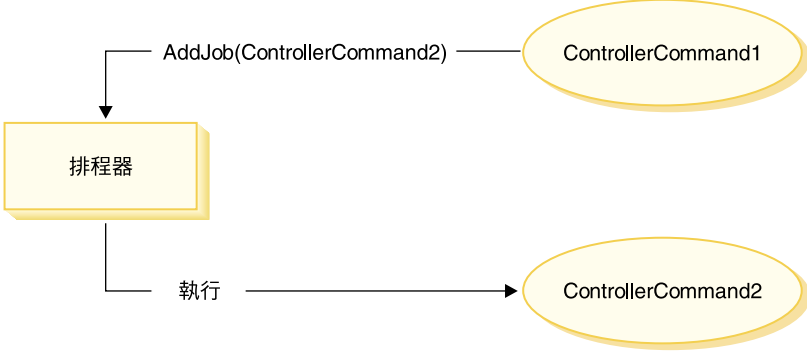


圖 28.

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，從屬站通常會輪詢 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 40 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，如果從屬站使用 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 施行類別（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 施行類別。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 施行類別。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 施行類別，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。這會當成通用的轉遞檢視畫面般施行。

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，且會施行一個延伸 com.ibm.commerce.TaskCommand 介面的介面。如同第 116 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，而非作業指令的施行類別。這可讓您擁有作業指令的多種施行方式（每一家商店一種）而呼叫端不需顧慮要呼叫哪個施行類別。

定義於介面中的所有方法必須置於施行類別中施行。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。

除了施行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，作業指令本身可施行 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：

- 新增其它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。

新增新商業邏輯到控制程式指令中

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，施行類別名稱爲 ExistingControllerCmdImpl。如今因某項商業需求，您必須在現有指令中新增新商業邏輯。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。

新增新商業邏輯的第一步是建立新施行類別，且此施行類別是從原始施行類別延伸而來。在本例中，您將建立從 ExistingControllerCmdImpl 延伸而來的新 ModifiedControllerCmdImpl 類別。新施行類別應施行原始介面 (ExistingControllerCmd)。

在新施行類別中，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案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，以新增新商業邏輯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案例化新作業指令，以新增新商業邏輯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施行類別，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施行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

不是新增新商業邏輯到控制程式指令的開頭或尾端）。在此情況下，您必須將您想改寫之作業指令的案例化，換成新作業指令的案例化，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施行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，來案例化作業指令。指令 factory 會使用作業指令的介面名稱，然後根據指令登錄，決定正確的施行類別。因此，如果要更換已案例化的作業指令，您必須建立新作業指令施行類別，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施行類別。相關資訊請參閱第 132 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施行類別。舉例來說，建立一個由 ExistingControllerCmdImpl 延伸而來並施行 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 線上說明中的「參照」區段。

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：

- 新增其它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施行類別。下列各節將進一步詳述。

新增新商業邏輯到作業指令中

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，施行類別名稱爲 ExistingTaskCmdImpl。如今因某項商業需求，您必須在現有指令中新增新商業邏輯。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。

新增新商業邏輯的第一步是建立新施行類別，且此施行類別是從原始施行類別延伸而來。在本例中，您將建立從 ExistingTaskCmdImpl 延伸而來的新 ModifiedTaskCmdImpl 類別。新施行類別應施行原始介面 (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。

下列的虛擬程式碼示範如何新增新商業邏輯到現有作業指令中：

```
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，以便讓新施行類別連結現有介面。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施行類別。這個新的施行類別必須繼承自現有的作業指令，但是不要施行現有的介面。此外，在新的施行類別中，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 線上說明中的「參照」區段，以確定新施行類別所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 VisualAge for Java 產生）可提供一種簡單方法來存取實體 Bean 中的資訊。當您修改實體 Bean 時（例如：新增新欄位、新商業方法或新搜尋器），只要重新產生存取 Bean，即會將更新反映於存取 Bean 中。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，因而不需進行編碼讓資料 Bean 採用實體 Bean 中的新屬性。

如果您想在非從實體 Bean 衍生而來的資料 Bean 中新增新屬性，您可以使用 Java 繼承特性來延伸現有資料 Bean。舉例來說，如果您想新增新欄位到 OrderDataBean 中，請依照下列的方法定義 MyOrderDataBean：

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  在此施行新欄位
     }

}
```

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
{

}
```

VisualAge for Java 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：合約與 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擷取。

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：

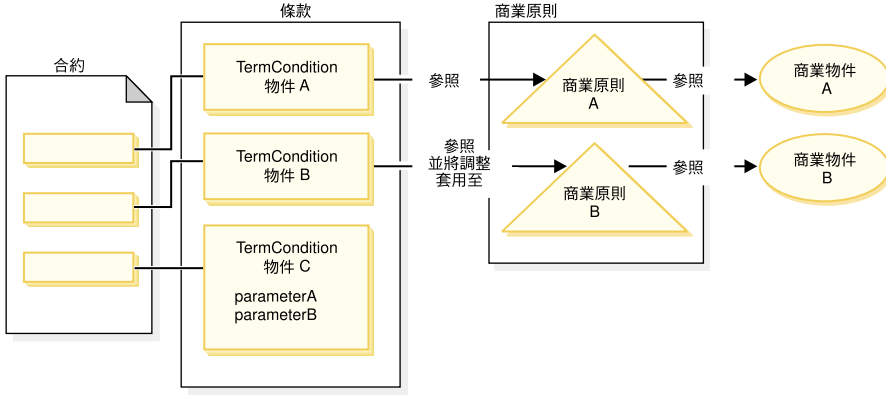


圖 29.

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，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，TermCondition 物件 B 可能代表價格條款物件，套用商業原則 B 所定義之價格的 50% 折扣。在此情況下，商業原則 B 是價格原則，商業物件 B 是定義主要種類的交易狀態的交易狀態儲存器。

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 137 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
  用以施行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()` 指令取得價格條款。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施行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：

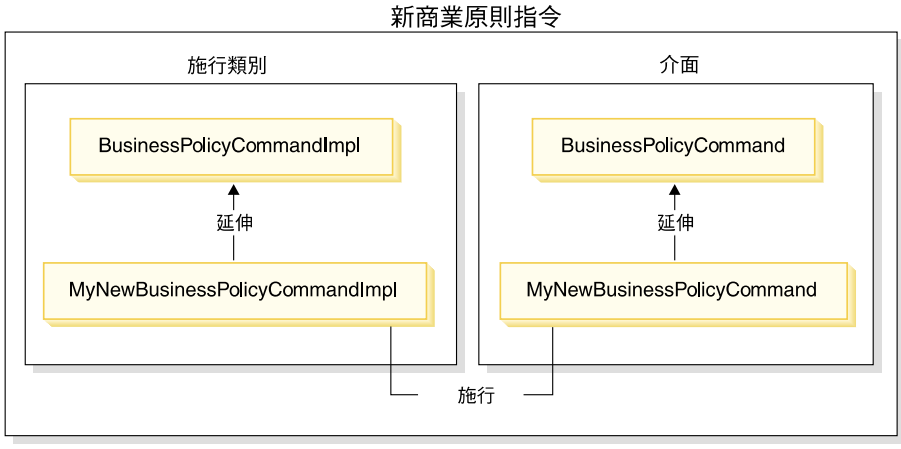


圖 30.

如上圖所示，如果要建立新商業原則指令，您將建立新施行類別，且這個施行類別是從 WebSphere Commerce BusinessPolicyCmdImpl 施行類別延伸而來。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。另外要注意的是，當安裝範例時，則任何唯一識別碼（如 CONTRACT_ID）的值皆可能和此處所示的不同。

CONTRACT 表格範例資料

下列表格顯示「工具屋」 CONTRACT 資料庫表格中的相關範例資料。請注意爲了顯示的目的，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

直欄名稱	範例資料
MEMBER_ID	-2001
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

TERMCOND 表格範例資料

下列表格顯示「工具屋」TERMCOND 資料庫表格中的相關範例資料。請注意，爲方便閱讀，資料庫直欄標頭會顯示在第一欄，表格中的範例資料列則顯示於第二與第三欄。

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

下列表格顯示「工具屋」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

下列表格顯示「工具屋」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

下列表格顯示「工具屋」TRADEPOSCN 資料庫表格中的相關範例資料。

	直欄名稱			
	READEPOSCN_ID	MEMBER_ID	NAME	TYPE
範例資料列	10051	-2001	ToolTech	S

SHIPMODE 表格範例資料

下列表格顯示「工具屋」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。
   下列的作業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. 關聯條款物件至新商業原則
   藉由關聯現有的條款物件至新商業原則或建立新條款物件來完成。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 DTD（文件類型定義）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。

建立新商業原則

一般而言，建立新商業原則所涉及的作業包括：將唯一商業原則登錄在資料庫中，以及建立新商業原則指令。

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。雖然施行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施行此處指定的介面。

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.');
```

建立新商業原則類型的最後一個步驟是，您可以編寫一或多個新商業原則類型介面。這些介面是供隸屬此商業原則類型之領域下的商業原則指令施行。舉例來說，在「工具屋」範例商店中，「價格」即被定義成一種商業原則類型。因此，會有 com.ibm.commerce.price.commands.ResolvePriceListsCmd 與 com.ibm.commerce.price.commands.RetrievePricesCmd 介面供所有和價格有關的商業原則指令施行。

如果您沒有商業原則指令用來對新商業原則類型執行作業，則不會要求您建立新介面。這種情況相當少見；在大部份情況下，當您建立新商業原則類型時，您亦必須建立新商業原則類型介面。

在您建立商業原則類型介面時，新介面必須自 com.ibm.commerce.command.BusinessPolicyCommand 介面延伸而來。

撰寫新商業原則指令

如果要建立新商業原則指令，您必須建立新指令，且這個指令會施行指令之相關商業原則類型的介面。新指令亦必須從 com.ibm.commerce.command.BusinessPolicyCommandImpl 施行類別延伸而來。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）明確設定其它必要的內容。

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，施行類別名稱爲 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();
}
```

此商業原則指令的施行類別被定義成：

```
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。
      /* 一旦案例化，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它任何必要內容。
      */

      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"));
         }
}
```

用以呼叫新商業原則指令的指令可採用類似下列方式來定義：

```
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，您必須登錄原則與施行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
  施行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。您可以關聯現有的條款物件至新的商業原則 (藉由擷取 B2BTrading.dtd 檔中現有的條款物件和新商業原則之間的關係來完成)，或是建立新的條款物件來關聯新的商業原則。

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 B2BTrading.dtd 檔，以反映新條款。
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 陳述式範例顯示如何更新綱目：

```
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');
```

在合約文件類型定義中登錄新條款

B2BTrading.dtd 是文件類型定義 (DTD) 檔，指定可在商業原則內使用的各種條款。如果要建立新條款可在合約中使用，必須更新此檔案以併入新的條款。

當您建立了新條款，必須將新條款新增到 TermCondition 定義中，並建立一個新元素來說明此條款。

如果要更新 B2BTrading.dtd 檔，請執行下列步驟：

1. 導覽至下列目錄：
   - Windows *drive*:\WebSphere\CommerceServer\xml\trading
   - AIX /usr/WebSphere/CommerceServer/xml/trading
   - Solaris /opt/WebSphere/CommerceServer/xml/trading
   - Linux /opt/WebSphere/CommerceServer/xml/trading
   - 400 /QIBM/ProdData/WebCommerce/xml/trading
2. 開啓 B2BTrading.dtd 檔。
3. 以新條款更新 TermCondition 定義。舉例來說，下列 TermCondition 定義中以粗體字顯示更新部份。

   ```
   <!ELEMENT TermCondition (TermConditionDescription?,Participant*,
   CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
   PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
   PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
   OrderTC|MyTC))>
   ```

   請注意，折行只爲了方便閱讀。
4. 此時新增新元素到 B2BTrading.dtd 檔中。舉例來說，下列顯示有關新增 MyTC 元素的更新，而此元素會參照商業原則，且有兩個必要屬性。

   ```
   <!ELEMENT MyTC (MySubTC)>
   <!ELEMENT MySubTC (PolicyReference)>
   <!ATTLIST MySubTC
      attr1 CDATA #REQUIRED
      attr2 CDATA #REQUIRED
   >
   ```
5. 儲存檔案。

為條款建立新 CMP Enterprise Bean

您必須爲條款物件建立新 CMP Enterprise Bean。此 Bean 是針對條款子類型而建。

請注意，在您建立新 Enterprise Bean 時，一般而言，是將 Bean 置於您自己的 EJB 群組中，而非將之置於內含 WebSphere Commerce 實體 Bean 的 EJB 群組之一中。不過，在此情況下，由於條款的所有新實體 Bean 皆必須繼承自 WebSphere Commerce TermCondition Bean，因此您必須將新條款 Bean 置於 WCS Contract EJB 群組中。

如果要爲條款物件建立新 CMP Enterprise Bean，請在 VisualAge for Java 中執行下列步驟：

1. 執行下列步驟，使用精靈來建立新 Enterprise Bean：
   a. 在 VisualAge for Java 的「工作台」中，選取 EJB 標籤。
   b. 選取並以滑鼠右鍵按一下 **WCS Contract** EJB 群組，並選取**新增 > 採用繼承的 Enterprise Bean**。
      會開啓「建立採用繼承的 Enterprise Bean」引導精靈。
   c. 在引導精靈中，輸入您 Bean 的適當資訊。舉例來說，下表顯示範例值。

屬性	值
Bean 名稱	MySubTC
繼承自	TermCondition
套件	com.ibm.commerce.contract.objects
Bean 類別	MySubTCBean
遠端介面	MySubTC
發源介面	MySubTCHome

   d. 按一下**新增**以新增 CMP 欄位到 Bean 中，並視需要爲 Bean 建立新欄位。以本例來說，請使用下列資訊建立兩個新 CMP 欄位：

屬性	值
欄位名稱	attr1
欄位類型	字串
使用 getter 與 setter 方法存取	啓用
將 getter 與 setter 方法引介到遠端介面中	啓用

屬性	值
欄位名稱	attr2

屬性	值
欄位類型	整數
使用 getter 與 setter 方法存取	啓用
將 getter 與 setter 方法引介到遠端介面中	啓用

e. 按一下**完成**。

2. 下個步驟是將新 Bean 中的欄位映射至 TERMCOND 表格中的直欄。如果要建立這個映射資訊，請執行下列步驟：

   a. 從 **EJB** 功能表中選取**開啓到 > 綱目映射**。
   會開啓「映射瀏覽器」。

   b. 在映射瀏覽器的「資料儲存庫映射」畫面中，按兩下 **WCS Contract**。

   c. 在「持續性類別」畫面中，按兩下 **TermCondition**，然後選取 **MySubTC**。

   d. 從「表格映射」功能表中選取**新表格映射 > 新增單一繼承表格映射**。
   會開啓「單一繼承表格映射編輯程式」。

   e. 在**判別碼之值**欄位中，輸入 TCSUBTYPE_ID 值。以本例來說，請輸入 `'MySubTC'`（請包含引號），並按一下**確定**。

   f. 確定「持續性類別」畫面中 **MySubTC** 仍被選取。在「表格映射」畫面中，選取並以滑鼠右鍵按一下 TERMCOND 表格。選取「編輯內容映射」。
   會開啓「內容映射編輯程式」。

   g. 在「內容映射編輯程式」中，按如下所示來設定屬性：

類別屬性	映射類型	表格直欄
attr1	Simple	STRINGFIELD2
attr2	Simple	INTEGERFIELD1

然後按一下**確定**。

   h. 從「資料儲存庫映射」功能中，選取「儲存資料儲存庫映射」。在儲存映射時，請輸入下列資訊：

屬性	值
專案	`IBM WCS Enterprise Bean`
套件	保留的 `WCSContract EJB`
類別名稱	`WCSContractMap`

按一下**完成**，並關閉「映射瀏覽器」。

3. 依下列所示，在新 Enterprise Bean 中（亦即，在 MySubTCBean 中），建立一個新 ejbCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement) 方法：

```
public void ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException {
   _initLinks();
   super.ejbCreate (argTradingId, argElement);
   this.attr1= null;
   this.attr2= null;
}
```

4. 按如下所示建立新 ejbPostCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement) 方法：

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException
   {
   parseXMLElement(argElement);
   }
```

5. 依下列所示，改寫 MySubTCBean 中的 parseXMLElement(org.w3c.dom.Element argElement) 方法：

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException,
   javax.ejb.RemoveException
   {
      super.parseXMLElement(argElement);
      if (argElement == null)
     return;

      String nodeName = argElement.getNodeName();
      if (nodeName.equals("TCCopy"))
     return;
      // 取得元素 "MyTC"
      Element eMyTC = ContractUtil.getElementByTag(argElement,"MyTC");

      // 從元素 "MyTC" 取得元素 "MySubTC"
      Element eMySubTC = ContractUtil.getElementByTag(eMyTC ,"MySubTC");
      this.attr1 = eMySubTC .getAttribute("attr1").trim();
      this.attr2 = new Integer (eMySubTC .getAttribute("attr2").trim());

      // 從 "MySubTC" 取得元素 "PolicyReference"
      Element ePolicyReference = ContractUtil.getElementByTag(eMySubTC,
```

```
            "PolicyReference");
        parseElementPolicyReference(ePolicyReference);

}
```

6. 依下列所示，改寫 MySubTCBean 中的 createNewVersion(Long argNewTradingId) 方法：

```
public Long createNewVersion(Long argNewTradingId) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   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.toString());
   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;
}
```

7. 依下列所示，改寫 MySubTCBean 中的 getXMLString() 方法：

```
public String getXMLString() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException
   {
      String xmlTC = "      <MyTC>"  +
                     "%XML_POLICYREFERENCE%" +
                     "         <MySubTC attr1=\"" + this.attr1 +
                     "\"  attr2=\"" + this.attr2.toString() + "\"/>"
                     "          '>" +
                     "      <MYTC></MYTC>" ;
      String xmlPolicy = getXMLStringForElementPolicyReference(
         "ProductSet") ;
      xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
         xmlPolicy );

      return xmlTC;
}
```

8. 依下列所示，改寫 MySubTCBean 中的 markForDelete() 方法：

```
public void markForDelete() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   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
}
```

9. 確定 ejbCreate 方法已新增到發源介面中，且其它所有已修改的方法已新增到遠端介面中。
10. 下個步驟是建立 MySubTC 實體 Bean 的存取 Bean；請執行下列步驟：
    a. 以滑鼠右鍵按一下 **MySubTC** 實體 Bean，並選取**新增 > 存取 Bean**。會開啓「建立存取 Bean」引導精靈。
    b. 確定您已輸入下列資訊：

表 1.

屬性	值
EJB 群組	WCSContract
Enterprise Bean	MySubTC
存取 Bean 名稱	MySubTCAccessBean
存取 Bean 類型	實體 Bean 的 Copy Helper

然後按一下**下一步**。

c. 從**選取零引數建構子的發源方法**下拉清單中選取 **findByPrimaryKey(TermConditionKey)**
d. 在 **initKey_referenceNumber**（位於「起始內容」直欄中）方面，將 **Converter** 設爲 com.ibm.commerce.base.objects.WCStringConverter，並按一下**下一步**。
e. 針對所有已新增的新欄位方面，確定已選出 **CopyHelper**，並將每一個的 Converter 值設爲 com.ibm.commerce.base.objects.WCStringConverter。按一下**完成**。
在完成程式碼的產生後，您即可檢視新程式碼；方法是切換至**內容**標籤，展開 **IBM WCS Enterprise Bean** 專案，然後展開 **com.ibm.commerce.contract.objects** 套件。

11. 回到 EJB 標籤，以滑鼠右鍵按一下 **MySubTC** Enterprise Bean，並選取**產生部署程式碼**。
12. 您也必須爲上層 Bean (TermCondition Bean) 與所有的同層 Bean （WCS Contract EJB 群組中，其名稱中含有 “TC” 的其它所有 Bean），重新產生部

署程式碼。請注意，如果您曾新增新欄位，或曾修改現有 TermCondition Bean 的遠端介面，您得爲其本身與其所有下層 Bean 重新產生存取 Bean。
如果要重新產生部署程式碼，請執行下列步驟：

a. 選取 TermCondition Bean，以及選取其名稱中含有 "TC" 的其它所有 Bean（例如 DisplayCustomizationTC、FulfillmentTC 與 InvoiceTC 即爲部份的同層 Bean）。

b. 全部選出這些 Bean，然後以滑鼠右鍵按一下以選取**產生部署程式碼**。

13. 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。分別是：

- 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，就此方法的預設施行而言，合約中至少需有一項 PriceTC。
- otherValidateCheck()
  就此方法的預設施行而言爲空的。您可以另外新增其它不屬於前兩種方法範圍的檢驗。

14. 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：

a. 在本例中，新部署指令介面爲 MySubTCDeployedCmd，施行類別爲 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');
```

b. 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 的值爲 MySubTCAccessBean，且 getTargetStoreId() 方法會傳回要部署合約之商店的識別碼。

c. 在相同範例中，建立 MySubTCDeployCmdImpl 施行類別。此施行類別必須從 com.ibm.commerce.contract.commands.DeployTCCmdImpl 延伸而來。以下說明新指令施行類別：

```
public class MySubTCDeployCmdImpl
   extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
   implements MySubTCDeployCmd
   {
       // 客戶代碼
   }
```

更新 WebSphere Commerce Accelerator 使用新條款

一旦您建立新條款後，可更新 WebSphere Commerce Accelerator，以便用來建立內含這些新條款的新合約。如果要更新 WebSphere Commerce Accelerator 以達到此目的，請執行下列步驟：

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 *drive*:\WebSphere\CommerceServer\samples\contract\ Extensions.js
- Windows *drive*:\WebSphere\CommerceServerDev\samples\contract\ Extensions.js
- AIX /usr/WebSphere/CommerceServer/samples/contract/Extensions.js
- Solaris /opt/WebSphere/CommerceServer/samples/contract/Extensions.js
- Linux /opt/WebSphere/CommerceServer/samples/contract/Extensions.js
- 400 /QIBM/ProdData/WebCommerce/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 *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wctools.war\ javascript\tools\contract
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/ javascript/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/ javascript/tools/contract
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/ javascript/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/javascript/tools/contract

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。其做法必須和您在 B2BTrading.dtd 檔中所做的延伸一致。請見下列的程式碼片段：

```
function submitMyTC(termsAndConditions) {

   var tcModel = get("ContractMyTCModel");

   if (tcModel != null) {

      var myTC = new Object();
      myTC.MyTC = new Object();
      myTC.MyTC.MySubTC = new Object();
      myTC.MyTC.MySubTC.attr1 = tcModel.attr1;
      myTC.MyTC.MySubTC.attr2 = tcModel.attr2;

      myTC.MyTC.PolicyReference = new Object();
      myTC.MyTC.PolicyReference.policyName =
         tcModel.policyList[tcModel.selectedPolicyIndex].policyName;
      myTC.MyTC.PolicyReference.policyType = "ProductSet";
      myTC.MyTC.PolicyReference.storeIdentity =
```

```
            tcModel.policyList[tcModel.selectedPolicyIndex].storeIdentity;
         myTC.MyTC.PolicyReference.Member =
            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";
         }

         termsAndConditions[termsAndConditions.length] = 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 *drive*:\WebSphere\CommerceServer\samples\contract\ContractMyTC.jsp
- Windows *drive*:\WebSphere\CommerceServerDev\samples\contract\ContractMyTC.jsp
- AIX /usr/WebSphere/CommerceServer/samples/contract/ ContractMyTC.jsp
- Solaris /opt/WebSphere/CommerceServer/samples/contract/ ContractMyTC.jsp
- Linux /opt/WebSphere/CommerceServer/samples/contract/ ContractMyTC.jsp
- 400 /QIBM/ProdData/WebCommerce/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 *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wctools.war\tools\contract
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract

- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract

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 *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\properties\com\ibm\commerce\tools\contract\properties\ ContractRB_*locale*.properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties

- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties

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 *drive*:\WebSphere\CommerceServer\xml\tools\contract\ContractNotebook.xml
- AIX /usr/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- Solaris /opt/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- Linux /opt/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/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：

```
<TermCondition>
   <MyTC>
      <MySubTC attr1="adc" attr2="123" />
         <PolicyReference policyName = "Product Set 1"
                          policyType = "ProductSet"
                          storeIdentity = "StoreGroup1" >
            <Member>
               <User distinguishName = "uid=wcsadmin,o=Root Organization"/>
            </Member>
         </PolicyReference>
      </MyTC>
 </TermCondition>
```

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，以判斷施行該原則的指令。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

完全整合合約模型的延伸到您的商業程序的下一步就是建立包含參照新商業原則的條款的合約。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。有關建立合約的詳細資訊，請參閱 WebSphere Commerce 線上說明。

合約自訂實務

本節提供下列合約自訂實務所涉及的步驟概觀：

- 啓用折讓

折讓實務

在這個範例實務中，會建立定額折讓。由於「工具屋」範例商店並無符合折讓實務的條款與原則類型，因此必須建立這些。此外，必須建立新商業原則，以及用來儲存折讓代碼的資料庫表格。

施行此折讓實務包含下列的高階步驟：

1. 建立 XREBATECODE 資料庫表格與對應的 XRebateCodeBean 實體 Bean，以用來存取此表格中的資訊。
2. 執行下列的子作業，建立 5DollarRebate 新商業原則：
   a. 建立對應的新商業原則類型。這是定義新商業原則指令將施行的介面 (RebatePolicyCmd)。
   b. 建立新 CalculateRebateCmdImpl 商業原則指令。
   c. 將新商業原則指令與商業原則類型登錄在資料庫中。
3. 執行下列的子作業，爲折讓建立新條款 (RebateTC)：
   a. 將 RebateTC 條款登錄在資料庫中。
   b. 更新 `B2BTrading.dtd` 檔，以反映新 RebateTC。
   c. 爲 RebateTC 建立新 Enterprise Bean。
   d. 更新 WebSphere Commerce Accelerator 以反映新 RebateTC。
4. 建立使用 RebateTC 的新合約。
5. 將新商業原則整合到購物流程中。

後續各節將詳述上述的每一個步驟。

步驟 1：建立新表格與 Enterprise Bean
由於現有資料庫綱目不含折讓金額與代碼的指定，因此必須建立新表格。一般而言，當建立新表格時，亦會建立新實體 Bean，以用來存取此表格中的資訊。

以本例來說，假設所建立的資料庫表格爲 XREBATECODE。

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，將建立新 CMP 實體 Bean (XRebateCodeBean)。有關建立此 Bean 的詳細資訊，請參閱第 54 頁的『建立新 CMP Enterprise Bean』。

步驟 2：建立 “5DollarRebate” 商業原則
若要建立這個新商業原則，您必須執行下列步驟：

1. 建立新商業原則類型介面。此爲 `CalculateRebateCmdImpl` 所要施行的 `RebatePolicyCmd` 介面。
2. 建立 `CalculateRebateCmdImpl` 新商業原則指令。
3. 將新商業原則與商業原則指令登錄在資料庫中。

建立 “Rebate” 商業原則類型: 由於尚無一個對應至折讓的現有商業原則類型，因此必須建立一個新的。建立新商業原則類型，涉及到定義原則類型，並登錄在資料庫中。下列表格必須更新：

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

就此實務來說，爲了建立 REBATE 新原則類型，將使用下列的 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.');
```

如此一來，下列表格顯示 PLCYTYCMIF 表格中，顯示原則類型與相關商業原則指令間之關係的相關直欄。

表 3. PLCYTYCMIF 表格的更新

	直欄名稱	
	POLICYTYPE_ID	BUSINESSCMDIF
範例資料	Rebate	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

您也必須編寫 RebatePolicyCmd 新介面。此介面必須從 com.ibm.commerce.command.BusinessPolicyCommand 介面延伸而來。依上述表格的建議，請將此介面納入您自己的套件中。

建立 CalculateRebateCmdImpl 商業原則指令: 爲了建立新商業原則指令，您必須建立一個新指令 CalculateRebateCmdImpl，且這個指令是從 com.ibm.commerce.command.BusinessPolicyCommandImpl 施行類別延伸而來。此指令應施行您在上述步驟中所建的 RebatePolicyCmd 介面。

請注意，在本例中，介面名稱與指令名稱並不相同。選擇這些名稱，是有意指出施行「折讓」商業原則類型的商業原則指令可能很多。每一種施行（亦即，每一個商業原則指令）都會以唯一方式來施行「折讓」。

指令的邏輯，視客戶如何拿取商品的特定施行方式而定。此外，此 CalculateRebateCmdImpl 應由應用程式中個別的控制程式或作業指令所呼叫。

登錄新商業原則與新商業原則指令: 新商業原則必須登錄在資料庫中。您也必須登錄新商業原則與新商業原則指令間的關係。

如果要登錄此資訊，您可以使用 com.ibm.commerce.contract.commands.PolicyAddCmd 指令。以下針對此實務，顯示 PolicyAdd 指令的用法範例：

```
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，在輸入內容方面，URL 保留字元必須換成其 ASCII 碼。因此，符號 =（等號）應換成 “%3D”， & 換成 “%26”，空格字元換成 “%20”。前述範例中所用的日期格式爲 yyyy-mm-dd hh:mm:ss，並以 ASCII 碼取代 URL 保留字元。

下表顯示在執行更新後，受影響資料庫表格中的相關直欄。

表 4. POLICY 表格的更新

	直欄名稱				
	POLICY_ID	POLICY NAME	POLICYTYPE_ID	STOREENT_ID	PROPERTIES
範例資料	301	5 Dollar Rebate	Rebate	-1	rebatecode_id= 201

請注意，在此亦假定起始日期與結束日期值皆設爲空值。

表 5. POLICYCMD 表格的更新

	直欄名稱		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
範例資料	301	com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl	空值

因此，目前您已有一個和 CalculateRebateCmd 商業原則指令相關的新商業原則 “5DollarRebate”。

步驟 3：建立 “RebateTC” 條款

若要建立 “RebateTC” 條款，需執行下列步驟：

1. 將 RebateTC 條款登錄在資料庫中。
2. 更新 B2BTrading.dtd 檔，以反映新 RebateTC。
3. 爲 RebateTC 建立新 Enterprise Bean。
4. 更新 WebSphere Commerce Accelerator 以反映新 RebateTC。

將 “RebateTC” 條款登錄在資料庫中: 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將 RebateTC 登錄在資料庫中：

```
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);
```

下表取自 TCTYPE 與 TCSUBTYPE 表格中的相關直欄。

表 6. TCTYPE 表格的更新

	直欄名稱
	TCTYPE_ID
範例資料	RebateTC

表 7. TCSUBTYPE 表格的更新

	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

更新 B2BTrading.dtd 檔，以反映新 RebateTC: 爲了讓新條款可用於合約中，您必須更新 B2BTrading.dtd 檔，以包含新條款。在您更新此檔案時，您必須將新條款新增到 TermCondition 定義中，然後建立一個說明該條款的新元素。

下列的粗體文字範例是顯示如何新增新 RebateTC 到 TermCondition 定義中：

```
<!ELEMENT TermCondition (TermConditionDescription?,Participant*,
CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
OrderTC|RebateTC))>
```

請注意，折行只爲了方便閱讀。

接著，您必須在檔案中加入說明此條款的新段落。以下是 RebateTC 的範例：

```
<!ELEMENT RebateTC (PolicyReference?)>
```

為 RebateTC 建立新 Enterprise Bean: 您必須爲新 RebateTC 建立新 Enterprise Bean。這個新 Bean 必須繼承自 WebSphere Commerce TermCondition Bean。

一般而言，條款的新 Enterprise Bean 是以子類型命名。請注意，在本例中，條款子類型和條款類型相同，因此 Bean 的名稱和條款類型同名。

下表顯示必須建立的新 Bean 的一些一般資訊。有關 Bean 的詳述（包括置換的方法），請參閱第 147 頁的『爲條款建立新 CMP Enterprise Bean』。

表 8.

屬性	值
Bean 名稱	RebateTC
繼承自	TermCondition
套件	com.ibm.commerce.contract.objects
Bean 類別	RebateTCBean
遠端介面	RebateTC
發源介面	RebateTCHome

更新 WebSphere Commerce Accelerator 以包含 RebateTC: 一旦您建立新條款後，可更新 WebSphere Commerce Accelerator，以便用來建立內含這些新條款的新合約。有關如何更新此工具的資訊，請參閱第 153 頁的『更新 WebSphere Commerce Accelerator 使用新條款』。

步驟 4：建立新合約

您必須建立一個包含 "RebateTC" 條款並參照 "5DollarRebate" 商業原則的新合約。您可以使用 WebSphere Commerce Accelerator 或 XML 來建立新合約。有關用以建立新合約的這些方法的說明，請參閱 WebSphere Commerce 線上說明。

下表顯示在建立合約後，對 TERMCOND 與 POLICYTC 資料庫表格中之相關直欄所做的更新。

表 9. TERMCOND 表格的更新

	直欄名稱		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
範例資料	25	901	RebateTC

表 10. POLICYTC 表格的更新

	直欄名稱	
	POLICY _ID	TERMCOND_ID
範例資料	301	901

步驟 5：將新商業原則整合到購物流程中

在此實務中，假設是要新增新網頁到商店中，以容許客戶登入並要求折讓。當客戶按一下以要求折讓時，應會呼叫一個會呼叫 RebatePolicyCmd 新介面的指令。

舉例來說，可能有一個會呼叫 `RebatePolicyCmd` 的 ClaimRebateCmd 新控制程式指令。接著，會找出正確的商業原則，並套用 “5DollarRebate” 商業原則（就本例而言）。

第 3 篇 開發環境

第 8 章 開發工具與部署

本章介紹自訂 WebSphere Commerce 應用程式所用的一些主要開發工具。其中說明將自訂程式碼從 VisualAge for Java 部署到 WebSphere Commerce Server 的程序。此外亦會說明如何將程式碼部署到 Commerce Studio，以便在開發 JSP 範本時能發揮自訂程式碼的優點。

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。

WebSphere Commerce Studio Business Developer Edition 與 WebSphere Commerce Studio Professional Developer Edition 皆可讓您選擇包含範例 WebSphere Commerce 商店，以用於 VisualAge for Java 的 WebSphere Test Environment 元件中。這可簡化開發環境的架構，這是因爲程式開發人員不必使用 WebSphere Commerce 中的工具來建立商店，再將商店資產移至開發環境中。

開發環境另一個關鍵元件爲 WebSphere Commerce 程式碼的儲存庫。在完成產品安裝後，這個儲存庫必須匯入到 VisualAge for Java 工作區中。而在匯入這個儲存庫後，程式開發人員必須執行和 WebSphere Test Environment 有關的一些架構步驟。

在完成所有安裝與架構作業後，程式開發人員會有一個獨立的開發機器，供其建立與測試自訂的 WebSphere Commerce 程式碼。程式開發人員不必將 WebSphere Commerce 安裝在開發機器上。

WebSphere Commerce Studio

WebSphere Commerce Studio Business Developer Edition 與 WebSphere Commerce Studio Professional Developer Edition 皆包含 VisualAge for Java 企業版 4.0。此版 VisualAge for Java 含有一些特性，像是用來開發與除錯進階 JSP 範本的強大工具，可協助您開發商店前端資產。此外，在強化與 WebSphere Studio 間的整合下，使得新增內容到這些 JSP 範本的速度加快，進而提高程式設計師與 Web 開發

人員的產能。在商店後端應用程式碼的開發方面，它可支援 Enterprise JavaBeans 技術與連結性，而可和其它系統（如：CICS® TS、MQSeries、SAP R/3 等）整合。此外，所整合的 VisualAge for Java 企業版 WebSphere Test Environment 可讓程式開發人員在不需結束 VisualAge for Java 下執行 WebSphere Commerce 功能。也就是說，您用不著將程式碼部署到 WebSphere Commerce Server 中，即可測試自訂的 WebSphere Commerce 程式碼。

註: VisualAge for Java 企業版 4.0 的 WebSphere Test Environment 元件所執行的是 WebSphere Application Server V3.5.4 中的應用程式。請注意，WebSphere Commerce Business Edition 與 WebSphere Commerce Professional Edition 所用的是 WebSphere Application Server V4.0。在此項版本間的差異下，如果您要將新或修改過的 Enterprise Bean 部署到執行於 WebSphere Application Server V4.0 的 WebSphere Commerce 案例中，您必須使用 WebSphere Application Server V4.0 所附的 EJBDeploy 工具，在 VisualAge for Java 外產生部署程式碼。事實上，此部署程式碼必須產生於已安裝 WebSphere Application Server V4.0 的機器上。詳細說明請參閱第 171 頁的『EJB 部署程式碼相關資訊』。

如果您想完成第 181 頁的第 4 篇, 『指導教學』中的指導教學，您必須安裝 WebSphere Commerce Studio Business Developer Edition 或 WebSphere Commerce Studio Professional Developer Edition。有關安裝 Commerce Studio 與架構 VisualAge for Java 的進一步資訊，請參閱 *WebSphere Commerce Studio Business Developer Edition 安裝手冊*或 *WebSphere Commerce Studio Professional Developer Edition 安裝手冊*。

VisualAge for Java 的特性與功能

本*程式設計手冊*不會教您如何使用 VisualAge for Java。在該產品方面，本手冊通常偏向於說明如何在 VisualAge for Java 中執行作業，較少說明如何完成程式設計作業以建立 WebSphere Commerce 的電子商務應用程式。因此，如果您是初次使用 VisualAge for Java，您可藉由執行本書中的指導教學，開始學習如何執行 VisualAge for Java 中的一些作業。不過，您必須參閱 VisualAge for Java 文件、指導教學與課程，以熟悉此工具的使用。

舉例來說，（選用）使用 VisualAge for Java 中的除錯器指導教學主題所提供的快速簡介，會說明在執行程式碼期間如何使用除錯器來檢視作業指令中的變數值。VisualAge for Java 的除錯器元件是一種強大的除錯工具，因此您應參閱 VisualAge for Java 文件，以取得其特性與用法的詳述。

WebSphere Commerce 程式碼儲存庫

爲了建立 WebSphere Commerce 應用程式的自訂程式碼，您必須將 WebSphere Commerce 程式碼儲存庫匯入到 VisualAge for Java 工作區中。 WebSphere Commerce Business Edition Disk 2 CD 與 WebSphere Commerce Professional Edition Disk 2 CD 會提供儲存庫。現行儲存庫（本書出版時）的名稱爲 `WC_54.dat`。

程式碼部署

在您自訂 e-commerce 應用程式時，您可執行下列任何作業：

- 建立新指令、資料 Bean 或實體 Bean
- 延伸現有的 WebSphere Commerce 實體 Bean
- 修改現有指令或資料 Bean 的邏輯

當您在 VisualAge for Java 中開發程式碼時，您可以在 WebSphere Test Environment 中測試您的程式碼。在某些情況中，您必須將程式碼部署到開發環境外的 WebSphere Commerce Server 中。

在下列各節中，*目標 WebSphere Commerce Server* 是指您要部署自訂程式碼的 WebSphere Commerce Server。在某些測試實務內容中，您可以部署和 VisualAge for Java 在同一部機器上執行的 WebSphere Commerce Server。在其它的情況下，目標 WebSphere Commerce Server 是在另一部機器，它可能在不同的平台上執行。

下列各節說明部署各種不同自訂程式碼類型的高階步驟。 請利用它們來瞭解部署程序中所牽涉的步驟，並參閱第 315 頁的附錄 B, 『部署明細』以取得逐步指示。

除了下列各節說明自訂程式碼的部署外，如果您在開發環境中建立新存取控制原則，則必須在目標 WebSphere Commerce Server 中建立相同的存取控制原則。有關此程序的範例說明，請參閱第 254 頁的『載入新資源的存取控制原則』。

EJB 部署程式碼相關資訊

您必須知道 VisualAge for Java V4.0 之 WebSphere Test Environment 元件所用的是 WebSphere Application Server V3.5.4。在此版 WebSphere Application Server 中，是在 Enterprise JavaBeans (EJB) V1.0 規格層次支援 Enterprise Bean。因此，如果要在 WebSphere Test Environment 中執行任何 Enterprise Bean，您用以產生您 Enterprise Bean 之部署程式碼的 VisualAge for Java 工具，需符合 EJB V1.0 規格。

相對地，WebSphere Commerce Business Edition 與 WebSphere Commerce Professional Edition 所用的是 WebSphere Application Server V4.0。WebSphere Application Server V4.0 可支援 EJB V1.1 規格。因此，執行於 WebSphere Test

Environment 中之 Enterprise Bean 的部署程式碼有別於執行於 WebSphere Application Server V4.0 中之 Enterprise Bean 的部署程式碼。

這對開發與部署的影響是：

- 如果要在 WebSphere Test Environment 中測試 Enterprise Bean，請使用 VisualAge for Java 的工具來產生符合 WebSphere Test Environment 中所用之 WebSphere Application Server V3.5.4 需求的部署程式碼。如果要產生此程式碼，請以滑鼠右鍵按一下 Enterprise Bean，並選取**產生部署程式碼**。請注意，這不會建立 JAR 檔。
- 如果要將 Enterprise Bean 部署到 WebSphere Application Server V4.0 中，您必須執行下列步驟：
  1. 使用 VisualAge for Java 中的工具，將 Enterprise Bean 匯出至本文件中所說的 *EJB 1.1 匯出 JAR 檔*。這個 JAR 檔會將程式碼包裝成 EJBDeploy 工具所用的格式。如果要建立此檔案，請以滑鼠右鍵按一下內含所要部署之 Enterprise Bean 的 EJB 群組，並選取**匯出 > EJB 1.1 JAR**。請注意，當您建立此 JAR 檔時，必須爲將部署程式碼的機器選取適當的資料庫類型。
  2. 將這個 EJB 1.1 匯出 JAR 檔傳送到執行 WebSphere Application Server V4.0 的目標伺服器上。
  3. 在目標伺服器上，執行 EJBDeploy 工具並將 EJB 1.1 匯出 JAR 檔當成輸入執行，以便爲符合 Enterprise JavaBeans V1.1 規格的部署程式碼建立一個新 JAR 檔。

部署 Enterprise Bean 還涉及其它步驟。詳細資訊請參閱第 173 頁的『新實體 Bean 的部署』與第 175 頁的『已修改之 WebSphere Commerce 公用實體 Bean 的部署』。有關使用 EJBDeploy 工具的詳細說明，請參閱第 326 頁的『產生部署程式碼』。

新指令與資料 Bean 的部署

在您建立新指令與資料 Bean 時，您應將之放在一個適當套件中（其名稱符合您應用程式的要求）。舉例來說，您可以建立一個新套件 `com.mycompany.mycommands`，以便放置您的新指令。此套件必須儲存在某個有別於 WebSphere Commerce 專案（IBM WCS Commerce Server 與 IBM WC Enterprise Bean）的專案中。舉例來說，您可以建立一個新專案 `My Project`。請謹慎做好程式碼的組織，以確定部署能順利完成。

在您將所有的新指令與資料 Bean 儲存在自己的專案的情況下，部署作業涉及如下的高階步驟：

1. 使用開發機器，爲專案建立一個 JAR 檔。請從 VisualAge for Java 的「專案」頁面中，使用工具將專案匯出到 JAR 檔中。請爲您的程式碼取一個適當的 JAR

檔名稱；例如 CustomCommands.jar。此外，您必須使用 VisualAge for Java 外的工具重新包裝 JAR 檔，以確定已包含部署到 WebSphere Application Server 時所需的所有命名資訊。相關資訊請參閱第 315 頁的『自訂指令和資料 Bean 的 JAR 檔案』。

2. 將 JAR 檔、JSP 範本和其它商店資產複製到目標 WebSphere Commerce Server 的適當目錄中。相關資訊，請參閱第 322 頁的『將資產儲存到目標 WebSphere Commerce Server』。
3. 將新指令登錄在目標 WebSphere Commerce Server 上的指令登錄中。相關資訊請參閱第 26 頁的『指令登錄組織架構』。
4. 使用 WebSphere Application Server 管理主控台來停止並重新啓動 WebSphere Commerce 企業應用程式。有關啓動與停止這個應用程式的說明，請參閱適當的 *WebSphere Commerce 安裝手冊*。

新實體 Bean 的部署

在您建立新實體 Bean 時，您必須將之建於有別於內含 WebSphere Commerce 實體 Bean 之 EJB 群組的 EJB 群組中。您也必須使用自己的專案，並確定此專案不含任何指令或資料 Bean 的程式碼。舉例來說，您可以建立一個新 EJB 群組 MyEntityBeans 與 MyEntityBeansProject 專案。假設專案中含有新實體 Bean 程式碼以外的程式碼，則部署可能不會成功。

一旦您已滿意實體 Bean 在 WebSphere Test Environment 中的運作方式，您必須加以部署。下列資訊提供部署步驟的概觀：

1. 使用開發機器，爲新的 EJB 群組建立一個新的 EJB 1.1 匯出 JAR 檔。從 VisualAge for Java 的 EJB 頁面中，以滑鼠右鍵按一下新的 EJB 群組，並選擇將程式碼匯出到 EJB 1.1 JAR 檔中。請爲您的程式碼取一個適當的檔案名稱；例如 MyEntityBeans_DT.jar。相關資訊，請參閱第 317 頁的『爲新實體 Bean 建立 JAR 檔』。
2. 爲新的 EJB 專案建立一個新的施行 JAR 檔。如果要建立此 JAR 檔，請選取「專案」頁面，然後以滑鼠右鍵按一下新的 EJB 專案，再選擇將程式碼匯出到 JAR 檔中。請爲您的程式碼取一個適當的名稱；例如 MyEntityBeansImpl.jar。此外，您必須使用 VisualAge for Java 以外的工具重新包裝施行 JAR 檔，以確定已包含部署到 WebSphere Application Server 時所需的所有命名資訊。相關資訊，請參閱第 317 頁的『爲新實體 Bean 建立 JAR 檔』。
3. 將 JAR 檔複製到目標 WebSphere Commerce Server 上的適當目錄中。相關資訊，請參閱第 322 頁的『將資產儲存到目標 WebSphere Commerce Server』。
4. 在目標 WebSphere Commerce Server 上，使用 WebSphere Application Server 所提供的 EJB 部署工具，爲新 Enterprise Bean 產生部署程式碼。此工具會以

您在步驟 1 中所建的 JAR 檔做爲輸入，並且會建立一個對應的 JAR 檔，而此 JAR 檔中含有您 EJB 群組中之所有 Enterprise Bean 的部署程式碼。相關資訊請參閱第 326 頁的『產生部署程式碼』。

5. 在目標 WebSphere Commerce Server 上，修改部署程式碼之 JAR 檔中所含之 Enterprise Bean 的交易隔離層次。請使用 WebSphere Commerce 所提供的 `modifyIsolationLevel` 指令行公用程式，將交易隔離層次設爲適合您資料庫類型的層次。相關資訊請參閱第 328 頁的『修改實體 Bean 的交易隔離層次』。
6. 在目標 WebSphere Commerce Server 上，載入任何您所建新資源的存取控制原則。請使用 WebSphere Commerce `acpload` 與 `acpnlsload` 指令來載入原則資訊。有關載入新資源之存取控制原則的範例說明，請參閱第 9 章, 『指導教學：建立新商業邏輯』中的第 254 頁的『載入新資源的存取控制原則』一節。
7. 在目標 WebSphere Commerce Server 上，從 WebSphere Application Server 匯出目前的 WebSphere Commerce 企業應用程式。在匯出完成後，會建立一個內含整個應用程式的 .ear 檔。如要匯出目前的應用程式，請開啓 WebSphere Application Server 管理主控台，並選取 WebSphere Commerce 企業應用程式，然後選取匯出選項。當完成時，將會建立一個 `WC_Enterprise_App_`*`instanceName`*`.ear` 檔。相關資訊請參閱第 330 頁的『匯出現行的 WebSphere Commerce 企業應用程式』。
8. 在目標 WebSphere Commerce Server 上，匯出現行 WebSphere Commerce 企業應用程式（執行於 WebSphere Application Server 中）中所含之 Enterprise Bean 的架構資訊。此資訊是以 WebSphere Application Server 所提供之 `XMLConfig` 指令行公用程式的 `-export` 選項匯出到 XML 檔中。在所產生的 XML 檔（在此稱爲 `OutputFile.xml` 檔）中，企業應用程式內所含的每一個 Enterprise Bean 各有一個架構資訊段落。您必須在這個檔案中，爲每一個您將部署的新 Enterprise Bean 各新增一個新段落。相關資訊請參閱第 331 頁的『匯出 Enterprise Bean 的架構資訊』。
9. 使用 WebSphere Application Server 所提供應用程式組譯工具，將新 Enterprise Bean 新增到您的企業應用程式中。透過這項工具，您可以開啓現行應用程式的 `WC_Enterprise_App_`*`instanceName`*`.ear`，並將任何新 Enterprise Bean 匯入至應用程式中。此外，您可設定新 Enterprise Bean 的類別路徑，以包含任何相依的 JAR 檔，並將施行 JAR 檔當成檔案新增到應用程式中。最後，您可使用此工具爲您 Enterprise Bean 中所含的方法架構 WebSphere Application Server 安全特性。在完成這些步驟後，請儲存應用程式，這會爲您的企業應用程式建立一個新 `.ear` 檔。相關資訊請參閱第 336 頁的『將新 Enterprise Bean 組譯到企業應用程式中』。
10. 將新企業應用程式匯入至 WebSphere Application Server 中。此步驟由下列三項子作業組成：

a. 使用 WebSphere Application Server 管理主控台來停止並移除原始的 WebSphere Commerce 企業應用程式。相關資訊請參閱第 345 頁的『停止並移除企業應用程式』。

b. 使用 `XMLConfig` 指令行公用程式的 `-import` 選項，將新企業應用程式匯入至 WebSphere Application Server 中。相關資訊請參閱第 346 頁的『匯入企業應用程式』。

c. 使用 WebSphere Application Server 管理主控台來重新整理檢視畫面，以顯示新企業應用程式，然後啓動新應用程式。相關資訊請參閱第 348 頁的『啓動企業應用程式』。

部署現有指令與資料 Bean 的延伸

用以延伸現有指令的方法端視所需的修改類型而定。有關延伸的方法請參閱第 129 頁的『自訂現有指令』。一般而言，修改現有邏輯牽涉新類別的建立，而此新類別是繼承需要自訂的類別。請視需要改寫超類別的方法，以取代或修改邏輯。

在您自訂資料 Bean 時，您也會建立一個由現有資料 Bean 延伸而來的新類別。請在新類別中進行必要的修改。

在您建立這些新類別時，請確定這些類別儲存在您自己的一個套件中，且該套件儲存在您自己的一個專案中。

由於延伸實際上是由子類別來處理，因此指令與資料 Bean 延伸的部署作業，和新指令與資料 Bean 的部署作業一樣。詳細資訊請參閱第 172 頁的『新指令與資料 Bean 的部署』。

已修改之 WebSphere Commerce 公用實體 Bean 的部署

在您修改 WebSphere Commerce 公用 Enterprise Bean 時，將會修改到 WebSphere Commerce 程式碼。因此，自訂實體 Bean 的部署技術將和用於新實體 Bean 的部署技術稍有不同。以下提供部署步驟的概觀：

1. 爲內含已修改之實體 Bean 的 WebSphere Commerce EJB 群組建立一個 EJB 1.1 匯出 JAR 檔。選取 VisualAge for Java 工作台中的 EJB 標籤，並選取適當的 EJB 群組，然後選擇要將該群組匯出至 EJB 1.1 匯出 JAR 檔。在您命名 JAR 檔時，您可使用 `Cust_`*`EJBGroupName`*`-ejb_DT.jar` 命名慣例；其中 *`EJBGroupName`* 爲已修改之 EJB 群組的名稱，_DT 字尾則是附加於 JAR 檔名尾端。舉例來說，在您爲 WCSUser EJB 群組命名 JAR 檔時，可命名爲 `Cust_WCSUser-ejb_DT.jar`。請注意，_DT 字尾純粹是提醒您稍後必須將這個

JAR 檔傳到 EJBDeploy 工具，以產生 EJB 群組中之 Bean 的部署程式碼。相關資訊，請參閱第 319 頁的『爲自訂 WebSphere Commerce 實體 Bean 建立 JAR 檔』。

2. 爲所有 WebSphere Commerce EJB 群組建立一個內含從屬站程式碼的新從屬站 JAR 檔。選取所有 WebSphere Commerce EJB 群組（名稱開頭全爲 WCS），並選擇將之匯出到從屬站 JAR 檔中。相關資訊，請參閱第 319 頁的『爲自訂 WebSphere Commerce 實體 Bean 建立 JAR 檔』。
3. 將 JAR 檔複製到目標 WebSphere Commerce Server 上的適當目錄中。相關資訊，請參閱第 322 頁的『將資產儲存到目標 WebSphere Commerce Server』。
4. 在目標 WebSphere Commerce Server 上，使用 WebSphere Application Server 所提供的 EJBDeploy 工具，爲您要部署之 EJB 群組中的 Enterprise Bean 產生部署程式碼。此工具會以您在步驟 1 中所建的 JAR 檔做爲輸入，並且會建立一個對應的 JAR 檔，而此 JAR 檔中含有您 EJB 群組中之所有 Enterprise Bean 的部署程式碼。相關資訊請參閱第 326 頁的『產生部署程式碼』。
5. 在目標 WebSphere Commerce Server 上，修改部署程式碼之 JAR 檔中所含之 Enterprise Bean 的交易隔離層次。請使用 WebSphere Commerce 所提供的 `modifyIsolationLevel` 指令行公用程式，將交易隔離層次設爲適合您資料庫類型的層次。相關資訊請參閱第 328 頁的『修改實體 Bean 的交易隔離層次』。
6. 在目標 WebSphere Commerce Server 上，從 WebSphere Application Server 匯出目前的 WebSphere Commerce 企業應用程式。在匯出完成後，會建立一個內含整個應用程式的 .ear 檔。如要匯出目前的應用程式，請開啓 WebSphere Application Server 管理主控台，並選取 WebSphere Commerce 企業應用程式，然後選取匯出選項。當完成時，將會建立一個 `WC_Enterprise_App_`*`instanceName`*`.ear` 檔。相關資訊請參閱第 330 頁的『匯出現行的 WebSphere Commerce 企業應用程式』。
7. 在目標 WebSphere Commerce Server 上，匯出現行 WebSphere Commerce 企業應用程式（執行於 WebSphere Application Server 中）中所含之 Enterprise Bean 的架構資訊。此資訊是以 WebSphere Application Server 所提供之 `XMLConfig` 指令行公用程式的 `-export` 選項匯出到 XML 檔中。在所產生的 XML 檔（在此稱爲 `OutputFile.xml` 檔）中，企業應用程式內所含的每一個 Enterprise Bean 各有一個架構資訊段落。相關資訊請參閱第 331 頁的『匯出 Enterprise Bean 的架構資訊』。在此檔案中，您必須將說明您已修改之 Enterprise Bean 的段落改爲指向您的新 `Cust_`*`EJBGroupName`*`-ejb.jar` 檔。
8. 在目標 WebSphere Commerce Server 上，使用應用程式組譯工具將修改過的 EJB 群組整合到企業應用程式中。使用此工具，開啓現行應用程式的 `.ear` 檔。一旦開啓後，請執行下列作業：

a. 記下您已修改之 EJB 群組原始版本的類別路徑資訊。舉例來說，如果您修改了 WCSUser EJB 群組中的 User Bean，請將 WCSUser 群組的類別路徑資訊複製到一個文字檔中。

b. 刪除已修改的 EJB 群組原始版本（例如，刪除 WCSUser 群組）。

c. 匯入已修改的新 EJB 群組版本。

d. 將原始的類別路徑資訊套用在您剛匯入的 EJB 群組上。

e. 爲已修改之 EJB 群組中之所有 Enterprise Bean 所含的方法，架構 WebSphere Application Server 安全特性。

f. 將應用程式儲存到新 `.ear` 檔中。

相關資訊請參閱第 341 頁的『將已修改的 Enterprise Bean 組譯到企業應用程式中』。

9. 將新企業應用程式匯入至 WebSphere Application Server 中。此步驟由下列三項子作業組成：

   a. 使用 WebSphere Application Server 管理主控台來停止並移除原始的 WebSphere Commerce 企業應用程式。相關資訊請參閱第 345 頁的『停止並移除企業應用程式』。

   b. 使用 `XMLConfig` 指令行公用程式的 `-import` 選項，將新企業應用程式匯入至 WebSphere Application Server 中。相關資訊請參閱第 346 頁的『匯入企業應用程式』。

   c. 使用 WebSphere Application Server 管理主控台來重新整理檢視畫面，以顯示新企業應用程式，然後啓動新應用程式。相關資訊請參閱第 348 頁的『啓動企業應用程式』。

部署新資料 Bean 以用於 Commerce Studio 中

如果您使用 Commerce Studio 來開發 JSP 範本，您必須將新資料 Bean 部署到 Commerce Studio 中。尤其是您必須爲資料 Bean 建立一個 JAR 檔。

請以滑鼠右鍵按一下資料 Bean 套件並選擇將之匯出到 JAR 檔中，以建立此 JAR 檔。在選項中請選擇包含下列：

- **類別**
- **資源**
- **Bean**

此外，選取**選取所參照的類型與資源**，以包含資料 Bean 所需的指令與資源。

一旦您匯出 JAR 檔後，您必須更新 Commerce Studio 的類別路徑，以包含此 JAR 檔。

在您需要修改現有的 WebSphere Commerce 資料 Bean 時，您必須建立一個由需要自訂之資料 Bean 延伸而來的新資料 Bean。所以事實上您是在建立一個新資料 Bean，並且是按本節所述般部署。

部署自訂的公用實體 Bean 以用於 Commerce Studio 中

如果您要修改任何公用的實體 Bean，並使用 Commerce Studio 來開發 JSP 範本，您必須針對所有公用實體 Bean 建立一個新從屬站 JAR 檔，並修改 Commerce Studio 的類別路徑，以便將新 JAR 檔名稱置於原始 JAR 檔名稱之前。當您在 Page Designer 工具中使用資料 Bean 時，Commerce Studio 會用到從屬站 JAR 檔。

如果要建立此 JAR 檔，請使用 VisualAge for Java 中的工具。請從 VisualAge for Java 的 EJB 頁面中，選出*所有的* WebSphere Commerce EJB 群組（而非只選出您所修改的），並選擇要將之匯出到從屬站 JAR 檔中。在匯出完成後，請確定您已將此新 JAR 檔指定於 Commerce Studio 類別路徑的開頭。

日誌檔

在產品安裝、程式碼開發以及程式碼部署的階段中，會產生日誌檔。本節列出在這些階段中您可查閱的部份日誌檔。

Commerce Studio 日誌檔

Commerce Studio 會在安裝程序期間建置日誌檔。如果要存取這些日誌，請開啓下列檔案：

```
C:\Winnt\WCStudioInstall.log
```

Commerce Studio 的 WebSphere Test Environment 架構日誌

如果您選擇要執行後端開發工作，則在安裝 WebSphere Commerce Studio 期間，將會針對 WebSphere Test Environment 執行一些架構步驟。舉例來說，如果您選擇要包含範例商店（執行於 VisualAge for Java 的 WebSphere Test Environment 元件中），則會建立和大量載入程序以及資料庫建立有關的日誌檔。如果要存取這些日誌，請導覽至下列目錄：

drive:\WebSphere\CommerceServerDev\instances*instance_name*\logs

在 WebSphere Test Environment 中執行 WebSphere Commerce 元件時產生的日誌檔

當您在 WebSphere Test Environment 中執行商店或測試個別元件時，可能會產生追蹤檔與訊息檔。這些檔案的預設目錄爲：

drive:\WebSphere\CommerceServerDev\instances*instance_name*\logs

執行 modifyIsolationLevel 指令時產生的日誌

當您部署 Enterprise Bean 時，您可使用 modifyIsolationLevel 指令來修改

JAR 檔中之每個 Enterprise Bean 的交易隔離層次。產生的日誌檔會儲存在您執行指令時所指定的日誌檔中。相關資訊請參閱第 328 頁的『修改實體 Bean 的交易隔離層次』。

VisualAge for Java 中的日誌記載

當您在 WebSphere Test Environment 中執行程式碼時，「主控台」視窗即如同一個作用中日誌般執行。此外，您可以修改 EJB 伺服器的追蹤層次，以增加「主控台」視窗中所能找到的資訊量。在 EJB 伺服器內容中，會將此資訊指定成追蹤層次。

測試付款方法

在預設的情況下，在 WebSphere Test Environment 中執行的「時尚館」範例商店使用的是測試付款方法。我們已包含此測試付款方法，因此，您可在 WebSphere Test Environment 中完成購物流程，而不需呼叫遠端 Payment Manager。使用此付款方法下單的訂單狀態爲 'M'（表示已起始付款，且正在等待處理）。

此測試付款方法只能讓您完成一項交易，而不能讓此付款方法所提交的訂單可供日後處理用。因此，測試付款方法應只在 WebSphere Test Environment 中使用。

付款相關指令的施行類別屬於商業原則指令類型。因此，選擇要使用哪種施行類別是由商業原則控制。在預設的情況下，在 WebSphere Test Environment 中執行的「時尚館」範例商店含有一個商業原則 TestPaymentMethod，其所用付款相關指令的施行如下：

- `com.ibm.commerce.payment.commands.DoPayment`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.CheckPaymentAccept`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoCancel`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoDeposit`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoRefund`**`Test`**`CmdImpl`

在此必須強調，這些指令僅供測試結帳程序用。上述一系列指令的提供是爲求周延，不過 DoDepositTestCmdImpl 僅爲指令片段，如果您呼叫將會擲出異常狀況。請勿試著使用任何訂單管理功能來處理這些訂單。如果您希望能測試這些其它功能，您可以將 WebSphere Test Environment 架構成使用遠端 Payment Manager。

要提醒您的是，當您將程式碼部署到目標 WebSphere Commerce Server 中時，請勿將測試付款方法的相關商業原則從您的開發環境複製到目標機器中。此外，您不應將測試付款方法的相關指令登錄在目標 WebSphere Commerce Server 上。

有關停用測試付款方法的進一步資訊，請參閱 *WebSphere Commerce Business Edition 安裝手冊*或 *WebSphere Commerce Professional Edition 安裝手冊*中的『架構 VisualAge for Java』一章。

使用遠端 Payment Manager

如果您的開發工作還包括下單後的訂單處理（例如，修改訂單管理程序中的某個步驟），您必須將執行於 WebSphere Test Environment 中的商店架構成使用遠端 Payment Manager。有關詳細的架構步驟，請參閱 *WebSphere Commerce Studio, Business Developer Edition 安裝手冊*。該文件中亦會教您如何將您以測試付款方法所下的訂單從資料庫中移除。

第 4 篇 指導教學

本節提供一些指導教學，協助您熟悉您電子商務應用程式的自訂。所提供的指導教學如下：

- 建立新商業邏輯
  本指導教學示範建立新商業邏輯中的開發程序。其中包含下列的子作業：
  - 準備範例專案
  - 撰寫新指令
  - 建立新資料 Bean 並從要求取得內容
  - 使用實體 Bean
  - 建立新 Enterprise Bean
- 更新現有的商業邏輯
  本指導教學說明更新現有 WebSphere Commerce 商業邏輯的開發程序。其中包含下列的子作業：
  - 延伸現有的 WebSphere Commerce 控制程式指令
  - 修改現有的 WebSphere Commerce 公用實體 Bean
  - 建立新作業指令，以延伸某個現有的 WebSphere Commerce 作業指令。

指導教學中含有一些必須輸入到 VisualAge for Java 中的程式碼區段。建議您從下列網站取得本文件的 PDF 版本：

Business

```
http://www.ibm.com/software/webservers/commerce/wc_be
   /lit-tech-general.html
```

Professional

```
http://www.ibm.com/software/webservers/commerce/wc_pe
   /lit-tech-general.html
```

您可以從「程式設計手冊」的 PDF 版本中剪貼程式碼片段。

第 9 章 指導教學：建立新商業邏輯

指導教學環境

本書中的指導教學會要求您必須已安裝 WebSphere Commerce Business Edition 5.4 版或 WebSphere Commerce Professional Edition 5.4 版。此外，當您安裝產品時，您必須選擇安裝下列選項：

- **使用 WebSphere Studio 開發商店前端資產**
- **使用 VisualAge for Java 開發商店後端邏輯**
- **建立資料庫**
- **包含範例商店**

請按 *WebSphere Commerce Studio, Business Developer Edition 安裝手冊*或 *WebSphere Commerce Studio Professional Developer Edition 安裝手冊*中的指示，以取得完整的安裝與架構資訊。

在您開始進行指導教學前，您必須能在 WebSphere Test Environment 中執行範例商店，並在商店中完成購物。

指導教學中的程式碼部署步驟

本書中所含的指導教學會有一些選用步驟，用以說明如何將自訂程式碼部署到目標 WebSphere Commerce Server 中。它會假設目標 WebSphere Commerce Server 是在有別於您開發環境之機器上的 Windows NT 或 Windows 2000 中執行。請注意，如果您以 WebSphere Commerce Studio Business Developer Edition 做爲開發環境，您應部署到 WebSphere Commerce Business Edition 中。如果您以 WebSphere Commerce Studio Professional Developer Edition 做爲開發環境，則應部署到 WebSphere Commerce Professional Edition 中。

此外，在目標 WebSphere Commerce Server 上您必須已公佈一家以「時尚館」範例商店爲基礎的商店。在該商店中，您必須能夠完成一項交易。您可以使用遠端或本端 Payment Manager 來進行您的付款處理程序。

如果您要部署到的目標 WebSphere Commerce Server，是位於和您開發環境相同的機器上或者是在不同的作業系統上執行時，請參閱第 169 頁的第 8 章, 『開發工具與部署』 與第 315 頁的附錄 B, 『部署明細』，取得適當的部署資訊。

準備範例專案

在本節中，您將匯入 WC_SAMPLE_54.dat 儲存庫；此儲存庫含有『建立新商業邏輯』指導教學的起點。

如果要準備環境，請執行下列步驟：

1. 請確定您所用的是 WebSphere Commerce 程式碼的 WC_54.dat 儲存庫。您可在 WebSphere Commerce Business Edition V5.4 Disk 2 CD 或 WebSphere Commerce Professional Edition V5.4 Disk 2 CD 中找到。
2. 執行下列步驟，將範例專案匯入到您的工作區中：
   a. 將下列 CD 插入到您開發機器的光碟機中：
      - Business WebSphere Commerce Business Edition, V5.4 Disk 2
      - Professional WebSphere Commerce Professional Edition, V5.4 Disk 2
   b. 開啓 VisualAge for Java。
   c. 從**檔案**功能表中，選取**匯入**。
      這會開啓「匯入引導精靈」。
   d. 選取匯入**儲存庫**，然後按**下一步**。
   e. 在「從另一儲存庫中匯入」視窗中，執行下列步驟：
      1) 選取**本端儲存庫**。
      2) 在**儲存庫名稱**欄位中，輸入
         CD_drive:\repository\samples\programguide\WC_SAMPLE_54.dat
         其中 *CD_drive* 爲光碟機。
      3) 選取**專案**並按一下**明細**。選取 **_WCSamples** 專案。選取 **WC Sample 5.4** 版，並按一下**確定**。
      4) 請確定已選取**將最近的專案修訂版新增至工作區**。
      5) 按一下**完成**，開始匯入。
         匯入專案可能需要幾分鐘。
3. 執行下列步驟，來確定工作區擁有者是設爲 WCS 程式開發人員。
   a. 從**工作區**功能表中，選取**變更工作區擁有者**。
   b. 選取 **WCS 開發人員**並按一下**確定**。
4. 將練習用的 JSP 範本複製到適當目錄中，以便在 WebSphere Test Environment 中使用。如果要複製這些檔案，請執行下列步驟：
   a. 在下列目錄中找出 Sample.jsp 與 Sample_All.jsp 檔：
      CD_drive:\repository\samples\programguide\
      其中 *CD_drive* 爲前往下列目錄的光碟機：

b. 將這兩個檔案複製到下列目錄中：

```
vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment
     \hosts\default_host\default_app\web
```

其中 *vaj_drive* 爲您要安裝 VisualAge for Java 的磁碟機。

5. 將存取控制原則檔案複製到適當目錄中。這些檔案用以載入您在指導教學期間所建之新資源的新存取控制原則。如果要複製這些檔案，請執行下列步驟：

a. 切換至下列目錄：

```
CD_drive:\repository\samples\programguide\
```

b. 在該目錄中找出下列檔案：

- SampleCmdACPolicy.xml
  此 XML 檔中含有您在建立新控制程式指令時所用的存取控制原則。
- SampleACPolicy.xml
  此 XML 檔中含有您在建立新 Enterprise Bean 時所用的存取控制原則。
- SampleACPolicy_*locale*.xml
  其中 *locale* 爲語言識別碼。此 XML 檔中含有存取控制原則的說明。

c. 將上述檔案複製到下列目錄中：

```
drive:\WebSphere\CommerceServerDev\xml\policies\xml
```

其中 *drive* 爲您安裝 WebSphere Commerce Studio 的磁碟機。

6. 執行下列動作測試您的環境，以確定您已可開始進行指導教學：

a. 依照第 313 頁中所描述來啓動永久名稱伺服器。

b. 依照第 314 頁中所描述來啓動 EJB 伺服器。

c. 依照第 314 頁中所描述來啓動 Servlet 引擎。

d. 開啓瀏覽器並輸入下列的 URL：

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
   storeId=store_ID&catalogId=catalog_Id&langId=-1
```

其中 *store_ID* 爲您範例商店的識別碼（10001 爲範例值），*catalog_Id* 爲您範例商店型錄的識別碼（10001 爲範例值）。
當顯示範例商店的首頁時，請選取一項產品並購買。您必須能夠到達購物流程中的「訂單確認」頁面。

如果要驗證您商店的 storeId 值，可參閱 STOREENT 表格。

e. 關閉所有瀏覽器，並停止 WebSphere Test Environment。

此時您已可開始進行指導教學。

撰寫控制程式指令

本節教您如何撰寫新控制程式指令。在您完成本練習後，您會有一個新的工作指令 *MyNewControllerCmd*。此指令供所有商店使用，且每一家商店所用的指令施行方式相同。

在您建立新控制程式指令時，會有下列三個基本步驟：

1. 在指令登錄架構中登錄指令。
2. 建立指令的介面。
3. 建立指令的施行類別。

有關指令的詳細資訊，請參閱第 20 頁的『指令設計型樣』。

開始進行此指導教學前

您應該已完成第 184 頁的『準備範例專案』中的步驟。

如果要撰寫新指令，請執行下列步驟：

1. 撰寫控制程式指令的第一個步驟是建立您指令的名稱。在本範例中，指令的名稱爲 *MyNewControllerCmd*。
2. 指令必須登錄在指令登錄組織架構中。新控制程式指令的登錄程序將涉及在 URLREG 表格中建立項目。

   DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以登錄指令：

   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**使用陳述式終止字元**勾選框，並確定所指定的字元是分號 (;)。
   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 your_database_name;
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
    DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
    'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
    'This is a new controller command for test/education purposes.',
    null)
```

   其中 *your_database_name* 爲資料庫名稱；請按一下「執行」圖示。此指令是供所有商家使用（亦即，將 STOREENT_ID 之值設爲 0）。

Oracle 如果您所用的是 Oracle 資料庫，請執行下列步驟以登錄指令：

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**主電腦字串**欄位中，輸入您的連接字串。

e. 在 SQL Plus 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
    DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
    'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
    'This is a new controller command for test/education purposes.',
    null);
```

並按 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。

註: 爲求簡明，在本練習中，新指令只有一個施行類別，因此所有的商店皆採用同一種施行類別。此施行類別直接指定於介面的程式碼中。因此，您不需將介面與施行類別間的映射登錄於 CMDREG 表格中。若在指導教學環境外，您應將控制程式指令登錄於 CMDREG 表格以及 URLREG 表格中。

3. 控制程式指令必須傳回檢視畫面。您將建立的控制程式指令會傳回 SampleViewTask 檢視畫面。您必須將 SampleViewTask 檢視畫面登錄在 VIEWREG 表格中。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟以登錄檢視畫面：

a. 在 Script 視窗中輸入下列資訊，以便在 VIEWREG 表格中建立項目（請注意，您可能需先將上述的 SQL 陳述式從視窗中清除）：

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('SampleViewTask',-1, 0,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=Sample.jsp','This is a sample view for the
    Bonus Point exercise', 0, null)
```

然後按一下「執行」圖示。

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：

a. 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 ('SampleViewTask',-1, 0,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=Sample.jsp','This is a sample view for the
   Bonus Point exercise', 0, null);
```

並按 Enter 鍵，以執行 SQL 陳述式。

b. 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。

4. 在「VisualAge for Java 工作台」視窗中，展開 **_WCSamples** 專案。
5. 展開 **com.ibm.commerce.sample.commands** 套件，以滑鼠右鍵按一下 **MyNewControllerCmd** 介面，然後選取**新增 > 欄位**。
這時會開啓「建立欄位」引導精靈。
6. 建立一個欄位來指定介面的預設施行類別，方式如下：
   a. 在**欄位名稱**欄位中，輸入 defaultCommandClassName。
   b. 從**欄位類型** 下拉清單中，選取**字串**。
   c. 在「初始值」欄位中，輸入 "com.ibm.commerce.sample.commands.MyNewControllerCmdImpl"。（請務必包括雙引號。）
   d. 按一下**完成**。
   會產生新欄位的程式碼。

當您進行此指導教學時，只要您改寫存在於超類別中的欄位或方法，即可能出現一則警告，指出新欄位或方法將隱藏繼承的欄位或方法。在此情況下，請按一下**是**繼續進行。

7. 展開 **MyNewControllerCmdImpl** 類別，並選取其 **performExecute** 方法，以檢視其原始碼。
8. 在 performExecute 方法的原始碼中，取消區段 1 與區段 5 的註解。區段 1 是將下列程式碼引入方法中：

```
// 為輸出建立新 TypedProperties。
TypedProperty rspProp = new TypedProperty();
```

區段 5 是將下列程式碼引入方法中：

```
// 查看控制程式指令如何呼叫 JSP

rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
setResponseProperties(rspProp);
```

儲存您的工作 (**Ctrl + S**)。前面描述的程式碼片段是設定控制程式指令所要傳回的檢視畫面名稱。

9. 必須爲這個新控制程式指令指定指令層次的存取控制。在本例中，指令層次的存取控制原則將指出容許所有使用者執行指令。此原則指定於 SampleCmdACPolicy.xml 檔中。如果要載入新存取控制原則，請執行下列步驟：
   a. 在指令提示下，切換至下列目錄：
      drive:\WebSphere\CommerceServerDev\bin
   b. 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* 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

      舉例來說，您可發出下列指令：

      ```
      acpload VAJ_Demo user password SampleCmdACPolicy.xml
      ```

10. 執行下列步驟，將新 _WCSamples 專案新增到 Servlet 引擎的類別路徑中：
    a. 從**工作區**功能表中，選取**工具 > WebSphere Test Environment**。這時會開啓 WebSphere Test Environment 控制中心。
    b. 按一下 **Servlet 引擎**與**編輯類別路徑**。在「Servlet 引擎類別路徑」視窗中，按一下**全選**，然後按一下**確定**。
11. 執行下列步驟，測試您的新指令：
    a. 依照第 313 頁中所描述來啓動永久名稱伺服器。
    b. 依照第 314 頁中所描述來啓動 EJB 伺服器。
    c. 依照第 314 頁中所描述來啓動 Servlet 引擎。
    d. 開啓瀏覽器並輸入下列的 URL：

       ```
       http://localhost:8080/webapp/wcs/stores/servlet/
          MyNewControllerCmd
       ```

       在數分鐘過後，瀏覽器會顯示一頁，其中顯示如下的「範例 JSP」：

圖 31.

12. 爲現行狀態下的程式碼建立一個版本。這可讓您隨時可將程式碼還原回此一狀態。如果要建立版本，請執行下列步驟：
    a. 選取 **com.ibm.commerce.sample.commands** 與 **com.ibm.commerce.sample.databeans** 套件（在高亮度顯示情況下按住 `Ctrl` 鍵，並選取多個套件），以滑鼠右鍵按一下**管理 > 建立開放修訂版**。
    b. 以滑鼠右鍵按一下 **_WCSamples** 專案，並選取**管理 > 建立開放版本**。
    c. 再以滑鼠右鍵按一下 **_WCSamples** 專案，並選取**管理 > 版本**。會開啓「建立所選項目的版本」視窗。
    d. 選取**一個名稱**圓鈕，並輸入 `mySample 1.2 Completed`，然後按一下**確定**。

到目前爲止，您已新增新指令並在整合的測試環境中簡潔地測試過。

修改 MyNewControllerCmd

在上一節中您已建立 MyNewControllerCmd。在本練習中，您將更仔細地檢查新指令的內容，以便清楚瞭解如何撰寫自己的自訂控制程式指令。

一開始先檢查您所建立之程式碼的結構。 MyNewControllerCmd 介面爲 ControllerCommand 介面的延伸。它亦將該施行類別定義爲當成預設使用。當指令未登錄在 CMDREG 表格中，或者未在該表格中指定施行類別時，即會採用此類別。

您亦建立了 MyNewControllerCmdImpl 類別。該施行類別爲最終含有您想施行之商業邏輯（或者會呼叫作業指令以執行個別的商業作業）的類別。您的施行類別含有下列方法：

MyNewControllerCmdImpl 中的方法	說明
MyNewControllerCmdImpl()	建構子方法。
validateParameters()	作爲伺服器端驗證指令的輸入參數用。
isGeneric()	決定一般使用者可否呼叫該指令。
isRetriable()	決定在資料庫回復後是否要重試指令。
performExecute()	含有您指令的商業邏輯。

下列各節將進一步詳述如何更新您的新控制程式指令。

將變數傳遞給 JSP 範本

在本節中，您將修改 MyNewControllerCmd，以傳遞變數給 JSP 範本。 爲了顯示變數，您必須使用新資料 Bean，其名稱爲 DataBeanSampleBean。爲了讓變數顯示在 JSP 範本中，請執行下列步驟：

1. 在「VisualAge for Java 工作台」視窗中，展開 **_WCSamples** 專案。
2. 展開 **com.ibm.commerce.sample.commands** 套件，然後展開 **MyNewControllerCmdImpl** 類別，並選取其 **performExecute** 方法。
3. 在 performExecute 方法的原始碼中，取消區段 2 的註解。這會將下列程式碼引入方法中：

   ```
   // 查看控制程式指令如何將變數傳遞給 JSP

   // 將控制程式指令的其它參數新增到
   // rspProp 中以便回應
   //
   rspProp.put("ControllerParm1", "Hello world");
   rspProp.put("ControllerParm2", "Have a nice day!");
   ```

   上述的程式碼片段會建立兩個新參數，並且會被放到將傳遞給 JSP 範本的內容中。 儲存您的工作 (**Ctrl + S**)。
4. DataBeanSampleBean 資料 Bean 是供 JSP 範本用來顯示變數。雖然已先爲您建立此 Bean，但需依照下列的方法修改原始碼：
   a. 展開 **com.ibm.commerce.sample.databeans** 套件。
   b. 展開 **DataBeanSampleBean** 類別，並選取 **setRequestProperties** 方法，以檢視其原始碼。
   c. 在 setRequestProperties 方法的原始碼中，取消區段 1 的註解。這會將下列程式碼引入方法中：

```
// 將輸入 TypedProperties 複製到本端環境

requestProperties = aParam;
```

儲存您的工作。上述的程式碼片段會在本端環境下複製 aParam（定義於超類別中）中的值。 此是用來取得要求物件中的內容。

5. 執行下列步驟，以更新 Sample.jsp 檔，以便使用新資料 Bean 並顯示變數：
   a. 使用文字編輯器開啓 Sample.jsp 與 Sample_All.jsp 檔。這些檔案位於下列目錄中：
      vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host \default_app\web
   b. 從 Sample_All.jsp，將程式碼中的 <!-- SECTION 1 --> 到 <!-- END OF SECTION 1 --> 標記間的區段 1 複製到 Sample.jsp 中。這會將下列程式碼引入 JSP 範本中：

```
<!-- SECTION 1 -->
<%
DataBeanSampleBean testBean = new DataBeanSampleBean ();
com.ibm.commerce.beans.DataBeanManager.activate (testBean, request);
%>
<!-- END OF SECTION 1 -->
```

      此程式碼區段將會設定資料 Bean 的案例。

   c. 從 Sample_All.jsp，將程式碼中之 <!-- SECTION 2 --> 到 <!-- END OF SECTION 2 --> 標記間的區段 2 複製到 Sample.jsp 中。這會將下列程式碼引入 JSP 範本中：

```
<!-- SECTION 2 -->
<%
TypedProperty prop = testBean.getRequestProperties();
out.print("<B>List of name value pairs in TypedProperties object</B><P>");
// 將要求內容轉換成查詢字串
for (Enumeration pns = prop.keys(); pns.hasMoreElements();) {
     String paramName = (String) pns.nextElement();
     // 請勿將 URL 參數新增到查詢字串中
     Object val = prop.get(paramName,null);
     if (val != null) {
          if (val.getClass().isArray()) {
          // 讓陣列平順
         String[] oarray = (String[]) val;
          int len = java.lang.reflect.Array.getLength(val);
          for (int i = 0; i < len; i++) {
          out.print(paramName + "[" + i + "]=" + oarray[i] + "<br>");
          }
     } else {
     // 假設其為字串
     out.print(paramName + "=" + val.toString() + "<br>");
     }
   }
```

```
}
%>
<P>
<!-- END OF SECTION 2 -->
```

儲存此檔案。

此程式碼區段是採用 testBean 資料 Bean 物件的 getRequestProperties 方法。它會循環過所有內容，並將之顯示在瀏覽器中。

6. 執行下列步驟，測試您所作的修改，以驗證變數會顯示在 JSP 範本中：
   a. 確定 WebSphere Test Environment 正在執行。
   b. 在瀏覽器中輸入下列 URL：

      ```
      http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
      ```

      這會顯示範例 JSP 檔，其中顯示控制程式指令的內容。其輸出類似如下：

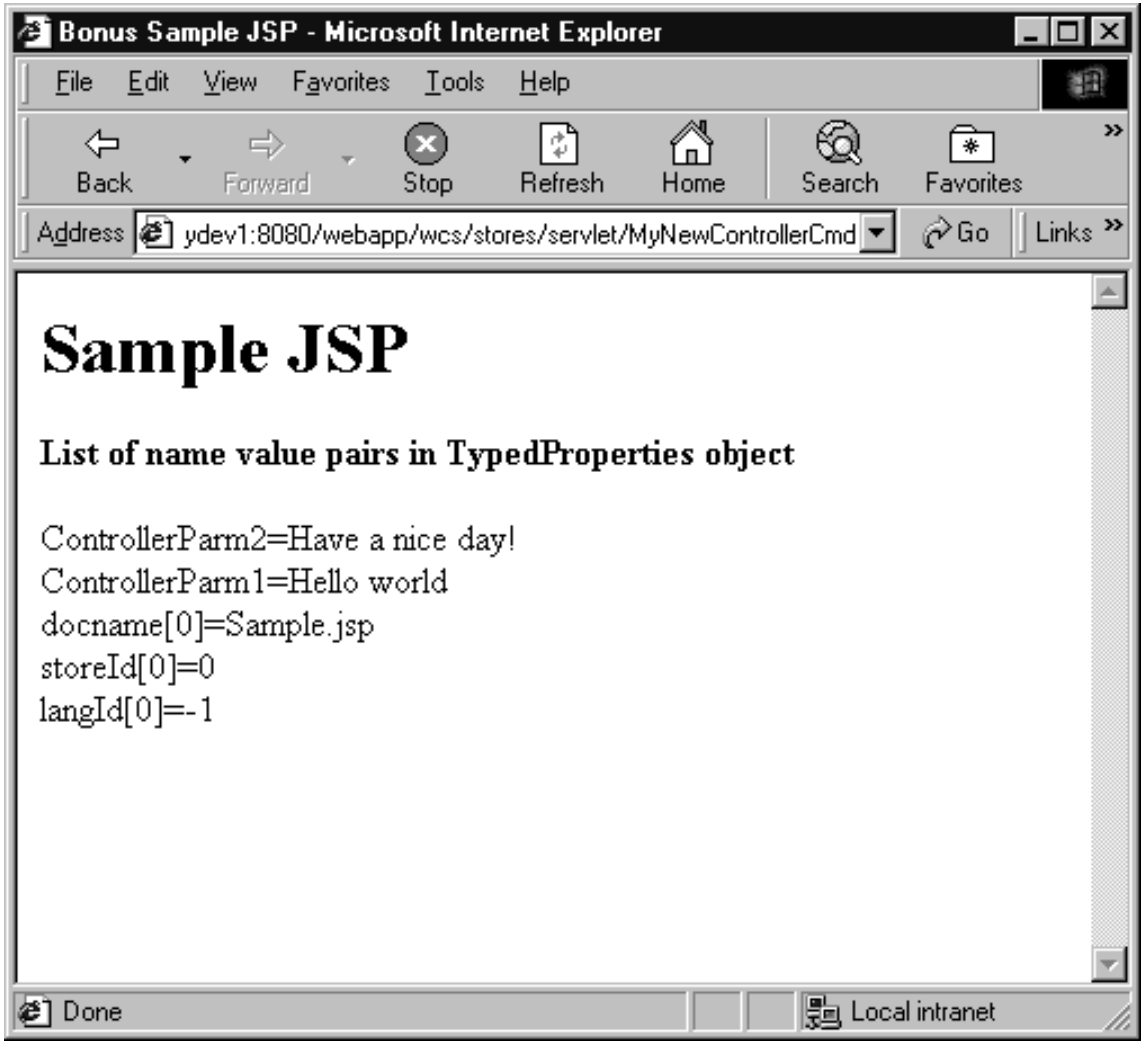


圖 32.

7. 爲現行狀態下的程式碼建立一個版本。爲 `mySample 1.3 Completed` 版本命名。有關建立程式碼之版本的詳述，請參閱步驟（第 190 頁的12）。

修改 validateParameters 方法

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 ECException {
     }
```

在本節中，您將在指令中新增自己的自訂參數檢查，然後將參數傳給 JSP 範本。

在修改 validateParameters 方法時，您可新增欄位到參數所用的類別中。

建立新欄位: 將新的欄位加入到現有的介面或類別時，您可以使用 VisualAge for Java 中的「建立欄位」引導精靈。本節說明建立新欄位的同屬步驟。當您必須新增欄位到介面和類別時，請使用指導教學中下列各節的資訊。

如果要建立新欄位，請執行下列步驟：

1. 以滑鼠右鍵按一下您要新增欄位的介面或類別，然後選取**新增 > 欄位**。這時會開啓「建立欄位」引導精靈。
2. 在**欄位名稱**欄位中，輸入新欄位的名稱。
3. 如果指定欄位類型，請執行下列其中一個步驟：
   - 從**欄位類型**下拉清單中，選取欄位類型。
   - 如果必要的欄位類型不在清單中，請按一下**瀏覽**。在**型樣**欄位中，輸入欄位類型的名稱（或部分名稱），然後按一下**確定**。

   註: 在指導教學中，只要指定 `String` 欄位類型，則其是出自 `java.lang` 套件。
4. 在**初始值**欄位中，輸入欄位的初始值。對於屬於字串的初始值，請務必加上雙引號（" "）。
5. 必要時，在**存取修改元**值中選取適當的圓鈕（public、protected、none 或 private）。
6. 如果想要替欄位產生 getter 和 setter 方法，請選取**以 getter 和 setter 方法存取**勾選框。如果選取這個勾選框，就必須同時指定 getter 和 setter 方法的內容，方式如下：
   - 若是 getter 方法，請選取其中一個 public、protected、private 或 none 圓鈕。
   - 若是 setter 方法，請選取其中一個 public、protected、private 或 none 圓鈕。
7. 按下**完成**。

如果要修改 validateParameters 方法，請執行下列步驟：

1. 您必須在 MyNewControllerCommandImpl 類別中建立兩個新欄位。第一個欄位是用於輸入字串，而第二個是用於輸入整數。如果要建立這些欄位，請執行下列步驟：
   a. 展開 **com.ibm.commerce.sample.commands** 套件。
   b. 選取 **MyNewControllerCmdImpl** 類別。
   c. 使用以下的值，新增一個欄位到類別中。 有關如何建立新欄位的詳細步驟，請參閱『建立新欄位』。

屬性名稱	值
欄位名稱	inputString
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

d. 使用以下的值，爲輸入整數建立另一個欄位：

屬性名稱	值
欄位名稱	inputInteger
欄位類型	Integer **註:** 按一下**瀏覽**，然後輸入 Integer。不要選擇 int。
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

2. 選取 MyNewControllerCmdImpl 類別中的 **performExecute** 方法。
3. 在 performExecute 方法的原始碼中，取消區段 3 的註解，以便將下列程式碼引入方法中：

```
// 查看控制程式指令如何將輸入變數傳遞給 JSP
     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());
```

此程式碼是將控制程式指令中的變數傳遞給資料 Bean。儲存您的工作。

4. 選取 MyNewControllerCmdImpl 類別中的 **validateParameters** 方法。
5. 取消 validateParameters 原始碼中之區段 1 的註解，以便將下列程式碼引入方法中：

```
// 取消檢查參數的註解

TypedProperty prop = getRequestProperties();
```

```
// 擷取必要參數
//
try {
          setInputString(prop.getString("input1"));
} catch (ParameterNotFoundException e) {
          throw new ECApplicationException(
               ECMessage._ERR_CMD_MISSING_PARAM,
               "MyControllerCmdImpl","validateParameters",
               ECMessageHelper.generateMsgParms(e.getParamName()));
}

// 擷取選用整數
// 若無輸入值，請將 input2 設為 0
//
setInputInteger(prop.getInteger("input2", 0));
```

儲存您的作業。

上述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在此，若不是，則會擲出異常狀況。由於第二個參數爲選用的，若該參數遺漏或類型不正確，此程式碼會將該參數的值設爲 0。

6. 執行下列步驟來測試指令：
   a. 確定 Persistent 名稱伺服器、EJB 伺服器和 Servlet 引擎都已經在執行中。
   b. 在您的瀏覽器中，輸入下列 URL：
      - 情況 1：遺漏參數；請輸入：

        ```
        http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
        ```

        沒有參數可傳遞給指令，因此會顯示一個一般應用程式錯誤，指出遺漏該參數。結果會顯示在下列的螢幕畫面中：

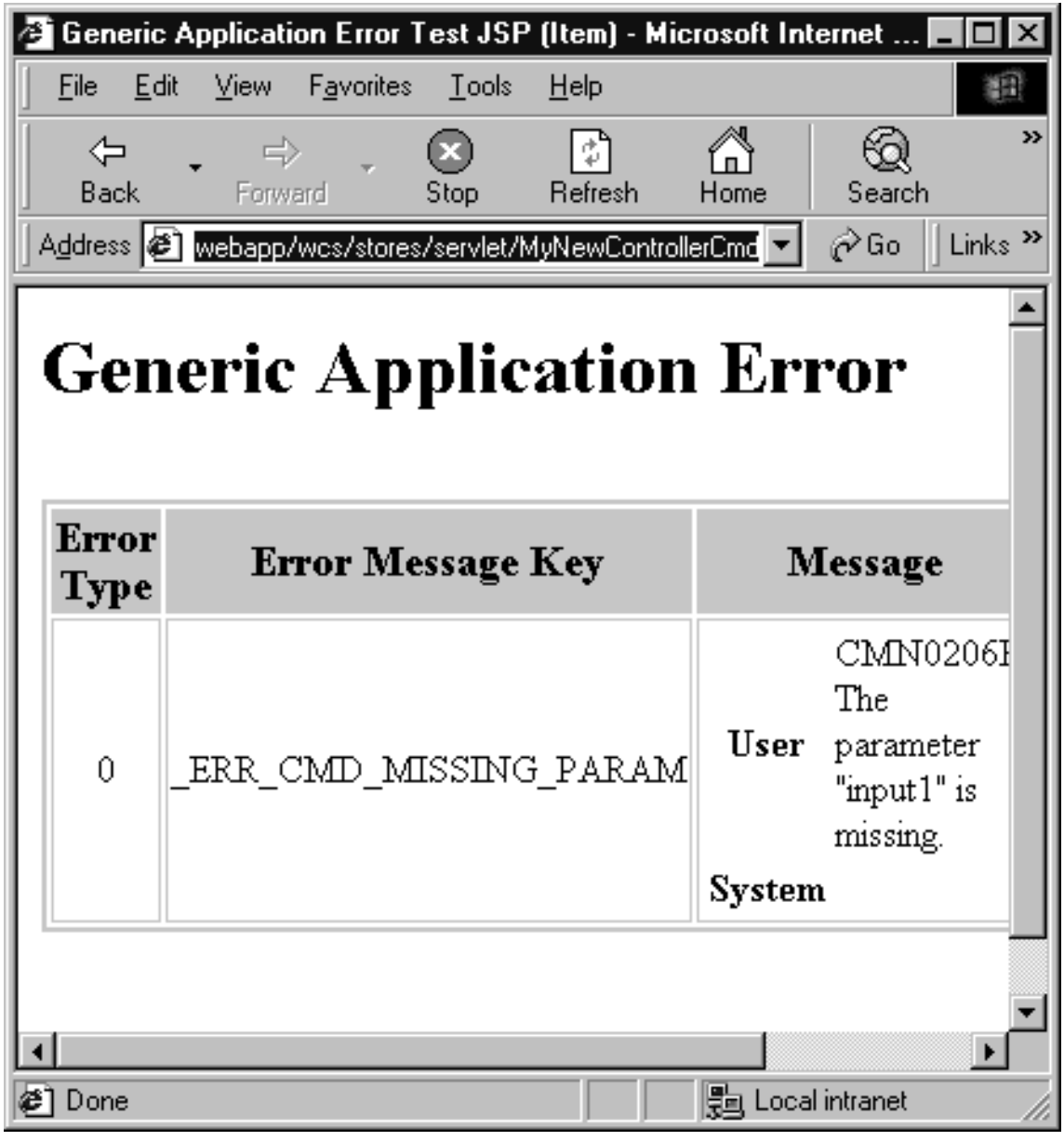


圖 33.

註: 如果出現「找不到頁面」錯誤，表示您的 Servlet 引擎可能已停止。請檢查 WebSphere Test Environment 控制中心取得詳細資訊。如果所出現的是「範例 JSP」頁面而非「一般應用程式錯誤」頁面，您可能需停止再重新啓動 Servlet 引擎，或者在瀏覽器中重新載入頁面。

- 情況 2：第一個參數有效，第二個參數遺漏；
  請輸入：

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc
  ```

  儘管省略了第二個參數，此指令的結果是顯示「範例 JSP」頁面。下列螢幕是顯示此結果。而螢幕後面會解釋未傳回錯誤之因。

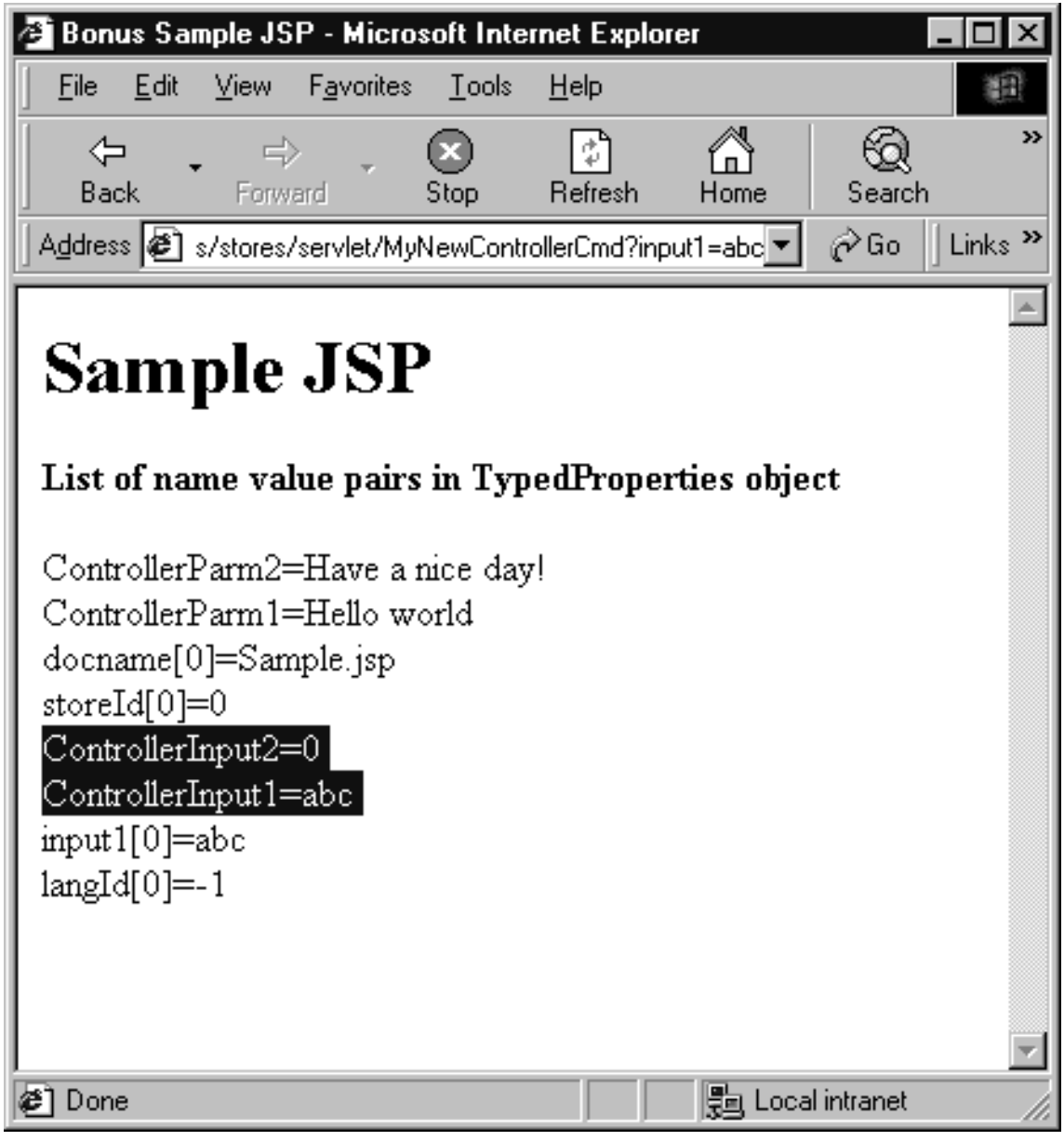


圖 34.

由於使用了 `getInteger` 方法，因而未傳回錯誤。換句話說，程式碼中的 `setInputInteger(prop.getInteger("input2", 0))` 行將 input2 的預設值設爲 0。當參數遺漏或類型不正確時，即會採用此預設值。爲了強迫對此參數進行類型檢查，請將程式碼改爲

setInputInteger(prop.getInteger("input2"))，並重新輸入 URL （請記得重新整理您的瀏覽器）。應會出現「一般應用程式錯誤」頁面。

註: 如果您測試您在程式碼中所做的 setInputInteger(prop.getInteger("input2")) 修改，在您繼續進行下個測試情況前，請將之切換回 setInputInteger(prop.getInteger("input2", 0))。

- 情況 3：第一個參數有效，第二個參數無效；
  請輸入：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=abc
```

此指令的結果是顯示「範例 JSP」頁面，且第二個參數的值（整數）會設爲預設值 0。如同上述測試情況中所述，這是使用 getInteger 方法的結果。結果會顯示在下列的螢幕畫面中：

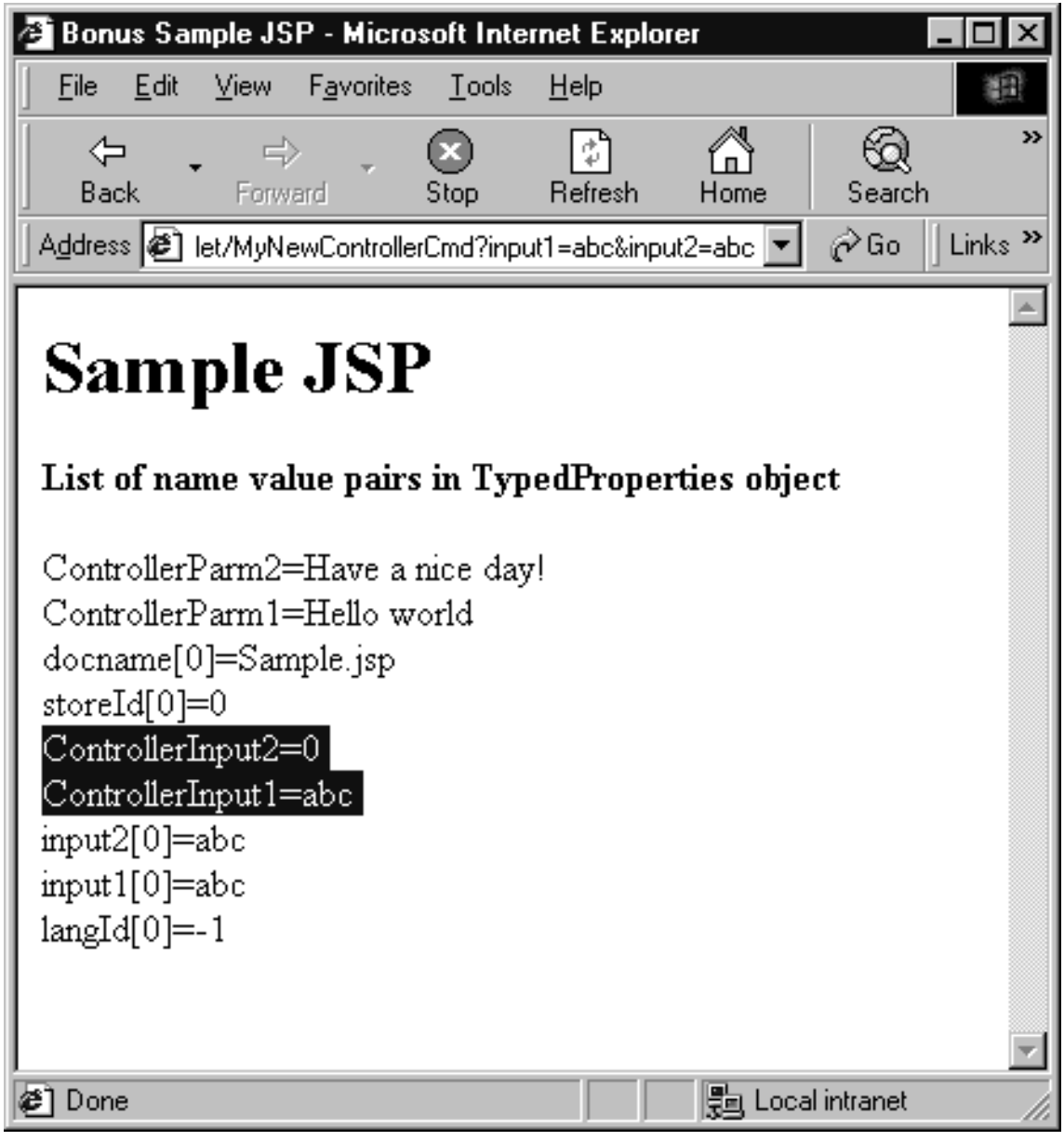


圖 35.

- 情況 4：兩個參數皆有效；請輸入：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

此指令的結果是顯示「範例 JSP」頁面，並同時顯示所輸入的輸入值。結果會顯示在下列的螢幕畫面中：

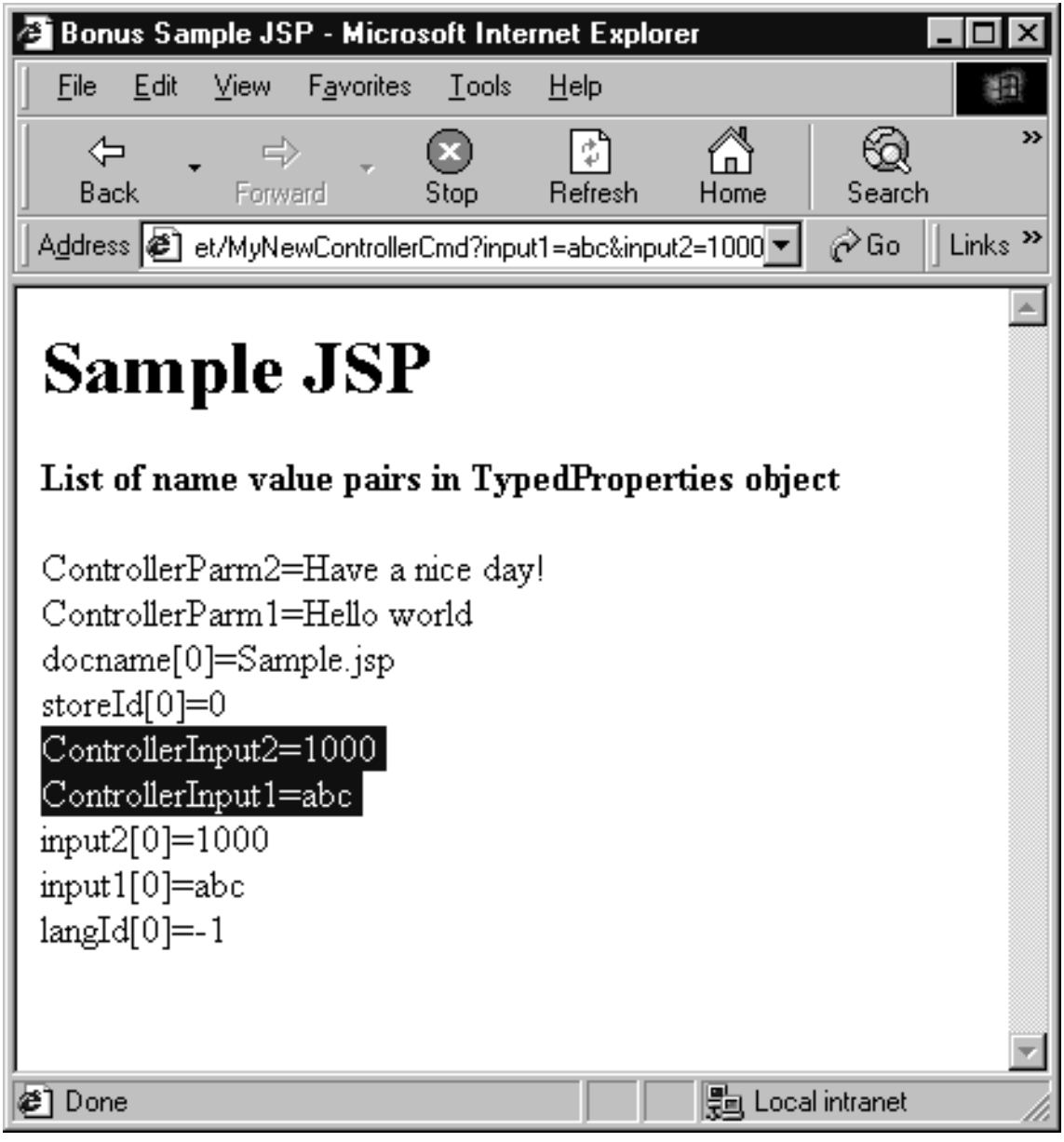


圖 36.

7. 爲現行狀態下的程式碼建立一個版本。爲 mySample 1.4 Completed 版本命名。有關如何建立程式碼的版本詳述，請參閱步驟（第 190 頁的12）。

建立作業指令

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。 OrderProcessCmd 控制程式指令所呼叫的這些作業指令之一是 CalculateOrderTaxTotalCmd。

MyNewControllerCmdImpl 目前尚不會呼叫任何作業指令。本項練習分爲兩個階段。在第一個階段中，您將建立新作業指令。在第二個階段中，您將修改您控制程式指令的 performExecute 方法，以呼叫作業指令。

下列的程式碼片段顯示已移除所有註解之 MyNewControllerCmdImpl 類別中的目前 performExecute 方法：

```
public void performExecute() throws ECException {

     super.performExecute();

  TypedProperty rspProp = new TypedProperty();
     rspProp.put("ControllerParm1", "Hello world");
     rspProp.put("ControllerParm2", "Have a nice day!");

     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());

     rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
  setResponseProperties(rspProp);
}
```

撰寫作業指令程式碼: 本節教您如何撰寫新作業指令。建立完整的新作業指令將牽涉到介面與施行類別的建立。在您建立作業指令時，介面應爲 com.ibm.commerce.commands.TaskCommand 的延伸。而施行類別應爲 com.ibm.commerce.command.TaskCommandImpl 的延伸。

在您完成本練習後，您會有一個新的工作指令 *MyNewTaskCmd*。 此指令供所有商店使用，且每一家商店所用的指令施行方式相同。

在指導教學的這個部分中，您會學習將欄位和方法新增至新作業指令的介面中。您之前已經爲 MyNewControllerCmdImpl 類別建立了新的欄位。請試著自行爲這個介面建立欄位，但如果您需要更多詳細資訊，請參閱第 195 頁的『建立新欄位』。

建立方法: 本節說明新增方法到現有類別與介面的同屬步驟。請閱讀此處的指示，然後當指導教學需要您參閱新方法時，再回過頭來參閱它們。

如果要建立新方方法，請執行下列步驟：

1. 以滑鼠右鍵按一下您要新增方法的介面或類別，然後選取**新增 > 方法**。
   這時會開啓「建立方法」引導精靈。
2. 確定**建立新方法**已經選取，然後按**下一步**。
3. 在**方法名稱**欄位中，輸入新方法的名稱。
4. 執行以下其中一個步驟來指定方法的傳回類型：
   - 從**傳回類型**下拉清單中，選取適當的傳回類型。例如，選取 `String`。
   - 如果傳回類型不在清單中，請按一下**瀏覽**。然後在**型樣**欄位中，輸入傳回類型，然後按一下**確定**。

   註: 在指導教學中，只要將傳回類型指定爲 `String`，則其是出自 `java.lang` 套件。
5. 如果方法接受參數，請按一下**新增**。在「參數」視窗中，指定參數名稱以及其它必要的資訊，然後按一下**新增**。在新增所有的參數後，按一下**關閉**。
6. 按**下一步**。
   這時會開啓「屬性」視窗。
7. 如果方法擲回異常狀況，請按一下「屬性」視窗中的**新增**。在**型樣**欄位中，輸入異常狀況的名稱，然後按一下**新增**。 在加入所有的異常狀況之後，按一下**關閉**。
8. 按一下**完成**。
   這時會產生方法的程式碼。

如果要建立 MyNewTaskCmd 指令，請執行下列步驟：

1. 展開 **com.ibm.commerce.sample.commands** 套件。
2. 以滑鼠右鍵按一下
   MyNewTaskCmd 介面，然後選取**新增 > 欄位**。
3. 使用「建立欄位」引導精靈，建立一個用以指定介面所用之預設施行類別的欄位。請使用下表中的值。如果您需要建立欄位的進一步明細，請參閱第 195 頁的『建立新欄位』。

屬性名稱	值
欄位名稱	`defaultCommandClassName`
欄位類型	`String`
初始值	`"com.ibm.commerce.sample.commands.MyNewTaskCmdImpl"`

產生欄位的程式碼時，會以下列方式出現：

```
java.lang.String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";
```

由於整個網站採用同一施行類別，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施行方式。 如果您有指令具有多種施行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，以建立介面與施行類別間的映射。

4. 執行下列步驟來將新的方法加入 MyNewTaskCmd 介面：
   a. 以滑鼠右鍵按一下 **MyNewTaskCmd** 介面並選取**新增 > 方法**。使用「建立方法」引導精靈，使用您在下列步驟中指定的值來建立新方法。如果需要建立方法的詳細資訊，請參閱第 205 頁的『建立方法』。
   b. 使用下列的值，建立一個新方法來擷取客戶的紅利積點結餘：

屬性名稱	值
方法名稱	getOldBonusPoint
傳回類型	String
參數	無
異常狀況	無

註: 可能會出現錯誤，指出在 MyNewTaskCmdImpl 施行類別中並未施行剛建立之繼承的抽象方法。這會在後續的步驟中更正。

   c. 建立一個新方法來擷取從作業指令所輸出的使用者 ID。建立這個方法時，請使用以下的值：

屬性名稱	值
方法名稱	getTask_output_userId
傳回類型	String
參數	無
異常狀況	無

   d. 建立一個新方法來擷取作業指令的輸出值。建立這個方法時，請使用以下的值：

屬性名稱	值
方法名稱	getTask_output1
傳回類型	String

屬性名稱	值
參數	無
異常狀況	無

e. 建立一個新方法來設定作業指令的第一個輸入值。建立這個方法時，請使用以下的值：

屬性名稱	值
方法名稱	setTask_input1
傳回類型	void
參數名稱	newTask_input1
參照類型	String
異常狀況	無

f. 建立一個新方法來設定作業指令的第二個輸入值。建立這個方法時，請使用以下的值：

屬性名稱	值
方法名稱	setTask_input2
傳回類型	void
參數名稱	newTask_input2
原始類型	int
異常狀況	無

5. 執行下列步驟，將新的欄位加入 MyNewTaskCmdImpl 類別：
   a. 以滑鼠右鍵按一下 **MyNewTaskCmdImpl** 類別並選取**新增 > 欄位**。使用「建立欄位」引導精靈，使用您在下列步驟中指定的值來建立新欄位。如果需要有關建立新欄位的額外資訊，請參閱第 195 頁的『建立新欄位』。
   b. 使用下列的值，在施行類別中建立一個新的欄位：

屬性名稱	值
欄位名稱	task_input1
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選

屬性名稱	值
Getter	public
Setter	public

c. 使用下列的值，在施行類別中建立一個新的欄位：

屬性名稱	值
欄位名稱	task_input2
欄位類型	int
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

d. 使用下列的值，在施行類別中建立一個新的欄位：

屬性名稱	值
欄位名稱	task_output_userId
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

e. 使用下列的值，在施行類別中建立一個新的欄位：

屬性名稱	值
欄位名稱	oldBonusPoint
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選

屬性名稱	值
Getter	public
Setter	public

f. 使用下列的值，在施行類別中建立一個新的欄位：

屬性名稱	值
欄位名稱	task_output1
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

6. 選取 **MyNewTaskCmdImpl** 類別中的 **performExecute** 方法，以檢視其原始碼。
7. 在原始碼中，取消區段 1 的註解，以便將下列程式碼引入方法中：

```
// 修改 task_input1 並在 NVP 清單中查看之

    setTask_output1( "Hello ! " + getTask_input1() );
```

儲存您的工作。
上述的程式碼區段是讓新屬性成爲指令的輸出以提供使用。

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此方式修改控制程式指令。

1. 在「工作台」中，選取 **MyNewControllerCmdImpl** 類別中的 **performExecute** 方法。
2. 在「原始碼」窗格中取消區段 4 的註解，以呼叫作業指令。這會將下列程式碼引入方法中：

```
// 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());
       // 為作業指令設定輸入參數
       cmd.setTask_input1(getInputString());
```

```
            cmd.setTask_input2(getInputInteger().intValue());
            // 對所有指令而言此為必要的
            cmd.setCommandContext(getCommandContext());
            // 呼叫指令的 performExecute 方法
            cmd.execute();
            // 擷取作業指令產生的輸出參數

            rspProp.put("task_output1", cmd.getTask_output1());
            if (cmd.getTask_output_userId() != null) {
                 rspProp.put("task_output_userId",
                 cmd.getTask_output_userId());
            }
            if (cmd.getOldBonusPoint() != null) {
                 rspProp.put("task_output_oldBonusPoint",
                 cmd.getOldBonusPoint());
            }

} catch (ECException ex) {
            // 原封不動地擲出異常狀況
            throw (ECException) ex;
}
```

儲存您的作業。

上述的程式碼片段會使用指令 Factory 來建立新作業指令。 接著它會設定指令環境定義，呼叫作業指令加以執行，並擷取作業指令中的輸出參數。

3. 按如下輸入您控制程式指令的 URL，以測試該指令：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

範例 JSP 中會列出要求物件中的「名稱-值」對清單，其中包括作業的輸出值。其應類似如下：

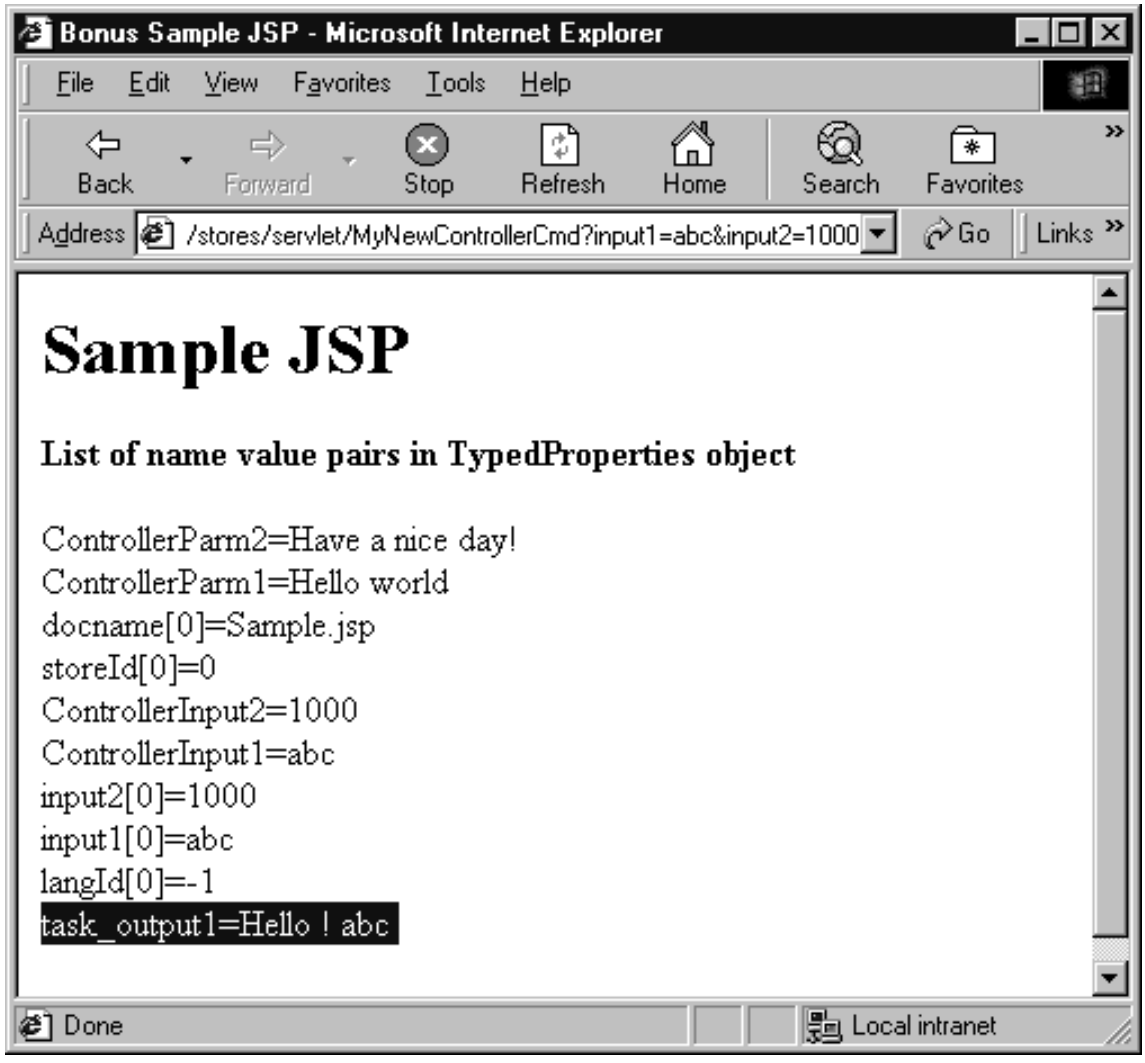


圖 37.

4. 爲現行狀態下的程式碼建立一個版本。爲 `mySample 1.5 Completed` 版本命名。有關如何建立程式碼的版本詳述，請參閱步驟第 190 頁的12。

驗證使用者 ID

下一步驟是將作業指令修改成使用 UserRegistryAccessBean，以驗證使用者所輸入的值是否是一位已登錄使用者所輸入的。除了修改作業指令外，您也必須修改 DataBeanSampleBean。

修改 MyNewTaskCmdImpl 以進行使用者 ID 的驗證: 您必須修改 MyNewTaskCmdImpl 類別中的 performExecute 方法，以便使用 UserRegistryAccessBean 來驗證使用者的 ID。如果要在 performExecute 方法中加入此項新功能，請執行下列步驟：

1. 在「工作台」中，選取 **MyNewTaskCmdImpl** 類別中的 **performExecute** 方法。
2. 在「原始碼」窗格中取消 performExecute 方法中之區段 2 的註解。這會將下列程式碼引入方法中：

```
// 使用 UserRegistryAccessBean 來檢查成員參考號碼

    String refNum;
    UserRegistryAccessBean rrb = new UserRegistryAccessBean();

  try {
        rrb = rrb.findByUserLogonId(getTask_input1());
        refNum = rrb.getUserId();
    } catch (javax.ejb.FinderException e) {

    return;

    } 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");
    }

     setTask_output_userId(refNum);
```

儲存您的工作。

修改 DataBeanSampleBean: 您必須修改 DataBeanSampleBean，以使用您預先定義的輸入參數。如果要修改這個 Bean，請執行下列步驟：

1. 在「工作台」中展開 **com.ibm.commerce.sample.databeans** 套件。
2. 執行下列步驟，將新的欄位加入資料 Bean：
   a. 以滑鼠右鍵按一下 **DataBeanSampleBean** 類別，然後選取**新增 > 欄位**。使用「建立欄位」引導精靈，使用您在下列步驟中指定的值來建立新欄位。如果需要有關建立欄位的額外資訊，請參閱第 195 頁的『建立新欄位』。

b. 使用以下的值，將新欄位加入資料 Bean 中：

屬性名稱	**值**
欄位名稱	input1
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

c. 使用以下的值，將新欄位加入資料 Bean 中：

屬性名稱	**值**
欄位名稱	input2
欄位類型	int
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

d. 使用以下的值，將新欄位加入資料 Bean 中：

屬性名稱	**值**
欄位名稱	task_output_userId
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

3. 選取 DataBeanSampleBean 類別的 **populate** 方法，以檢視其原始碼。

4. 在原始碼中，取消區段 1A 與區段 1B 的註解。這會將下列程式碼引入方法中：

```
//// 區段 1A //////////
// 設定其它資料欄位

try {
   setInput1( getRequestProperties().getString("input1"));
   setInput2( getRequestProperties().getIntValue("input2", 0)  );
   setTask_output_userId(getRequestProperties().getString
     ("task_output_userId"));

//// 區段 1A 結束 /////

   //// 區段 2 ////////////
   /*
   // 將資料 Bean 案例化為 BonusAccessBean
   setTask_output_oldBonusPoint(getRequestProperties().getString(
    "task_output_oldBonusPoint"));
   */
   //// 區段 2 結束 ////

//// 區段 1A //////////
// 將資料 Bean 案例化為 BonusAccessBean 並
// 設定其它資料欄位

   }
catch (ParameterNotFoundException e){}

//// 區段 1B 結束 ////
```

儲存您的工作。

上述的程式碼片段是使用 getRequestProperties 方法，從控制程式指令取得資料欄位的值。

修改 Sample.jsp 以進行使用者 ID 的驗證: 您必須修改現行的 JSP 範本，以進行使用者 ID 的驗證。如果要修改 Sample.jsp 檔，請執行下列步驟：

1. 以文字編輯器開啓 Sample.jsp 與 Sample_All.jsp 檔。
2. 從 Sample_All.jsp，將程式碼中的 <!-- SECTION 3 --> 到 <!-- END OF SECTION 3 --> 標記間的區段 3 複製到 Sample.jsp 中。下列程式碼會引入到 JSP 範本中。

```
<!-- SECTION 3 -->

<B>Your first input is &lt; <%=testBean.getInput1()%> &gt;</B>

<%
String userId = testBean.getTask_output_userId();

if (userId == null) {

%>
```

```
<UL>
     <LI> This is not a registered user id.
</UL>

<%

} else {

%>

<B>
<UL>
     <LI> 'lt; <%=testBean.getInput1()%> &gt; is a registed user id.
     <LI> The member reference number of this user is <%=userId%>.
</UL>
</B>

<%

}

%>

<B>Your second input is < <%=testBean.getInput2()%> ></B> <P>

<!-- END OF SECTION 3 -->
```

儲存 Sample.jsp 檔。

3. 開啓瀏覽器並輸入下列的 URL：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

瀏覽器應顯示範例 JSP 檔，其中指出 input1 的值不是有效的使用者 ID；請見如下的螢幕畫面：

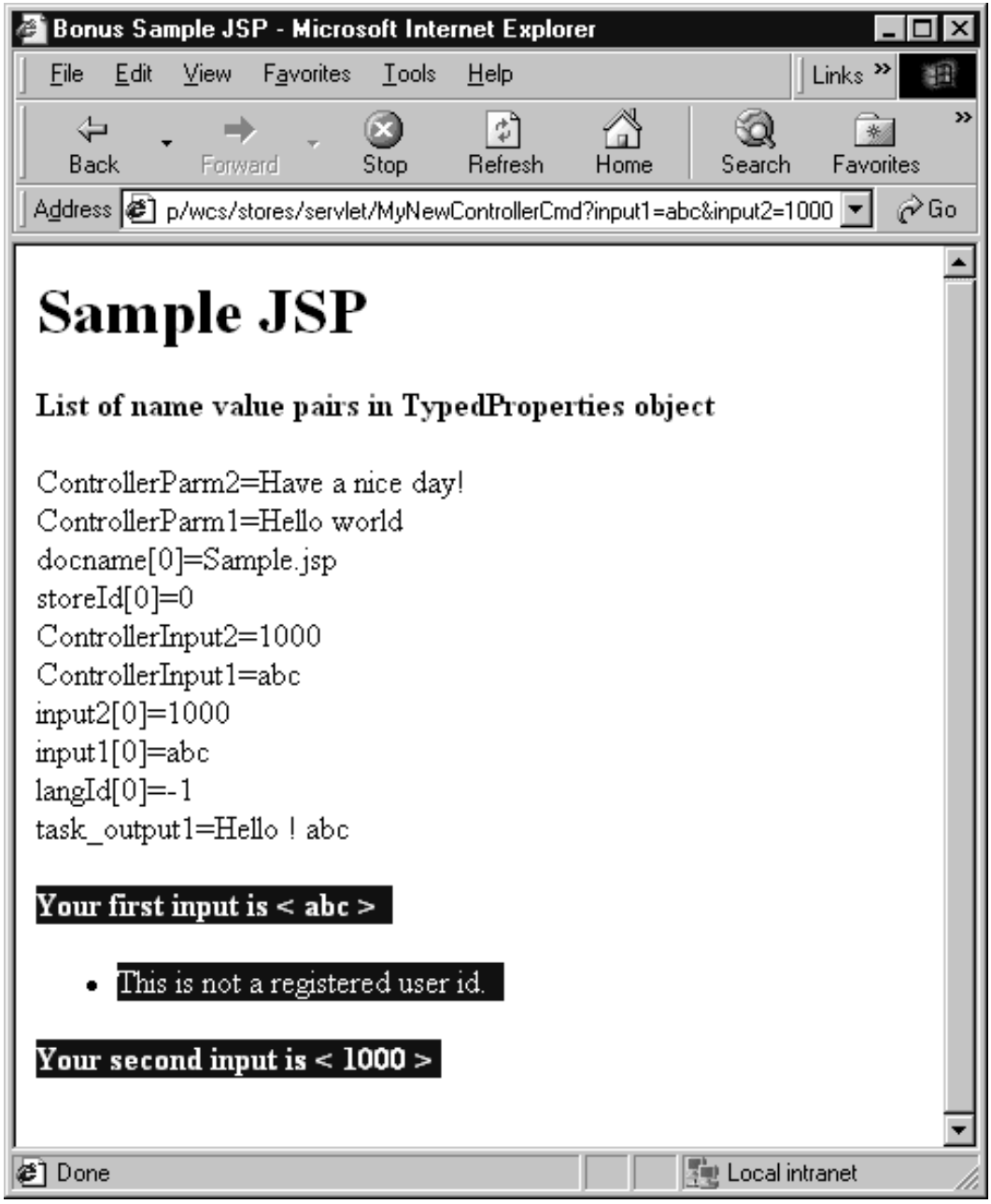


圖 38.

4. 當 input1 的值爲有效的使用者 ID 時，如果要查看結果，請輸入下列 URL：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=wcsadmin&input2=1000
```

這會顯示在下列的螢幕畫面中：

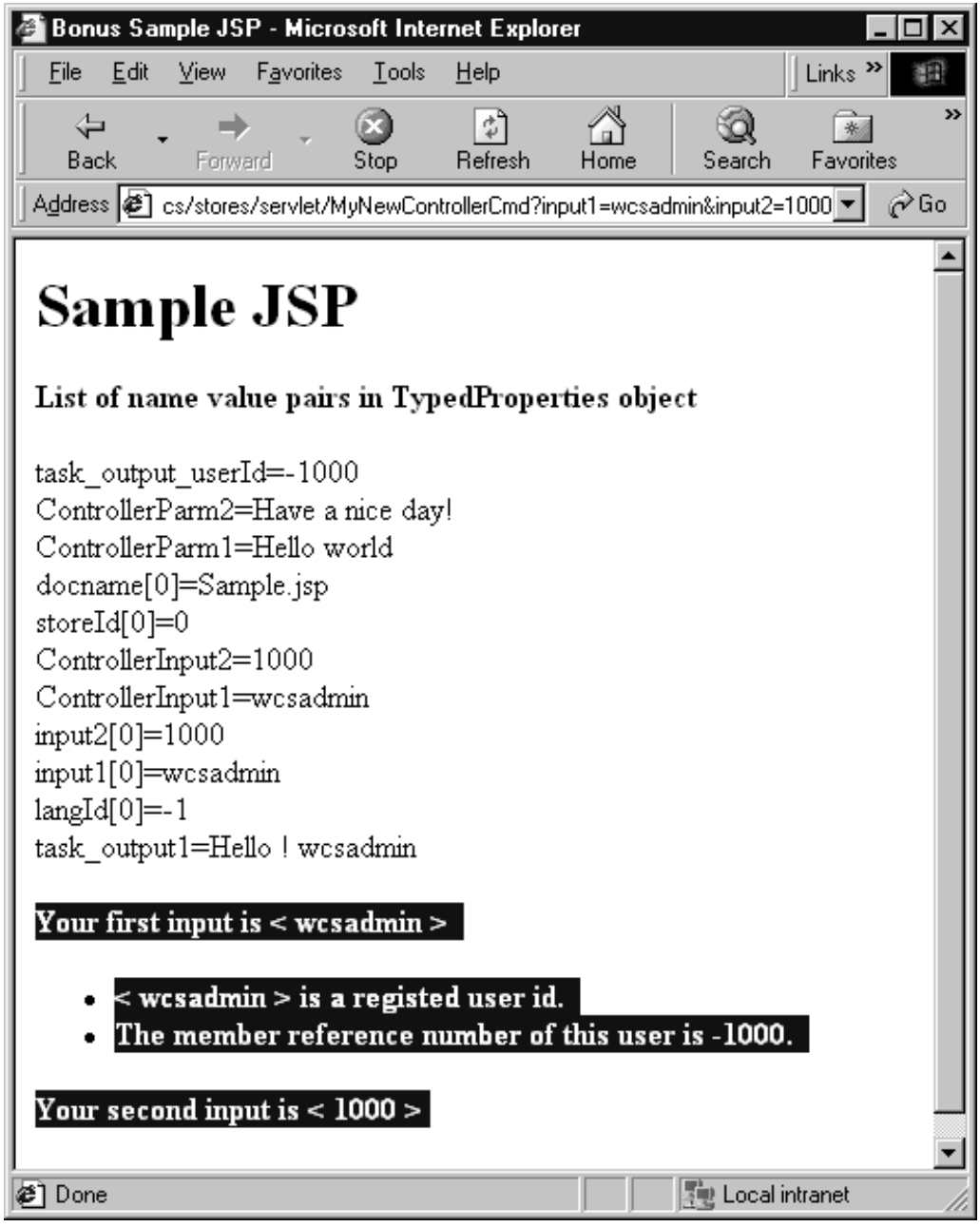


圖 39.

5. 爲現行狀態下的程式碼建立一個版本。爲 `mySample 1.6 Completed` 版本命名。有關如何建立程式碼的版本詳述，請參閱步驟第 190 頁的12。

建立新實體 Bean

本節說明如何使用 VisualAge for Java 來建立新實體 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，一旦您建立資料庫表格，您必須建立一個實體 Bean（爲一個 Enterprise Bean) 以存取資料。

在本範例中，您將使用 VisualAge for Java 的引導精靈來建立此實體 Bean。

建立新資料庫表格

在準備建立實體 Bean 的過程中，您必須先建立新資料庫表格。所要建立的表格稱爲 Bonus。

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 的「指令中心」（**開始 > 程式集 > IBM DB2 > 指令中心**），並按一下 **Scripting** 標籤。
2. 在「Script」視窗中，輸入下列指令：

   ```
   connect to your_database_name;
   CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
   BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
   constraint f_memberid foreign key (MEMBERID)
   references users (users_id) on delete cascade)
   ```

   其中 *your_database_name* 是您的資料庫名稱。然後按一下「執行」圖示。現在 Bonus 表格已建立完成。

   註: 若有人先前曾使用此資料庫來執行本範例，您必須在建立 Bonus 表格前，先發出下列指令：

   ```
   drop table Bonus
   ```

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**主電腦字串**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 Bonus (MEMBERID NUMBER NOT NULL,
BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
constraint f_memberid foreign key (MEMBERID)
references users (users_id) on delete cascade);
```

並按 Enter 鍵，以執行 SQL 陳述式。此時即已建立 BONUS 表格。

註: 若有人先前曾使用此資料庫來執行本範例，您必須在建立 Bonus 表格前，先發出下列指令：

```
drop table Bonus;
```

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 實體 Bean

一旦建立資料庫，即可準備開始建立新實體 Bean。以下的步驟將使用 VisualAge for Java。如果要建立新實體 Bean，請執行下列步驟：

1. 建立 EJB 群組。EJB 群組爲一種邏輯群組，可讓您組織您的 Enterprise Bean。您可以對 EJB 群組執行整體作業，而這會同樣反映在常駐於該群組中的所有 Enterprise Bean 上。舉例來說，當您選出一個要匯出到 EJB JAR 檔中的 EJB 群組，那麼也將會匯出所有的 Enterprise Bean。
   在本指導教學中，您將建立一個 EJB 群組，以組織與您 Bonus 表格之自訂有關的所有 Enterprise Bean。
   如果要建立 EJB 群組，請執行下列步驟：
   a. 在「工作台」中按一下 **EJB** 標籤。
   b. 從 **EJB** 功能表中選取**新增 > EJB 群組**。
      這會開啓「新增 EJB 群組」引導精靈。
   c. 在**專案**欄位中輸入 `_WCSamplesEntityBeansProject`。

      註: 爲了部署，實體 Bean 程式碼必須儲存在其本身的專案中。
   d. 在**建立指名的新 EJB 群組**欄位中，輸入 `WCSSamplesEntityBeans`，並按一下**完成**。
2. 建立新實體 Bean。
   如果要建立新實體 Bean，請執行下列步驟：
   a. 在 Enterprise Bean 窗格中，以滑鼠右鍵按一下 **WCSSamplesEntityBeans** EJB 群組，並選取**新增 > Enterprise Bean**。
      這會開啓「建立 Enterprise Bean」引導精靈。

b. 輸入下列資訊：

屬性	值
Bean 名稱	Bonus **註:** 命名慣例是採用與實體 Bean 所存取之表格相同的名稱，來呼叫該實體 Bean。
Bean 類型	具有由儲存器所管理的持續性（CMP）欄位的實體 Bean
建立新 Bean 類別	enable
專案	_WCSamplesEntityBeansProject
套件	com.ibm.commerce.sample.objects
類別名稱	BonusBean
超類別	com.ibm.commerce.base.objects.ECEntityBean

然後按**下一步**。

c. 按一下**新增 CMP 欄位到 Bean 文字框**旁的**新增**按鈕，以便在 BONUS 表格中新增 MEMBERID 直欄的欄位。
這會開啓「建立 CMP 欄位」引導精靈。

d. 輸入下列資訊：

屬性	值
欄位名稱	memberId
欄位類型	Long **註:** 您必須使用 *Long* 資料類型，而非使用 *long*。
關鍵字欄位	enable

然後按一下**完成**。

e. 再按一下**新增**，以便在 BONUS 表格中新增 BONUSPOINT 直欄的欄位。

f. 以下列資訊建立另一個欄位：

屬性	值
欄位名稱	bonusPoint
欄位類型	Integer **註:** 您必須使用 *Integer* 資料類型，而非使用 *int*。
使用 getter 與 setter 方法存取	enable
將 getter 與 setter 方法引介到遠端介面中	enable

然後按一下「建立 CMP 欄位」視窗中的**完成**。

g. 執行下列步驟，以使用存取控制保護 Enterprise Bean：

1) 按一下**遠端介面應從哪些介面延伸而來**？旁的**新增**。
2) 在**型樣**欄位中輸入 com.ibm.commerce.security.Protectable，並按一下**新增**。按一下**關閉**，關閉視窗。

h. 再按一下**完成**。

Bonus 實體會建爲一個 Enterprise Bean。

3. 執行下列步驟，以設定實體 Bean 的隔離層次：
   a. 以滑鼠右鍵按一下 **Bonus** Bean，然後選取**內容**。
   b. 從**隔離層次**下拉清單中，選取 **TRANSACTION_READ_COMMITTED**，然後按一下**確定**。
4. 在您建立新 Enterprise Bean 時，VisualAge for Java 會在 Bean 中產生一個 EntityContext 欄位以及對應的 corresponding getEntityContext() 與 setEntityContext(EntityContext) 方法。根據 WebSphere Commerce 程式設計模型，您的新 Bean 是從 com.ibm.commerce.base.objects.ECEntityBean 類別延伸而來，且 ECEntityBean 會提供本身的施行給這個欄位與這些方法。由於不應改寫 EntityContext、getEntityContext 與 setEntityContext，此時您必須將產生的欄位與方法自 Bean 中刪除。
   如果要刪除產生的 EntityContext 欄位與其 getter 與 setter 方法，請執行下列步驟：
   a. 在「類型」窗格中，選取 **BonusBean** 類別。
   「成員」窗格中會顯示此類別的欄位與方法。

   註: 如果看不到「類型」窗格，請按一下「內容」窗格中的 C/I（類別/介面）圖示。這會開啓「類型」窗格。

   b. 在「成員」窗格中，執行下列步驟：
      1) 以滑鼠右鍵按一下 **entityContext** 欄位，並選取**刪除**。
      2) 以滑鼠右鍵按一下 **getEntityContext()** 方法，並選取**刪除**。
      3) 以滑鼠右鍵按一下 **setEntityContext(EntityContext)** 方法，並選取**刪除**。
   c. 儲存您的工作 (Ctrl+S)。
5. 執行下列步驟，將新 getMemberId 方法新增到 Enterprise Bean 中：
   a. 以滑鼠右鍵按一下 **BonusBean** 類別，並選取**新增 > 方法**。
   會開啓「建立方法」引導精靈。
   b. 使用下表中指定的值，來建立新方法。有關建立新方法的詳細資訊，請參閱第 205 頁的『建立方法』。

表 11.

屬性名稱	**值**
方法名稱	getMemberId
傳回類型	Long
參數	無
異常狀況	無

c. 當產生新方法時，請檢視原始碼。

d. 在預設的情況下，方法中含有下列程式碼：

```
return null;
```

將此程式碼改爲

```
return memberId;
```

e. 以滑鼠右鍵按一下 **getMemberId** 方法，並選取**新增至 > EJB 遠端介面**，以便將新方法新增到遠端介面中。

6. 在 BonusBeanFinderHelper 中新增新 FinderHelper 欄位。
此介面含有一個對應至 FinderHelper 方法（將在下個步驟中建立）的搜尋子句。
如果要新增 FinderHelper 欄位，請執行下列步驟：

a. 在「類型」窗格中，按一下 **BonusBeanFinderHelper** 介面。

b. 修改「原始碼」窗格中的程式碼，使其類似如下：

```
public interface BonusBeanFinderHelper {
public static final String
                    findByMemberIdWhereClause = " (MEMBERID = ?) ";
}
```

註: "WhereClause" 的語法相當重要；它必須與 FinderHelper 方法中所用的方法名稱相符。在本範例中，findByMemberIdWhereClause 中的 "findByMemberId" 與您在下個步驟中所建的方法名稱（findByMemberId）一模一樣。

7. 在 BonusHome 介面中新增新 FinderHelper 方法。
如果要新增新 FinderHelper 方法，請執行下列步驟：

a. 在「類型」窗格中，以滑鼠右鍵按一下 **BonusHome** 介面並選取**新增 > 方法**。
這會開啓「建立方法」引導精靈。

b. 選取**建立新方法**，然後按**下一步**。

c. 在 **MethodName** 欄位中輸入 findByMemberId。

d. 在 **ReturnType** 欄位中輸入 Bonus。

e. 按一下**此方法應有哪些參數**？旁的**新增**。
這會開啓「參數」視窗。

f. 在**名稱**欄位中輸入 argMemberId。

g. 選取**參照類型**，並輸入 Long。按一下**新增**，然後按一下**關閉**。

h. 按**下一步**。

i. 按一下**此方法可能擲出的異常狀況**欄位旁的**新增**，並在**型樣**欄位中輸入 RemoteException，然後按一下**新增**。 這會在「屬性」視窗中新增 java.rmi.RemoteException 異常狀況。（「屬性」視窗可能位於「異常狀況」視窗後面。）

j. 在「異常狀況」視窗的**型樣**欄位中，輸入 FinderException，按一下 **新增**，然後按一下**關閉**。javax.ejb.FinderException 異常狀況會列於「屬性」視窗中。

k. 按一下**完成**。

8. 在 EJB 中新增新 ejbCreate 方法。此方法會引介到發源介面中，以便能在產生的存取 Bean 中用到此方法。如果要建立此方法，請執行下列步驟：

a. 選取「類型」窗格中的 **BonusBean** 類別。

b. 按一下「成員」窗格中的 **ejbCreate(long)**。

c. 修改程式碼以符合下列：

```
public void ejbCreate(java.lang.Long argMemberId,
   Integer argBonusPoint)
throws javax.ejb.CreateException, java.rmi.RemoteException {
    initLinks();
// 在此應起始設定所有的 CMP 欄位。
memberId=argMemberId;
            bonusPoint=argBonusPoint;
}
```

d. 儲存程式碼，且 VisualAge for Java 會在「成員」窗格中建立一個新方法，名稱爲 **ejbCreate(Long, Integer)**。

e. 以滑鼠右鍵按一下原始的 **ejbCreate(Long)** 方法，並選取**刪除**。

9. 新增新方法到發源介面中。這可在存取 Bean 類別中提供該方法。其做法是執行下列步驟：

a. 以滑鼠右鍵按一下 BonusBean 類別中的 **ejbCreate(Long, Integer)** 方法，並選取**新增至 > EJB 發源介面**。

10. 執行下列步驟，以更新 getOwner 方法：

a. 選取「類型」窗格中的 **BonusBean** 類別。

b. 按一下「成員」窗格中的 **getOwner()** 方法。

註: 如果您選擇要檢視繼承的方法，您將會看到兩個 getOwner() 方法。其中一個是繼承自 ECEntityBean 類別。在此步驟中您不應選取此方法。請確定您所選的是 BonusBean 類別的特定 getOwner 方法。

c. getOwner 方法的原始碼類似如下：

```
public Long getOwner()
  throws Exception, java.rmi.RemoteException
   {
      return null;
   }
```

您必須變更方法所傳回的值。下列以粗體字顯示您應變更的程式碼部份：

```
public Long getOwner()
  throws Exception, java.rmi.RemoteException
   {
      return getMemberId();
   }
```

儲存您的工作。

d. 按一下「成員」窗格中的 **fulfills(Long, String)** 方法。

註: 如果您選擇要檢視繼承的方法，您將會看到兩個 fulfills(Long, String) 方法。其中一個是繼承自 ECEntityBean 類別。在此步驟中您不應選取此方法。請確定您所選的是 BonusBean 類別的特定 fulfills(Long, String) 方法。

e. fulfills(Long, String) 方法的原始碼類似如下：

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException
  {
      return false;
  }
```

您必須指定使用者必須執行的關係。若要達到此目的，您必須變更下列以粗體字顯示的程式碼部份：

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException
  {

      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }      return false;
  }
```

儲存您的工作。

11. 將 BONUS 資料庫表格映射至 BonusBean。將資料庫綱目映射至 BonusBean 實體的第一個步驟將牽涉到使用 VisualAge for Java 中的工具來建立資料庫綱目。請執行下列步驟以建立綱目：
    a. 在「工作區」視窗的 **EJB** 功能表中，選取**開啓到 > 資料庫綱目**。
    b. 從**綱目**功能表中選取**匯入/匯出綱目 > 匯入資料庫中的綱目**。
    這會開啓「必要資訊」視窗。
    c. 在**綱目名稱**欄位中輸入 `WCSSamples`，並按一下**確定**。
    這會開啓「資料庫連線資訊」視窗。
    d. 填寫下列資訊：

屬性	DB2 **DB2 值**	Oracle **Oracle 值**
連線類型	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
資料來源	`jdbc:db2:`*`wcs_db_name`*	`jdbc:oracle:thin:@`*`hostname`*`:`*`port`*`:`*`SID`*
使用者名稱	*`wcs_db_user_name`*	*`wcs_db_user_name`*
密碼	*`wcs_db_password`*	*`wcs_db_password`*

    按如下所述置換各值：

    - DB2 *`wcs_database_name`* 爲您 WebSphere Commerce 資料庫的名稱
    - Oracle *`hostname`* 爲 Oracle 主電腦名稱
    - Oracle *`port`* 爲 Oracle 資料庫的埠號（例如 `1521`）。
    - *`wcs_db_user_name`* 爲資料庫的使用者名稱。
    - *`wcs_db_password`* 爲資料庫密碼。

    按一下**確定**。
    這會開啓「選取表格」視窗。
    e. 從**限定元**清單中，選取您資料庫的限定元（可能是您資料庫的使用者名稱或機器名稱），然後按一下**建置表格清單**。將會載入可用的資料庫表格清單。
    f. 從「表格」畫面中選取 **Bonus**，並按一下**確定**。請靜待一下。
    g. 在綱目瀏覽器中按一下剛剛新增的表格，這會出現表格中的兩個直欄。
    h. 以滑鼠右鍵按一下 **Bonus** 表格，並選取**編輯表格**。
    會開啓「表格編輯器」。

i. 移除**限定元**欄位中的任何項目。移除限定元資訊，可讓您將程式碼部署到使用不同資料庫的其它機器中。

j. Oracle 依下列所示，修改直欄資料類型：

1) 選取 **MEMBERID** 直欄，並按一下**編輯**。從「類型」下拉清單中選取 **BIGINT**，並按一下**確定**。

2) 選取 **BONUSPOINT** 直欄，並按一下**編輯**。從「類型」下拉清單中選取 **INTEGER**，並按一下**確定**。

k. 按一下**確定**，以結束「表格編輯器」。

l. 從**綱目**功能表中選取**儲存綱目**。
這會開啓「儲存綱目」視窗。

m. 輸入下列資訊：

屬性	值
專案	_WCSamplesEntityBeansProject
套件	com.ibm.commerce.sample.objects
類別名稱	WCSSamplesSchema

然後按一下**完成**，並關閉綱目瀏覽器。

12. 一旦建立綱目後，您即可建立綱目映射。如果要在 BONUS 表格與 BonusBean 實體間建立此映射，請執行下列步驟：

a. 從 **EJB** 功能表中選取**開啓到 > 綱目映射**。這會出現映射瀏覽器。

b. 在映射瀏覽器中，從**資料儲存庫映射**功能表中選取**新 EJB 群組映射**。
這會開啓「新資料儲存庫映射」視窗。

c. 填寫下列資訊：

屬性	值
名稱	WCS 範例
EJB 群組	WCSSamplesEntityBeans
綱目	WCSSamples

然後按一下**確定**。

d. 在「資料儲存庫映射」畫面中，按一下 **WCS 範例**。

e. 在「持續性類別」畫面中，按一下 **Bonus**。

f. 從**表格映射**功能表中選取**新表格映射 > 新增表格映射且不繼承**。

g. 從**表格**下拉清單中，選取 **Bonus**，然後按一下**確定**。

h. 在「表格映射」畫面中選取 **Bonus**，然後以滑鼠右鍵按一下它，並選取**編輯內容映射**。
這會開啓「內容映射編輯器」。

i. 依照下列的說明設定屬性：

類別屬性	映射類型	表格直欄
memberId	Simple	MEMBERID
bonusPoint	Simple	BONUSPOINT

然後按一下**確定**。

j. 從**資料儲存庫映射**功能表中，選取**儲存資料儲存庫映射**。
這會開啓「儲存資料儲存庫映射」。

k. 輸入下列資訊：

屬性	值
專案	_WCSamplesEntityBeansProject
套件	com.ibm.commerce.sample.objects
類別名稱	WCSSamplesMap

然後按一下**完成**，並關閉映射瀏覽器。

13. 一旦建立 BonusBean 實體，且正確映射綱目後，您必須爲實體 Bean 建立存取 Bean。此存取 Bean 可方便應用程式存取 Bonus 實體 Bean 中所含的資訊。 VisualAge for Java 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 Bonus 實體 Bean 建立存取 Bean，請執行下列步驟：

a. 在「工作台」中選取 EJB 標籤，並以滑鼠右鍵按一下 **Bonus** Enterprise Bean，並選取**新增 > 存取 Bean**。
這會開啓「建立存取 Bean」引導精靈視窗（可能要稍待一下）。

b. 確定您已輸入下列資訊：

屬性	值
EJB 群組	WCSSamplesEntityBeans
Enterprise Bean	Bonus
存取 Bean 名稱	BonusAccessBean
存取 Bean 類型	實體 Bean 的 Copy Helper

然後按**下一步**。

c. 從**選取零引數建構元的發源方法**下拉清單中，選取 **findByMemberId(Long)**。

d. 在 **init_argMemberId**（位於「起始內容」直欄中）方面，將 **Converter** 設爲 com.ibm.commerce.base.objects.WCSStringConverter，並按**下一步**。

e. 在 bonusPoint 方面，確定已選出 **CopyHelper**，並將 **Converter** 值設爲 com.ibm.commerce.base.objects.WCSStringConverter，並按一下**完成**。

註: 您不必對 memberId 欄位進行任何修改。

f. 當出現「順利產生程式碼」訊息時，請按一下**確定**。

您可以切換至**專案**標籤，展開 **_WCSamplesEntityBeansProject** 專案，並展開 **com.ibm.commerce.sample.objects**，以檢視新產生的程式碼。新類別 BonusAccessBean 會出現在套件內。

14. 下個步驟是產生部署程式碼。
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）施行方式與施行類別，以及爲 CMP Bean 產生 JDBC persister 與 finder 類別。此外亦會產生 Java ORB、Stub 與 Tie 類別，以透過 IIOP 進行 RMI 存取，並爲發源與遠端介面產生 Stub。
如果您所選的 EJB 群組中含有一個 CMP Enterprise Bean，或者您選取一個個別的 CMP Enterprise Bean，則亦會產生下列項目：
    - 將產生到 Persister 類別中的建立表格字串
    - 與表格間相映射的 Persister 施行方式

    如果要產生部署程式碼，請執行下列步驟：

    a. 在 Enterprise Bean 畫面中選取 EJB 標籤，並以滑鼠右鍵按一下 **Bonus** Enterprise Bean，然後選取**產生部署程式碼**。程式碼的產生可能需要幾分鐘。

15. 在測試 Bonus enterprise bean 之前，您必須先建立一個新的 EJB 伺服器，其中包含所有的 WebSphere Commerce EJB 群組以及新的 WCSSamplesEntityBeans EJB 群組。這些群組必須在同一部伺服器上執行，才能維持異動範圍。一旦建立好新的伺服器，就必須加以啓動。
如果要建立並啓動新的 EJB 伺服器，請執行下列步驟：

    a. 確定所有的 EJB 群組皆已收合。

    b. 在 Enterprise Bean 窗格中，選取所有的 WebSphere Commerce EJB 群組以及您的新 EJB 群組（亦即，選取所有開頭爲 WCS 的 EJB 群組）。

    c. 在選出這些 EJB 群組的情況下，按一下滑鼠右鍵並選取**新增至 > 伺服器架構**。
    這會開啓「EJB 伺服器架構」視窗（可能需要稍待一下）。

d. 以滑鼠右鍵按一下新建立的 EJB 伺服器（例如 **EJB Server (server2)**），並選取**內容**，然後填寫下列各項：

屬性	DB2 **DB2 值**	Oracle **Oracle 值**
資料來源	WebSphere Commerce DB2 DataSource *instance_name*	WebSphere Commerce Oracle DataSource *instance_name*
連線類型	<DataSource>	<DataSource>
使用者名稱	*wcs_db_user_name*	*wcs_db_user_name*
密碼	*wcs_db_password*	*wcs_db_password*
交易逾時	1200	1200
交易休止逾時	600000	600000

然後按一下**確定**。

註: 資料來源的值必須符合 *instance_name*.xml 檔中指定的資料來源值。

視您開發機器的硬體（例如：處理器速度）而定，您可能需增加**交易逾時**與**交易休止** EJB 伺服器內容的值。

e. 如果 WebSphere Test Environment 已在執行中，請依照第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所描述的，加以停止並停止永久名稱伺服器與其他 EJB 伺服器。

f. 依照第 313 頁的『啓動與停止永久名稱伺服器』中所描述的，啓動永久名稱伺服器。

g. 依照第 314 頁的『啓動與停止 EJB 伺服器』中所描述的，啓動新 EJB 伺服器。

16. 一旦您啓動 EJB 伺服器，即可啓動測試從屬站。當使用測試從屬站時，您將在資料庫中建立新記錄。如果要啓動測試從屬站，並建立此記錄，請執行下列步驟：

a. 在選取 EJB 標籤的情況下，以滑鼠右鍵按一下 **Bonus** Enterprise Bean，並選取**執行測試從屬站**。

b. 在「查看 EJB」視窗中，按一下**查看**。
這會開啓 Bonus 視窗。

c. 按一下 **create(Long, Integer)**。在「明細」窗格中，填妥下列資訊：

屬性	值
Long	-1000

屬性	值
Integer	100

然後按一下「EJB 測試從屬站」視窗中的「呼叫」圖示。

註: 第一個屬性必須與任何已登錄使用者的 memberId 相符。您可以查看 USERS 表格來決定 memberId。

17. 直接查詢資料庫，以驗證資料庫記錄是否建立正確。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟：

a. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）

b. 選取**非作用**標籤。

c. 輸入 `connect to wcs_database_name`，並按一下「執行」圖示。

d. 輸入 `SELECT * FROM Bonus`，並按一下「執行」圖示。
應會傳回下列各值。

直欄	值
MEMBERID	-1000
BONUSPOINT	100

上述記錄是由您的 EJB 測試從屬站所建。

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（**開始 > 程式集 > Oracle - OraHome81 > 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**主電腦字串**欄位中，輸入您的連接字串。

e. 在 SQL Plus 視窗中輸入下列：

```
select * from BONUS;
```

並按 Enter 鍵，以執行 SQL 陳述式。
應會傳回下列值：

直欄	值
MEMBERID	-1000
BONUSPOINT	100

18. 執行下列步驟，以使用測試從屬站來驗證 Bonus Enterprise Bean 是否能順利存取資料庫記錄：
    a. 在「Bonus」視窗中，選取 **Home** 標籤。
    b. 在「方法」畫面中按一下 **findByMemberId(Long)**。
    c. 在 **Long** 欄位中輸入 `-1000`，並按一下「呼叫」圖示。
    d. 先選取**遠端**標籤，再展開**方法**，之後按一下 **getBonusPoint**，然後按一下「呼叫」圖示。
       「詳細資訊」畫面中會顯示產生的整數 `100`。
    e. 關閉「Bonus」和「EJB 測試從屬站」兩個視窗。

整合 Bonus 實體 Bean 與 MyNewControllerCmd

在上一節中，您使用 VisualAge for Java 中產生的測試從屬站來測試新 Bonus 實體 Bean。在本節中，您將整合 Bonus 實體 Bean 與 MyNewControllerCmd 邏輯。一旦 Java 程式碼有所更新，即會更新 Sample.jsp 範本而建立一個介面，以容許更新購物者的紅利積點結餘。

整合 Bonus 實體 Bean 將牽涉到如下的高階步驟：

1. 修改 MyNewTaskCmdImpl 類別的 performExecute 方法，以計算新紅利積點，並將紅利積點儲存到 BONUS 表格中。
2. 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。
3. 爲新資源建立新存取控制原則。
4. 修改 DataBeanSampleBean，使其從 Bonus 實體 Bean 的存取 Bean 延伸而來。藉由讓資料 Bean 由存取 Bean 延伸而來，可讓資料 Bean 繼承存取 Bean 中的所有屬性。
5. 修改 DataBeanSampleBean 中的方法。
6. 修改 WebSphere Test Environment 中之 Servlet 引擎的類別路徑，以包含新 _WCSamplesEntityBeansProject。
7. 修改 Sample.jsp 範本，讓使用者可輸入紅利積點並顯示結果。

修改 MyNewTaskCmdImpl 以計算紅利積點: MyNewTaskCmdImpl 是作爲 Bonus 實體 Bean 與 MyNewControllerCmd 間的整合點（這是因爲 MyNewControllerCmd 會呼叫 MyNewTaskCmd）。

如果要修改 MyNewTaskCmdImpl 以執行紅利積點的計算，請執行下列步驟：

1. 在「VisualAge for Java 工作台」視窗中，展開 **_WCSamples** 專案。
2. 展開 **com.ibm.commerce.sample.commands** 套件，並選取 **MyNewTaskCmdImpl** 類別以檢視其原始碼。

3. 取消下列 import 陳述式的註解：

```
import com.ibm.commerce.sample.objects.*;
```

儲存您的工作 (**Ctrl + S**)。

4. 選取 **MyNewTaskCmdImpl** 類別中的 **performExecute** 方法。
5. 在 performExecute 方法的原始碼中，取消區段 3 的註解。這會將下列程式碼引入方法中：

```
// 使用 BonusAccessBean 來更新新紅利積點

String newBonusPoint = null;
BonusAccessBean bb = new BonusAccessBean();
try {
     if (refNum != null) {
          bb.setInit_argMemberId(refNum);
          bb.refreshCopyHelper();
          oldBonusPoint = bb.getBonusPoint();
     }
} catch (javax.ejb.FinderException e) {
   try {
          bb = new BonusAccessBean(new Long(refNum),new Integer(0));
          oldBonusPoint = "0";
     } catch (javax.ejb.CreateException ec) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
               this.getClass().getName(), "performExecute");
     } catch (javax.naming.NamingException ec) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
               this.getClass().getName(), "performExecute");
     } catch (java.rmi.RemoteException ec) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_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");
}


try {
     if (oldBonusPoint != null) {
          int newBP =  Integer.parseInt(oldBonusPoint) + getTask_input2();
          newBonusPoint = Integer.toString( newBP );
          bb.setBonusPoint( newBonusPoint )   ;
          newBonusPoint=bb.getBonusPoint();
          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");
}
```

儲存您的工作。

新增 getResources 方法到 MyNewControllerCmdImpl 類別中: 在本節中，您將新增 getResources 方法到 MyNewControllerCmdImpl 類別中。此方法會傳回指令在處理程序期間所用的資源清單。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. 選取**專案**標籤，並展開 **_WCSamples** 專案。
2. 展開 **com.ibm.commerce.sample.commands** 套件。
3. 選取 **MyNewControllerCmdImpl** 類別，以檢視其原始碼。
4. 在原始碼中，取消存取控制區段的註解。此區段如下列的程式碼片段所示：

```
public AccessVector getResources() throws ECException{

// 使用 UserRegistryAccessBean 來檢查成員參考號碼

   String refNum;
   String methodName="getResources";

   com.ibm.commerce.user.objects.UserRegistryAccessBean rrb =
      new com.ibm.commerce.user.objects.UserRegistryAccessBean();

   try {
      rrb = rrb.findByUserLogonId(getInputString());
          refNum = rrb.getUserId();
   }
   catch (javax.ejb.FinderException e) {

      throw new ECSystemException(ECMessage._ERR_BAD_USER_NAME,
         this.getClass().getName(),methodName);

   }
   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);
   }


   //尋找此使用者的 Bonus Bean
String newBonusPoint = null;
   com.ibm.commerce.sample.objects.BonusAccessBean bb =
      new com.ibm.commerce.sample.objects.BonusAccessBean();
   try {
     if (refNum != null) {
          bb.setInit_argMemberId(refNum);
          bb.refreshCopyHelper();
      }
   }
catch (javax.ejb.FinderException e) {

   // 使用者沒有 Bonus 物件，因而會在建立時
   // 傳回存放 Bonus 物件的儲存器
   return new AccessVector(rrb);
   }
   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 new AccessVector(bb);

}
```

儲存您的工作。 (Ctrl+S)。

一旦您儲存上述的程式碼區段，VisualAge for Java 即會將欄位與存取元和這個特定檢視畫面隔開。請注意，此時它們會出現在 MyNewControllerCmdImpl 類別下，且方法會標上 M。

註: 本指導教學爲力求簡明，而將資源物件建在此 getResources 方法中。在實際的應用程式中，最好將資源物件建立在 validateParameters 方法中，並將之儲存成案例變數。如此一來，getResources 與 performExecute 方法便可重複使用物件。

為新資源設置存取控制原則: 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 實體 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。舉例來說，如果使用者登入成 wcsadmin 使用者，使用者只能修改 wcsadmin 的紅利積點。

設置 AllUsersUpdateBonusResourceGroup 原則涉及下列步驟：

1. 使用 acpload 指令載入 SampleACPolicy.xml 檔。
2. 使用 acpnlsload 指令載入 SampleACPolicy_***locale***.xml 說明。
3. 重新整理原則登錄。請注意，只有在載入存取控制原則期間 Servlet 引擎正在執行時，才需執行此步驟。

如果要設置 AllUsersUpdateBonusResourceGroup 原則，請執行下列步驟：

1. 在指令提示下，切換至下列目錄：
   drive:\WebSphere\CommerceServerDev\bin
2. 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

舉例來說，您可發出下列指令：

```
acpload VAJ_Demo user password SampleACPolicy.xml
```

3. 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 VAJ_Demo user password SampleACPolicy_en_US.xml
```

4. 如果 WebSphere Test Environment 的 Servlet 引擎正在執行，請停止再重新啓動，以重新整理原則登錄。

修改紅利積點的 DataBeanSampleBean: 在本節中，您將執行下列步驟，而將 DataBeanSampleBean 修改成延伸 BonusAccessBean：

1. 在「工作台」中展開 **com.ibm.commerce.sample.databeans** 套件。
2. 執行下列步驟，將新的欄位加入資料 Bean 中：
   a. 以滑鼠右鍵按一下 **DataBeanSampleBean** 類別，然後選取**新增 > 欄位**。使用「建立欄位」引導精靈，根據下列步驟的說明來建立新的欄位。 如果需要有關建立欄位的額外資訊，請參閱第 195 頁的『建立新欄位』。
   b. 使用以下的值，將新欄位加入資料 Bean 中：

屬性名稱	**值**
欄位名稱	task_output_oldBonusPoint
欄位類型	String
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

按下**完成**。

3. 按一下 **DataBeanSampleBean** 類別來檢視其原始碼。在原始碼中，取消下列 import 陳述式的註解：

```
import com.ibm.commerce.sample.objects.*;
```

儲存您的工作。

4. 在 DataBeanSampleBean 類別的原始碼中，取消區段 1 的註解，並將區段 2 標成註解行。這會變更 Bean 使其從 BonusAccessBean 延伸而來。在進行這些修改後，程式碼將類似如下：

```
/// 區段 1 ////////////////////////////////////////////////

// 將資料 Bean 延伸為 BonusAccessBean

public class DataBeanSampleBean
   extends com.ibm.commerce.sample.objects.BonusAccessBean
   implements SmartDataBean {


//////////////////////////////////////////////////////////////

/// 區段 2 ////////////////////////////////////////////////
/*
// 將資料 Bean 延伸為 BonusAccessBean

     public class DataBeanSampleBean implements SmartDataBean {
*/

//
//////////////////////////////////////////////////////////////
```

儲存您的工作。

5. 選取 **setTask_output_userId(String)** 方法以檢視其原始碼。找出下面這一行的程式碼：

```
     public void setTask_output_userId(java.lang.String newTask_output_
userId) {
task_output_userId = newTask_output_userId;
```

在前一行之後，輸入下面的程式碼，以便將新的 BonusAccessBean 案例化：

```
//////////////////////////////////////
// 區段 A : 案例化 BonusAccessBean

if (task_output_userId != null)
     this.setInit_argMemberId(newTask_output_userId);

//////////////////////////////////////
```

儲存您的工作。

6. 選取 **populate()** 方法，以檢視其原始碼。取消區段 2 的註解，以設定 BonusAccessBean 的案例。這會將下列程式碼引入方法中：

```
setTask_output_oldBonusPoint(getRequestProperties().getString(
    "task_output_oldBonusPoint"));
```

修改類別路徑: 您必須先修改類別路徑以包含建給新 Bonus 實體 Bean 的新專案，您才能在 WebSphere Test Environment 中測試您所修改的指令。如果要修改此類別路徑，請執行下列步驟：

1. 從 VisualAge for Java 的**工作區**功能表中選取**工具 > WebSphere Test Environment**。
   這會開啓 WebSphere Test Environment 控制中心。
2. 按一下 **Servlet 引擎**。
3. 如果 Servlet 引擎正在執行，請按一下**停止 Servlet 引擎**，然後按一下**編輯類別路徑**。
4. 按一下**全選**，然後按一下**確定**。

修改紅利積點的 Sample.jsp 範本: 如果要修改顯示範本，請執行下列步驟：

1. 以文字編輯器開啓 Sample.jsp 與 Sample_All.jsp 檔。
2. 從 Sample_All.jsp，將程式碼中的 <!-- SECTION 4 --> 到 <!-- END OF SECTION 4 --> 標記間的區段 4 複製到 Sample.jsp 中。下列程式碼會引入到 JSP 範本中。

```
<!-- SECTION 4 -->

<h1>
Bonus Administration
</h1>

<%
if (userId != null) {
%>

<B>
<UL>
   <LI> The bonus point before update is
      <%=testBean.getTask_output_oldBonusPoint()%>
   <LI> The bonus point after update is
      <%=testBean.getBonusPoint()%>
</UL>
</B>

<%
}
%>

<br>
<B>Input to command:</B><P>

<FORM NAME=Bonus ACTION="MyNewControllerCmd">
<TABLE>
   <TR>
         <TD>
```

```
            <B>Logon ID </B>
          </TD>
          <TD>
            <input type=text name=input1 value='<%=testBean.getInput1()%>'>
          </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>

<!-- END OF SECTION 4 -->
```

儲存 Sample.jsp 檔。

3. 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。您必須將 Sample.jsp 檔複製到商店的目錄結構中，這是因爲一旦您登入商店， Web 控制程式會在商店目錄中搜尋 Sample.jsp 檔。請將 Sample.jsp 檔從
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web
   複製到
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*。

   註: 對範例商店而言， *store_directory* 的值爲 InFashion。

4. 確定 WebSphere Test Environment 正在執行中（請參閱第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』）。
5. 執行下列步驟，以便以 wcsadmin 使用者登入：
   - 在瀏覽器中輸入下列 URL：

     ```
     http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
          storeId=store_Id&catalogId=catalog_Id&langId=-1
     ```

   - 按一下**登錄**鏈結。
     會出現「登錄或登入」頁面。

- 在**電子郵件位址**欄位中輸入 wcsadmin。
- 在**密碼**欄位中輸入 wcsadmin 使用者的密碼，然後按一下**登入**。

註: 原始密碼爲 wcsadmin，不過當您在安裝程序期間執行 contractPublish 指令時，即會變更此密碼。請步驟的說明請見下列文件：

– Business *WebSphere Commerce Studio Business Developer Edition 安裝手冊*

– Professional *WebSphere Commerce Studio Professional Developer Edition 安裝手冊*

6. 在登入完成後，請在同一瀏覽器中輸入下列 URL：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=wcsadmin&input2=1000
```

將會出現類似下面的畫面，其中含有先前所有的輸出參數，以及一份可讓您更新使用者之紅利積點結餘的套表；請見如下的螢幕畫面：

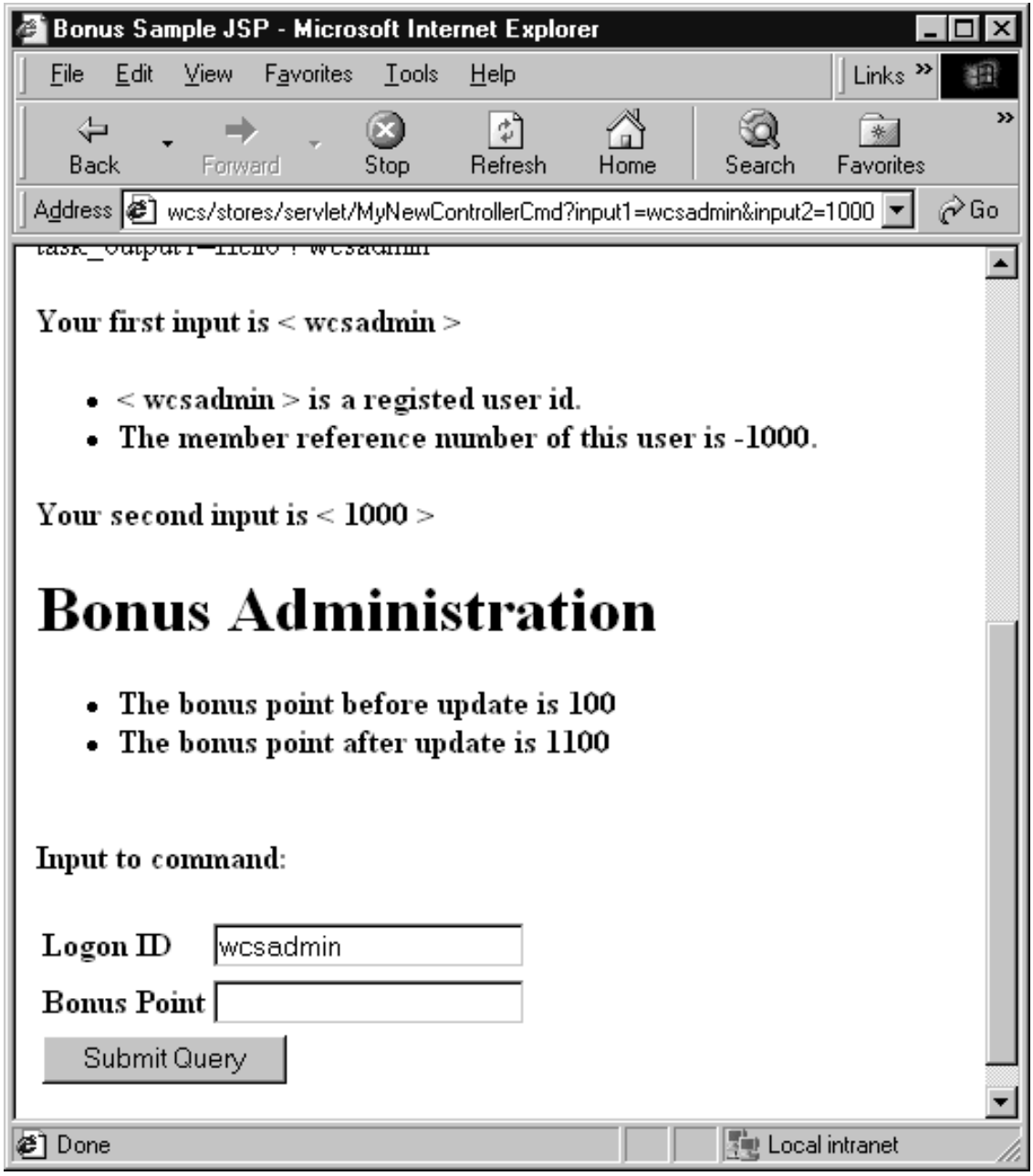

Bonus Sample JSP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Back Forward Stop Refresh Home Search Favorites
Address wcs/stores/servlet/MyNewControllerCmd?input1=wcsadmin&input2=1000
Go
Your first input is < wcsadmin >
< wcsadmin > is a registed user id.
The member reference number of this user is -1000.
Your second input is < 1000 >
Bonus Administration
The bonus point before update is 100
The bonus point after update is 1100
Input to command:
Logon ID
wcsadmin
Bonus Point
Submit Query
Done
Local intranet

圖 40.

7. 爲現行狀態下的程式碼建立一個版本。爲 mySample 1.7 Completed 版本命名。在您建立程式碼的版本時，請確定兩個專案皆有選出。有關如何建立程式碼的版本詳述，請參閱步驟（第 190 頁的12）。

（選用）使用 VisualAge for Java 中的除錯器

本節教您如何在程式碼中新增岔斷點，並啓動 VisualAge for Java 的「除錯器」元件。其中包含「除錯器」的介紹。有關此一強大特性的詳述，請參閱 VisualAge for Java 的線上說明。

本節爲選用內容，因爲在此不會介紹指導教學所需的任何新程式碼。而是教您建立一個岔斷點，以便在遇到岔斷點時驗證一些變數的值，然後再移除岔斷點。

在程式碼中新增岔斷點

如果要在程式碼中新增岔斷點，請執行下列步驟：

1. 執行下列步驟，以確定您工作區中有啓用岔斷點：
   a. 從**視窗**功能表中選取**除錯**。確定**全域啓用岔斷點**已選取。
2. 選取**專案**標籤。
3. 展開 **_WCSamples** 專案。
4. 展開 **com.ibm.commerce.sample.commands** 套件。
5. 展開 **MyNewTaskCmdImpl** 類別。
6. 選取 **performExecute** 方法，以檢視其原始碼。
7. 在「原始碼」窗格中，將游標置於（並按一下）程式碼中之下行的開頭：

   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   將游標留在此位置。
8. 從**編輯**功能表中選取**岔斷點**。
9. 在「架構：岔斷點 #1」視窗中選取**在所選的執行緒中**，並按一下**確定**。
10. 確定 WebSphere Test Environment 正在執行中（請參閱第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』）。
11. 執行下列步驟，以便以 wcsadmin 使用者登入：
    - 在瀏覽器中輸入下列 URL：

      ```
      http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
          storeId=store_Id&catalogId=catalog_Id&langId=-1
      ```

    - 按一下**登錄**鏈結。
      會出現「登錄或登入」頁面。
    - 在**電子郵件位址**欄位中輸入 wcsadmin。

- 在**密碼**欄位中輸入 wcsadmin 使用者的密碼，然後按一下**登入**。

12. 在完成登入程序後，請輸入下列 URL：

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=wcsadmin&input2=1000
```

13. 當一到達程式碼中的該岔斷點處時，即會開啓「除錯器」視窗。

驗證變數的值

本節說明如何驗證 task_input1 與 task_output1 變數在執行指令期間於各點處的值。

當因 MyNewTaskCmdImpl 之 performExecute 方法中的岔斷點而開啓「除錯器」時，請切換至該視窗中，並執行下列步驟：

1. 在「變數」窗格中展開**此項**。
2. 按一下 **task_input1**。
   在「值」窗格中，會顯示此變數在執行期間一到此點時的值。應該是顯示 wcsadmin。

如果要驗證 task_output1 變數值的設定一如預期，您必須讓「除錯器」繼續執行 performExecute 方法，以到達程式碼中設定該變數之處。其做法如下：

1. 使用「跳過副程序」功能，逐步在程式碼中移動。其做法有下列兩種：
   - 從**已選取**功能表中選取**跳過副程序**。
   - 按一下 F6。
   - 按一下「跳過副程序」圖示。

   就本例的目的而言，請按 F6 四次。這會執行設有 task_output1 變數的程式碼。
2. 在「變數」窗格中按一下 task_output1。
   在「值」窗格中，會顯示此變數在執行期間一到此點時的值。應該是顯示 Hello ! wcsadmin。
3. 您可以按一下「回復」圖示，以完成指令的執行。

移除岔斷點

如果要將岔斷點從程式碼中移除，請執行下列步驟：

1. 切回到「工作台」視窗中。
2. 請確定您可看到 MyNewTaskCmdImpl 類別之 performExecute 方法的原始碼。
3. 在「原始碼」窗格中，導覽至程式碼中的下行：

```
setTask_output1( "Hello ! " + getTask_input1() );
```

   請注意，在窗格的左邊界中會有一個藍點標示出岔斷點。

4. 按兩下該藍點，以移除岔斷點。

此時，岔斷點已從您的程式碼中移除。

整合 MyNewControllerCmd 與 WebSphere Test Environment 中的範例商店

在本節中，您將新增範例商店首頁的鏈結（會呼叫 MyNewControllerCmd）。如果要執行此項整合步驟，請執行下列步驟：

1. 在文字編輯器中，開啓 sidebar.jsp 檔。此檔案位於下列目錄中：
   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*\include
   其中 *vaj_drive* 爲您安裝 VisualAge for Java 的磁碟機，*store_directory* 爲範例商店的目錄名稱。
2. 將下列程式碼插入到最後一個 </table> 標籤之前，以便在表格中新增內含 MyNewControllerCmd 鏈結的另一列：

   ```
   <tr>
   <td>
   <a href = "MyNewControllerCmd?input1=wcsadmin&input2=1000">
        MyNewControllerCmd</a>
   </td>
   </tr>
   ```

   儲存您的工作。
3. 確定 WebSphere Test Environment 正在執行。
4. 在瀏覽器中輸入下列 URL，以測試此項整合：

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

   按一下**登錄**鏈結。
   會出現「登錄或登入」頁面。
5. 在**電子郵件位址**欄位中輸入 wcsadmin。
6. 在**密碼**欄位中輸入 wcsadmin 使用者的密碼，然後按一下**登入**。
7. 在使用者登入後，按一下側邊導覽窗格中的 **MyNewControllerCmd** 鏈結。會出現範例 JSP。

   註: 如果側邊導覽窗格未顯示新鏈結，則可能是快取了 sidebar.jsp。請將之從快取中移除，並重新載入頁面。有關刪除已編譯 JSP 檔的進一步資訊，請參閱第 349 頁的附錄 C, 『VisualAge for Java 的相關要訣』。在您刪除已編譯的 JSP 檔後，您可能需重新啓動 Servlet 引擎。

（選用）將新商業邏輯部署到遠端 WebSphere Commerce Server 中

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，您必須已在遠端 WebSphere Commerce Server 上建立一家商店（以「時尚館」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。

為新指令邏輯建立 JAR 檔

您必須建立一個內含新指令與資料 Bean 邏輯的 JAR 檔。如果要建立此 JAR 檔，請執行下列步驟：

1. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所述，停止 WebSphere Test Environment。
2. 選取「專案」標籤，並選取 **_WCSamples** 專案。
3. 在專案呈高亮度顯示的情況下，按一下滑鼠右鍵並選取**匯出**。
   這會開啓「匯出」引導精靈。
4. 選取 **Jar 檔**，並按**下一步**。
5. 在「JAR 檔」欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev\mytemp\wcssamples_1.jar
   其中 *drive* 爲安裝 Commerce Studio 的磁碟機。
6. 按如下選取屬性：

屬性	**值**
類別	勾選
java	未勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
7. 按下**完成**。
8. 當出現提示時，請確認新目錄的建立。

由於所建的 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中，導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp
2. 輸入 mkdir temp1。
3. 輸入 cd temp1。
4. 按如下設定路徑：
   set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio4\bin;
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 jar xvf ../wcssamples_1.jar
6. 輸入 jar cvf ../wcssamples.jar * （請記得拿掉名稱中的 _1）。

為新 EJB 群組建立 JAR 檔

您必須爲您的新 EJB 群組建立 JAR 檔。如果要建立此檔案，請執行下列步驟：

1. 選取 EJB 標籤，並以滑鼠右鍵按一下 **WCSSamplesEntityBeans** EJB 群組，然後選取**匯出 > EJB 1.1 JAR**。
   這會開啓「匯出到 EJB 1.1 JAR 檔」引導精靈。
2. 在 **JAR 檔**欄位中輸入
   drive:\WebSphere\CommerceServerDev\mytemp\sampleEntityBeans_DT.jar。
3. 按如下選取屬性：

屬性	**值**
類別	勾選
java	未勾選
資源	勾選
目標資料庫	DB2 如果您是部署到 DB2 資料庫，請選取 **DB2 for NT, V7.1**。 Oracle 如果您是部署到 Oracle 資料庫，請選取 **Oracle, V8**。
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
4. 按下**完成**。

此時會建立 JAR 檔。

JAR 檔名稱中已加上 "_DT" 字尾，如此可提醒您在將之部署到 WebSphere Commerce 應用程式前，必須透過 WebSphere Application Server 所提供的「EJB 部署工具」執行此 JAR 檔。

為新 Enterprise Bean 建立施行 JAR 檔

您必須建立一個內含 Bonus Enterprise Bean 之施行程式碼的 JAR 檔。如果要建立此檔案，請執行下列步驟：

1. 選取「專案」標籤，並以滑鼠右鍵按一下 **_WCSSamplesEntityBeansProject**，然後選取**匯出**。
   這會開啓「匯出」引導精靈。
2. 選取 **Jar 檔**，並按**下一步**。
3. 在 **JAR 檔**欄位中輸入
   drive:\WebSphere\CommerceServerDev\mytemp\sampleImpl_1.jar。
4. 按如下選取屬性：

屬性	**值**
類別	勾選
java	未勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
5. 按下**完成**。

此時會建立 JAR 檔。 關閉 VisualAge for Java。

由於所建的 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中，導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp
2. 輸入 mkdir temp2。
3. 輸入 cd temp2。
4. 按如下設定路徑：
   set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio4\bin;
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 jar xvf ../sampleImpl_1.jar
6. 輸入 jar cvf ../sampleImpl.jar * （請記得拿掉名稱中的 _1）。

將 JSP 檔複製到目標 WebSphere Commerce Server 中

您得將 Sample.jsp 與更新過的 sidebar.jsp 檔複製到將部署程式碼之商店的正確目錄中。

在您將更新過的 JSP 範本複製到目標 WebSphere Commerce Server 前，您可在該機器上建立幾份原始 JSP 範本的備份。亦即，將現有檔案重新命名爲 sidebar.jsp.bak。

如果要複製這些檔案，請執行下列步驟：

1. 在開發機器上，導覽至下列目錄：
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*；其中 *vaj_drive* 爲您安裝 VisualAge for Java 的磁碟機，*store_directory* 爲商店的目錄名稱。複製 Sample.jsp。
2. 將 Sample.jsp 貼到目標 WebSphere Commerce Server 上的下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\store_directory
   ```

   其中 *drive* 爲安裝 WebSphere Commerce 的磁碟機， *store_directory* 爲商店的目錄名稱， *instance_name* 爲您 WebSphere Commerce 案例的名稱。
3. 在開發機器上，導覽至下列目錄：
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*\include
   複製 sidebar.jsp
4. 將 sidebar.jsp 貼到目標 WebSphere Commerce Server 上的下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\store_directory\include
   ```

將 JAR 檔複製到目標 WebSphere Commerce Server 中

您必須將 JAR 檔從開發機器複製到目標 WebSphere Commerce Server 上的適當目錄中。

此外，您必須針對內含新 EJB 群組的 JAR 檔額外執行兩個處理步驟。首先，您必須針對該檔案執行 WebSphere Application Server 中的 EJB 部署工具。然後，針對該檔案執行 modifyIsolationLevel 指令。如此一來，在將 JAR 檔複製到目標 WebSphere Commerce Server 的這個步驟中，即會將此特定 JAR 檔儲存在暫時目錄中，以待進行這些進一步的步驟。

如果要複製這些檔案，請執行下列步驟：

1. 在開發機器上導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp，然後找出下列檔案：
   - wcssamples.jar
   - sampleImpl.jar
   - sampleEntityBeans_DT.jar

   其中 *drive* 爲您安裝 WebSphere Commerce Studio, Business Developer Edition 的磁碟機。

   上述每一個檔案皆必須複製到目標 WebSphere Commerce Server 的特定目錄中。請仔細閱讀下列步驟，以確定每一個檔案皆儲存在正確位置中。
2. 將 wcssamples.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\AppServer\InstalledApps\
   WC_Enterprise_App_instance_name.ear\wcstores.war\WEB-INF\lib
   ```

   其中 *drive* 爲您安裝 WebSphere Commerce Business Edition 的磁碟機，*instance_name* 爲您案例的名稱（例如 demo）。
3. 建立您將放置 JAR 檔的暫時程式庫目錄。亦即，建立下列目錄：

   ```
   drive:\WebSphere\CommerceServer\temp\lib
   ```
4. 將 sampleImpl.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\CommerceServer\temp\lib
   ```
5. 將 sampleEntityBeans_DT.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\CommerceServer\temp
   ```

執行 EJB 部署工具

您必須先產生 Enterprise Bean 的部署程式碼，才能順利在測試或正式作業伺服器中執行您的 Enterprise Bean。 WebSphere Application Server 所提供的 Enterprise JavaBeans 部署工具（亦稱爲 EJB 部署工具）會提供一個指令行介面，讓您用來產生 Enterprise Bean 部署程式碼。

如果要執行此工具，請執行下列步驟：

1. 在指令提示下，切換至下列目錄：

   ```
   drive:\WebSphere\CommerceServer\temp
   ```
2. 輸入下列指令，暫時將工具新增到系統路徑中：

   ```
   PATH=drive:\WebSphere\AppServer\deploytool;%PATH%
   ```
3. 依下列所示輸入 ejbdeploy 指令：

```
ejbdeploy EJBGroupJARFile WorkingDir OutputJARFile -nowarn -keep -35 -cp
   ClassPathOfDepJARFiles
```

其中：

- *EJBGroupJARFile* 爲您想產生部署程式碼之 Enterprise Bean 的完整 JAR 檔名稱。在本例中，此爲
  drive:\WebSphere\CommerceServer\temp\sampleEntityBeans_DT.jar。
- *WorkingDir* 爲目錄名稱，用來儲存產生程式碼時需使用的暫存檔。
- *OutputJARFile* 爲輸出 JAR 檔的完整名稱。就本例而言，請輸入
  drive:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar。
- -nowarn 爲選用參數，用以抑制警告與參考訊息的出現。
- -keep 爲選用參數，用以在執行 ejbdeploy 指令後保留工作目錄。
- -35 爲必要參數，表示將針對 WebSphere Application Server 3.5 版所提供之「EJB 部署工具」中所用的 CMP 實體 Bean，採用相同的由上往下映射規則。
- -cp *ClassPathOfDepJARFiles* 爲任何相依 JAR 檔的類別路徑。就本例而言，請輸入

  ```
  "drive:\WebSphere\CommerceServer\temp\lib\sampleImpl.jar
  drive:\WebSphere\AppServer\installedApps\
     WC_Enterprise_App_instanceName.ear\lib\wcsejbimpl.jar"
  ```

  註: 您必須以引號 (“”) 括住類別路徑。

修改 Bonus Bean 的交易隔離層次

在本步驟中，您將使用 modifyIsolationLevel 指令來修改 Bonus Bean 的交易隔離層次。此工具亦會將 Bean 的隔離層次設爲您特定資料庫類型所要的層次。

如果要執行 modifyIsolationLevel 指令，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 上，開啓指令視窗。
2. 切換至下列目錄：
   drive:\WebSphere\CommerceServer\bin
3. 您必須發出採用下列一般語法的 modifyIsolationLevel 指令：

   ```
   modifyIsolationLevel -jarFile jar_file_name.jar
      -logFile log_file_name -dbType db_type
   ```

   其中

   - *jar_file_name.jar* 爲內含自訂程式碼的 JAR 檔名稱
   - *log_file_name* 爲應記載資訊的完整檔案名稱
   - *db_type* 爲您所用的資料庫類型。輸入 DB2 或 ORACLE

以下是設有所有指定值的 modifyIsolationLevel 指令範例：

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

如果指令視窗中沒有顯示異常狀況，表示指令執行成功。在完成後，請記下您部署 JAR 檔變更的時間戳記。

註: 參數名稱有區分大小寫。亦即，jar**F**ile 不等於 jar**f**ile。請確定您所輸入的參數名稱無誤。

更新目標資料庫

由於您要將新商業邏輯部署到使用有別於 WebSphere Test Environment 所用資料庫的目標 WebSphere Commerce Server 上，因此您必須更新目標資料庫，以反映對指令登錄所做的變更以及建立 BONUS 表格。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟，以更新目標資料庫：

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**使用陳述式終止字元**勾選框，並確定所指定的字元是分號 (**;**)。
4. 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 your_database_name;
   insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
       DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
       'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
       'This is a new controller command for test/education purposes.',
       null)
   ```

   其中 *your_database_name* 爲資料庫名稱；請按一下「執行」圖示。此指令是供所有商家使用（亦即，將 STOREENT_ID 之值設爲 0）。
5. 在 Script 視窗中輸入下列字串，以便在 VIEWREG 表格中建立項目：

   ```
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
      ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
      'docname=Sample.jsp','This is a sample view for the Bonus Point
      exercise', 0, null)
   ```

   然後按一下「執行」圖示。
6. 在 Script 視窗中輸入下列資訊，以便建立 BONUS 表格：

```
CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
constraint f_memberid foreign key (MEMBERID)
references users (users_id) on delete cascade)
```

按一下「執行」圖示。

上述步驟是將 MyNewControllerCmd 與 SampleViewTask 登錄在指令登錄中，並建立 BONUS 表格。

Oracle 如果您所用的是 Oracle 資料庫，請執行下列步驟，以更新目標資料庫：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

   ```
   insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
       DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
       'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
       'This is a new controller command for test/education purposes.',
       null);
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。
6. 在 SQL Plus 視窗中輸入下列資訊，以便在 VIEWREG 表格中建立項目：

   ```
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
      ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
      'docname=Sample.jsp','This is a sample view for the Bonus Point
      exercise', 0, null);
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。
7. 在 SQL Plus 視窗中輸入下列資訊，以便建立 BONUS 表格：

   ```
   CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
   BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
   constraint f_memberid foreign key (MEMBERID)
   references users (users_id) on delete cascade);
   ```

   並按 Enter 鍵，以執行 SQL 陳述式。
8. 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。

載入新資源的存取控制原則

在指導教學中，您將建立一個新 Enterprise Bean（即 Bonus Bean），且其爲可保護的資源。因而，此資源會有一個相關的存取控制原則。此外，您也已建立一個可供所有使用者執行的新控制程式指令。而在您於開發機器上工作時，您已將存取控制原則資訊載入該機器中。此時，您必須將相同的存取控制原則資訊載入到目標 WebSphere Commerce Server 中。

如果要設置存取控制原則，請執行下列步驟：

1. 將下列 CD 插入光碟機中：
   - Business WebSphere Commerce Business Edition V5.4 Disk 2 CD
   - Professional WebSphere Commerce Professional Edition V5.4 Disk 2 CD
2. 切換至下列目錄：

   CD_drive:\repository\samples\programguide\
3. 在該目錄中找出下列檔案：
   - SampleCmdACPolicy.xml
     此 XML 檔中含有新控制程式指令所用的存取控制原則。
   - SampleACPolicy.xml
     此 XML 檔中含有您在建立新 Enterprise Bean 時所用的存取控制原則。
   - SampleACPolicy_*locale*.xml
     其中 *locale* 爲語言識別碼。此 XML 檔含有存取控制原則的說明。
4. 將上述三個檔案複製到下列目錄中：
   drive:\WebSphere\CommerceServer\xml\policies\xml
5. 如果要載入 SampleCmdACPolicy.xml 檔，請在指令提示下切換至下列目錄：
   drive:\WebSphere\CommerceServer\bin
   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

   舉例來說，您可發出下列指令：

   acpload mall user password SampleCmdACPolicy.xml

6. 發出 acpload 指令並將 SampleACPolicy.xml 檔指定成輸入檔，以載入 SampleACPolicy.xml 檔。舉例來說，您可發出下列指令：

   ```
   acpload mall user password SampleACPolicy.xml
   ```

7. 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 mall user password SampleACPolicy_en_US.xml
   ```

請注意，爲了讓原則生效，一般而言需要重新整理登錄。但就本例而言，您不必執行此步驟，這是因爲您將在 WebSphere Application Server 中停止再重新啓動 WebSphere Commerce Server 應用程式，而此屬 Enterprise Bean 部署步驟中的環節。如果您的情況並非如此，您可以使用 WebSphere Commerce 中的「管理主控台」來更新登錄。有關「管理主控台」的詳細資訊，請參閱 WebSphere Commerce 線上說明。

▶ 400 如果您是部署到執行於 iSeries 機器上的 WebSphere Commerce 案例中，您將使用不同的指令來載入存取控制原則資訊。請使用 LODWCSAC 指令（而非 acpload 指令）與 LODWCSACD 指令（而非 acpnlsload 指令）。

LODWCSAC 指令的語法爲：

```
LODWCSAC DATABASE(dbName) SCHEMA(schemaName)
PASSWD(instancePassword) INSTROOT('instanceRoot')
INFILE('inputFile')
```

其中

- *dbName* 爲定義在 WRKRDBDIRE 指令的關聯式資料庫名稱。
- *schemaName* 爲案例的資料庫綱目名稱（和案例名稱同名）。
- *instancePassword* 爲案例密碼。
- *instanceRoot* 爲案例的起始位置。例如，案例起始位置爲：

  /QIBM/UserData/WebCommerce/instances/*instanceName*
- *inputFile* 爲內含存取原則之輸入 XML 檔的完整名稱

LODWCSACD 指令的語法爲：

```
LODWCSACD DATABASE(dbName) SCHEMA(schemaName)
PASSWD(instancePassword) INSTROOT('instanceRoot')
INFILE('inputFile')
```

您可將存取控制原則的 XML 檔儲存在下列目錄中：

/QIBM/UserData/WebCommerce/instances/*instanceName*

此外，在您存取控制原則的 XML 檔中，您必須使用存取控制 DTD 的完整路徑。存取控制原則的 DTD 儲存在 /QIBM/ProdData/WebCommerce/xml/policies/dtd 目錄中。

舉例來說，如果您將教學指導用的存取控制原則部署到執行於 iSeries 機器上的 WebSphere Commerce 案例中，您必須在教學指導用之存取控制原則的 XML 檔中，將 DTD 規格從：

```
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">
```

改爲

```
<!DOCTYPE Policies SYSTEM "/QIBM/ProdData/WebCommerce/
xml/policies/dtd/accesscontrolpolicies.dtd">
```

有關 WebSphere Commerce 存取控制模型的詳細資訊，請參閱 *WebSphere Commerce 存取控制手冊*。

從 WebSphere Application Server 匯出現行企業應用程式

在此步驟中，您將從 WebSphere Application Server 匯出現行企業應用程式，以便在應用程式組譯工具中開啓。

如果要匯出現行企業應用程式，請執行下列步驟：

1. 建立目錄，以便將現行企業應用程式匯出到此。請注意，您並不會呼叫這個「暫時」目錄，這是因爲在您確定您已滿意自訂程式碼部署後的行爲模式前，您應不願冒著在常態系統維護期間檔案被刪的風險。如果要建立此目錄，請在指令提示下執行下列步驟：
   a. 導覽至下列目錄：

      drive:\WebSphere\CommerceServer\

   b. 輸入下列指令：

      ```
      mkdir working
      ```

      這會建立 *drive*:\WebSphere\CommerceServer\working 目錄。
2. 開啓 WebSphere Application Server 管理主控台。
3. 展開 **WebSphere 管理網域**。
4. 展開**企業應用程式**。
5. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **示範**應用程式，並選取**匯出應用程式**。
6. 在**匯出目錄**欄位中輸入 *drive*:\WebSphere\CommerceServer\working。這會將整個應用程式（包括所有資源）匯出到 WC_Enterprise_App_*instanceName*.ear 檔中（其中 *instanceName* 爲您 WebSphere Commerce 案例的名稱）。
7. 按一下**確定**。匯出應用程式可能需要幾分鐘。

匯出企業應用程式的 XML 架構資訊

您也必須匯出企業應用程式的 XML 架構資訊。如果要匯出此資訊，您可使用 WebSphere Application Server 所提供的 XMLConfig 指令行公用程式。

如果要匯出此架構資訊，請執行下列步驟：

1. 將 was.export.app.xml 檔從下列目錄：

   drive:\WebSphere\CommerceServer\xml\config

   複錄到下列目錄中：

   drive:\WebSphere\CommerceServer\working

2. 以文字編輯器開啓 was.export.app.xml 檔。在此檔案中，將所有 $Enterprise_Application_Name$ 換成

   WebSphere Commerce Enterprise Application - *instanceName*

   其中 *instanceName* 爲 WebSphere Commerce 案例的名稱（例如 demo）。儲存此檔案。

   註: 您所插入的值必須符合 WebSphere 進階管理主控台所顯示的案資訊。
3. 在指令提示下，切換至下列目錄：

   drive:\WebSphere\CommerceServer\working
4. 輸入下列指令，呼叫 XMLConfig 工具以執行局部匯出：

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
      -adminNodeName wasHostName
   ```

   其中 *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。此外，OutputFile.xml 爲因執行此指令而建立的檔案，was.export.app.xml 爲您在步驟 2 中所修改的檔案。

在您匯出現行企業應用程式中的相關 Enterprise Bean 資訊後，您必須在 XML 檔中加上一個新段落，以說明 Bonus Bean。

如果要新增一個說明 Bonus Bean 的新段落，請執行下列步驟：

1. 導覽至下列目錄：

   drive:\WebSphere\CommerceServer\working
2. 以文字編輯器開啓 OutputFile.xml 檔。
3. 找出 <ear-file-name> 標籤，並將值換成：

   drive:\WebSphere\CommerceServer\working\
      WC_Enterprise_App_*instanceName*.ear
4. 您也必須加入有關 Bonus Bean 的新段落；請見下列範例：

   ```
   <ejb-module name="WCSSamplesEntityBeans">
      <jar-file>sampleEntityBeans.jar</jar-file>
      <module-install-info>
         <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
            WebSphere Commerce Server - demo/</application-server-full-name>
      </module-install-info>
      <ejb-module-binding>
         <data-source>
            <jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
            <default-user>user</default-user>
            <default-password>password</default-password>
         </data-source>
         <enterprise-bean-binding name="Bonus_Binding">
   ```

```
            <jndi-name>democom/ibm/commerce/sample/objects/Bonus</jndi-name>
        </enterprise-bean-binding>
    </ejb-module-binding>
</ejb-module>
```

其中

- *user* 爲您資料庫使用者名稱。
- *password* 爲您資料庫使用者的密碼。

註:

a. 上述範例中的折行只爲了方便閱讀。

b. 確定 **$hostName$** 值符合現行管理節點伺服器名稱。此外，請確定此行中沒有回車字元。

c. `<application-server-full-name>` 規格必須以單行顯示。

d. 如果您使用 Oracle 資料庫，您必須修改資料來源資訊。請將下行（取自上述程式碼片段）：

```
<jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
```

改爲：

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource demo</jndi-name>
```

e. 當您部署自己的應用程式時（例如，本教學指導以外的），請確定 XML 檔中指定的 Enterprise Bean JNDI 名稱符合 VisualAge for Java 中所用的 JNDI 名稱，只是其前頭多了 WebSphere Commerce 案例名稱。

5. 儲存 `OutputFile.xml` 檔。

將新 EJB 群組組譯到企業應用程式中

在此步驟中，您將在應用程式組譯器工具中開啓您的企業應用程式。一旦在該工具內開啓後，您可執行下列步驟，將新 Bonus Bean 新增到企業應用程式中：

1. 匯入新 Bonus Bean。新 EJB 群組的 JAR 檔儲存在企業應用程式的「EJB 模組」區段中。
2. 設定 Bonus Bean 的類別路徑，以包含施行 JAR 檔。
3. 將施行 JAR 檔新增到應用程式中。此 JAR 檔儲存在企業應用程式的「檔案」區段中。
4. 爲 Bonus Bean 中的方法設置 WebSphere Application Server 安全特性。

如果要將新 EJB 群組組譯到企業應用程式中，請執行下列步驟：

1. 執行下列步驟，以備份現行企業應用程式：
   a. 在指令提示下，切換至下列目錄：

drive:\WebSphere\CommerceServer\working

b. 輸入下列指令：

```
copy WC_Enterprise_App_instanceName.ear
   WC_Enterprise_App_instanceName.ear.bak
```

2. 開啓 WebSphere Application Server 管理主控台。
3. 從**檔案**功能表中選取**工具 > 應用程式組譯工具**。
4. 如果「歡迎使用」視窗開啓，請選取**取消**，以關閉該視窗。
5. 執行下列步驟，以開啓您要工作的企業應用程式：
   a. 從**檔案**功能表中，選取**開啓**。
   b. 在**檔案名稱**欄位中輸入：

      drive:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_*instanceName*.ear

      然後按一下**開啓**。等待應用程式開啓，再繼續進行後續步驟。需要等待幾分鐘。
6. 以滑鼠右鍵按一下 **EJB 模組**，並選取**匯入**。
7. 在**檔案名稱**欄位中輸入：

   drive:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar

   然後按一下**開啓**。在「確認值」視窗中按一下**確定**。
8. 一旦匯入 sampleEntityBeans.jar 檔後，請捲動到 **WCSSamplesEntityBeans** EJB 群組，並選取此群組。
   此群組的資訊會顯示在右窗格中。
9. 在新 Enterprise Bean 的類別路徑欄位中，輸入任何相依的 JAR 檔。就本例而言，請輸入

   lib/sampleImpl.jar lib\wcsejbimpl.jar
10. 按一下**套用**。
11. 執行下列步驟，以新增 sampleImpl.jar 檔到應用程式中：
    a. 以滑鼠右鍵按一下企業應用程式的**檔案**節點，並選取**新增檔案**。企業應用程式的**檔案**節點較靠近階層樹狀結構底端。請注意，在企業應用程式中另有元件的其它「檔案」節點，但您必須選取整個應用程式的「檔案」節點。
    b. 在「新增檔案」視窗中按一下**瀏覽**。
    c. 導覽至 *drive*:\WebSphere\CommerceServer\temp。
    d. 標示出此目錄，並按一下**選取**。

e. 回到「新增檔案」視窗。請注意，會顯示 *drive*:\WebSphere\CommerceServer\temp 目錄的內容。請標示 lib 目錄。lib 目錄的內容會出現在右窗格中。

f. 在右窗格中選取 sampleImpl.jar 檔，並按一下**新增**。檔案會出現在「所選檔案」中。

g. 按一下**確定**。

12. 執行下列步驟，以架構 Bonus Bean 的安全：

a. 展開「EJB 模組」節點，找出 **WCSSamplesEntityBeans** 節點並展開。

b. 展開**實體 Bean**。

c. 展開 **Bonus**。

d. 按一下**方法延伸**，然後在右窗格中執行下列步驟：

1) 按一下**進階**標籤。

2) 確定**安全身份**已選取。

3) 針對每一個方法，確定**使用 EJB 伺服器的身份**已選取。

4) 按一下**套用**（如果您有做任何修改的話）。

e. 在左導覽窗格中，以滑鼠右鍵按一下 WCSamplesEntityBeans EJB 群組下的**安全職務**，並選取**新建**，然後執行下列步驟：

1) 在**名稱**欄位中，輸入 WCSecurityRole，並按一下**套用**。請注意，如果此職務已存在，便不需執行此步驟。

f. 在左導覽窗格中，以滑鼠右鍵按一下 WCSamplesEntityBeans EJB 群組下的**方法許可權**，並選取**新建**，然後執行下列步驟：

1) 在**方法許可權名稱**欄位中輸入 WCMethodPermission。

2) 在**方法**選擇區中按一下**新增**。
這會開啓「新增方法」視窗。

3) 依序展開 **sampleEntityBeans.jar**，**Bonus** 以及方法的每一個**起源**與**遠端**清單。

4) 按住 Shift 鍵，並選取每一個起源方法，並按一下**確定**。

5) 重複方法選擇程序，以新增遠端方法（如有遠端方法的話）。

6) 在「職務」選擇區中，按一下**新增**，並選取 WCSecurityRole，然後按一下**確定**。

7) 分別針對每一項更新，按一下**套用**。

13. 從**檔案**功能表中，選取**儲存**。

14. 關閉「應用程式組譯器工具」。

在完成此步驟後，您便已建立一個新企業應用程式，且其含有先前的所有邏輯以及您的新商業邏輯。這全含在新修改的 WC_Enterprise_App_*instanceName*.ear 檔中。

將新企業應用程式匯入至 WebSphere Application Server 中

下列的高階步驟是將新企業應用程式匯入到 WebSphere Application Server 中：

1. 停止目前正在 WebSphere Application Server 中執行的企業應用程式，然後移除之。這些步驟是在 WebSphere Application Server 管理主控台中進行。
2. 使用 XMLConfig 指令行公用程式匯入新應用程式。
3. 重新整理 WebSphere Application Server 管理主控台，然後啓動新企業應用程式。

下列各節將進一步說明上述每一個步驟。

停止並移除現行企業應用程式

如果要停止並移除 WebSphere Application Server 中的現行企業應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**。
3. 展開**節點**。
4. 展開 *nodeName* （其中 *nodeName* 爲節點名稱。）
5. 展開**應用程式伺服器**。
6. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server -** *instanceName*，並選取**停止**。
7. 展開**企業應用程式**。
8. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce 企業應用程式 - 示範**應用程式，並選取**停止**。
9. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce 企業應用程式 - 示範**應用程式，並選取**移除**。
10. 當出現提示指出是否應匯出應用程式時，請選取**否**。

使用 XMLConfig 匯入新企業應用程式

如果要使用 XMLConfig 指令行公用程式來匯入新企業應用程式，請執行下列步驟：

1. 導覽至下列目錄：

drive:\WebSphere\CommerceServer\working

2. 在指令提示下，輸入下列指令以便將企業應用程式匯入到 WebSphere Application Server 中：

   ```
   xmlConfig -import OutputFile.xml -adminNodeName was_hostname
   ```

   其中 *was_hostname* 爲內含現行應用程式的 WebSphere Application Server 節點名稱。

註： > 400 如果您是部署到執行 iSeries 的 WebSphere Commerce 案例中，在您匯入應用程式後，您得額外執行一個步驟，以修改目錄許可權。有關如何修改這些許可權的詳述，請參閱第 346 頁的『匯入企業應用程式』。

啓動新企業應用程式

在您使用 XMLConfig 指令行公用程式匯入新企業應用程式後，即可使用 WebSphere Application Server 管理主控台來執行重新整理，然後啓動新應用程式。

如果要重新整理主控台並啓動新應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**
3. 標明出**節點**。
4. 按一下**重新整理所選子樹**圖示。
5. 執行下列以啓動 WebSphere Commerce 應用程式：
   - 展開**應用程式伺服器**。
   - 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server -** *instanceName*，並選取**啓動**。

測試 MyNewControllerCmd

下一步驟是測試您商店（執行於 WebSphere Application Server 環境中）中的新邏輯。如果要測試 MyNewControllerCmd，請執行下列步驟：

1. 在瀏覽器中輸入下列 URL，以測試此項整合：

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

   其中 *store_Id* 爲您商店的識別碼，*catalog_Id* 爲您商店型錄的識別碼。

2. 按一下**登錄**鏈結。
   會出現「登錄或登入」頁面。
3. 在**電子郵件位址**欄位中輸入 wcsadmin。
4. 在**密碼**欄位中，輸入此網站所用 wcsadmin ID 的密碼，然後按一下**登入**。

5. 在使用者登入後，按一下側邊導覽窗格中的 **MyNewControllerCmd** 鏈結。會出現範例 JSP。

若未出現更新過的 sidebar.jsp 檔，請執行下列步驟以清除您的快取：

1. 將快取檔案從下列目錄中刪除：
   drive:\WebSphere\CommerceServer\instances*instanceName*\cache
2. 將任何相關的快取檔案從下列目錄中刪除：

   ```
   drive:\WebSphere\AppServer\temp\hostName\
   WebSphere_Commerce_Server_instanceName\
   WebSphere_Commerce_Enterprise_Application_-_instanceName\
      wcstores.war\storeName
   ```

   ```
   drive:\WebSphere\AppServer\temp\hostName\
   WebSphere_Commerce_Server_instanceName\
   WebSphere_Commerce_Enterprise_Application_-_instanceName\
   wcstores.war
   ```
3. 清除您瀏覽器的快取。

如果 JSP 頁面的編譯時間過長，則該頁面可能不會顯示。在此情況下，請重新載入頁面。

第 10 章 修改與延伸現有的商業邏輯

以下的指導教學說明如何延伸或修改現有的 WebSphere Commerce 商業邏輯。

延伸現有的控制程式指令

在本節中，您將延伸現有的 OrderProcess 控制程式指令，以便讓交易中累計的總紅利積點顯示在訂單確認頁面上。

註: 此指導教學的目的是顯示修改現有控制程式指令的程序。其目的不在於提供最好的方式讓您來修改購物流程中的訂單程序步驟。事實上，WebSphere Commerce 中所提供的 `ExtOrderProcess` 作業指令可用來修改購物流程中的訂單程序步驟。

開始進行此指導教學前

您應該已完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』中的步驟。

下列摘要說明延伸 OrderProcess 指令時會涉及的步驟：

1. 建立用來儲存自訂程式碼的新套件。再次提醒您，所有自訂程式碼（關於指令與資料 Bean）必須儲存在有別於 WebSphere Commerce 程式碼的專案與套件中。
2. 建立從現有 `OrderProcessCmdImpl` 指令延伸而來的新 `OrderProcessCmdBonusImpl` 類別。
3. 新增欄位與方法到 `OrderProcessCmdBonusImpl` 類別中。
4. 修改指令登錄以使用 `OrderProcessCmdBonusImpl` 類別。
5. 修改 confirmation.jsp 範本，以顯示新商業邏輯。
6. 在 WebSphere Test Environment 中測試新商業邏輯。
7. （選用）將新商業邏輯部署到遠端 WebSphere Commerce Server 上的商店中。

為 *OrderProcessCmdBonusImpl* 建立新套件

如果要建立用以儲存 `OrderProcessCmdBonusImpl` 指令的新套件，請執行下列步驟：

1. 在「VisualAge for Java 工作台」視窗中，確定您已選取「專案」標籤。

2. 以滑鼠右鍵按一下 **_WCSamples** 專案，並選取**新增 > 套件**。
   會開啓「新增套件」引導精靈。
3. 確定您已啓用**建立指名的新套件**圓鈕，並輸入
   com.ibm.commerce.sample.order。
4. 按一下**完成**。

建立 *OrderProcessCmdBonusImpl* 類別

如果要建立新 OrderProcessCmdBonusImpl 類別，請執行下列步驟：

1. 以滑鼠右鍵按一下 com.ibm.commerce.sample.order 套件，並選取**新增 > 類別**。
   會開啓「建立類別」引導精靈。
2. 確定您已選取**建立新類別**圓鈕。
3. 在**類別名稱**欄位中，輸入 OrderProcessCmdBonusImpl。
4. 如果要指定超類別，請按一下**瀏覽**，然後在**型樣**欄位中，輸入
   com.ibm.commerce.order.commands.OrderProcessCmdImpl，再按一下**確定**。
5. 按**下一步**。
6. 如果要指定應匯入的套件，請按一下**新增套件**。在**型樣**欄位中，輸入下列套件：
   - com.ibm.commerce.datatype，然後按一下**新增**
   - com.ibm.commerce.exception，然後按一下**新增**
   - com.ibm.commerce.order.commands，然後按一下**新增**
   - com.ibm.commerce.order.objects，然後按一下**新增**
   - com.ibm.commerce.ras，然後按一下**新增**
   - com.ibm.commerce.server，然後按一下**新增**
   - com.ibm.commerce.sample.objects，然後按一下**新增**
   - javax.ejb，然後按一下**新增**
   - java.io，然後按一下**新增**
   - java.math，然後按一下**新增**，再按一下**關閉**。
7. 如果要指定類別應施行的介面，請按一下**新增**。在「型樣」欄位中，輸入下列介面：
   - OrderProcessCmd，並按一下**新增**，然後按一下**關閉**。
8. 按一下**完成**。

新增欄位與方法到 *OrderProcessCmdBonusImpl* 中

您必須新增兩個欄位與 performExecute 方法到類別中。

如果要新增 theOrder 欄位到 OrderProcessCmdBonusImpl 類別中，請執行下列步驟：

1. 以滑鼠右鍵按一下 OrderProcessCmdBonusImpl 類別，並選取**新增 > 欄位**。會開啓「建立欄位」引導精靈。
2. 使用下列屬性新增欄位到類別中。有關如何建立新欄位的詳細步驟，請參閱第 195 頁的『建立新欄位』。

屬性名稱	**值**
欄位名稱	theOrder
欄位類型	OrderAccessBean
初始值	保持空白。
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

然後按一下**完成**。

如果要新增 bonusPercentAmount 欄位到 OrderProcessCmdBonusImpl 類別中，請執行下列步驟：

1. 以滑鼠右鍵按一下 OrderProcessCmdBonusImpl 類別，並再次選取**新增 > 欄位**。
2. 使用下列屬性新增欄位到類別中。有關如何建立新欄位的詳細步驟，請參閱第 195 頁的『建立新欄位』。

屬性名稱	**值**
欄位名稱	bonusPercentAmount
欄位類型	double
初始值	1000
存取修改元	private
其它修改元	維持全不勾選。
使用 getter 與 setter 方法存取	勾選
Getter	public
Setter	public

然後按一下**完成**。

如果要新增 performExecute 方法到 OrderProcessCmdBonusImpl 類別中，請執行下列步驟：

1. 以滑鼠右鍵按一下 **OrderProcessCmdBonusImpl** 類別，並選取**新增 > 方法**。會開啓「建立方法」引導精靈。
2. 確定您已選取**建立新方法**圓鈕，然後按**下一步**。
3. 在**方法名稱**欄位中，輸入 performExecute。
4. 從**傳回類型**下拉清單中選取 void。按**下一步**。
5. 如果要指定方法可擲出的異常狀況，請按一下**新增**。在**型樣**欄位中，輸入 ECException，按一下**新增**，然後按一下**關閉**。
6. 按一下**完成**。
   會產生方法，會顯示方法的原始碼。
7. 您必須修改原始碼。請在 performExecute 方法的原始碼中找出下行：

```
public void performExecute() throws
   com.ibm.commerce.exception.ECException {
```

在上行之後輸入下列程式碼：

您可以從「程式設計手冊」的 PDF 版本中剪貼這個程式碼。建議您一開始先複製 VisualAge for Java Scrapbook 視窗中的程式碼（請參閱 VisualAge for Java 線上說明以取得其它明細），然後查看程式碼，確定在剪貼作業中沒有任何字元遺失或修改。在驗證程式碼之後，將它複製到目標位置上。請注意，將文字複製到另一編輯器中，可能會修改部份字元。

```
   final String methodName = "performExecute";
   ECTrace.entry(ECTraceIdentifiers.COMPONENT_ORDER,
   this.getClass().toString(), methodName);

// 按正常般執行所有訂單處理程序
   super.performExecute();

// *** 更新紅利積點資訊 ***

// 提取訂單資訊
theOrder = new OrderAccessBean();
theOrder.setInitKey_orderId(getOrderRn().toString());

int bonusPt; // 此訂單的紅利積點
int bonusTotal; // 總紅利積點
   BigDecimal subtotal;  // 小計
   BigDecimal bonusdeter;  // 紅利行列值
   BigDecimal ans;

// 決定紅利積點 = 小計 * 紅利行列值
   try {
```

```
        subtotal = theOrder.getTotalProductPriceInEJBType();
        bonusdeter = new BigDecimal(bonusPercentAmount);
        ans = subtotal.multiply(bonusdeter);
        bonusPt = Math.round(ans.floatValue());

         System.out.println("subtotal is: " + subtotal +
                " bonus deter is: " + bonusdeter + " ans is: " + ans);
         System.out.println("Bonus Percent amount = " +
            bonusPercentAmount);
         System.out.println("Bonus calculated is: "+ bonusPt);
        }


// 各種異常狀況
catch (Exception ex) {
        throw new ECSystemException(ECMessage._ERR_GENERIC,
          this.getClass().toString(),methodName,
          ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
       }

   // *** 使用上例中所建的 Bean 來更新
   //     BONUS 表格中的紅利積點 ***
BonusAccessBean bonusBean = new BonusAccessBean();
   bonusBean.setInit_argMemberId(
      getCommandContext().getUserId().toString());
   try {
// 新紅利值 = 此訂單的紅利積點 + 舊有的紅利積點
bonusTotal = bonusPt + Integer.parseInt(bonusBean.getBonusPoint());
bonusBean.setBonusPoint(String.valueOf(bonusTotal));
        bonusBean.commitCopyHelper();

        System.out.println("In try, BonusTotal calculated is: "+
                 bonusTotal);

        }
// 各種異常狀況
catch (FinderException e) // 使用者尚未設定點數
   {
      // 在 bonus 表格中建立一列
bonusTotal = bonusPt;
   try {
          BonusAccessBean bonusBeanNew = new
             BonusAccessBean(getCommandContext().getUserId(),
                new Integer(bonusTotal));

             System.out.println("In catch, BonusTotal calculated is: "+
                bonusTotal);

   }
catch (Exception ex) {
        throw new ECSystemException(ECMessage._ERR_GENERIC,
          this.getClass().toString(), methodName,
          ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
       }
}
```

```
catch (Exception ex) {
       throw new ECSystemException(ECMessage._ERR_GENERIC,
         this.getClass().toString(), methodName,
         ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
   }
   // *** 設定檢視畫面明細 ***

   // 提取 setResponse 內容，
  //  並新增 JSP 頁面所需的紅利參數
TypedProperty resp = getResponseProperties();
resp.put("bonus", new Integer(bonusPt).toString());
setResponseProperties(resp);
   ECTrace.exit(ECTraceIdentifiers.COMPONENT_ORDER,
   this.getClass().toString(), methodName);
```

8. 儲存您的工作。

修改指令登錄以使用 *OrderProcessCmdBonusImpl*

在本例中，每當需要進行訂單處理程序時，您想在訂單處理程序中使用新施行類別。爲了達到此目的，您必須更新指令登錄，讓原始 OrderProcess 介面連結新 OrderProcessCmdBonusImpl 施行類別。

DB2 如果您使用 DB2 資料庫，請執行下列步驟以更新指令登錄：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
2. 選取 Script 標籤，並在 Script 視窗中輸入下列資訊，以便在 CMDREG 表格中建立必要項目：

```
connect to your_database_name;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

其中 *your_database_name* 爲資料庫名稱；請按一下「執行」圖示。此指令是供所有商家使用（亦即，將 STOREENT_ID 之值設爲 0）。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以更新指令登錄：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便在 URLREG 表格中建立必要項目：

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

按 Enter 鍵以執行 SQL 陳述式。
此指令是供所有商家使用（亦即，將 STOREENT_ID 之值設爲 0）。

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。

修改 *confirmation.jsp* 範本

您必須修改 confirmation.jsp 範本，以顯示您已新增到訂單程序商業程序中的新商業邏輯。如果要修改顯示範本，請執行下列步驟：

1. 導覽至下列目錄：
   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*。
2. 複製一份 confirmation.jsp，並命名爲 confirmation.jsp.bak
3. 以文字編輯器開啓 confirmation.jsp。
4. 在現有的 import 陳述式後面，加上以下的指令：

```
<%@ page import="com.ibm.commerce.datatype.*"   %>
```

5. 在 JSP 範本的下行：

```
String orderRn = jhelper.getParameter("orderId");
```

後緊接著新增下列：

```
String bonus = ((TypedProperty)request.getAttribute(
   ECConstants.EC_REQUESTPROPERTIES)).getString("bonus");
```

6. 在 JSP 範本中找出下列區段：

```
<tr>
<td align="left" valign="middle">
<font class="product"><%=infashiontext.getString("GRAND_TOTAL")%>
   </font></td>
<td align="right" valign="middle">
<font class="strongprice"><%=orderBean.getGrandTotal()  %></font></td>
```

然後新增下列：

```
</tr>
<tr>
<td align="left" valign="middle">
<font class="text">Bonus Points</font></td>
<td align="right" valign="middle">
<font class="strongtext"><%=bonus  %></font></td>
```

7. 儲存您的工作。

在 WebSphere Test Environment 中測試 *OrderProcessCmdBonusImpl*

此時，您可以使用 WebSphere Test Environment 來測試您的新商業邏輯。如果要測試 OrderProcessCmdBonusImpl 指令，請執行下列步驟：

1. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所述，啓動 WebSphere Test Environment。
2. 開啓瀏覽器並輸入您商店的 URL。舉例來說，輸入下列 URL：

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=10001&catalogId=10001&langId=-1
   ```

3. 選取並購買產品。
4. 在您購買產品後，訂單確認會顯示訂單所獲得的紅利積點數。

（選用）將自訂商業邏輯部署到遠端 WebSphere Commerce Server 中

當您在 WebSphere Test Environment 中測試完您的商業邏輯，並滿意您的程式碼後，您可將程式碼部署到遠端 WebSphere Commerce Server 上的商店中。在本指導教學中，自訂作業包括：

- 新 OrderProcessCmdBonusImpl 類別。
- 更新的 confirmation.jsp 範本檔。
- 更新的指令登錄。

因此，程式碼部署包含下列步驟：

1. 使用 VisualAge for Java 中的工具，爲指令邏輯建立一個 JAR 檔。
2. 將 JAR 檔與 JSP 範本複製到目標 WebSphere Commerce Server 上的適當目錄中。
3. 更新目標 WebSphere Commerce Server 上的指令登錄。

測試付款方法的相關注意事項

在預設的情況下，在 WebSphere Test Environment 中執行的範例商店使用的是測試付款方法。使用此測試付款方法，可讓您在 WebSphere Test Environment 中完成購物流程，而不需呼叫 Payment Manager。此測試付款方法只能讓您完成一項交易，而不能讓此付款方法所提交的訂單可供日後處理用。因此，測試付款方法應只在 WebSphere Test Environment 中使用。

請確定您可在將部署此自訂程式碼的商店中完成交易。付款處理程序可使用本端或遠端 Payment Manager 來完成。

有關測試付款方法的進一步資訊，請參閱第 179 頁的『測試付款方法』。

為指令邏輯建立 JAR 檔

您必須將指令邏輯組裝到 JAR 檔中，以便將之部署到目標 WebSphere Commerce Server 中。由於 OrderProcessCmdBonusImpl 儲存在 _WCSamples 專案中，您將針對該專案建立一個 JAR 檔。

如果要建立 JAR 檔，請執行下列步驟：

1. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所述，停止 WebSphere Test Environment。
2. 以滑鼠右鍵按一下 **_WCSamples** 專案，並選取**匯出**。
   會開啓「匯出」引導精靈。
3. 選取 **Jar 檔**，並按**下一步**。
4. 在「JAR 檔」欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev\mytemp_b\wcssamplesb_1.jar
   其中 *drive* 爲安裝 Commerce Studio 的磁碟機。
5. 按如下選取屬性：

屬性	值
類別	勾選
java	不勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
6. 按一下**完成**。

由於所建的 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中，導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp_b
2. 輸入 mkdir temp1。
3. 輸入 cd temp1。
4. 按如下設定路徑：
   set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio4\bin;
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 jar xvf ../wcssamplesb_1.jar

6. 輸入 jar cvf ../wcssamplesb.jar *

將資產儲存到目標 WebSphere Commerce Server

指令邏輯的 JAR 檔以及已修改的 confirmation.jsp 範本必須置於目標 WebSphere Commerce Server 的適當目錄中。

如果要將 JAR 檔儲存在遠端 WebSphere Commerce Server 的適當目錄中，請執行下列步驟：

1. 在開發機器上開啓指令視窗，並導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp_b
   然後找出 wcssamplesb.jar 檔。
2. 將此檔案複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instanceName.ear\wcstores.war\WEB-INF\lib
   ```

如果要將 confirmation.jsp 範本儲存在目標 WebSphere Commerce Server 的適當目錄中，請執行下列步驟：

1. 在開發機器中導覽至下列目錄：
   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*
2. 將 confirmation.jsp 範本複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instanceName.ear\wcstores.war\storeDir
   ```

更新指令登錄

如果您要將 OrderProcessCmdBonusImpl 指令部署到採用有別於 WebSphere Test Environment 所用資料庫的目標 WebSphere Commerce Server 中，您必須更新目標資料庫，以反映您對指令登錄所做的變更。

DB2 如果您所用的是 DB2 資料庫，請執行下列步驟，以更新目標 WebSphere Commerce Server 的資料庫：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**）。
2. 選取 Script 標籤，並在 Script 視窗中輸入下列資訊，以便在 CMDREG 表格中建立必要項目：

   ```
   connect to your_target_database_name;
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0
   ```

其中 *your_target_database_name* 爲您的資料庫名稱，然後按一下「執行」圖示。

Oracle 如果您使用 Oracle 資料庫，請執行下列步驟以更新指令登錄：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便在 CMDREG 表格中建立必要項目：

   ```
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0;
   ```

   按一下 Enter 鍵以執行 SQL 陳述式。
   此指令是供所有商家使用（亦即，將 STOREENT_ID 之值設爲 0）。
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 WebSphere Application Server 中重新啓動企業應用程式

在您藉由將檔案資產置於適當目錄中並更新指令登錄，以新增指令邏輯到企業應用程式後，您必須停止再重新啓動企業應用程式，以便讓變更生效。

如果要停止再重新啓動企業應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**。
3. 展開**節點**。
4. 展開 ***nodeName*** （其中 ***nodeName*** 爲節點名稱。）
5. 展開**應用程式伺服器**。
6. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server - 示範**應用程式，並選取**停止**。
7. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server - 示範**應用程式，並選取**啓動**。

測試您在「時尚館」（執行於 WebSphere Application Server 中）內的新邏輯

此時您可以驗證您在「時尚館」商店（執行於 WebSphere Application Server 中）內的新商業邏輯。

如果要執行此項最終的驗證，請執行下列步驟：

1. 在啓動 WebSphere Commerce Server 案例後，開啓瀏覽器並輸入您商店首頁的 URL。舉例來說，輸入下列 URL：

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

   其中 *store_Id* 爲您商店的識別碼，*catalog_Id* 爲您商店型錄的識別碼。
2. 選取並購買產品。
3. 在您購買產品後，訂單確認會顯示訂單所獲得的紅利積點數。

修改現有實體 Bean，並延伸現有的作業指令

註: 此指導教學的目的是顯示修改現有實體 Bean 與延伸現有作業指令時所用的程序。其目的不在於提供最好的方式讓您來修改產品計價。有關折扣的進一步資訊，請參閱 *WebSphere Commerce Calculation Framework Guide*。

在『第 9 章, 『指導教學：建立新商業邏輯』』中，會建立一整組的新商業邏輯。其中包括：建立新控制程式指令、新 JSP 範本、新資料庫表格、存取表格所用的新 Enterprise Bean、對應的存取 Bean 以及資料存取 Bean。所有這些邏輯合在一起，就可以建立一個簡化的紅利積點應用程式，讓您利用藉由新控制程式指令啓動的 JSP 範本，來更新使用者紅利積點的結餘。

下列圖解說明了在『第 9 章, 『指導教學：建立新商業邏輯』』中使用 BONUS 表格的方法：

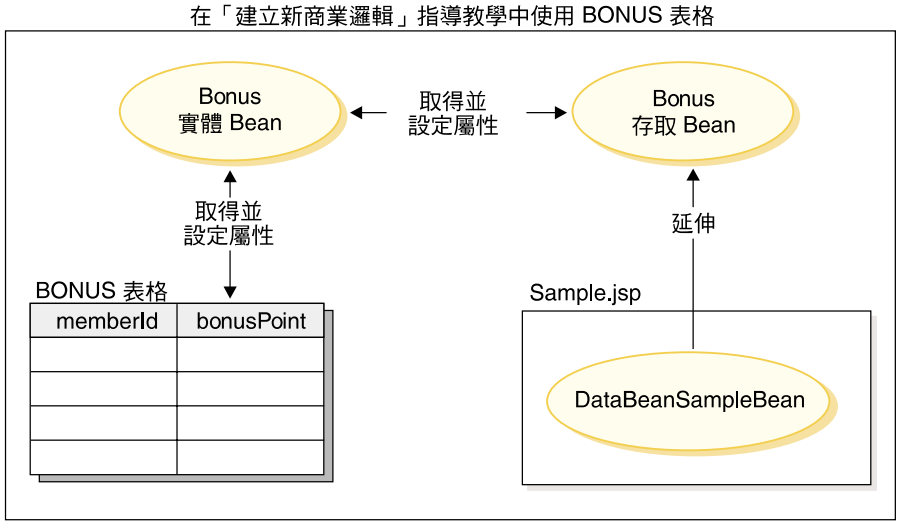


圖 41.

在下個指導教學中，將以不同方式來使用 BONUS 表格。具體而言，所用的自訂 User 實體 Bean 方式，可讓應用程式認爲 BONUS 表格的 BONUSPOINT 直欄實際上是 USERS 表格中的某個直欄。當您在 USERS 表格中建立新記錄時，就會自動將對應的記錄插入 BONUS 表格中。

爲了建立這種*表格結合*，必須將新的 CMP 欄位新增至 User 實體 Bean 中。這個 CMP 欄位會使用 VisualAge for Java「映射瀏覽器」中的「次要表格映射」特性來映射到 BONUS 表格中的 BONUSPOINT 直欄。

爲了將修改過的 User 實體 Bean 整合到購物流程中，會爲產品建立一個新價格。這個新價格會考量購物者目前的紅利積點結餘。您可以藉由延伸 GetProductContractUnitPriceCmd 作業指令來建立新價格。在您延伸此指令時，您將建立從 GetProductContractUnitPriceCmd 介面延伸而來的新介面。新介面將新增其它屬性（關於紅利價格）。您也將建立從 GetContractUnitPriceCmdImpl 施行類別延伸而來的新施行類別。此新施行類別稱爲 “GetNewContractUnitPriceCmdImpl”。新施行類別會呼叫其超類別的 performExecute 方法，並新增商業邏輯以判斷新紅利價格。

下圖顯示在下個指導教學中如何使用 BONUS 表格。

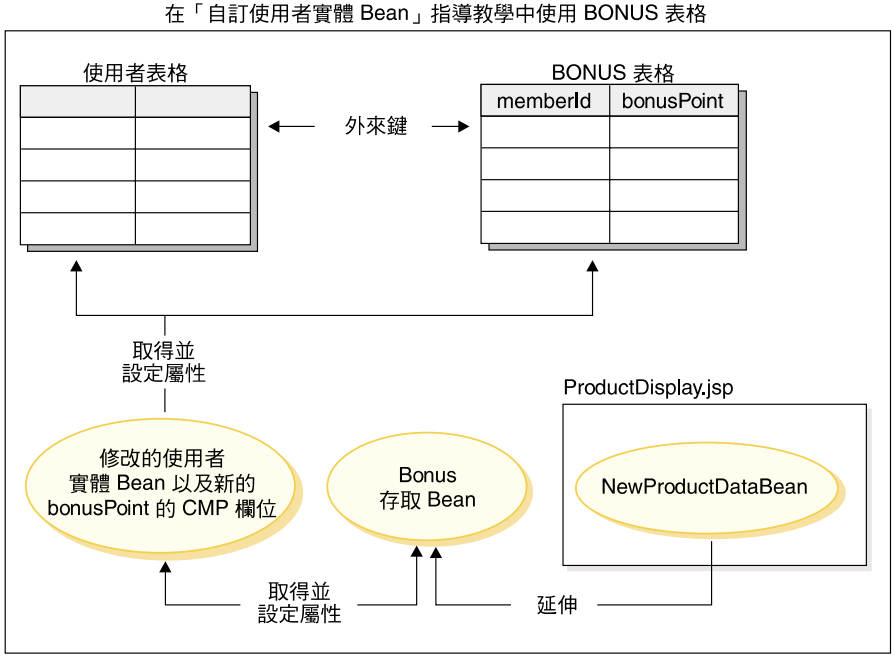


圖 42.

本指導教學包括下列步驟：

1. 將新的 CMP 欄位加入 User 實體 Bean。
2. 建立 BONUS 表格並移入資料。
3. 更新 WCSUser 資料庫綱目以及表格映射，以併入新的 BONUS 表格。
4. 建立 BONUS 和 USER 表格之間的關係。
5. 建立 BONUS 表格映射。
6. 爲 User 實體 Bean 產生部署程式碼和存取 Bean。
7. 使用測試從屬站，以初步測試已更新的實體 Bean。
8. 建立新作業指令介面與施行類別。新作業指令是從 `GetBaseUnitProductPriceCmd` 延伸而來。此指令中最重要的新特性是施行類別之 performExecute() 方法中的邏輯。此方法會計算產品的新*紅利價格*。會建立此紅利價格，因此價格會因購物者的紅利積點結餘而下降，直到抵達所算出價格的 20% 最大折扣爲止。下表所示的範例爲此價格所用的折扣公式：

算出的價格	紅利積點結餘	最大折扣（算出價格的 20%）	新紅利價格
$1000	100	$200	$900 由於紅利積點結餘低於最大折扣，因此標價可以減去紅利積點。
$1000	300	$200	$800 由於紅利積點結餘超過最大折扣，標價會減價 $200 的最大折扣。

9. 建立從 ProductDataBean 延伸而來的 NewProductDataBean。新增新方法到此 Bean 中，以方便在產品顯示頁面中使用到新紅利價格。
10. 更新「時尚館」商店的產品顯示 JSP 範本，以顯示紅利價格。
11. 測試「時尚館」商店（執行於 WebSphere Test Environment 中）內的商業邏輯。
12. （選用）將更新過的商業邏輯部署到遠端 WebSphere Commerce Server 中，並在 WebSphere Test Environment 外進行測試。

開始進行此指導教學前

如果您尚未完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，在您開始進行本指導教學前，必須先執行第 184 頁的『準備範例專案』中所述的步驟。

新增新 bonusPoint 欄位到 User 實體 Bean 中

在本節中，您可以使用 VisualAge for Java 的 EJB 工具，將新 CMP 欄位新增到實體 Bean 中。新欄位名稱爲 *bonusPoint*，其最後映射至 BONUS 表格中的 BONUSPOINT 直欄。

如果要新增新 CMP 欄位到 User 實體 Bean 中，請執行下列步驟：

1. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所述，停止 Servlet 引擎、EJB 伺服器以及永久名稱伺服器（如果這些正在執行的話）。
2. 在「工作台」中按一下 **EJB** 標籤。

3. 展開 **WCSUser** EJB 群組。
4. 以滑鼠右鍵按一下 **User** Bean，並選取**新增 > CMP 欄位**。
   會開啓「建立 CMP 欄位」引導精靈。
5. 以下列內容來建立新 CMP 欄位：

內容	值
欄位名稱	bonusPoint
欄位類型	int
初始值	0
使用 getter 與 setter 方法存取	enable
將 getter 與 setter 方法引介到遠端介面中	enable
Getter	public
Setter	public

然後按一下**完成**。

BonusPoint 欄位會顯示在「內容」窗格中。可能會出現一些警告，但在您重新產生實體 Bean 的程式碼時這些會修正。

建立 BONUS 表格並移入資料

在本指導教學中，會使用一個資料庫表格以記錄使用者的紅利積點。

如果您完成了第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，就表示您已經瞭解這個表格。在此情況下，您必須更新表格，以便在 USERS 表格中針對每一位使用者各新增一列。如果您尚未完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，您必須建立表格並移入資料。我們會提供這些實務的相關指示。

▶ DB2 如果您所用的是 DB2 資料庫，且您需要*更新* BONUS 表格，請執行下列步驟：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**），然後按一下 **Script** 標籤。
2. 在**指令**欄位中，輸入

   ```
   connect to your_database_name
   ```

   其中 *your_database_name* 是您的資料庫名稱，然後按一下「執行」圖示。
3. 在**指令**欄位中，輸入下列指令，然後按一下「執行」圖示：

```
INSERT INTO BONUS
(SELECT USERS_ID, 0
FROM USERS
WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
```

此時 BONUS 表格已更新。

DB2 如果您所用的是 DB2 資料庫，且您需要*建立與移入資料*到 BONUS 表格中，請執行下列步驟：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**），然後按一下 **Script** 標籤。
2. 在**指令**欄位中，輸入

   ```
   connect to your_database_name
   ```

   其中 *your_database_name* 是您的資料庫名稱，然後按一下「執行」圖示。
3. 在**指令**欄位中，輸入下列指令，然後按一下「執行」圖示：

   ```
   CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
   BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
   constraint f_memberid foreign key (MEMBERID)
   references users (users_id) on delete cascade)
   ```

   此時 BONUS 表格已建立。
4. 如果要在表格中移入資料，請在**指令**欄位中輸入下列資訊，並按一下「執行」圖示：

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. 關閉 DB2 指令中心。

Oracle 如果您所用的是 Oracle 資料庫，而您需要*更新* BONUS 表格，請執行下列步驟：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便更新 BONUS 表格：

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

按一下 Enter 鍵，以執行 SQL 陳述式。
此時 BONUS 表格已更新。

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。

Oracle 如果您所用的是 Oracle 資料庫，而您需要*建立並移入資料*到 BONUS 表格中，請執行下列步驟：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便建立 BONUS 表格：

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
constraint f_memberid foreign key (MEMBERID)
references users (users_id) on delete cascade);
```

按一下 Enter 鍵，以執行 SQL 陳述式。
此時 BONUS 表格已建立。

6. 輸入下列指令，以便將資料移入到 BONUS 表格中：

```
insert into BONUS (select USERS_ID, 0 from USERS);
```

7. 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。

更新綱目與表格映射

在下列各節中，您將以新 BONUS 表格更新 WCSUser 綱目，建立新表格的外來鍵關係，以及在 User 實體 Bean 的欄位與 BONUS 表格的直欄間建立表格映射。

建立 BONUS 表格綱目

如果要建立表格綱目，請執行下列步驟：

1. 以滑鼠右鍵按一下 **User** 實體 Bean，然後選取**開啓到 > 資料庫綱目**。
會開啓「綱目瀏覽器」視窗。
2. 從**綱目**清單中，選取 **WCS 使用者**。

3. 從**表格**功能表中，選取**新表格**。
   會開啓「表格編輯器」。
4. 在**名稱**欄位中，輸入 BONUS。
5. 將**限定元**和**實體名稱**兩個欄位留白。
6. 在「表格直欄」區段中，按一下**新增**。會開啓「直欄編輯器」。依下列所示，建立一個直欄：

屬性	**值**
名稱	memberId
實體名稱	讓此欄位留白。
類型	BIGINT
類型明細	VapConverter
容許空值	不勾選

   然後按一下**確定**。
7. 在「表格直欄」清單中選取 **memberId**，然後按一下 **>>** 來使用這個直欄作爲主要鍵。
8. 依下列所示，建立另一個直欄（再按一下**新增**）：

屬性	**值**
名稱	bonusPoint
實體名稱	讓此欄位留白。
類型	INTEGER
類型明細	VapConverter
容許空值	勾選

   然後按一下**確定**。
9. 在「表格編輯器」中按一下**確定**。
   不久，**BONUS** 即會列在「綱目瀏覽器」視窗的**表格**清單中。
10. 如果要驗證，請按一下**表格**清單中的 **BONUS**，並確定**直欄**清單中有出現這兩個直欄。

建立外來鍵關係

在本節中，您可以建立 BONUS 和 USERS 表格之間的外來鍵關係。

如果要建立外來鍵關係，請執行下列步驟：

1. 在「綱目瀏覽器」視窗中，按一下 **WCS 使用者**綱目。
   這時會顯示此綱目的表格和外來鍵關係。

2. 從**外來鍵**功能表中，選取**新增外來鍵關係**。
   會開啓「外來鍵關係編輯器」。
3. 在**名稱**欄位中，輸入 F_User_Bonus。
4. 確定**限制元已經在資料庫中**勾選框已經勾選。
5. 從**主要鍵表格**下拉清單中，選取 USERS
6. 從**外來鍵表格**下拉清單中，選取 BONUS
7. 按一下**外來鍵**標頭底下的空白欄位，以顯示下拉清單。從此清單中，選取 **memberId**，然後按一下**確定**。
   F_User_Bonus 會出現在外來鍵關係的清單中。
8. 從**綱目**功能表中選取**儲存綱目**，然後按一下**完成**。
9. 關閉「綱目瀏覽器」。

建立 BONUS 表格映射

在本節中，您可以在 BONUS 表格中的 BONUSPOINT 直欄與 User 實體 Bean 中的 bonusPoint 欄位之間建立映射。

如果要建立 BONUS 表格映射，請執行下列步驟：

1. 確定您已經開啓「工作台」，並已選取 EJB 標籤。
2. 從 **EJB** 功能表中，選取**開啓到 > 綱目映射**。
   會開啓「映射瀏覽器」。
3. 從「資料儲存庫映射」清單中，選取 **WCS 使用者**。
4. 在**持續性類別**清單中，執行下列步驟：
   a. 按兩下**成員**。
   b. 選取**使用者**（請注意，只有您在步驟 4a 中展開「成員」後，**使用者**才會出現在**成員**下。）
5. 從**表格映射**功能表中，選取**新增表格映射 > 新增次要表格映射**。
   會開啓「次要表格映射」視窗。
6. 從**表格**下拉清單中，選取 **BONUS**。
7. 從「外來鍵關係」下拉清單中，選取 **F_User_Bonus**，然後按一下**確定**。
   之後，BONUS（次要）映射即會出現在「表格映射」清單中。
8. 標示並以滑鼠右鍵按一下 **BONUS**（**次要**）表格映射，然後選取**編輯內容映射**。
   這時會開啓「內容映射編輯器」。
9. 向下捲動至顯示 bonusPoint 類別屬性的這一列。選取 **bonusPoint**。
10. 按一下**映射類型**直欄中的對應項目，然後從下拉清單中選取**簡式**。
11. 按一下**表格直欄**直欄中的對應項目，然後從下拉清單中選取 **bonusPoint**。

12. 不要改變其他所有欄位，然後按一下**確定**。
    會產生新的內容映射，不久

    ```
    (a) bonusPoint (bonusPoint)
    ```

    就會出現在「映射瀏覽器」的**內容映射**直欄中。
13. 以滑鼠右鍵按一下 **WCS 使用者** datastore 映射，選取**儲存 datastore 映射**，然後按一下**完成**。
14. 關閉「映射瀏覽器」。

此時，使用者 Bean 包含錯誤，表示有些抽象類別並未施行。這些錯誤會在產生部署的程式碼時修正。

產生部署的程式碼與存取 Bean

由於您已經修改了 User 實體 Bean 的程式碼，您必須重新產生其部署程式碼以及其存取 Bean。VisualAge for Java 中的工具可以簡化此項程式碼產生步驟。

如果要執行這個步驟，請執行下列步驟：

1. 確定您已開啓「工作台」，並選取 EJB 標籤。
2. 展開 **WCSUser** EJB 群組。
3. 以滑鼠右鍵按一下 **User** Bean，並選取**產生部署程式碼**。
   如果出現一個訊息視窗，詢問您是否要建立一版套件時，請按一下**是**。
   產生程式碼將需要幾分鐘。
4. 一旦產生部署程式碼後，請再次以滑鼠右鍵按一下 **User** Bean，並選取**新增 > 存取 Bean**。會開啓「建立存取 Bean」引導精靈。
5. 接受「選取存取 Bean 內容」頁面中的預設值，然後按**下一步**。
6. 接受「定義零引數建構子」頁面中的預設值，然後按**下一步**。
7. 在「選取及自訂 Copy Helper 的 Bean 內容」頁面中，下捲至「Enterprise Bean」直欄中的 **bonusPoint**。勾選 Bean 的 **Copy Helper**，並將轉換器設定成 `com.ibm.commerce.base.objects.WCSStringConverter`。
8. 按一下**完成**。
9. 當出現「順利產生程式碼」訊息時，請按一下**確定**。

使用測試從屬站來測試所做的修改

測試從屬站可用來測試新自訂的 User 實體 Bean。如果要啓動測試從屬站，並測試實體 Bean，請執行下列步驟：

1. 依照第 313 頁的『啓動與停止永久名稱伺服器』中所描述的，啓動永久名稱伺服器。

2. 根據第 314 頁的『啓動與停止 EJB 伺服器』的說明，啓動上面有 WCSUser EJB 群組的 EJB 伺服器。
3. 在 Enterprise Beans 窗格中，展開 **WCSUser** EJB 群組。以滑鼠右鍵按一下 **User** 實體 Bean，並選取**執行測試從屬站**。
4. 在「查看 EJB」視窗中，按一下**查看**。
5. 從方法清單中，選取 **findByPrimaryKey(MemberKey)** 。
6. 在「明細」窗格中，按一下 **<空值>**，然後在引數方框中輸入 `-1000` ，然後按一下「EJB 測試從屬站」視窗中的「呼叫」圖示。

   請注意，在此處輸入的值必須和 USERS 表格的 USERS_ID 直欄中的值相符。

   完成執行這個方法以後，就會在「方法」窗格的樹狀結構檢視畫面中，顯示 ID 爲 -1000 的使用者資訊。在這個檢視畫面中，會有一個節點名稱爲 *methods*。
7. 展開 **Methods** 節點。
8. 按一下 **getBonusPoint()**，然後按一下「呼叫」圖示。
   這個方法會擷取客戶的紅利積點結餘。會傳回目前的紅利積點結餘。
9. 按一下「明細」窗格中的 **setBonusPoint(int)**，輸入 `100`，然後按一下「呼叫」圖示。
10. 按一下 **getBonusPoint()**，然後按一下「呼叫」圖示。
    這時會傳回 100 的值。這表示資料庫已經順利被使用者 Enterprise Bean 所更新。
11. 關閉「User」和「EJB 測試從屬站」兩個視窗。

建立 *GetNewProductContractUnitPriceCmd* 介面

在這個自訂作業中，您會根據客戶的紅利積點結餘，建立一個新的商業邏輯來計算折扣後的價格。這個計算是由新的作業指令來執行。在本節中，您會爲這個新的作業指令建立一個介面。

如果要爲新的作業指令建立介面，請執行下列步驟：

1. 在「工作台」中，先選取「專案」標籤，然後展開 **_WCSamples** 專案。
2. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.commands** 套件，並選取**新增 > 介面**。
3. 請確定**建立新介面**已選取，然後在**介面名稱**欄位中，輸入 `GetNewProductContractUnitPriceCmd`。
4. 按一下**新增**來選取應該展開的介面，並在**型樣**欄位中，輸入 `com.ibm.commerce.price.commands.GetProductContractUnitPriceCmd`，按一下**新增**，然後按一下**關閉**。
5. 按**下一步**。

6. 如果要新增適當的 import 陳述式，請按一下**新增套件**，並執行下列步驟：
   a. 在**型樣**欄位中，輸入 com.ibm.commerce.price.utils，然後按一下**新增**。
   b. 在**型樣**欄位中，輸入 com.ibm.commerce.exception，並按一下**新增**。
   c. 按一下**關閉**。
7. 確定**將介面變成公用**已選取。
8. 按一下**完成**。
   新介面的原始碼會顯示在「來源」窗格中。
9. 執行下列步驟，爲 getBonusPrice() 方法建立一個方法簽章：
   a. 以滑鼠右鍵按一下 **GetNewProductContractUnitPriceCmd** 介面，然後選取**新增 > 方法**。
      會開啓「建立方法」引導精靈。
   b. 確定**建立新方法**已經選取，然後按**下一步**。
   c. 在「方法名稱」欄位中，輸入 getBonusPrice。
   d. 如果要選取方法的傳回類型，請按一下**瀏覽**。在**型樣**欄位中，輸入 MonetaryAmount，然後按一下**確定**，再按**下一步**。
   e. 如果要選取方法可以擲出的異常狀況，請按一下**新增**。在**型樣**欄位中，輸入ECSystemException，然後按一下**新增**，再按一下**關閉**。
   f. 按一下**完成**。
10. 新增一個欄位到介面中，以指定指令的預設施行類別，方式如下：
    a. 以滑鼠右鍵按一下 **GetNewProductContractUnitPriceCmd** 介面，然後選取**新增 > 欄位**。
       會開啓「建立欄位」引導精靈。
    b. 在**欄位名稱**欄位中輸入 defaultCommandClassName。
    c. 從**欄位類型**下拉清單中，選取**字串**。
    d. 在**起始值**欄位中輸入

       ```
       "com.ibm.commerce.sample.commands.
          GetNewContractUnitPriceCmdImpl"
       ```

       然後按一下**完成**。

       註:

       1) 在您輸入此值時，必須包含雙引號。
       2) 如果出現警告訊息指出超類別中的某欄位將因此新欄位的建立而隱藏，請按一下**是**繼續進行。
11. 執行下列步驟，以新增欄位到指定指令名稱的介面中：

a. 以滑鼠右鍵按一下 **GetNewProductContractUnitPriceCmd** 介面，然後選取**新增 > 欄位**。
會開啓「建立欄位」引導精靈。

b. 在**欄位名稱**欄位中輸入 NAME。

c. 從**欄位類型**下拉清單中，選取**字串**。

d. 在**起始值**欄位中輸入

```
"com.ibm.commerce.sample.commands.
   GetNewProductContractUnitPriceCmd"
```

然後按一下**完成**。

建立 *GetNewContractUnitPriceCmdImpl* 施行類別

您必須爲新的作業指令建立施行類別。此施行類別會施行您最近建立的介面，並且包含作業指令的商業邏輯。

如果要建立 GetNewContractUnitPriceCmdImpl 施行類別，請執行下列步驟：

1. 在「工作台」中，先選取「專案」標籤，然後展開 **_WCSamples** 專案。
2. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.commands** 套件，然後選取**新增 > 類別**。
這時會開啓「建立類別」引導精靈。
3. 確定**建立新類別**已選取，然後按如下建立類別：
   a. 在**類別名稱**欄位中，輸入 GetNewContractUnitPriceCmdImpl。
   b. 如果要指定超類別，請按一下**瀏覽**，然後在**型樣**欄位中，輸入 com.ibm.commerce.price.commands.GetContractUnitPriceCmdImpl，再按一下**確定**。
   c. 按**下一步**。
   d. 如果要指定應匯入的套件，請按一下**新增套件**。在**型樣**欄位中，輸入下列套件：
      - com.ibm.commerce.command，然後按一下**新增**
      - com.ibm.commerce.exception，然後按一下**新增**
      - com.ibm.commerce.price.commands，然後按一下**新增**
      - com.ibm.commerce.price.utils，然後按一下**新增**
      - com.ibm.commerce.ras，然後按一下**新增**
      - com.ibm.commerce.server，然後按一下**新增**
      - java.math，然後按一下**新增**，再按一下**關閉**。

e. 如果要指定類別應施行的介面，請按一下**新增**。在「型樣」欄位中，輸入以下介面：
   - GetContractSpecialPriceCmd，然後按一下**新增**。
   - GetContractUnitPriceCmd，然後按一下**新增**。
   - GetProductContractUnitPriceCmd，然後按一下**新增**。
   - GetNewProductContractUnitPriceCmd，按一下**新增**，然後按一下**關閉**。

f. 按一下**完成**。

4. 以滑鼠右鍵按一下 **GetNewContractUnitPriceCmdImpl** 類別，然後選取**新增 > 欄位**。會開啓「建立欄位」引導精靈。依下列所示輸入資訊：
   a. 在**欄位名稱**欄位中，輸入 bonusPrice。
   b. 如果要選取欄位類型，請按一下**瀏覽**。在「型樣」欄位中，輸入 MonetaryAmount，然後按一下**確定**。
   c. 按一下**完成**。
5. 新增新 performExecute() 方法到類別中。此方法將執行下列動作：
   - 呼叫超類別 (GetContractUnitPriceCmdImpl) 的 performExecute() 方法
   - 設定 thisClass 與 methodName 的值，供異常狀況處理機制使用
   - 案例化 StoreAccessBean
   - 案例化 UserAccessBean
   - 取得原始產品價格
   - 計算最大適用折扣
   - 計算新紅利價格（此價格的類型爲 *double*）
   - 取得商店的貨幣類型
   - 捨入紅利價格，並且將它儲存成正確的貨幣金額

   如果要新增這個方法，請以滑鼠右鍵按一下 **GetNewContractUnitPriceCmdImpl** 類別，然後選取**新增 > 方法**。會開啓「建立方法」引導精靈。 依下列所示輸入資訊：
   a. 確定**建立新方法**已經選取，然後按**下一步**。
   b. 在**方法名稱**欄位中，輸入 performExecute。
   c. 從**傳回類型**下拉清單中，選取 **void**，然後按**下一步**。
   d. 如果要選取方法可以擲出的異常狀況，請按一下**新增**。在**型樣**欄位中，輸入 ECException，然後按一下**新增**，再按一下**關閉**。
   e. 按一下**完成**。
   f. 在原始碼的下一行後面：

```
      public void performExecute () throws com.ibm.commerce.exception.
ECException {
```

新增以下的程式碼：

您可以從「程式設計手冊」的 PDF 版本中剪貼這個程式碼。建議您一開始先複製 VisualAge for Java Scrapbook 視窗中的程式碼（請參閱 VisualAge for Java 線上說明以取得其它明細），然後查看程式碼，確定在剪貼作業中沒有任何字元遺失。在驗證程式碼之後，將它複製到目標位置上。請注意，將文字複製到另一編輯器中，可能會修改部份字元。

```
super.performExecute();

//取得並設定此類別名稱與方法，
//以便在發生異常狀況時使用。
final String thisClass = GetContractUnitPriceCmdImpl.class.getName();
final String methodName = "performExecute";

//取得商店存取 Bean
Integer storeId = getStoreId();
com.ibm.commerce.common.objects.StoreAccessBean storeAB =
     getCommandContext().getStore(storeId);

//取得使用者存取 Bean
com.ibm.commerce.user.objects.UserAccessBean bonusAB =
     new com.ibm.commerce.user.objects.UserAccessBean();

// 從 GetContractUnitPriceCmdImpl 取得算出的價格
MonetaryAmount priceOrg = super.getPrice();
double dblPriceOrg = priceOrg.getValue().doubleValue();

//計算可套用在此產品上的最大紅利
double dblBonusPrice; // = dblPriceOrg;
double dblMaxBonusPoint = 0;

try {
bonusAB.setInitKey_MemberId(super.getUserId().toString());
bonusAB.refreshCopyHelper();
double dblMaxDed = dblPriceOrg * 0.2;
dblMaxBonusPoint = (new java.math.BigDecimal(bonusAB.getBonusPoint()).
doubleValue());
if (dblMaxBonusPoint > dblMaxDed) dblMaxBonusPoint = dblMaxDed;
} catch (javax.ejb.CreateException ex) {
throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION, thisClass,
methodName, ex);
} catch (javax.ejb.FinderException ex) {

} catch (javax.naming.NamingException ex) {
throw new ECSystemException(ECMessage._ERR_GENERIC, thisClass,
methodName, ex);
} catch (java.rmi.RemoteException ex) {
```

```
throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION, thisClass,
methodName, ex);
}


//將最大的適用紅利套用在此產品價格上
dblBonusPrice = dblPriceOrg - dblMaxBonusPoint ;

//取得此商店的貨幣
CommandContext context = getCommandContext();
String requestedCurrency = Helper.getCurrency( context, storeAB );

//捨入並以 MonetoryAmount 類型傳回紅利價格
bonusPrice = new MonetaryAmount(new BigDecimal(dblBonusPrice),
requestedCurrency);
CurrencyManager.getInstance().roundCustomized(bonusPrice, storeAB);
```

儲存您的工作。

6. 選取 **GetNewContractUnitPriceCmdImpl** 類別中的 **getBonusPrice()** 方法，以檢視其原始碼。在原始碼中，將

```
return null;
```

改爲

```
return bonusPrice;
```

儲存您的工作。

建立 *NewProductDataBean* 資料 Bean

getCalculatedBonusPrice() 方法必須新增到現有 WebSphere Commerce ProductDataBean 中。由於您不得實際修改 ProductDataBean 的程式碼，因此您必須建立一個從 ProductDataBean 延伸而來的新資料 Bean，然後將方法新增到新資料 Bean 中。

如果要建立新資料 Bean，請執行下列步驟：

1. 在「工作區」中，先選取「專案」標籤，然後展開 **_WCSamples** 專案。
2. 以滑鼠右鍵按一下 **com.ibm.commerce.sample.databeans** 套件，並選取**新增 > 類別**。這時會開啓「建立類別」引導精靈。依下列所示輸入資訊：
   a. 在**類別名稱**欄位中，輸入 NewProductDataBean。
   b. 如果要指定超類別，請按一下**瀏覽**，然後在**型樣**欄位中，輸入 com.ibm.commerce.catalog.beans.ProductDataBean。按一下**確定**，然後按**下一步**。
   c. 將適當的 import 陳述式加入類別中，方法是按一下**新增套件**，然後執行下列步驟：

- 輸入 com.ibm.commerce.beans，然後按一下**新增**。
- 輸入 com.ibm.commerce.catalog.objects，然後按一下**新增**。
- 輸入 com.ibm.commerce.command，然後按一下**新增**。
- 輸入 com.ibm.commerce.datatype，然後按一下**新增**。
- 輸入 com.ibm.commerce.exception，然後按一下**新增**。
- 輸入 com.ibm.commerce.price.beans，然後按一下**新增**。
- 輸入 com.ibm.commerce.ras，然後按一下**新增**。
- 輸入 com.ibm.commerce.sample.commands，然後按一下**新增**。
- 輸入 com.ibm.commerce.server，然後按一下**新增**。
- 輸入 java.util，然後按一下**新增**，再按一下**關閉**。

d. 按一下**完成**。

3. 以滑鼠右鍵按一下 **NewProductDataBean** 類別，並選取**新增 > 方法**。這會開啓「新增方法」引導精靈。
4. 確定**建立新方法**已經選取，然後按**下一步**。
5. 在**方法名稱**欄位中，輸入 getCalculatedBonusPrice。
6. 如果要選取傳回類型，請按一下**瀏覽**。在**型樣**欄位中，輸入 PriceDataBean，再按一下**確定**，然後按**下一步**。
7. 如果要選取方法可以擲出的異常狀況，請按一下**新增**。在**型樣**欄位中，輸入 ECSystemException，然後按一下**新增**，再按一下**關閉**。
8. 按一下**完成**。
9. 在原始碼中，將 return null; 取代爲下列的值：

```
PriceDataBean ibnPrice = null;
try {
     GetNewProductContractUnitPriceCmd comm =
        (GetNewProductContractUnitPriceCmd) CommandFactory.createCommand
        (GetNewProductContractUnitPriceCmd.NAME,
             getCommandContext().getStoreId());

     ECTrace.trace(ECTraceIdentifiers.COMPONENT_CATALOG,
     this.getClass().getName(), "getCalculatedBonusPrice",
     "Getting Price for CatalogEntry: " + getProductID());

comm.setCatEntryId(new Long(getProductID()));
comm.setCommandContext(getCommandContext());
comm.execute();
ibnPrice = new PriceDataBean(comm.getBonusPrice(), getCommandContext().
getStore(),getCommandContext().getLanguageId());
} catch (Exception e) {
throw new ECSystemException(ECMessage._ERR_RETRIEVE_PRICE, this.getClass()
```

```
.getName(), "getCalculatedBonusPrice",e);
}

return ibnPrice;
```

儲存您的工作。

新增新紅利價格到產品顯示範本中

下一步是將新的紅利價格新增至產品顯示範本，使購物者可以看見自訂的價格。一旦更新顯示範本之後，就會顯示新的折扣價格。

範例商店使用 ProductDisplay.jsp 範本來顯示產品。因此，您必須用新的資訊來更新這個範本，才能顯示新的價格。

如果要更新顯示範本，請執行下列步驟：

1. 導覽至下列目錄：
   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_name*
2. 複製一份 ProductDisplay.jsp 檔，並將之命名爲 ProductDisplay.jsp.bak。
3. 以文字編輯器開啓 ProductDisplay.jsp。
4. 在 <%@ page import="com.ibm.commerce.common.beans.*" %> 後面，新增下列 import 陳述式：

   ```
   <%@ page import="com.ibm.commerce.sample.commands.*" %>
   <%@ page import="com.ibm.commerce.sample.databeans.*" %>
   ```
5. 將所有出現的 ProductDataBean 換成 NewProductDataBean。
6. 找出下行：

   ```
   <font class="price"><%=product.getCalculatedContractPrice()%></font>
      <br><br>
   ```

   在這一行的後面，插入下列的程式碼，以擷取並顯示產品的紅利價格。

   ```
   <font class="price"><%=product.getCalculatedBonusPrice()%>
   Bonus Price </font>
   <br><br>
   ```
7. 儲存這個檔案。

註： 如果您將 *WebSphere Commerce 程式設計手冊* PDF 版中的區段剪貼至顯示範本中，請確定在該程序期間您未修改任何字元。

測試 Enterprise Bean 的延伸

在本節中，您可藉由檢視「時尚館」範例商店中的產品，來測試您所做的 Enterprise Bean 延伸。這時會顯示新的紅利價格。

請注意，爲了簡化這個範本，所有的購物者（已登錄或購物訪客）都可以檢視紅利價格。對於沒有紅利積點的購物者而言，紅利價格和一般價格相同。

如果要測試 Enterprise Bean 的延伸以及查看顯示的紅利價格，請執行下列步驟：

1. 執行下列步驟，驗證 Servlet 引擎的路徑中已含有 _WCSamples 專案：
   a. 從 VisualAge for Java 的**工作區**功能表中選取**工具 > WebSphere Test Environment**。
      會開啓 WebSphere Test Environment 控制中心。
   b. 按一下 **Servlet 引擎**。
   c. 如果 Servlet 引擎正在執行，請按一下**停止 Servlet 引擎**，然後按一下**編輯類別路徑**。
   d. 如果尚未選取 **_WCSamples**，此時請選取，並按一下**確定**。
2. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中的說明，啓動 WebSphere Test Environment。永久名稱伺服器與 EJB 伺服器可能已在執行中，在此情況下，您只需啓動 Servlet 引擎即可。
3. 開啓瀏覽器，然後輸入下列的 URL

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
       storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

4. 按一下「服務」標頭下的**登錄**鏈結，然後按一下「新客戶」標頭下的**登錄**。使用電子郵件位址 wctester@wc 與密碼 wctester1，登錄新客戶。在其它欄位中填入測試值，並按一下**提交**。讓瀏覽器維持開啓狀態。
5. DB2 開啓 DB2 指令中心，並執行下列步驟：
   a. 按一下**互動**標籤
   b. 在**指令**欄位中執行下列步驟：
      1) 輸入

         ```
         connect to your_database_name
         ```

         其中 *your_database_name* 是您 WebSphere Commerce 的資料庫名稱，然後按一下「執行」圖示。
      2) 輸入 `select users_id from userreg where logonid = 'wctester@wc'`，並按一下「執行」圖示。
   c. 「查詢結果」標籤會顯示您在步驟 4 中爲客戶所登錄的項目。請在此記下客戶的
      USERS_ID：_____
   d. 更新最近登錄的客戶的紅利積點結餘。按一下「互動」標籤，然後在**指令**欄位中，輸入下列指令：

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
```

其中 *users_id* 是步驟 5c 中的值。按一下「執行」圖示。

6. Oracle 執行下列步驟，更新測試使用者的紅利積點結餘：
   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**主電腦字串**欄位中，輸入您的連接字串。
   e. 輸入 `select users_id from userreg where logonid = 'wctester@wc';`
   f. 會出現您在步驟 4 中所登錄的客戶項目。請在此記下客戶的 USERS_ID：_____
   g. 輸入下列指令，以更新最近登錄之客戶的紅利積點結餘：

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id;
      ```

      其中 *users_id* 是步驟 6f 中的值。
   h. 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。
7. 在瀏覽器中，按一下**男仕時尚精品**鏈結，以檢視商店的「男仕時尚精品」部。
8. 按一下號召特價品鏈結，以檢視產品頁面。頁面中會顯示一般價格以及根據客戶的紅利積點結餘所算出的折扣價格。

註: 如果所出現的是堆疊追蹤而非紅利價格，則您可能需停用快取。其做法是依下列所示在 *instance_name*.xml 檔中將 CacheDaemon 元件值設爲 False：

```
<component compClassName="com.ibm.commerce.cache.daemon.
CacheDaemonComponent"
                enable="false"
                name="CacheDaemon" />
```

在變更 *instance_name*.xml 檔中的值後，您必須停止再重新啓動 WebSphere Test Environment 中的 Servlet 引擎。

（選用）將自訂商業邏輯部署到遠端 WebSphere Commerce Server 中

本節說明如何將修改好的實體 Bean 以及新的作業指令，部署到執行於 WebSphere Test Environment 外的商店中。

部署作業包括：爲 WebSphere Commerce 公用 Enterprise Bean、指令與資料 Bean 邏輯建立 JAR 檔，將 JAR 檔置於目標伺服器的適當目錄中，停止 WebSphere Commerce 案例，修改類別路徑，使用 XMLConfig 公用程式來部署 Enterprise Bean，以及重新啓動案例。

為新價格指令建立 JAR 檔

您必須爲 **_WCSamples** 專案建立一個 JAR 檔，才能部署新作業指令。如果要建立此 JAR 檔，請在開發機器上執行下列步驟：

1. 按第 313 頁的附錄 A, 『啓動與停止 WebSphere Test Environment』中所述，停止 WebSphere Test Environment。
2. 選取「專案」標籤，並選取 **_WCSamples** 專案。
3. 在專案反白的情況下，以滑鼠右鍵按一下並選取**匯出**。
   這會開啓「匯出」引導精靈。
4. 選取 **Jar 檔**，並按**下一步**。
5. 在 **Jar 檔**欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev\mytemp_c\wcssamplesc_1.jar
   其中 *drive* 爲安裝 WebSphere Commerce 的磁碟機。
6. 按如下選取屬性：

屬性	**值**
類別	勾選
java	不勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
7. 按一下**完成**。

由於所建的 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中，導覽至下列目錄：
   drive:\WebSphere\CommerceServerDev\mytemp_c
2. 輸入 `mkdir temp3`。
3. 輸入 `cd temp3`。

4. 按如下設定路徑：
   `set PATH=%PATH%;drive:\WebSphere\WebSphereStudio4\bin;`
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 `jar xvf ../wcssamplesc_1.jar`。
6. 輸入 `jar cvf ../wcssamplesc.jar *` （請記得拿掉名稱中的 `_1`）。

為 WCSUser EJB 群組建立 JAR 檔

您必須爲內含已修改 Enterprise Bean 的 EJB 群組，建立一個 EJB 1.1 匯出 JAR 檔。因此，在您建立 JAR 檔時必須選取下列群組：

- WCSUser

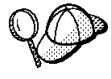
如果要爲 WCSUser EJB 群組建立 JAR 檔，請執行下列步驟：

1. 先選取 EJB 標籤，然後標示出 **WCSUser** EJB 群組。
2. 以滑鼠右鍵按一下 **WCSUser** EJB 群組，並選取**匯出 > EJB 1.1 JAR**。
   會開啓「匯出至 EJB 1.1 JAR 檔」引導精靈。
3. 在 **JAR 檔**欄位中輸入 *drive*:\WebSphere\CommerceServerDev\mytemp_c\CustomizedWCSUserDeployed_DT.jar。
4. 按如下選取屬性：

屬性	值
類別	勾選
java	不勾選
資源	勾選
目標資料庫	DB2 如果您是部署到 DB2 資料庫，請選取 **DB2 for NT, V7.1**。 Oracle 如果您是部署到 Oracle 資料庫，請選取 **Oracle, V8**。
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
5. 按一下**完成**。

此時會建立 JAR 檔。



JAR 檔名稱中已加上 "_DT" 字尾，如此可提醒您在將之部署到 WebSphere Commerce 應用程式前，必須透過 WebSphere Application Server 所提供的「EJB 部署工具」執行此 JAR 檔。

建立 *wcsejsclient.jar* 檔

如果要建立從屬站 JAR 檔，請執行下列步驟：

1. 先選取 EJB 標籤，然後標示出所有 WebSphere Commerce EJB 群組（其名稱的開頭爲 WCS）。在這些群組皆反白的情況下，以滑鼠右鍵按一下並選取**匯出 > 從屬站 JAR**。
   這會開啓「匯出」引導精靈。
2. 在 **JAR 檔**欄位中輸入
   drive:\WebSphere\CommerceServerDev\mytemp_c\wcsejsclient.jar
3. 按如下選取屬性：

屬性	值
Bean	勾選
類別	勾選
java	不勾選
資源	勾選
在 .class 檔中納入除錯屬性	勾選

   在其它屬性方面則接受預設值。
4. 按一下**完成**。

此時會建立 JAR 檔。

將已更新的 JSP 範本複製到目標商店目錄中

在第 293 頁的『新增新紅利價格到產品顯示範本中』中，您已經更新了顯示範本，以反映出新建立的價格。在這個步驟中，您會將更新過的 JSP 範本從 WebSphere Test Environment 目錄架構複製到商店在 WebSphere Test Environment 外部執行時所用的目錄中。

1. 在開發機器上，導覽至下列目錄：
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*
   其中 *vaj_drive* 爲您安裝 VisualAge for Java 的磁碟機， *store_directory* 爲範例商店的目錄名稱。
   複製 ProductDisplay.jsp 檔。
2. 將 ProductDisplay.jsp 檔貼到下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\store_directory
   ```

   其中 *drive* 爲安裝 WebSphere Commerce 的磁碟機， *store_directory* 爲商店的目錄名稱， *instance_name* 爲您 WebSphere Commerce 案例的名稱。

將 JAR 檔複製到目標 WebSphere Commerce Server 中

您必須將 JAR 檔從開發機器複製到目標 WebSphere Commerce Server 上的適當目錄中。如果要複製這些檔案，請執行下列步驟：

1. 在開發機器上導覽至下列目錄：

   drive:\WebSphere\CommerceServerDev\mytemp_c

   ，然後找出下列檔案：

   - wcssamplesc.jar
   - CustomizedWCSUserDeployed_DT.jar
   - wcsejsclient.jar

   其中 *drive* 爲您安裝 WebSphere Commerce Studio, Business Developer Edition 的磁碟機。

   上述每一個檔案皆必須複製到目標 WebSphere Commerce Server 的特定目錄中。請仔細閱讀下列步驟，以確定每一個檔案皆儲存在正確位置中。

2. 將 wcssamplesc.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\WEB-INF\lib
   ```

   其中 *drive* 爲您安裝 WebSphere Commerce Business Edition 的磁碟機，*instance_name* 爲您案例的名稱（例如 demo）。

3. 將 wcsejsclient.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   drive:\WebSphere\CommerceServer\temp\lib

4. 將 CustomizedWCSUserDeployed_DT.jar 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

   drive:\WebSphere\CommerceServer\temp

執行 EJB 部署工具

您必須針對內含新 EJB 群組的 JAR 檔執行 EJB 部署工具。此工具內含於 WebSphere Application Server 中。

如果要執行此工具，請執行下列步驟：

1. 在指令提示下，切換至下列目錄：

   drive:\WebSphere\CommerceServer\temp

2. 輸入下列指令，暫時將工具新增到系統路徑中：

   PATH=*drive*:\WebSphere\AppServer\deploytool;%PATH%

3. 依下列所示輸入 ejbdeploy 指令：

```
ejbdeploy EJBGroupJARFile WorkingDir OutputJARFile -nowarn -keep -35 -cp
   ClassPathOfDepJARFiles
```

其中：

- *EJBGroupJARFile* 爲您 EJB 群組的 JAR 檔名稱。在本例中，此爲 CustomizedWCSUserDeployed_DT.jar。
- *WorkingDir* 爲工作目錄。
- *OutputJARFile* 爲輸出 JAR 檔的名稱。就本例而言，請輸入 CustomizedWCSUserDeployed.jar。
- -nowarn 爲選用參數，用以抑制警告與參考訊息的出現。
- -keep 爲選用參數，用以在執行 ejbdeploy 指令後保留工作目錄。
- -35 爲必要參數，表示將針對 WebSphere Application Server 3.5 版所提供之「EJB 部署工具」中所用的 CMP 實體 Bean，採用相同的由上往下映射規則。
- -cp *ClassPathOfDepJARFiles* 爲任何相依 JAR 檔的類別路徑。如果您曾修改現有 WebSphere Commerce Enterprise Bean，則相依 JAR 檔的類別路徑中必須包含 wcsejsclient.jar、wcsejbimpl.jar 與 xml4j.jar 檔。因此請輸入：

```
"drive:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
drive:\WebSphere\AppServer\InstalledApps\
WC_Enterprise_App_instanceName.ear\lib\wcsejbimpl.jar;
drive:\WebSphere\AppServer\InstalledApps\
WC_Enterprise_App_instanceName.ear\lib\xml4j.jar;"
```

修改實體 Bean 的交易隔離層次

在本步驟中，您會使用 modifyIsolationLevel 指令，將實體 Bean 的交易隔離層次修改成您特定資料庫類型的必要層次。

如果要執行 modifyIsolationLevel 指令，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 中，使用指令提示導覽至下列目錄：

```
drive:\WebSphere\CommerceServer\bin
```

2. 您必須發出採用下列一般語法的 modifyIsolationLevel 指令：

```
modifyIsolationLevel -jarFile jar_file_name.jar
   -logFile log_file_name -dbType db_type
```

其中

- *jar_file_name.jar* 爲內含自訂程式碼的 JAR 檔名稱
- *log_file_name* 爲應記載資訊的完整檔案名稱

- *db_type* 爲您所用的資料庫類型。輸入 DB2 或 ORACLE

以下是設有所有指定值的 modifyIsolationLevel 指令範例：

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

註: 參數名稱有區分大小寫。亦即，jarFile 不等於 jarfile。請確定您所輸入的參數名稱無誤。

如果指令視窗中沒有顯示異常狀況，表示指令執行成功。在完成後，請記下您部署 JAR 檔變更的時間戳記。

更新目標資料庫

如果您要部署到採用有別於 WebSphere Test Environment 所用資料庫的目標 WebSphere Commerce Server 中，您必須依下列所示更新目標資料庫：

- 如果您已完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，您必須更新表格，以便在 USERS 表格中針對每一位使用者各新增一列。
- 如果您尚未完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，您必須建立表格並移入資料。

如果您尚未完成第 183 頁的第 9 章, 『指導教學：建立新商業邏輯』，您必須建立表格並移入資料。我們會提供這些實務的相關指示。

DB2 如果您所用的是 DB2 資料庫，且您需要*更新* BONUS 表格，請執行下列步驟：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**），然後按一下 **Script** 標籤。
2. 在 **Script** 欄位中輸入

   ```
   connect to your_database_name
   ```

   其中 *your_database_name* 爲資料庫名稱，然後按一下「執行」圖示。
3. 在 **Script** 欄位中，輸入下列指令，然後按一下「執行」圖示：

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
   ```

   此時 BONUS 表格已更新。

DB2 如果您所用的是 DB2 資料庫，且您需要*建立與移入資料*到 BONUS 表格中，請執行下列步驟：

1. 開啓 DB2 指令中心（**開始 > 程式集 > IBM DB2 > 指令中心**），然後按一下 **Script** 標籤。
2. 在 **Script** 欄位中輸入

   ```
   connect to your_database_name
   ```

   其中 *your_database_name* 爲資料庫名稱，然後按一下「執行」圖示。
3. 在 **Script** 欄位中，輸入下列指令，然後按一下「執行」圖示：

   ```
   CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
   BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
   constraint f_memberid foreign key (MEMBERID)
   references users (users_id) on delete cascade)
   ```

   此時 BONUS 表格已建立。
4. 如果要在表格中移入資料，請在 **Script** 欄位中輸入下列資訊，並按一下「執行」圖示：

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. 關閉 DB2 指令中心。

Oracle 如果您所用的是 Oracle 資料庫，而您需要*更新* BONUS 表格，請執行下列步驟：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便更新 BONUS 表格：

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   按一下 Enter 鍵，以執行 SQL 陳述式。
   此時 BONUS 表格已更新。
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。

Oracle 如果您所用的是 Oracle 資料庫，而您需要*建立並移入資料*到 BONUS 表格中，請執行下列步驟：

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**主電腦字串**欄位中，輸入您的連接字串。
5. 在 SQL Plus 視窗中輸入下列資訊，以便建立 BONUS 表格：

   ```
   CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
   BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
   constraint f_memberid foreign key (MEMBERID)
   references users (users_id) on delete cascade);
   ```

   按一下 Enter 鍵，以執行 SQL 陳述式。
   此時 BONUS 表格已建立。
6. 輸入下列指令，以便將資料移入到 BONUS 表格中：

   ```
   insert into BONUS (select USERS_ID, 0 from USERS);
   ```
7. 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。

從 WebSphere Application Server 匯出現行企業應用程式

在此步驟中，您將從 WebSphere Application Server 匯出現行企業應用程式，以便稍後可在應用程式組譯工具中開啓。

如果要從 WebSphere Application Server 匯出現行企業應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**。
3. 展開**企業應用程式**。
4. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，展開**示範**應用程式，並選取**匯出應用程式**。
5. 在**匯出目錄**欄位中輸入 *drive*:\WebSphere\CommerceServer\working。
   這會將整個應用程式（包括所有資源）匯出到 WC_Enterprise_App_*instanceName*.ear 檔中（其中 *instanceName* 爲您 WebSphere Commerce 案例的名稱）。

註: 如果已存在現有的 WC_Enterprise_App_*instanceName*.ear 檔，您可以更名舊檔案或加以改寫。

匯出企業應用程式的 XML 架構資訊

您也必須匯出企業應用程式的 XML 架構資訊。如果要匯出此資訊，您可使用 WebSphere Application Server 所提供的 XMLConfig 指令行公用程式。

如果要匯出此架構資訊，請執行下列步驟：

1. 在指令提示下，切換至下列目錄：

   drive:\WebSphere\CommerceServer\working

2. 輸入下列指令，呼叫 XMLConfig 工具以執行局部匯出：

   ```
   xmlConfig -export OutputFileB.xml -partial was.export.app.xml
      -adminNodeName wasHostName
   ```

   其中 *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。此外，OutputFileB.xml 爲因執行此指令而建立的檔案。

在您匯出現行企業應用程式中的相關 Enterprise Bean 資訊後，您必須更新 OutputFileB.xml 檔，以指向內含已修改 User Bean 程式碼的 JAR 檔。

如果要更新 OutputFileB.xml 檔，請執行下列步驟：

1. 導覽至下列目錄：

   drive:\WebSphere\CommerceServer\working

2. 以文字編輯器開啓 OutputFileB.xml 檔。
3. 找出 <ear-file-name> 標籤，並將值換成：

   ```
   drive:\WebSphere\CommerceServer\working\
      WC_Enterprise_App_instanceName.ear
   ```

4. 找出 WCSUser EJB 群組的 <ejb-module> 段落，並將 <jar-file> 標籤中的值改爲 CustomizedWCSUserDeployed.jar
5. 儲存 OutputFileB.xml 檔。

將已修改 EJB 群組組譯到企業應用程式中

在此步驟中，您將在應用程式組譯器工具中開啓您的企業應用程式。一旦在該工具內開啓後，您可執行下列步驟，將已修改的 User Bean 置於企業應用程式中：

1. 複製一份現有 WCSUser EJB 群組版本的類別路徑。
2. 移除現有的 WCSUser EJB 群組版本。
3. 匯入新的 WCSUser EJB 群組版本。新 EJB 群組的 JAR 檔儲存在企業應用程式的「EJB 模組」區段中。
4. 設定 WCSUser EJB 群組的類別路徑。

如果要將新 EJB 群組組譯到企業應用程式中，請執行下列步驟：

1. 執行下列步驟，以備份現行企業應用程式：
   a. 在指令提示下，切換至下列目錄：

      drive:\WebSphere\CommerceServer\working

   b. 輸入下列指令：

      ```
      copy WC_Enterprise_App_instanceName.ear
         WC_Enterprise_App_instanceName.ear.bak2
      ```

2. 開啓 WebSphere Application Server 管理主控台。
3. 從**工具**功能表選取**應用程式組譯工具**。（如果「歡迎使用」視窗開啓，請選取**取消**，以存取主控台。）
4. 執行下列步驟，以開啓您要工作的企業應用程式：
   a. 從**檔案**功能表中，選取**開啓**。
   b. 在**檔案名稱**欄位中輸入：

      ```
      drive:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_instanceName.ear
      ```

      然後按一下**開啓**。等待應用程式開啓，再繼續進行後續步驟。需要等待幾分鐘。
5. 按一下 **EJB 模組**。右窗格中會顯示企業應用程式中的 EJB 模組。
6. 按一下 **WCSUser** EJB 模組。
7. 按一下「一般」標籤，以檢視現有 WCSUser EJB 模組的類別路徑資訊。將此類別路徑資訊複製到文字檔中（例如 WCSUser_path.txt）。
8. 以滑鼠右鍵按一下 **WCSUser** EJB 模組，並選取**刪除**。
9. 以滑鼠右鍵按一下 **EJB 模組**，並選取**匯入**。
10. 在**檔案名稱**欄位中輸入：

    ```
    drive:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
    ```

    然後按一下**開啓**。在「確認值」視窗中按一下**確定**。
11. 一旦匯入 CustomizedWCSUserDeployed.jar 檔後，請捲動到 **WCSUser** EJB 群組，並選取此群組。
    此群組的資訊會顯示在右窗格中。
12. 開啓內含舊版 WCSUser EJB 群組之類別路徑資訊的文字檔。選取並複製類別路徑。
13. 在新 WCSUser EJB 群組的類別路徑欄位中，貼上此類別路徑資訊。
14. 按一下**套用**。
15. 從**檔案**功能表中，選取**關閉**。

16. 等待檔案關閉，然後從**檔案**功能表中選取**開啓**，重新開啓 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear 檔。
17. 執行下列步驟，以架構 User Bean 的安全：
    a. 展開「EJB 模組」節點，找出 **WCSUser** 節點並展開。
    b. 展開**實體 Bean**。
    c. 展開**使用者**。
    d. 按一下**方法延伸**，然後在右窗格中執行下列步驟：
        1) 按一下**進階**標籤。
        2) 確定**安全身份**已選取。
        3) 針對每一個方法，確定**使用 EJB 伺服器的身份**已選取。
        4) 按一下**套用**（如果您有做任何修改的話）。
    e. 在左導覽窗格中，以滑鼠右鍵按一下 WCSUser EJB 群組下的**安全職務**，並選取**新建**，然後執行下列步驟：
        1) 在**名稱**欄位中，輸入 WCSecurityRole，並按一下**套用**。請注意，如果此職務已存在，便不需執行此步驟。
    f. 在左導覽窗格中，以滑鼠右鍵按一下 WCSUser EJB 群組下的**方法許可權**，並選取**新建**，然後執行下列步驟：
        1) 在**方法許可權名稱**欄位中輸入 WCMethodPermission。
        2) 在**方法**選擇區中按一下**新增**。
        這會開啓「新增方法」視窗。
        3) 展開 **CustomizedWCSUserDeployed.jar**，並選取所有的 Enterprise Bean（選取時請按住 Shift 鍵）。按一下**確定**。此時，所有 Enterprise Bean 會顯示在 Enterprise Bean 直欄下，**所有方法**會顯示在「類型」直欄下。
        4) 在「職務」選擇區中，按一下**新增**，並選取 WCSecurityRole，然後按一下**確定**。
        5) 按一下**套用**，然後按一下**確定**。
18. 從**檔案**功能表中，選取**儲存**。
19. 關閉「應用程式組譯器工具」。

在完成此步驟後，您便已建立一個新企業應用程式，且其含有先前的所有邏輯以及您的新商業邏輯。這全含在新修改的 WC_Enterprise_App_*instanceName*.ear 檔中。

將新企業應用程式匯入至 WebSphere Application Server 中

下列的高階步驟是將新企業應用程式匯入到 WebSphere Application Server 中：

1. 停止目前正在 WebSphere Application Server 中執行的企業應用程式，然後移除之。這些步驟是在 WebSphere Application Server 管理主控台中進行。
2. 使用 XMLConfig 指令行公用程式匯入新應用程式。
3. 重新整理 WebSphere Application Server 管理主控台，然後啓動新企業應用程式。

下列各節將進一步說明上述每一個步驟。

停止並移除現行企業應用程式: 如果要停止並移除 WebSphere Application Server 中的現行企業應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**。
3. 展開**節點**。
4. 展開 *nodeName* （其中 *nodeName* 爲節點名稱。）
5. 展開**應用程式伺服器**。
6. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server -** *instanceName*，並選取**停止**。
7. 展開**企業應用程式**。
8. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Enterprise Server - 示範**應用程式，並選取**停止**。
9. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Enterprise Server - 示範**應用程式，並選取**移除**。
10. 當出現提示指出是否應匯出應用程式時，請選取**否**。

使用 XMLConfig 匯入新企業應用程式: 如果要使用 XMLConfig 指令行公用程式來匯入新企業應用程式，請執行下列步驟：

1. 導覽至下列目錄：

   ```
   drive:\WebSphere\CommerceServer\working
   ```

2. 在指令提示下，輸入下列指令以便將企業應用程式匯入到 WebSphere Application Server 中：

   ```
   xmlConfig -import OutputFileB.xml -adminNodeName was_hostname
   ```

   其中 *was_hostname* 爲內含現行應用程式的 WebSphere Application Server 節點名稱。

註: 400 如果您是部署到執行 iSeries 的 WebSphere Commerce 案例中，在您匯入應用程式後，您得額外執行一個步驟，以修改目錄許可權。有關如何修改這些許可權的詳述，請參閱第 346 頁的『匯入企業應用程式』。

啓動新企業應用程式: 在您使用 XMLConfig 指令行公用程式匯入新企業應用程式後，即可使用 WebSphere Application Server 管理主控台來執行重新整理，然後啓動新應用程式。

如果要重新整理主控台並啓動新應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**
3. 標示**節點**。
4. 按一下**重新整理所選子樹**圖示。
5. 執行下列以啓動 WebSphere Commerce 應用程式：
   - 展開**應用程式伺服器**。
   - 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server -** *instanceName*，並選取**啓動**。

在目標商店中測試新程式碼

如果要在執行於 WebSphere Application Server 的商店中測試修改過的實體 Bean 與新作業指令，請執行下列步驟：

1. 開啓瀏覽器並輸入下列的 URL：

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

   其中 *store_Id* 爲您商店的識別碼，*catalog_Id* 爲您商店型錄的識別碼。

   註: 如果您收到錯誤 500，您必須重新啓動 WebSphere Application Server，以便重新整理您的企業應用程式。

2. 按一下「服務」標頭下的**登錄**鏈結，然後按一下「新客戶」標頭下的**登錄**。使用電子郵件位址 `wctester@wc` 與密碼 `wctester1`，登錄新客戶。在其它欄位中填入測試值，並按一下**提交**。讓瀏覽器維持開啓狀態。

3. DB2 開啓 DB2 指令中心，並執行下列步驟：
   a. 按一下**互動**標籤
   b. 在**指令**欄位中執行下列步驟：
      1) 輸入

         ```
         connect to your_database_name
         ```

其中 *your_database_name* 是您 WebSphere Commerce 的資料庫名稱，然後按一下「執行」圖示。

2) 輸入 `select users_id from USERREG where LOGONID = 'wctester@wc'`，並按一下「執行」圖示。

c. 「查詢結果」標籤會顯示您在步驟 2 中爲客戶所登錄的項目。請在此記下客戶的

USERS_ID：_____

d. 更新最近登錄的客戶的紅利積點結餘。按一下「互動」標籤，然後在**指令**欄位中，輸入下列指令：

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
```

其中 *users_id* 是步驟 3c 中的值。按一下「執行」圖示。

4. Oracle 執行下列步驟，更新測試使用者的紅利積點結餘：

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**主電腦字串**欄位中，輸入您的連接字串。

e. 輸入 `select users_id from USERREG where LOGONID = 'wctester@wc';`，以判斷使用者 ID

f. 會出現您在步驟 2 中登錄的客戶項目。請在此記下客戶的

USERS_ID：_____

g. 輸入下列指令，以更新最近登錄之客戶的紅利積點結餘：

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id;
```

其中 *users_id* 是步驟 4f 中的值。

h. 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。

5. 在瀏覽器中，按一下**男仕時尚精品**鏈結，以檢視商店的「男仕時尚精品」部。

6. 按一下號召特價品鏈結，以檢視產品頁面。頁面中會顯示一般價格以及根據客戶的紅利積點結餘所算出的折扣價格。

第 5 篇 附錄與後記

附錄 A. 啓動與停止 WebSphere Test Environment

本節說明在 VisualAge for Java 中啓動 WebSphere Test Environment 的相關步驟。一般而言啓動此環境時，會要求您執行下列步驟：

1. 啓動永久名稱伺服器。
2. 啓動 EJB 伺服器。
3. 啓動 Servlet 引擎。

我們會在後續各節中分別說明這些步驟。

如果要停止測試環境，請顛倒步驟順序。

註: 爲了使用到 WebSphere Test Environment 中的所有特性，在啓動 WebSphere Test Environment 前，您應確定 WebSphere Application Server 不在執行中。有關停止 WebSphere Application Server 的說明，請參閱 *WebSphere Commerce 安裝手冊*。

啓動與停止永久名稱伺服器

永久名稱伺服器會接收從屬站針對 Enterprise Bean 所發出的查看要求。所有的 Enterprise Bean 皆會向永久名稱伺服器登錄。

如果要啓動或停止永久名稱伺服器，請執行下列步驟：

1. 從 VisualAge for Java 的**工作區**功能表中選取**工具 > WebSphere Test Environment**。
   這會開啓 WebSphere Test Environment 控制中心。
2. 按一下**永久名稱伺服器**，然後按一下下列其中一種動作：
   - **啓動名稱伺服器**。
     等待「主控台」視窗中出現「開啓商業用的伺服器」訊息。啓動此伺服器可能需要幾分鐘。
   - **停止名稱伺服器**。

啓動與停止 EJB 伺服器

EJB 伺服器爲一種高階的程序或應用程式，可提供執行期間環境以支援採用 Enterprise Bean 之伺服器應用程式的執行。

如果要啓動或停止 EJB 伺服器，請執行下列步驟：

1. 開啓「EJB 伺服器架構」視窗（按一下 EJB 標籤，然後從 **EJB** 功能表中選取 **開啓至 > 伺服器架構**）。
2. 以滑鼠右鍵按一下您想啓動或停止的 (例如，**EJB 伺服器 {伺服器 1}**)，並執行下列其中一種動作：
   - 選取**啓動伺服器**。
     等待「主控台」視窗中出現「開啓商業用的伺服器」訊息（您可能得選取「主控台」中的 **EJB 伺服器**，以檢視此伺服器的狀態）。啓動此伺服器可能得花 10 或 15 分鐘，此視您的電腦而定。
   - 選取**停止伺服器**

「主控台」爲 IDE 中之 Java 程式的標準輸入與輸出裝置。如果要查看某特定程式的輸出，請先在「所有程式」窗格中選取所要程式，此時其輸出會出現在「輸出」窗格中。

啓動與停止 Servlet 引擎

Servlet 引擎會整合 Web 伺服器與 WebSphere Application Server 功能，以建立一個測試用的環境。

如果要啓動或停止 Servlet 引擎，請執行下列步驟：

1. 從 VisualAge for Java 的**工作區**功能表中選取**工具 > WebSphere Test Environment**。
   這會開啓 WebSphere Test Environment 控制中心。
2. 按一下 **Servlet 引擎**，並執行下列之一：
   - **啓動 Servlet 引擎**。
     當啓動 Servlet 引擎後，主控台會顯示 `***Servlet 引擎已啓動***` 訊息。
   - **停止 Servlet 引擎**。

附錄 B. 部署明細

當您在 VisualAge for Java 中建立好自訂程式碼，並且在 WebSphere Test Environment 中測試完後，就必須將之部署到執行於 WebSphere Test Environment 外的 WebSphere Commerce 案例中。這個 WebSphere Commerce 案例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）。

本附錄說明將自訂程式碼部署到執行於 WebSphere Test Environment 外之 WebSphere Commerce 案例時所需執行的步驟。您應先參考第 171 頁的『程式碼部署』 ，以瞭解部署程序的高階步驟，然後再參考本附錄取得詳細說明。

映射至整合式檔案系統 (iSeries)

400 本節僅適用於當您的目標 WebSphere Commerce Server 是在 iSeries 平台上執行時。

如果您的目標 WebSphere Commerce Server 是在 iSeries 平台上執行，您必須將開發機器上的本端磁碟機映射至您 iSeries 伺服器上的整合式檔案系統 (IFS)。在本文件的後續各節中，*iSeries_drive* 是指映射至 IFS 的這個本端磁碟機。此外，*drive* 代表您開發機器上的本端磁碟機（未映射至 IFS）。

自訂指令和資料 Bean 的 JAR 檔案

自訂指令和資料 Bean 應儲存在一個有別於 WebSphere Commerce 程式碼的專案中。當您完成在 WebSphere Test Environment 中的測試後，您必須針對內含自訂指令和資料 Bean 程式碼的專案建立一個 JAR 檔，然後將該 JAR 檔置於目標 WebSphere Commerce Server 的適當目錄中。

如果要爲自訂指令和資料 Bean 建立一個 JAR 檔，請執行下列步驟：

1. 在 VisualAge for Java 的「工作台」中，選取**專案**標籤。
2. 以滑鼠右鍵按一下內含自訂指令和資料 Bean 程式碼的專案，然後選取**匯出**。這時會開啓「匯出」引導精靈。
3. 選取 **Jar 檔**，然後按**下一步**。
4. 在 **Jar 檔**欄位中輸入如下：
   drive:\WebSphere\CommerceServerDev*temp_directory*\ *jar_file_name*_1.jar

其中 *drive*:\WebSphere\CommerceServerDev*temp_directory*\ 是具有足夠可用空間可供 JAR 檔使用的暫存目錄， *jar_file_name*_1.jar 則是您的 JAR 檔名稱後面加上 _1。

5. 選取 JAR 檔的屬性，方式如下：

屬性	**值**
類別	勾選
java	不勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

在其它屬性方面則接受預設值。

6. 按一下**完成**。

由於所建的施行 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中導覽至您在先前步驟中儲存 *jar_file_name*_1.jar 的目錄。
2. 輸入 mkdir temp1。
3. 輸入 cd temp1。
4. 按如下設定路徑：
   set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio4\bin;
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 jar xvf ../*jar_file_name*_1.jar。
6. 輸入 jar cvf ../*jar_file_name*.jar * （請注意，已移除名稱中的 _1）。
7. 切換至 *drive*:\WebSphere\CommerceDev*temp_directory* 目錄。
8. 如果您要部署到本端 WebSphere Commerce 案例中，請將 *jar_file_name*.jar 複製到下列目錄中：

   drive:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib

   否則，請保留暫存目錄中的 JAR 檔，直到您要將所有的檔案資產轉送到目標 WebSphere Commerce Server 中爲止。

為新實體 Bean 建立 JAR 檔

在建立新的實體 Bean 時，您必須將程式碼儲存在有別於所有 WebSphere Commerce 程式碼的專案中。此外，亦必須與您的任何自訂指令以及資料 Bean 程式碼分開。新實體 Bean 必須建於有別於內含 WebSphere Commerce 公用實體 Bean 之 EJB 群組的 EJB 群組中。

部署新的實體 Bean 時，必須建立兩個 JAR 檔。第一個 JAR 檔爲 EJB 1.1 匯出 JAR，它是以您在選取 EJB 標籤時使用當時的可用工具建立而成的。第二個 JAR 檔含有實體 Bean 的施行程式碼，而且是以內含實體 Bean 程式碼的專案建立而成。此第二個 JAR 檔是您在 VisualAge for Java 工作台中選取「專案」標籤時使用當時的可用工具建立而成的。

建立 EJB 1.1 匯出 JAR 檔

如果要爲新實體 Bean 建立 EJB 1.1 匯出 JAR 檔，請執行下列步驟：

1. 在 VisualAge for Java 的「工作台」中，選取 **EJB** 標籤。
2. 以滑鼠右鍵按一下內含自訂實體 Bean 程式碼的 EJB 群組，然後選取**匯出 > EJB 1.1 JAR**。
   這時會開啓「匯出至 EJB 1.1 JAR 檔」引導精靈。
3. 在 **Jar 檔**欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev\temp_directory*jarFileName*_DT.jar
   其中 *drive*:\WebSphere\CommerceServerDev\temp_directory 爲具有足夠可用空間可供 JAR 檔使用的暫存目錄，而 *jarFileName*_DT.jar 爲您的 JAR 檔名稱後面加上字尾 _DT。

JAR 檔名稱中已加上 "_DT" 字尾，如此可提醒您在將之部署到 WebSphere Commerce 應用程式前，必須透過 WebSphere Application Server 所提供的「EJB 部署工具」執行此 JAR 檔。

4. 選取 JAR 檔的屬性，方式如下：

屬性	值
類別	勾選
java	不勾選
資源	勾選

屬性	值
目標資料庫	DB2 如果您要部署到 DB2 資料庫，請選取下列之一： • Windows AIX Solaris Linux **DB2 for NT, V7.1** • 400 **DB2 for AS/400, V4** Oracle 如果您要部署到 Oracle 資料庫，請選取 **Oracle, V8**
在 .class 檔中納入除錯屬性	勾選

在其它屬性方面則接受預設值。

5. 按一下**完成**。

此時會建立 JAR 檔。

建立施行 JAR 檔

如果要爲新的實體 Bean 建立施行 JAR 檔，請執行下列步驟：

1. 在 VisualAge for Java 的「工作台」中，選取**專案**標籤。
2. 以滑鼠右鍵按一下內含自訂實體 Bean 程式碼的專案，然後選取**匯出**。這時會開啓「匯出」引導精靈。
3. 選取 **Jar 檔**，然後按**下一步**。
4. 在 **Jar 檔**欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev*temp_directory**jarFileName*_1.jar
   其中 *drive*:\WebSphere\CommerceServerDev*temp_directory* 爲具有足夠可用空間可供 JAR 檔使用的暫存目錄， *jarFileName*_1.jar 爲您 JAR 檔名稱後面加上 _1。
5. 選取 JAR 檔的屬性，方式如下：

屬性	值
類別	勾選
java	不勾選
資源	勾選
Bean	勾選
在 .class 檔中納入除錯屬性	勾選

在其它屬性方面則接受預設值。

6. 按一下**完成**。

由於所建的施行 JAR 檔中未含完整的套件命名資訊，您必須使用另一種套裝公用程式（VisualAge for Java 以外）來重新套裝 JAR 檔。如果要重新套裝此檔案，請執行下列步驟：

1. 在指令視窗中導覽至您在先前步驟中儲存 *jarFileName*_1.jar 的目錄。
2. 輸入 mkdir temp1。
3. 輸入 cd temp1。
4. 按如下設定路徑：
   set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio4\bin;
   其中 *drive* 爲安裝 WebSphere Studio 的磁碟機。
5. 輸入 jar xvf ../*jarFileName*_1.jar。
6. 輸入 jar cvf ../*jarFileName*.jar * （請注意，已移除名稱中的 _1）。
7. 切換至 *drive*:\WebSphere\CommerceServerDev*temp_directory* 目錄。
8. 如果您要部署到本端 WebSphere Commerce 案例中，請將 *jarFileName*_1.jar 複製到下列目錄中：
   drive:\WebSphere\CommerceServer\temp\lib
   否則，請保留暫時目錄中的 JAR 檔，直到您要將所有的檔案資產轉送到目標 WebSphere Commerce Server 中爲止。

為自訂 WebSphere Commerce 實體 Bean 建立 JAR 檔

系統容許您延伸任何公用的 WebSphere Commerce 實體 Bean。這些 Bean 可在下列 EJB 群組中找到：

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder

- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

如果您延伸公用 WebSphere Commerce 實體 Bean，則您必須針對內含已修改 WebSphere Commerce 公用實體 Bean 的 EJB 群組建立一個 EJB 1.1 匯出 JAR 檔。

建立 EJB 1.1 匯出 JAR 檔

如果要爲內含已修改實體 Bean 的 EJB 群組建立 EJB 1.1 匯出 JAR 檔，請執行下列步驟：

1. 在 VisualAge for Java 的「工作台」中，選取 **EJB** 標籤。
2. 以滑鼠右鍵按一下內含自訂實體 Bean 程式碼的 EJB 群組，然後選取**匯出 > EJB 1.1 JAR**。
   這時會開啓「匯出至 EJB 1.1 JAR 檔」引導精靈。
3. 在 **Jar 檔**欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev*temp_directory*\Cust_*EJBGroupName*-ejb_DT.jar
   其中 *drive*:\WebSphere\CommerceServerDev*temp_directory* 爲具有足夠可用空間可供 JAR 檔使用的暫存目錄，Cust_*EJBGroupName*-ejb_DT.jar 爲您的 JAR 檔名稱後面加上字尾 _DT。

JAR 檔名稱中已加上 "_DT" 字尾，如此可提醒您在將之部署到 WebSphere Commerce 應用程式前，必須透過 WebSphere Application Server 所提供的「EJB 部署工具」執行此 JAR 檔。

4. 選取 JAR 檔的屬性，方式如下：

屬性	值
類別	勾選
java	不勾選
資源	勾選

屬性	值
目標資料庫	DB2 如果您要部署到 DB2 資料庫，請選取下列之一： • Windows AIX Solaris Linux **DB2 for NT, V7.1** • 400 **DB2 for AS/400, V4** Oracle 如果您要部署到 Oracle 資料庫，請選取 **Oracle, V8**
在 .class 檔中納入除錯屬性	勾選

在其它屬性方面則接受預設值。

5. 按一下**完成**。

建立從屬站 JAR 檔

如果要建立從屬站 JAR 檔，請執行下列步驟：

1. 先選取 EJB 標籤，然後標示出所有 WebSphere Commerce EJB 群組（其名稱的開頭爲 WCS）。在這些群組皆反白的情況下，以滑鼠右鍵按一下並選取**匯出 > 從屬站 JAR**。
   這會開啓「匯出」引導精靈。
2. 在 **JAR 檔**欄位中輸入下列：
   drive:\WebSphere\CommerceServerDev*temp_directory*\wcsejsclient.jar
3. 按如下選取屬性：

屬性	值
Bean	勾選
類別	勾選
java	不勾選
資源	勾選
在 .class 檔中納入除錯屬性	勾選

在其它屬性方面則接受預設值。

4. 按一下**完成**。

此時會建立 JAR 檔。

將資產儲存到目標 WebSphere Commerce Server

與自訂程式碼相關的資產必須複製到目標 WebSphere Commerce Server 上。這些資產包括自訂指令、資料 Bean 和實體 Bean 的 JAR 檔。您可能會有支援自訂作業的新 JSP 範本和圖形。

AIX Solaris Linux 您應使用您在執行 *WebSphere Commerce 安裝手冊之 "執行後置安裝 Script"* 一節中的步驟時所建的使用者，在目標 WebSphere Commerce Server 上執行所有部署步驟。依預設，此使用者爲 wasuser。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。

400 確定 /QIBM/UserData/WebCommerce/instances/*instanceName* 與 /QIBM/UserData/WebCommerce/instances/*instanceName*/working 目錄的權限已包含使用者 QEJB。請在這兩個目錄中新增這個使用者，並將資料權限設爲 *RWX。

下表摘要說明在執行 Windows NT 或 Windows 2000 的目標 WebSphere Commerce Server 中，儲存資產的標準目錄。

表 12.

資產類型	目標 WebSphere Commerce Server 中的目錄位置
指令與資料 Bean 邏輯的 JAR 檔	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
使用 VisualAge for Java 建立 EJB 1.1 匯出 JAR	*drive*:\WebSphere\CommerceServer\temp **註:** 此檔案會被當成輸入傳給 EJBDeploy 工具，以產生可用於 WebSphere Application Server V4.0 中的部署程式碼。
EJB 從屬站程式碼的 JAR 檔	*drive*:\WebSphere\CommerceServer\temp\lib **註:** 此檔案僅用於 EJBDeploy 工具的類別路徑中。
EJB 施行程式碼的 JAR 檔	*drive*:\WebSphere\CommerceServer\temp\lib **註:** 此檔案會當成檔案資產新增到企業應用程式中。
JSP 範本	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*storeDir*
影像	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*storeDir*\images

如果要將資產儲存在目標 WebSphere Commerce Server 中，請執行下列步驟：

1. 找出您指令與資料 Bean 程式碼的 JAR 檔。這些應位於開發機器上的下列目錄之一：

- Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
- Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*

2. 將您指令與資料 Bean 程式碼的 JAR 檔複製到目標 WebSphere Commerce Server 的下列其中一個目錄中：
   - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/WEB-INF/lib
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/WEB-INF/lib
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/WEB-INF/lib
   - 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/WEB-INF/lib
3. 在開發機器的下列目錄中，找出任何新 EJB 群組的 EJB 1.1 匯出 JAR 檔以及任何已修改的 WebSphere Commerce 實體 Bean：

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*
4. 將 EJB 1.1 匯出 JAR 檔複製到目標 WebSphere Commerce Server 的下列其中一個目錄中：
   - Windows *drive*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - Linux /opt/WebSphere/CommerceServer/temp
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/temp
5. 如果您已經建立好新的 Enterprise Bean，請在開發機器的下列目錄中，找出這些 Bean 的施行 JAR 檔：

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*
6. 將新 Enterprise Bean 的施行 JAR 檔複製到目標 WebSphere Commerce Server 的下列目錄中：

- Windows *drive*:\WebSphere\CommerceServer\temp\lib
- AIX /usr/WebSphere/CommerceServer/temp/lib
- Solaris /opt/WebSphere/CommerceServer/temp/lib
- Linux /opt/WebSphere/CommerceServer/temp/lib
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/ temp/lib

7. 如果您修改了現有 WebSphere Commerce 實體 Bean，請在開發機器的下列目錄中，找出從屬站 JAR 檔：

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*

8. 將從屬站 JAR 檔複製到目標 WebSphere Commerce Server 的下列其中一個目錄中：
   - Windows *drive*:\WebSphere\CommerceServer\temp\lib
   - AIX /usr/WebSphere/CommerceServer/temp/lib
   - Solaris /opt/WebSphere/CommerceServer/temp/lib
   - Linux /opt/WebSphere/CommerceServer/temp/lib
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/ temp/lib
9. 將所有的 JSP 範本複製到目標 WebSphere Commerce Server 的下列目錄中：
   - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*store_directory*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*
   - 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*

   其中 *store_directory* 是商店的目錄。
10. 將所有的影像複製到目標 WebSphere Commerce Server 的下列目錄中：

- Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*store_directory*\images
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images

其中 *store_directory* 是商店的目錄。

更新目標資料庫

如果您的目標 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所有更新。這包括：新登錄或已修改指令的任何更新，您所建之其它表格的更新，以及針對任何您所建之任何新資源所建立之存取控制原則的更新。

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）的進一步資訊，請參閱 *WebSphere Commerce 存取控制手冊*。

400 您需有一個公用程式以執行 SQL 陳述式。其做法之一是使用 Client Access Express V5R1（完整安裝）。如果要開啓此公用程式，請執行下列步驟：

1. 開啓**作業導引器**。
2. 在開啓「作業導引器」後，會要求您簽入到特定系統中。請確定您已選取目標 iSeries 機器，並使用 WebSphere Commerce 案例使用者設定檔與密碼。這可確保 WebSphere Commerce 案例使用者設定檔擁有您所建的所有新表格。
3. 在左畫面中按一下您的系統，然後以滑鼠右鍵按一下 **DATABASE**，並從下拉清單中選取**執行 SQL Script**。

這會開啓「執行 SQL Script」視窗。在此視窗中，您可以在 SQL 陳述式中剪貼，或者開啓 SQL Script。您可以使用「連線/JDBC 設定」選項，來設定預設綱目。

產生部署程式碼

本節說明如何使用 EJB 部署工具，爲 EJB 1.1 匯出 JAR 檔中所含的 Enterprise Bean 產生部署程式碼。此工具是由 WebSphere Application Server 提供。

如果要產生部署的程式碼，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 的指令提示下，切換至下列目錄：
   - Windows *drive*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - Linux /opt/WebSphere/CommerceServer/temp
2. 400 在目標 WebSphere Commerce Server 的指令提示下，輸入下列指令：

   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/instanceName/temp
   ```
3. 輸入下列指令，暫時將工具新增到系統路徑中：
   - Windows PATH=*drive*:\WebSphere\AppServer\deploytool;%PATH%
   - AIX PATH=/usr/WebSphere/AppServer/deploytool:$PATH
   - Solaris PATH=/opt/WebSphere/AppServer/deploytool:$PATH
   - Linux PATH=/opt/WebSphere/AppServer/deploytool:$PATH
   - 400 PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH

     CP=*ClassPathOfDepJARFiles*

     其中 *ClassPathOfDepJARFiles* 爲任何相依 JAR 檔的類別路徑。請注意，如果您曾修改現有 WebSphere Commerce Enterprise Bean，則相依 JAR 檔的類別路徑中必須包含 wcsejsclient.jar、wcsejbimpl.jar 與 xml4j.jar 檔。在下列的範例類別路徑中，現有 WebSphere Commerce Enterprise Bean 已修改：

     ```
     CP=/QIBM/UserData/WebCommerce/instances/instanceName/temp/lib/
     wcsejsclient.jar:
     /QIBM/UserData/WebASAdv4/WAS_AdminInstanceName/installedApps/
     ```

```
WC_Enterprise_App_instanceName.ear/lib/wcsejbimpl.jar:
/QIBM/UserData/WebASAdv4/WAS_AdminInstanceName/installedApps/
WC_Enterprise_App_instanceName.ear/lib/xml4j.jar
```

註: 上述類別路徑範例中的折行只爲了方便閱讀。您應以單行輸入類別路徑資訊。

4. Solaris Linux 爲了避免超過指令行介面中的字元限制，可使用 Script 來呼叫 ejbdeploy 指令。請執行下列步驟，以建立此 Script 並呼叫指令：
   a. 導覽至下列目錄：

      ```
      /opt/WebSphere/AppServer/bin
      ```

   b. 建立一個新檔案，並取一個適合 Script 的名稱。例如，將檔案命名爲 myejbd.sh。
   c. 在 myejbd.sh 檔中包含下列資訊：

      ```
      #!/bin/sh
      ./ejbdeploy.sh EJBGroupJARFile WorkingDir OutputJARFile
      -nowarn -keep -35 -cp ClassPathOfDepJARFiles
      ```

      其中，變數的定義和步驟 5（註：您未執行該步驟）中的定義相同。下列範例將顯示 Script 檔中的內容，並包含變數的值：

      ```
      #!/bin/sh
      ./ejbdeploy.sh
      /opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed_DT.jar
      . /opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed.jar
      -nowarn -keep -35 -cp "/opt/WebSphere/CommerceServer/temp/lib/
      wcsejsclient.jar:/opt/WebSphere/AppServer/installedApps/
      WC_Enterprise_App_demo.ear/lib/wcsejbimpl.jar:/opt/WebSphere/
      AppServer/installedApps/WC_Enterprise_App_demo.ear/lib/xml4j.jar"
      ```

      註: ./ejbdeploy.sh 指令後的所有折行只爲了方便閱讀。在您的檔案中，./ejbdeploy.sh 指令後面不應折行。

      儲存 Script 並讓其變成可執行檔。
   d. 輸入下列指令以呼叫 Script：

      ```
      ./myejbd.sh
      ```

5. Windows AIX 400 依下列所示輸入 ejbdeploy 指令：

   Windows 400

   ```
   ejbdeploy EJBGroupJARFile_DT WorkingDir
    OutputJARFile -nowarn -keep -35 -cp
      ClassPathOfDepJARFiles
   ```

   AIX

```
/usr/WebSphere/AppServer/bin/ejbdeploy.sh EJBGroupJARFile_DT WorkingDir
OutputJARFile -nowarn -keep -35
-cp ClassPathOfDepJARFiles
```

其中：

- *EJBGroupJARFile*_DT 爲您 EJB 群組的 JAR 檔名稱。舉例來說，如果您修改了 WCSUser EJB 群組中的 Bean，在 Windows 平台上的範例用法爲：

  ```
  drive:\WebSphere\CommerceServer\temp\CustomizedWCSUser_DT.jar
  ```

  400 在 iSeries 平台上的範例用法爲：

  ```
  /QIBM/UserData/WebCommerce/instances/instanceName/temp/
  CustomizedWCSUser_DT.jar
  ```

- *WorkingDir* 爲目錄名稱，用來儲存產生程式碼時需使用的暫存檔。
- *OutputJARFile* 爲輸出 JAR 檔的完整名稱。例如，您可輸入 CustomizedWCSUserDeployed.jar。
- -nowarn 爲選用參數，用以抑制警告與參考訊息的出現。
- -keep 爲選用參數，用以在執行 ejbdeploy 指令後保留工作目錄。
- -35 爲必要參數，表示將針對 WebSphere Application Server 3.5 版所提供之「EJB 部署工具」中所用的 CMP 實體 Bean，採用相同的由上往下映射規則。
- -cp *ClassPathOfDepJARFiles* 爲任何相依 JAR 檔的類別路徑。如果您曾修改現有 WebSphere Commerce Enterprise Bean，則相依 JAR 檔的類別路徑中必須包含 wcsejsclient.jar、wcsejbimpl.jar 與 xml4j.jar 檔。在下列的範例類別路徑中，現有 WebSphere Commerce Enterprise Bean 已修改：

  ```
  "drive:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
  drive:\WebSphere\AppServer\installedApps\
  WC_Enterprise_App_instanceName.ear \lib\wcsejbimpl.jar;
  drive:\WebSphere\AppServer\installedApps\
  WC_Enterprise_App_instanceName.ear\lib\xml4j.jar;"
  ```

  註： 400 在類別路徑值方面，請輸入 -cp $CP；其中 CP 則是先前您以適當類別路徑項目定義而成。此外，不一定要括上引號。

修改實體 Bean 的交易隔離層次

本節說明如何使用 modifyIsolationLevel 指令行公用程式，將您實體 Bean 的交易隔離層次設爲適合您資料庫類型的層次。

如果要執行 modifyIsolationLevel 指令，請執行下列步驟：

1. 在目標 WebSphere Commerce Server 中，使用指令提示導覽至下列目錄：

- Windows *drive*:\WebSphere\CommerceServer\bin
- AIX /usr/WebSphere/CommerceServer/bin
- Solaris /opt/WebSphere/CommerceServer/bin
- Linux /opt/WebSphere/CommerceServer/bin

2. 400 請輸入下列指令：

```
STRQSH
cd /QIBM/ProdData/WebCommerce/bin
```

3. 您必須發出採用下列一般語法的 modifyIsolationLevel 指令：

Windows AIX 400

```
modifyIsolationLevel -jarFile jar_file_name.jar
   -logFile log_file_name -dbType db_type
```

Solaris Linux

```
./modifyIsolationLevel.sh -jarFile jar_file_name.jar
   -logFile log_file_name -dbType db_type
```

其中

- *jar_file_name.jar* 爲內含自訂程式碼的 JAR 檔名稱
- *log_file_name* 爲應記載資訊的完整檔案名稱
- *db_type* 爲您所用的資料庫類型。輸入 DB2 或 ORACLE

以下是用於 Windows 平台上，並指定所有值的 modifyIsolationLevel 指令範例：

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

400 以下是指定所有值用於 iSeries 平台上的 modifyIsolationLevel 指令範例：

```
modifyIsolationLevel -jarFile
   /QIBM/UserData/WebCommerce/instances/instanceName/temp/
   CustomizedWCSUserDeployed.jar -logFile /QIBM/UserData/WebCommerce/
   instances/instanceName/logs/output.log -dbType DB2
```

請注意，在上述範例中，折行只爲了方便閱讀。

註: 參數名稱有區分大小寫。亦即，`jarFile` 不等於 `jarfile`。請確定您所輸入的參數名稱無誤。

如果指令視窗中沒有顯示異常狀況，表示指令執行成功。在完成後，請記下您部署 JAR 檔變更的時間戳記。

匯出現行的 WebSphere Commerce 企業應用程式

本節說明如何使用 WebSphere Application Server 管理主控台來匯出現行的 WebSphere Commerce 企業應用程式。

如果要從 WebSphere Application Server 匯出現行企業應用程式，請執行下列步驟：

1. 請確定目標 WebSphere Commerce Server 中已存在下列目錄：
   - Windows `drive:\WebSphere\CommerceServer\working`
   - AIX `/usr/WebSphere/CommerceServer/working`
   - Solaris `/opt/WebSphere/CommerceServer/working`
   - Linux `/opt/WebSphere/CommerceServer/working`
   - 400 `/QIBM/UserData/WebCommerce/instances/instanceName/working`

   如果沒有，請立即建立。

   註: AIX Solaris Linux 確定 `/working` 目錄的許可權已設定給您在執行 *WebSphere Commerce 安裝手冊*之"*執行後置安裝 Script*" 節的步驟時所建立的使用者。

   400 確定 `/QIBM/UserData/WebCommerce/instances/instanceName` 與 `/QIBM/UserData/WebCommerce/instances/instanceName/working` 目錄的權限已包含使用者 `QEJB`。請在這兩個目錄中新增這個使用者，並將資料權限設為 `*RWX`。
2. 開啓 WebSphere Application Server 管理主控台。
3. 展開 **WebSphere 管理網域**。
4. 展開**企業應用程式**。
5. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，展開**示範**應用程式，並選取**匯出應用程式**。
6. 在**匯出目錄**欄位中輸入下列：
   - Windows `drive:\WebSphere\CommerceServer\working`
   - AIX `/usr/WebSphere/CommerceServer/working`

- Solaris /opt/WebSphere/CommerceServer/working
- Linux /opt/WebSphere/CommerceServer/working
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working

這會將整個應用程式（包括所有資源）匯出到 WC_Enterprise_App_*instanceName*.ear 檔中（其中 *instanceName* 爲您 WebSphere Commerce 案例的名稱）。

匯出 Enterprise Bean 的架構資訊

本節說明如何使用 XMLConfig 指令行公用程式的 -export 選項，來匯出現有企業應用程式中之 Enterprise Bean 的架構資訊。

在您匯出現有應用程式中之 Bean 的資訊後，您必須手動加上您要新增至應用程式中之任何新 Bean 的資訊。

如果要匯出此架構資訊，請執行下列步驟：

1. 將 was.export.app.xml 檔從目標 WebSphere Commerce Server 的下列目錄：
   - Windows *drive*:\WebSphere\CommerceServer\xml\config
   - AIX /usr/WebSphere/CommerceServer/xml/config
   - Solaris /opt/WebSphere/CommerceServer/xml/config
   - Linux /opt/WebSphere/CommerceServer/xml/config
   - 400 /QIBM/ProdData/WebCommerce/xml/config

   複製到目標 WebSphere Commerce Server 的下列目錄中：
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working

     其中
     - *instanceName* 爲 WebSphere Commerce 案例的名稱。

2. Windows AIX Solaris Linux 以文字編輯器開啓 was.export.app.xml 檔。在此檔案中，將所有 $Enterprise_Application_Name$ 換成

   WebSphere Commerce Enterprise Application - *instanceName*

   其中 *instanceName* 爲 WebSphere Commerce 案例的名稱（例如 demo）。儲存此檔案。

   註: 您所插入的值必須符合 WebSphere Application Server 管理主控台中所示的案例資訊。

3. 400 以文字編輯器開啓 was.export.app.xml 檔。在此檔案中，將所有 $Enterprise_Application_Name$ 換成

   instanceName - WebSphere Commerce 企業應用程式

   其中 *instanceName* 爲 WebSphere Commerce 案例的名稱（例如 demo）。儲存此檔案。

   註: 您所插入的值必須符合 WebSphere Application Server 管理主控台中所示的案例資訊。

4. Windows AIX Solaris Linux 在指令提示下導覽至下列目錄：

   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working

5. 400 在指令提示下輸入下列：

   ```
   STRQSH
   PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
   cd /QIBM/UserData/WebCommerce/instances/instanceName/working
   ```

6. Windows AIX Solaris Linux 輸入下列指令，以呼叫 XMLConfig 工具來執行部份匯出：

   Windows

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
      -adminNodeName wasHostName
   ```

   AIX

```
/usr/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

Solaris Linux

```
/opt/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

其中

- *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。
- OutputFile.xml 爲因執行此指令而建立的檔案， was.export.app.xml 爲您在步驟 2 中所修改的檔案。
- *wasAdminPort* 爲 WebSphere Application Server 管理埠。

7. 400 輸入下列指令，呼叫 XMLConfig 工具以執行局部匯出：

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
-adminNodeName wasHostName -nameServiceHost wasHostName
-nameServicePort wasAdminPort -instance wasInstanceName
```

其中

- *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。

  註: *wasHostName* 的值有區分大小寫，且必須和 TCP/IP 架構中的值相符。（請使用指令行處理器來存取 CFGTCP，選項 12，以驗證主電腦名稱。）
- OutputFile.xml 爲因執行此指令而建立的檔案， was.export.app.xml 爲您在步驟 3 中所修改的檔案。
- *wasAdminPort* 爲 WebSphere Application Server 管理埠。
- *wasInstanceName* 爲 WebSphere Application Server 案例名稱。

在您匯出現行企業應用程式中之每一個 Bean 的架構資訊後，您必須加上一個新段落，以說明您要新增至應用程式中的每一個 Enterprise Bean。舉例來說，如果您有一個新實體 Bean “Bonus”，您必須新增一個專門說明此 Bonus Bean 的段落。此外，您必須更換架構檔中的變數，以指定您企業應用程式確切的 .ear 檔名。

如果要更新 OutputFile.xml 檔，請執行下列步驟：

1. 導覽至目標 WebSphere Commerce Server 的下列目錄中：
   - Windows *drive*:\WebSphere\CommerceServer\working

- AIX /usr/WebSphere/CommerceServer/working
- Solaris /opt/WebSphere/CommerceServer/working
- Linux /opt/WebSphere/CommerceServer/working
- 400 /QIBM/UserData/WebCommerce/instances/ *instanceName*/working

2. 以文字編輯器開啓 OutputFile.xml 檔。
3. 找出 <ear-file-name> 標籤，並將值換成：
   - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear
   - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
   - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
   - Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working/WC_Enterprise_App_*instanceName*.ear
4. 您也必須針對您要新增至企業應用程式中的每一個新 Bean，分別加入一個新段落。下列範例顯示如何新增新 Bean “Bonus”.

   Windows AIX Solaris Linux

```
<ejb-module name="yourEJBGroup">
   <jar-file>yourDeployedJarFile.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - instanceName/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource instanceName
   </jndi-name>
         <default-user>user</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="BeanBindingName">
         <jndi-name>instanceNameJNDINameOfBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

```
<ejb-module name="yourEJBGroup">
   <jar-file>yourDeployedJarFile.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         instanceName - WebSphere Commerce Server/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/instanceName WebSphere Commerce DB2 DataSource
   </jndi-name>
         <default-user>user</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="BeanBindingName">
         <jndi-name>instanceNameJNDINameOfBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

其中

- *yourEJBGroup* 爲內含您要新增到企業應用程式中之 Bean 的 EJB 群組名稱。
- *yourDeployedJarFile* 爲內含 EJB 群組之部署程式碼的 JAR 檔名。
- *instanceName* 爲您要部署程式碼的 WebSphere Commerce 案例名稱。
- *user* 爲您資料庫使用者名稱。
- *password* 爲您資料庫使用者的密碼。
- *BeanBindingName* 爲您 Enterprise Bean 的連結名稱。以 Bonus Bean 爲例，此爲 Bonus_Binding。
- *instanceNameJNDINameOfBean* 爲前頭加上 WebSphere Commerce 案例的 Enterprise Bean JNDI 名稱。 此 JNDI 名稱必須完全符合 Enterprise Bean 的名稱。例如，其值可爲 democom/ibm/commerce/sample/objects/Bonus。在本例中，“demo” 爲案例名稱，“com/ibm/commerce/sample/objects/Bonus” 爲 Enterprise Bean 的 JNDI 名稱。此值必須輸入成單行。
  您可以使用 VisualAge for Java 或應用程式組譯工具來驗證 JNDI 名稱。如果要使用 VisualAge for Java 來驗證 JNDI 名稱，請執行下列步驟：
  a. 以滑鼠右鍵按一下 Bean，並選取**內容**。
     會顯示 JNDI 名稱。

您可使用應用程式組譯工具來驗證 JNDI 名稱，不過，這會要求您已將新 Enterprise Bean 組譯到企業應用程式中。應用程式組譯工具可從 WebSphere Application Server 管理主控台的「工具」功能表中存取到。

如果要使用應用程式組譯工具來驗證 JNDI 名稱，請執行下列步驟：

a. 開啓內含 Bean 的 .ear 檔。

b. 展開內含 Bean 的 EJB 模組。

c. 選取 Bean。

d. 按一下**連結**標籤。
會顯示 JNDI 名稱。

請注意，無論您使用哪種方法來顯示 JNDI 名稱，在您於 XML 檔中新增資訊時，仍必須在其前加上 WebSphere Commerce 案例名稱。

註:

a. 上述段落中的折行只爲了方便閱讀。

b. 確定 **$hostName$** 值符合現行管理節點伺服器名稱。此外，請確定此行中沒有回車字元。

c. `<application-server-full-name>` 規格必須以單行顯示。

d. 如果您使用 Oracle 資料庫，您必須修改資料來源資訊。請將下行（取自上述程式碼片段）：

```
<jndi-name>jdbc/WebSphere Commerce DB2 DataSource instanceName
   </jndi-name>
```

改爲：

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource instanceName
   </jndi-name>
```

5. 如果您要部署已修改的 WebSphere Commerce 實體 Bean，您必須更新這些 Bean 的段落，以反映內含已修改 Bean 之部署程式碼的 JAR 檔名稱。

6. 儲存 `OutputFile.xml` 檔。

將新 Enterprise Bean 組譯到企業應用程式中

本節說明如何使用應用程式組譯工具，將新 Enterprise Bean 組譯到現有企業應用程式中。

400 由於應用程式組譯工具是在 Windows 平台上執行，當提示您輸入完整路徑名稱時，您必須參照映射至您 iSeries IFS 的磁碟機。這可稱爲 *iSeries_drive*。

AIX Solaris Linux 建議您增加 assembly.sh 檔中的記憶體資料堆大小的值，以防在儲存新或已修改的 ear 檔時發生記憶體不足現象。

此檔位於下列目錄中：

- AIX /usr/WebSphere/AppServer/bin
- Solaris /opt/WebSphere/AppServer/bin
- Linux /opt/WebSphere/AppServer/bin

如果要增加記憶體資料堆大小，請修改下行：

```
$JAVA_HOME/jre/bin/java
```

使其成爲：

```
$JAVA_HOME/jre/bin/java -mx512M
```

在此步驟中，您將在應用程式組譯器工具中，開啓您在匯出現行的 WebSphere Commerce 企業應用程式一節中所建之企業應用程式的 .ear 檔。一旦您在該工具內開啓該檔案，請執行下列作業，將新實體 Bean 新增到企業應用程式中：

1. 匯入內含新實體 Bean 的 EJB 群組。新 EJB 群組的 JAR 檔儲將儲存在企業應用程式的「EJB 模組」區段中。
2. 設定新實體 Bean 的類別路徑，以包含施行 JAR 檔。
3. 將施行 JAR 檔新增到應用程式中。此 JAR 檔將儲存在企業應用程式的「檔案」區段中。
4. 爲新實體 Bean 中的方法設置 WebSphere Application Server 安全特性。

如果要將新 EJB 群組組譯到企業應用程式中，請執行下列步驟：

1. Windows AIX Solaris Linux 在目標 WebSphere Commerce Server 上執行下列步驟，備份現有的企業應用程式：
   a. 在指令提示下，切換至下列目錄：
      - Windows *drive*:\WebSphere\CommerceServer\working
      - AIX /usr/WebSphere/CommerceServer/working
      - Solaris /opt/WebSphere/CommerceServer/working
      - Linux /opt/WebSphere/CommerceServer/working
   b. 複製一份現有的 WC_Enterprise_App_*instanceName*.ear 檔，並命名爲 WC_Enterprise_App_*instanceName*.ear.bak。

2. 400 執行下列步驟，以備份現行企業應用程式：
   a. 在指令提示下輸入：

      ```
      STRQSH
      cd /QIBM/UserData/WebCommerce/instances/instanceName/working
      cp WC_Enterprise_App_instanceName.ear
      WC_Enterprise_App_instanceName.ear.bak
      ```

   b. 爲了避免因久候您本端從屬站機器與執行 WebSphere Application Server 之 iSeries 機器間的資料轉送而浪費時間，請在您本端從屬站機器上建立下列目錄，然後將 WC_Enterprise_App_*instanceName*.ear 檔複製到這個目錄中：

      drive:\WebSphere\CommerceServer\working

      其中 *drive* 爲本端磁碟機。

      註: 如果本端機器中沒有 *drive*:\WebSphere\CommerceServer\working 目錄，請立即建立。

3. 開啓 WebSphere Application Server 管理主控台。
4. 從**工具**功能表選取**應用程式組譯工具**。
5. 如果「歡迎使用」視窗開啓，請選取**取消**，以關閉該視窗。
6. 執行下列步驟，以開啓您要工作的企業應用程式：
   a. 從**檔案**功能表中，選取**開啓**。
   b. 在**檔案名稱**欄位中輸入：
      - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear
      - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - 400 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear

      然後按一下**開啓**。等待應用程式開啓，再繼續進行後續步驟。需要等待幾分鐘。
7. 以滑鼠右鍵按一下 **EJB 模組**，並選取**匯入**。
8. 在**檔案名稱**欄位中輸入：

- Windows *drive*:\WebSphere\CommerceServer\temp\ *yourDeployedJarFile*.jar
- AIX /usr/WebSphere/CommerceServer/temp/ *yourDeployedJarFile*.jar
- Solaris /opt/WebSphere/CommerceServer/temp/ *yourDeployedJarFile*.jar
- Linux /opt/WebSphere/CommerceServer/temp/ *yourDeployedJarFile*.jar
- 400 *iSeries_drive*:\QIBM\UserData\WebCommerce\instances\ *instanceName*\temp*yourDeployedJarFile*.jar

其中

- *yourDeployedJarFile* 爲內含 EJB 群組之部署程式碼的 JAR 檔名
- *iSeries_drive* 爲映射至 iSeries IFS 的本端磁碟機。

請按一下**開啓**，然後在「確認值」視窗中按一下**確定**。

9. 一旦匯入 *yourDeployedJarFile*.jar 檔後，請捲動到 *yourEJBGroup* EJB 群組（其中 *yourEJBGroup* 爲您 EJB 群組的名稱），並選取此群組。
此群組的資訊會顯示在右窗格中。
10. 在新 Enterprise Bean 的類別路徑欄位中，輸入任何相依的 JAR 檔。舉例來說，您可依照如下輸入對應的施行 JAR 檔與 WebSphere Commerce 實體 Bean 的施行 JAR 檔：

    lib/*yourImplJarFile*.jar lib/wcsejbimpl.jar
11. 按一下**套用**。
12. 執行下列步驟，以新增您 EJB 群組的施行 JAR 檔到應用程式中：
    a. 以滑鼠右鍵按一下企業應用程式的**檔案**節點，並選取**新增檔案**。（企業應用程式的**檔案**節點較靠近階層樹狀結構底端。請注意，在企業應用程式中另有元件的其它「檔案」節點，但您必須選取整個應用程式的「檔案」節點。）
    b. 在「新增檔案」視窗中按一下**瀏覽**。
    c. 導覽至下列目錄：
       - Windows *drive*:\WebSphere\CommerceServer\temp
       - AIX /usr/WebSphere/CommerceServer/temp
       - Solaris /opt/WebSphere/CommerceServer/temp
       - Linux /opt/WebSphere/CommerceServer/temp
       - 400 *iSeries_drive*:\QIBM\UserData\WebCommerce\instances\ *instanceName*\temp

d. 標示出此目錄，並按一下**選取**。

e. 回到「新增檔案」視窗。請注意，會顯示暫時目錄的內容。請標示 `lib` 目錄。

`lib` 目錄的內容會出現在右窗格中。

f. 在右窗格中選取 *`yourImplJarFile`* 檔，並按一下**新增**。檔案會出現在「所選檔案」中。

g. 按一下**確定**。

13. 執行下列步驟，以架構您實體 Bean 的安全特性：

a. 展開「EJB 模組」節點，找出 *YourEJBGroup* 節點並展開。

b. 展開**實體 Bean**。

c. 展開 *yourEntityBean*，其中 *`yourEntityBean`* 爲您實體 Bean 的名稱。

d. 按一下**方法延伸**，然後在右窗格中執行下列步驟：

1) 按一下**進階**標籤。

2) 確定**安全身份**已選取。

3) 針對每一個方法，確定**使用 EJB 伺服器的身份**已選取。

4) 按一下**套用**（如果您有做任何修改的話）。

e. 在左導覽窗格中，以滑鼠右鍵按一下 *YourEJBGroup* EJB 群組下的**安全職務**，並選取**新建**，然後執行下列步驟：

1) 在**名稱**欄位中，輸入 `WCSecurityRole`，並按一下**套用**。請注意，如果此職務已存在，便不需執行此步驟。

f. 在左導覽窗格中，以滑鼠右鍵按一下 *`YourEJBGroup`* EJB 群組下的**方法許可權**，並選取**新建**，然後執行下列步驟：

1) 在**方法許可權名稱**欄位中輸入 `WCMethodPermission`。

2) 在**方法**選擇區中按一下**新增**。
這會開啓「新增方法」視窗。

3) 依序展開 *yourDeployedJarFile*.**jar**，**Bonus** 以及方法中的每一個**起源**與**遠端**清單。

4) 按住 Shift 鍵，並選取每一個起源方法，並按一下**確定**。

5) 重複方法選擇程序，以新增遠端方法（如果有遠端方法的話）。

6) 在「職務」選擇區中，按一下**新增**，並選取 `WCSecurityRole`，然後按一下**確定**。

7) 按一下**套用**。

14. 從**檔案**功能表中，選取**儲存**。

15. 關閉應用程式組譯工具。

16. 400 將新修改的 WC_Enterprise_App_*instanceName*.ear 檔從本端機器複製到執行 WebSphere Application Server 的 iSeries 機器中。亦即，複製下列目錄中的檔案：

    drive:\WebSphere\CommerceServer\working

    複錄到下列目錄中：

    iSeries_drive:\QIBM\UserData\WebCommerce\instances*instanceName*\working

    其中 *iSeries_drive* 爲您映射到至 iSeries IFS 的磁碟機字母。

在完成此步驟後，您便已建立一個新企業應用程式，且其含有先前的所有邏輯以及您的新商業邏輯。這全含在新修改的 WC_Enterprise_App_*instanceName*.ear 檔中。

將已修改的 Enterprise Bean 組譯到企業應用程式中

本節說明如何使用應用程式組譯工具，將已修改的 WebSphere Commerce Enterprise Bean 組譯到企業應用程式中。

在此步驟中，您將在應用程式組譯器工具中開啓您的企業應用程式。一旦在該工具內開啓後，您可執行下列步驟，將已修改的 WebSphere Commerce Enterprise Bean 納入到企業應用程式中：

1. 複製一份您已修改之 EJB 群組現有版本的類別路徑。
2. 移除您已修改之 EJB 群組的現有版本。
3. 匯入已修改的新 EJB 群組版本。新 EJB 群組的 JAR 檔儲存在企業應用程式的「EJB 模組」區段中。
4. 設定已修改之 EJB 群組的類別路徑。
5. 爲已修改之實體 Bean 中的方法設置 WebSphere Application Server 安全特性。

AIX Solaris Linux 建議您增加 assembly.sh 檔中的記憶體資料堆大小的值，以防在儲存新或已修改的 ear 檔時發生記憶體不足現象。

此檔位於下列目錄中：

- AIX /usr/WebSphere/AppServer/bin
- Solaris /opt/WebSphere/AppServer/bin
- Linux /opt/WebSphere/AppServer/bin

如果要增加記憶體資料堆大小，請修改下行：

```
$JAVA_HOME/jre/bin/java
```

使其成爲：

```
$JAVA_HOME/jre/bin/java -mx512M
```

如果要將已修改之 EJB 群組組譯到企業應用程式中，請執行下列步驟：

1. Windows AIX Solaris Linux 在目標 WebSphere Commerce Server 上執行下列步驟，備份現有的企業應用程式：
   a. 在指令提示下，切換至下列目錄：
      - Windows *drive*:\WebSphere\CommerceServer\working
      - AIX /usr/WebSphere/CommerceServer/working
      - Solaris /opt/WebSphere/CommerceServer/working
      - Linux /opt/WebSphere/CommerceServer/working

   b. 複製一份現有的 WC_Enterprise_App_*instanceName*.ear 檔，並命名爲 WC_Enterprise_App_*instanceName*.ear.bak。
2. 400 執行下列步驟，以備份現行企業應用程式：
   a. 在指令提示下輸入：

      ```
      STRQSH
      cd /QIBM/UserData/WebCommerce/instances/instanceName/working
      cp WC_Enterprise_App_instanceName.ear
      WC_Enterprise_App_instanceName.ear.bak
      ```

   b. 爲了避免因久候您本端從屬站機器與執行 WebSphere Application Server 之 iSeries 機器間的資料轉送而浪費時間，請在您本端從屬站機器上建立下列目錄，然後將 WC_Enterprise_App_*instanceName*.ear 檔複製到這個目錄

中：
drive:\WebSphere\CommerceServer\working
其中 *drive* 爲本端磁碟機。

註: 如果本端機器中沒有 *drive*:\WebSphere\CommerceServer\working 目錄，請立即建立。

3. 開啓 WebSphere Application Server 管理主控台。
4. 從**工具**功能表選取**應用程式組譯工具**。
5. 執行下列步驟，以開啓您要工作的企業應用程式：
   a. 從**檔案**功能表中，選取**開啓**。
   b. 在**檔案名稱**欄位中輸入：
      - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear
      - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - 400 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear

      然後按一下**開啓**。等待應用程式開啓，再繼續進行後續步驟。需要等待幾分鐘。
6. 按一下 **EJB 模組**。右窗格中會顯示企業應用程式中的 EJB 模組。
7. 按一下已修改之 EJB 群組的 EJB 模組。舉例來說，如果您修改了 WCSUser EJB 群組中的 Bean，請按一下 **WCSUser**。
8. 按一下「一般」標籤，以檢視類別路徑資訊。將此現行類別路徑資訊複製到文字檔中（例如 WCSUser_path.txt）。
9. 以滑鼠右鍵按一下 EJB 模組，並選取**刪除**。
10. 以滑鼠右鍵按一下 **EJB 模組**，並選取**匯入**。
11. 在**檔案名稱**欄位中輸入：
    - Windows *drive*:\WebSphere\CommerceServer\temp\Cust_*EJBGroupName*-ejb.jar

-
- AIX /usr/WebSphere/CommerceServer/temp/Cust_*EJBGroupName*-ejb.jar
- Solaris /opt/WebSphere/CommerceServer/temp/Cust_*EJBGroupName*-ejb.jar
- Linux /opt/WebSphere/CommerceServer/temp/Cust_*EJBGroupName*-ejb.jar
  400 *iSeries_drive*:\QIBM\UserData\WebCommerce\instances*instanceName*\temp\Cust_*EJBGroupName*-ejb.jar

然後按一下**開啓**。在「確認值」視窗中按一下**確定**。

12. 一旦匯入 *EJBGroupJARFile*.jar 檔後，請捲動到已修改的 EJB 群組，並選取此群組。
    此群組的資訊會顯示在右窗格中。
13. 開啓內含舊版 EJB 群組之類別路徑資訊的文字檔。選取並複製類別路徑。
14. 在已修改的 EJB 群組的**類別路徑**欄位中，貼上此類別路徑資訊。
15. 按一下**套用**。
16. 從**檔案**功能表中，選取**關閉**。
17. 等待檔案關閉，然後從**檔案**功能表中選取**開啓**，重新開啓 WC_Enterprise_App_*instanceName*.ear 檔。
18. 執行下列步驟，以架構已修改 Bean 的安全特性：
    a. 展開「EJB 模組」節點，找出已修改的 EJB 群組並展開。
    b. 展開**實體 Bean**。
    c. 展開已修改的 EJB 群組。
    d. 按一下**方法延伸**，然後在右窗格中執行下列步驟：
       1) 按一下**進階**標籤。
       2) 確定**安全身份**已選取。
       3) 針對每一個方法，確定**使用 EJB 伺服器的身份**已選取。
       4) 按一下**套用**（如果您有做任何修改的話）。
    e. 在左導覽窗格中，以滑鼠右鍵按一下已修改 EJB 群組下的**安全職務**，並選取**新建**，然後執行下列步驟：
       1) 在**名稱**欄位中，輸入 WCSecurityRole，並按一下**套用**。請注意，如果此職務已存在，便不需執行此步驟。
    f. 在左導覽窗格中，以滑鼠右鍵按一下已修改之 EJB 群組下的**方法許可權**，並選取**新建**，然後執行下列步驟：

1) 在**方法許可權名稱**欄位中輸入 WCMethodPermission。
2) 在**方法**選擇區中按一下**新增**。
這會開啓「新增方法」視窗。
3) 展開 *modifiedEJBGroup*，並選取所有 Enterprise Bean（按住 Shift 鍵然後選取）。按一下**確定**。此時，所有 Enterprise Bean 會顯示在 Enterprise Bean 直欄下，所有方法會顯示在「類型」直欄下。
4) 在「職務」選擇區中，按一下**新增**，並選取 WCSecurityRole，然後按一下**確定**。
5) 按一下**套用**。

19. 從**檔案**功能表中，選取**儲存**。
20. 關閉應用程式組譯工具。
21. 400 將新修改的 WC_Enterprise_App_*instanceName*.ear 檔從本端機器複製到執行 WebSphere Application Server 的 iSeries 機器中。亦即，複製下列目錄中的檔案：

    drive:\WebSphere\CommerceServer\working

    複錄到下列目錄中：

    iSeries_drive:\QIBM\UserData\WebCommerce\instances*instanceName*\working

    其中 *iSeries_drive* 爲您映射到至 iSeries IFS 的磁碟機字母。

在完成此步驟後，您便已建立一個新企業應用程式，且其含有先前的所有邏輯以及您的新商業邏輯。這全含在新修改的 WC_Enterprise_App_*instanceName*.ear 檔中。

停止並移除企業應用程式

本節說明如何使用 WebSphere Application Server 管理主控台，來停止目前正在執行的企業應用程式並將之移除。

如果要停止並移除您的企業應用程式，請執行下列步驟：

1. 開啓 WebSphere Application Server 管理主控台。
2. 展開 **WebSphere 管理網域**。
3. 展開**節點**。
4. 展開 ***nodeName*** （其中 ***nodeName*** 爲節點名稱。）
5. 展開**應用程式伺服器**。

6. Windows AIX Solaris Linux 以滑鼠右鍵按一下 WebSphere Commerce 應用程式伺服器。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server - *nodeName***，並選取**停止**。
7. 400 以滑鼠右鍵按一下 WebSphere Commerce 應用程式伺服器。舉例來說，以滑鼠右鍵按一下 ***instanceName* - WebSphere Commerce Server**，並選取**停止**。
8. 展開**企業應用程式**。
9. Windows AIX Solaris Linux 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce 企業應用程式 - *instanceName*** 應用程式，並選取**停止**。
10. 400 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 ***instanceName* - WebSphere Commerce 企業應用程式** 應用程式，並選取**停止**。
11. 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce 企業應用程式 -*instanceName***（或 ***instanceName* - WebSphere Commerce 企業應用程式**，若爲 iSeries 的話）應用程式，並選取**移除**。
12. 當出現提示指出是否應匯出應用程式時，請選取**否**。
13. 當出現提示指出是否應移除應用程式時，請選取**是**。

匯入企業應用程式

本節說明如何使用 XMLConfig 指令行公用程式來匯入企業應用程式。

如果要匯入新企業應用程式，請執行下列步驟：

1. Windows AIX Solaris Linux 輸入下列指令，呼叫 XMLConfig 工具，以便將企業應用程式匯入到 WebSphere Application Server 中：

   Windows

   ```
   xmlConfig -import OutputFile.xml -adminNodeName wasHostName
   ```

   AIX

   ```
   /usr/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
      -adminNodeName wasHostName
    -nameServiceHost wasHostName -nameServicePort wasAdminPort
   ```

   Solaris Linux

```
/opt/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
   -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

其中

- *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。
- *OutputFile.xml* 爲說明您所有 Enterprise Bean 的 XML 檔。
- *wasAdminPort* 爲 WebSphere Application Server 管理埠。

2. 400 輸入下列指令，呼叫 XMLConfig 工具，以便將企業應用程式匯入到 WebSphere Application Server 中：

```
STRQSH
PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
xmlConfig -import
/QIBM/UserData/WebCommerce/instances/instanceName/working/OutputFile.xml
-adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
-instance wasInstanceName
```

其中

- *OutputFile.xml* 爲說明您所有 Enterprise Bean 之 XML 檔的完整名稱。
- *wasHostName* 爲 WebSphere Application Server 中內含現行企業應用程式的節點名稱。

  註: 400 *wasHostName* 的值有區分大小寫，且必須和 TCP/IP 架構中的值相符。（請使用指令行處理器來存取 CFGTCP，選項 12，以驗證主電腦名稱。）
- *wasAdminPort* 爲 WebSphere Application Server 管理埠。
- *wasInstanceName* 爲 WebSphere Application Server 案例名稱。

400 在您試著執行 `XMLConfig -import` 指令時，您可能會收到如下的錯誤訊息：“無法展開 `/QIBM/UserData/WebAsAdv4/wasInstanceName/installedApps/WC_Enterprise_App_instanceName.ear` 下的 ear 檔”。如果您收到此訊息，請移除或重新命名上述目錄，並重新執行指令。

3. 400 在您匯入企業應用程式之後，必須執行 Script 來修改目錄許可權。如果要執行此 Script，請執行下列步驟：
   - 在指令提示下輸入：

```
STRSQH
cd /QIBM/ProdData/WebCommerce/bin
changeAuthority wasAdminInstanceName instanceName
```

其中

– *wasAdminInstanceName* 爲 WebSphere Application Server 管理案例的名稱。

– *instanceName* 爲 WebSphere Commerce 案例的名稱。

啓動企業應用程式

本節說明如何使用 WebSphere Application Server 管理主控台，重新整理檢視畫面然後啓動企業應用程式。

1. 開啓 WebSphere Application Server 管理主控台。
2. 依序展開 **WebSphere 管理網域**，**節點**與 *nodeName*。
3. 標明出 *nodeName* 節點。
4. 按一下**重新整理所選子樹**圖示。
5. 執行下列以啓動 WebSphere Commerce 應用程式：
   - 展開**應用程式伺服器**。
   - Windows AIX Solaris Linux 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 **WebSphere Commerce Server -** *instanceName*，並選取**啓動**。
   - 400 以滑鼠右鍵按一下您的 WebSphere Commerce 應用程式。舉例來說，以滑鼠右鍵按一下 *instanceName* **- WebSphere Commerce Server** 應用程式，並選取**啓動**。

附錄 C. VisualAge for Java 的相關要訣

本節說明與開發環境中之疑難排解、效能提升以及簡化作業有關的一些要訣。

變更 WebSphere Test Environment 中之 Servlet 引擎的內容

WebSphere Test Environment 中之 Servlet 引擎的內容由 default.servlet_engine 內容檔所控制。在此檔案中，您可以修改 Web 伺服器的文件起始目錄，以及變更 WebSphere Test Environment 所用之埠。

假設 WebSphere Test Environment 已停止運作，而重新啓動並非一個選項的話，您可能需變更埠。如果要變更埠，請執行下列步驟：

1. 以文字編輯器開啓 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine 檔。
2. 在 <transport> 段落中，將下行中的 8080 值改爲一個可用之埠。

   ```
   <arg name="port" value="8080"/>
   ```

3. 將下行中的 8080 值改成上述指定的相同埠。

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```

4. 儲存檔案。
5. 開啓 WebSphere Test Environment 控制中心並啓動 Servlet 引擎。

在預設的情況下，WebSphere Test Environment 的文件起始目錄爲 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\hosts\default_host \default_app\web。如果要變更成另一個目錄，請執行下列步驟：

1. 開啓 WebSphere Test Environment 控制中心並停止 Servlet 引擎。
2. 以文字編輯器開啓 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine 檔。
3. 在 <websphere-webgroup name="default_app"> 段落中，將 <document-root>$approot$/web</document-root> 改爲：

   ```
   <document-root>your_document_root</document-root>
   ```

   其中 *your_document_root* 爲您所要的文件起始目錄。

4. 將下行中的 8080 值改成上述指定的相同埠。

```
<hostname-binding hostname="localhost:8080" servlethost="default_host"/>
   <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
```

5. 儲存檔案。
6. 開啓 WebSphere Test Environment 控制中心並啓動 Servlet 引擎。

解決永久名稱伺服器的問題

如果您遇到永久名稱伺服器方面的問題，您可能得除去再重建永久名稱伺服器資料庫。有關建立此資料庫的詳述，請參閱 *Commerce Studio 安裝手冊*。

另外，如果出現一則訊息指出目前該埠已在使用中，請確定 WebSphere Application Server 不在執行中。

刪除已編譯的 JSP 檔

爲求效能，VisualAge for Java 會維護一個其中儲存著已編譯的 JSP 檔的專案資料夾。有些時候您可能需要刪除已編譯的 JSP 檔。舉例來說，如果您移除 JSP 檔中的資料 Bean，在您下次呼叫該 JSP 檔時可能會遇到錯誤。在此情況下，您可以刪除已編譯的 JSP 檔。

如果要刪除已編譯的 JSP 檔，請執行下列步驟：

1. 在 VisualAge for Java 中選取「專案」標籤。
2. 往下捲動到 **JSP 頁面編譯產生碼**專案。
3. 選取整個專案（如果您需要刪除許多已編譯的 JSP 檔），或者展開專案而只刪除必須重新編譯的 JSP 檔。

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。

本文件中可能包含著 IBM 所擁有之專利或專利申請案。 使用者不得享有本書內容之專利權。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.

若要查詢有關二位元組 (DBCS) 資訊的特許權限事宜，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 僅以現狀提供本書，而不提供任何明示或默示之保證 (包括但不限於可售性或符合特定效用的保證)。若有些地區在某些交易上並不允許排除上述保證，則該排除無效。

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 得隨時改進並 (或) 變動本出版品中所提及的產品及 (或) 程式。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供保證。該 Web 站上的資料，並非本 IBM 產品所用資料的一部分，因使用該 Web 站造成之損害，由 貴客戶自行負責。

IBM 得以各種適當的方式使用或散布由 貴客戶提供的任何資訊，而無需對您負責。

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，係一由產品的供應商，或其出版的聲明或其他公開管道取得。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。在說明的產品上市之前，這些資訊仍會變更。

本書中的範例包含了用於日常商業活動的資料及報告。爲了盡可能詳細，範例中涵蓋了個人、公司、品牌及產品的名稱。這些名稱全部都是虛構的，如果與眞實公司企業的名稱和地址雷同，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

下列詞彙爲 IBM 公司在美國及（或）其他國家的商標或註冊商標：

400	iSeries
AIX	MQSeries
AS/400	Net.Commerce
CICS	Net.Data
DB2	VisualAge
IBM	WebSphere

Windows 與 Windows NT 爲 Microsoft Corporation 在美國與（或）其他國家中的商標或註冊商標。

Oracle 是 Oracle Corporation 在美國與（或）其他國家的商標或註冊商標。

Solaris、Java 以及所有和 Java 相關的商標以及標誌都是 Sun Microsystems, Inc. 在美國及（或）其它國家的商標或註冊商標。

其它公司、產品及服務名稱可能是其他公司的商標或服務標記。

索引

索引順序以中文字，英文字，及特殊符號之次序排列。

〔六劃〕

〔七劃〕

〔八劃〕

〔九劃〕

〔十劃〕

〔十一劃〕

〔十二劃〕

〔十三劃〕

〔十四劃〕

〔十五劃〕

〔十六劃〕

〔十七劃〕

〔十九劃〕

〔二十三劃〕

C

E

F

J

M

S

U

V

W

讀者意見表

爲使本書盡善盡美，本公司極需您寶貴的意見；懇請您使用過後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號（∨）；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	
	實際畫面訊息與本書所提之畫面訊息是否一致	□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，版面安排是否方便使用	□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，是否便於查考	□是□否	
	索引語錄之排定，是否便於查考	□是□否	
	※評估意見爲 "否" 者，請於備註欄說明。		

其他： (篇幅不夠時，請另紙說明。)

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

GC40-0809-00

第 5.4 版

折疊線

台北市 110 基隆路一段 206 號

大臺灣國際商業機器股份有限公司
中華研發中心　軟體國際部　啟

寄件人　姓名：
　　地址：

廣告回信
台灣北區郵政管理局
登記證
北台字第 0587 號

（免貼郵票）

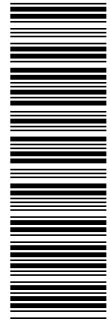
寄

折疊線

讀者意見表

Part Number: CT024TC

(1P) P/N: CT024TC



GC40-0809-00