IBM Toolbox for Java

IBM Toolbox for Java 為一組 Java(TM) 類別，可讓您使用 Java 程式來 存取 iSeries 及 AS/400e 伺服器中的資料。您可使用這些類別來撰寫使用 iSeries 中 資料的主從架構應用程式、Applet 及 Servlet。您也可在 iSeri Java 虛擬 機器 (Java VM) 中，執行使用 IBM Toolbox for Java 類別的 Java 應用程式。

IBM Toolbox for Java 使用 iSeries Host Servers 作為 對系統的存取點。因為 Toolbox for Java 使用 Java 內建的通信功能，所以您毋需 使用 IBM iSeries Access Express for Windows 來使用 Toolbox for Java。每一個伺服器均是以 伺服器的個別工作之方式執行，而每一個伺服器工作會在 socket 連接中傳送與接收資料串流。

請使用主導覽列或下列鏈結，來取得 IBM Toolbox for Java 的其它相關資訊：

V5R2 的新增功能
介紹重大的變更、強化的功能及其它附註項目。

列印 IBM Toolbox for Java 主題
檢視或下載 PDF 檔的 Toolbox for Java 主題。您也可下載 zip 套件形式的 Toolbox for Java 主題。

使用類別搜尋器
輕鬆及快速地依名稱及說明搜尋類別、依資料包顯示類別，或檢查按字母順序排列的所有 Toolbox for Java 類別清單。

IBM Toolbox for Java 入門
說明 IBM Toolbox for Java 安裝管理的相關資訊。介紹如何在工作站與伺服器上進行安裝。使用簡單的程式設計範例來介紹如何開始在應用程式中 Toolbox for Java 類別。

IBM Toolbox for Java 類別
介紹 IBM Toolbox for Java 套件中，可讓您使用 iSeries 及 AS/400e 伺服器資料的各種類別。此資訊包括說明、範例程式碼及技術資訊，可協助您建立 IBM Toolbox for Java 程式。

使用 Graphical Toolbox 建立自己的 GUI 畫面
介紹使用 Graphical Toolbox 建立以 Java 撰寫的自訂使用者介面畫面，將其納入 Java 應用程式、Applet 或「iSeries 領航員」外掛程式。

JavaBeans
介紹使用 Toolbox for Java 公用類別 (建立為 Javasoft JavaBean 標準) 來建立 JavaBeans。提供說明如何在程式中使用 JavaBeans 的範例。

JDBC
說明 Toolbox for Java 提供的 JDBC 支援。使用 JDBC，您的程式可對伺服器中的資料庫發出結構化查詢語言 (SQL) 陳述式，並處理來自伺服器中資料庫的結果。

使用 PCML 呼叫 iSeries 程式
介紹使用「程式呼叫語言 (PCML)」來協助您使用較少的 Java 程式碼便可呼叫 iSeries 程式。PCML 是一種標籤語法，以 XML 為基礎，能完整地說明由 Java 應用程式呼叫的 iSeries 程式之輸入及輸出參數。

Proxy 支援

介紹如何使用 IBM Toolbox for Java proxy 支援，包括使用 Sockets Layer (SSL) 通信協定來加密資料。

使 用 RFML 來定義及管理資料格式
使用「記錄格式標示語言 (RFML)」來區分資料格式規格 (Java 程式的商業邏輯)。RFML 是一種標籤語法，以 XML 為基礎並與 PCML 密切相關，可讓您的 Java 應用程式指定並操作特定類型記錄中的欄位。

安全性
介紹使用 Java Secure Socket Extension (JSSE) 及 Toolbox for Java 來提供透過 TCP/IP 執行任一應用程式通信協定的從屬站與伺服器之間，安全性資料的交換。

iSeries 系統除錯器
使用「iSeries 系統除錯器」「圖形式使用者介面 (GUI)」，進行在 iSeries 伺服器中執行的程式之除錯與測試。

系統內容
說明您如何使用系統內容，來配置 IBM Toolbox for Java 的不同層面，例如，定義 PROXY 伺服器或追蹤層次時所進行的配置。使用系統內容可使您不用重新編譯程式碼，便可執行執行時間配置。

IBM Toolbox for Java 2 Micro Edition
使用此項新的 Toolbox for Java 元件，為各種無線裝置撰寫 Java 程式。使用 ToolboxME for iSeries，您的無線裝置可直接存取 iSeries 伺服器資料與資源。

IBM Toolbox for Java 的 Javadocs
檢視 IBM Toolbox for Java 類別的 javadoc 參考資料。

常用問題集 (FAQ)
提出將 IBM Toolbox for Java 效能最佳化、執行疑難排解、使用 JDBC 等問題的解答

其餘資訊包括：

- Toolbox for Java 程式設計範例清單
- 程式設計秘訣有助您使用 Toolbox for Java
- 相關資訊，包括與 Java、Servlet、XML 等其它相關資訊的鏈結。

註：請閱讀 程式碼不保事項聲明 以取得重要的法律資訊。

V5R2 的新增功能

IBM Toolbox for Java 可以下列方式提供：

- IBM Toolbox for Java, 5722-JC1, 版本 5 版次 2 (V5R2) 授權程式, 可安裝在 OS/400 V4R5 與以上的版本。IBM Toolbox for Java 可以自從屬站連接至 OS/400 的 V4R5 與以上的版本。
- OS/400 也包含非圖形式的 IBM Toolbox for Java 類別, 這些類別已經過最佳化, 可在 iSeries Java 虛擬機器 (JVM) 上執行 IBM Toolbox for Java 類別時使用。因此, 假如您不需要使用此授權程式的圖形式功能時, 您仍然可以輕鬆地使用 IBM Toolbox for Java。如需詳細資訊, 請參閱Jar 檔案。
- IBM Toolbox for Java 目前也開放了原始碼。您可以從TOpen 網站中下載程式碼及取得資訊。

新套件

IBM Toolbox for Java 2 Micro Edition是IBM Toolbox for Java 的新套件。這組新ibm.as400.micro 套件 提供一組類別, 可以讓您撰寫在無線裝置 (如：個人數位助理和行動電話) 上所執行的 Java 程式。您必須分別下載 micro 套件。

iSeries 系統除錯器的工作站提供了新的圖形式除錯環境, 以便讓您應用於 iSeries 伺服器上執行的 ILE、Java、C 和 C++ 程式。

新類別

IBM Toolbox for Java V5R2 也在現有的套件中加入許多新類別。這些新類別可讓您：

- 使用ClusteredHashTable類別來共用及抄寫叢集的節點之間的暫時性資料
- 使用CommandPrompter 類別來提示給定指令的參數。
- 如需新的 JDBC(TM) 類別的詳細資訊, 請參閱新的 JDBC 類別及已強化的函數。
- RecordFormatDocument 類別可讓您使用新的「記錄格式標示語言 (RFML)」元件, 來指定記錄格式, 以及建立、讀取和寫入資料記錄。

已強化的類別

IBM Toolbox for Java V5R2 也包括對現有類別的加強功能。這些加強功能提供：

- AS400 物件的新增 Kerberos 支援, 它目前使用了 Java Generic Security Service (JGSS) 組織架構向 Toolbox for Jav 伺服器進行鑑別。
- SecureAS400 物件現在可以使用 Java Secure Socket Extention (JSSE) 組織架構, 將從屬站與伺服器之間的資料流程加密。
- 部份「程式呼叫語言 (PCML)」標籤和功能的變更：
  - <data> 標籤 的新屬性新增對 Unicode 字串的支援, 並可讓使用者指定如何調整字串的空間。
  - <program> 標籤 的新屬性和修改過的屬性現在可讓您在執行時間指定路徑, 並允許服務程式進入點名稱的 CCSID 規格。
  - 在追蹤作業方面, PCML 現在需要使用 Trace 類別(com.ibm.as400.access.Trace), 而非 PcmlMessageLog 類別。

新的 JDBC 類別及已強化的功能

IBM Toolbox for Java V5R2 JDBC 支援具有新類別和已強化的功能, 包括支援 JDBC 3.0 應用程式設計介面 (API)。重要的變更包括下列各項：

- AS400JDBCSavepoint 是一個新類別 (支援 JDBC 3.0), 它透過交易回轉來提供更好的控制

- AS400JDBCParameterMetaData 是一個新類別（支援 JDBC 3.0），可讓您擷取 PreparedStatement 和 CallableStatement 物件中的參數類型和內容
- 新增連接至獨立的輔助儲存體儲存區（IASP）的能力
- 二進位大型物件（blob）和 字元大型物件（clob）（支援 JDBC 3.0）中的新方法可讓您更新這些資料類型的值
- 新的 JDBC 內容延伸 meta 資料）則改進了 ResultSetMetaData 屬性的報告
- 其它改進 JDBC 支援的加強功能

新的 XML 元件

在 V5R2 中，IBM Toolbox for Java 新增了記錄格式標示語言（RFML），它是類似 PCML 的 XML 擴充。RFML 可讓您在 Java 程式中使用 XML， 以便指定資料緩衝區的格式和實體檔案記錄格式，並檢查擷取的緩衝區和記錄內容。

Graphical Toolbox額外的功能與特性

Graphical Toolbo納入了下列新特性：

- 將表格直欄的資料格顯示成核取框
- 指定對話框的最小高度和寬度，以確保適當地顯示對話元素
- 指定表格的第一個直欄包含動態的階層樹，其中的每一個資料格對應一個樹節點

如需這些特性的詳細資訊，請參閱 GUI Builder 線上說明。

相容性

IBM Toolbox for Java 不再支援在 Netscape(R) Navigator 或 Microsoft(R) Internet Explorer 的預設 JVM 中執行。針對使用 Toolbox f Java 類別而在瀏覽器上執行的 applet 而言，您應該安裝外掛程式，例如 Java 2 Runtime Environment (JRE) 1.3.0 plug-in 。

Toolbox for Java 不再包括 data400.jar。data400.jar 先前所含有的類別目前位於 jt400.jar 中。請從 CLASSPATH 陳述式中移 data400.jar。

當 SQLType 為 SMALLINT 時，ResultSet 和 CallableStatement 的 getObject() 方法目前將傳回 Integer 物件。這些方法先前是傳回 Short 物件。 如果您使用 readObject 來讀取 SMALLINT 直欄，則您必須修改您的 Java 應用程式，讓它容納傳回物件的新類型。

不同的錯誤報告，它們會在資料截斷 錯誤時產生不會導致應用程式失敗的警告。

您無法使用這個 IBM Toolbox for Java 版次，解除序列化您使用 V5R1 之前的版次所序列化的部份物件。

如果您是使用 Secure Sockets Layer (SSL) 來進行從屬站與伺服器之間的資料流程加密，則您必須使用下列其中一項：

- Java Secure Socket Extension (JSSE)
- IBM iSeries Client Encryption 授權程式 5722-CE2 或 5722-CE3 的 V5R1 或以上版本所隨附的 SSL 物件。這個 IBM Toolbox for Java 版次不會使用 iSeries Client Encryption V4R5 及更早版本。

IBM Toolbox for Java 繼續提供下列支援

- Swing 1.1，它是使用 GUI 類別或 Graphical Toolbox 所必要的
- Java 2 Platform 標準版 (J2SE)，繼續支援 Java Development Kit 1.1.8

您也應該複查 執行 IBM Toolbox for Java 的 OS/400 基本要求。

2002 年 9 月 26 日的新增功能

IBM Toolbox for Java 類別搜尋器
使用新類別搜尋器，您可以輕鬆及快速地依名稱及說明搜尋類別、依資料包顯示類別，或檢查按字母順序排列的所有 IBM Toolbox for Java 類別清單。 每一個類別的簡短說明均鏈結至其餘特定資訊。

如何查看新增內容或變更內容

為了協助您查看技術變更之處，此資訊（而非 javadocs）將使用下列符號：

- »標示新資訊或變更資訊的開頭
- «標示新資訊或變更資訊的結尾

若要尋找此版次的新增功能或變更功能的其它資訊，請參閱 使用者備忘錄  。

列印此主題

欲檢視或下載 PDF 版本，請選取 IBM Toolbox for Java PDF (大約有 5.4 百萬位元組或 811 頁)。

註： IBM Toolbox for Java 主題含有一些未包括在 PDF 檔中的資訊。

儲存 PDF 檔

如欲在您的工作站上儲存 PDF，供檢視或列印之用：

1. 在您的瀏覽器中以滑鼠右鍵按一下 PDF (以滑鼠右鍵按一下上述的鏈結)。
2. 按一下 將目標另存新檔
3. 導覽到您想要儲存 PDF 之處。
4. 按一下 儲存。

下載 Adobe Acrobat Reader

如果您需要 Adobe Acrobat Reader 來檢視或列印這些 PDF，您可以從 Adobe 網站 (www.adobe.com/products/acrobat/readstep.html) 下載一份副本。

如欲在您的工作站上儲存 PDF，供檢視或列印之用：

1. 在您的瀏覽器中開啟 PDF (按一下上述的鏈結)。
2. 在瀏覽器的功能表中，按一下檔案 。
3. 按一下 另存新檔
4. 導覽到您想要儲存 PDF 之處。
5. 按一下 儲存。

下載壓縮成套件的 IBM Toolbox for Java 資訊

您可以下載包含 javadocs 且壓縮成套件的 IBM Toolbox for Java 主題，它們位於 IBM Toolbox for Java and JTOpen 網站。

註：IBM Toolbox for Java 資訊具有未包括在這個壓縮套件內的文件鏈結。
這些鏈結將無法在您下載到工作站中的檔案內作用。

IBM Toolbox for Java 入門

使用 IBM Toolbox for Java，使得撰寫從屬站 Java Applets、Servlets 及 存取 iSeries 資源、資料及程式的應用程式變得更容易。

下列資訊有助於安裝以及使用 IBM Toolbox for Java：

≫管理安裝系統
檢視以不同的方式安裝與管理 Toolbox for Java 安裝系統時，對管理與效能的影響有多大。**≪**

安裝 Toolbox for Java
請讀取關於在主從架構環境中安裝 Toolbox for Java 的 OS/400 與 工作站基本要求。學習在 iSeries 伺服器與工作站中安裝 Toolbox for Java 的方法。

系統內容
請讀取關於系統內容的資訊，並學習如何利用系統內容來配置 IBM Toolbox for Java 不同層面的方法。

≫簡式程式設計範例
開始使用 Toolbox for Java。建立第一個 Toolbox for Java 程式，或複查簡式程式設計範例。範例顯示如何開始利用 Toolbox for Java 來使用 iSeries 伺服器中可用的資料與服務。**≪**

管理 IBM Toolbox for Java 安裝系統

您只需要在要使用 IBM Toolbox for Java 的從屬站系統中安 裝 IBM Toolbox for Java，或者在從屬站可存取的網路位置上進行安裝。 從屬站可以是個人電腦、專用工作站或 iSeries 系統。請不要忘記 iSeries 伺服器或伺服器的分割區可以配置成為從屬站。在後面的情況下，您必須在伺服器的從屬站分割區中安裝 Toolbox for Jav

您可以使用下列任一種方法（單獨一種或組合）來安裝與管理 Toolbox for Java：

- 個別管理安裝以及個別管理每一個從屬站中的 Toolbox for Java
- 網路管理從屬站安裝系統使用 AS400ToolboxInstaller 來安裝以及管理每一個從屬站中的 Toolbox for Java
- 網路管理單一安裝系統利用網路來安裝以及管理伺服器中的單一及共用的 Toolbox for Java 安裝系統

下一節簡短說明每一種方式對於效能與管理的影響。您所選擇的 Java 應用程式開發以及資源管理方式，會決定您需要使用的方法（或方法組合）。

個別管理

您可以選擇個別管理個別從屬站中的 Toolbox for Java 安裝系統。在個別從屬站中安裝 Toolbox for Java 的主要優點是可以縮減從屬站花在啟動使用 Toolbox for Java 類別之應用程式的時間。

主要的缺點則是必須個別管理這些安裝系統。必須有一個使用者或您建立的應用程式來負責追蹤及管理安裝在每個工作站 Toolbox for Java 版本。

網路管理從屬站安裝系統

您可以選擇使用網路與AS400ToolboxInstaller來 管理 Toolbox for Java 的從屬站安裝系統。因為每一個從屬站都有它自己 的 Toolbox for Java 副本，所以這一類的安裝方式具有減少從屬站花費在啟 動 Toolbox for Java 應用程式時間的相同優點。它也可以自動更新所有 的 Toolbox for Java 個別安裝系統。

這種方式的主要缺點是必須要建立並維護使用 AS400ToolboxInstaller 的 處理，來管理個別安裝系統

網路管理單一安裝系統

您也可以使用網路在伺服器中安裝及管理單一 Toolbox for Java 副本，讓所有的從屬站存取。這一種網路安裝系統具有下列優點：

- 所有從屬站都使用相同版本的 IBM Toolbox for Java
- 更新單一的 Toolbox for Java 安裝系統，便可以造福所有從屬站
- 個別從屬站除了設定相同的起始 CLASSPATH 之外，沒有維護的問題

這種安裝系統的缺點也會增加從屬站花費在啟動 Toolbox for Java 應用程式的時間。 您也要將從屬站 CLASSPATH 指向該伺服器。您可以使用與 OS/400 整合的iSeries NetServer或使用不同的方法，讓您可以存取 iSeries NetServer 中的檔案，像 是iSeries Access for Windows

安裝 IBM Toolbox for Java

IBM Toolbox for Java 的安裝方式是取決於您要如何管理安裝系統。
決定好要如何管理安裝系統之後，請確定您的環境符合下列基本要求：

- OS/400 基本要求
- 工作站基本要求

安裝 Toolbox for Java

決定好要如何管理安裝系統以及備妥執行 IBM Toolbox for Java 的基本要求之後，您就可以安裝 Toolbox for Jav

- 在伺服器中安裝 Toolbox for Java
- 在工作站中安裝 Toolbox for Java

IBM Toolbox for Java 的 OS/400 基本要求

決定好要如何管理安裝系統後，請確定您 的 OS/400 環境符合下列基本要求：

- 必要的 OS/400 選項
- 與其它授權程式的相依關係
- 與不同 OS/400 層次的相容性
- 在 OS/400 JVM 中執行時的原有最佳化
- 執 行 ToolboxME for iSeries 應用程式的基本要求

附註：開始使用 Toolbox for Java 之前，請先備妥與您的環境相關的工作站基本要求。

必要的 OS/400 選項

若要在主從架構環境中執行 IBM Toolbox for Java，您需要啟用 QUSER 使用者設定檔、啟動主電腦伺服器及啟動 TCP/IP：

- 必須啟用 QUSER 使用者設定檔，才能啟動主電腦伺服器。
- 主電腦伺服器接收並接受從屬站的連線要求。OS/400 基本選項有包含「OS/400 主電腦伺服器」選項 (授權產品 5722SS1)。如需詳細資訊，請參閱 主電腦伺服器管理
- 整合在 OS/400 中的 TCP/IP 支援可讓您將伺服器連接到網路。如需詳細資訊，請參閱 TCP/IP。

啟動必要的 OS/400 選項

請從 iSeries 指令行完成下列步驟，以啟動必要的 OS/400 選項：

1. 確定已啟用 QUSER 設定檔 。
2. 若要啟動 OS/400 主電腦伺服器，請使用「啟動主電腦伺服器」CL 指令。 鍵入 STRHOSTSVR *ALL ，再按 ENTER 。
3. 若要啟動 TCP/IP 分散式資料管理 (DDM) 伺服器，請使用「啟動 TCP/IP 伺服器」CL 指令。鍵入 STRTCPSVR SERVER(*DDM) ，再按 ENTER 。

檢查伺服器中是否已安裝 IBM Toolbox for Java

許多 iSeries 伺服器都已經有安裝 IBM Toolbox for Java 授權產品。

若要查看是否已經有安裝 Toolbox for Java，請完成下列步驟：

- 在「iSeries 領航員」中，選取並登入您要使用的系統。
- 在功能樹（左窗格）中，展開系統，然後展開配置與服務。
- 展開軟體，然後展開 已安裝的產品
- 在明細 窗格（右窗格）中，查看 5722jc1 產品直欄。如果您可以看到此產品，表示 IBM Toolboxfor Java 授權程式已經安裝在所選取的伺服器中。

  附註：另一種檢查是否有安裝 Toolbox for Java 的 方式，是否利用「CL 跳至功能表 GO指令（MENU(LICPGM)　），選項 11。

如果沒有安裝 Toolbox for Java，您可以安裝 IBM Toolbox for Java 授權產品。

如果已安裝了舊版的 Toolbox for Java，請先刪除目前安裝的版本，然後安裝 IBM Toolbox for Java 授權產品。為了避免可能發生的問題，請在刪除目前安裝的版本之前，先備份目前安裝的 Toolbox for Java 版本。

檢查 QUSER 設定檔

「OS/400 主電腦伺服器」是在 QUSER 使用者設定檔下啟動的，所以您首先必須確定 QUSER 設定檔已經啟用。

檢查 QUSER 設定檔

要使用指令行來檢查 QUSER 設定檔，請完成下列步驟：

1. 在 iSeries 指令行中，鍵入 DSPUSRPRF USRPRF(QUSER) 後再按 Enter 鍵。
2. 確定 Status 是 *ENABLED。如果設定檔的狀態不是 *ENABLED，請變更 QUSER 設定檔。

變更 QUSER 使用者設定檔

如果 QUSER 設定檔不是 *ENABLED，則需啟用它才能啟動 OS/400 主電腦伺服器。 此外，QUSER 設定檔密碼也不可為 *NONE。如果此為 TRUE，則需重設它。

若要使用指令行來啟用 QUSER 設定檔，請完成下列步驟：

1. 鍵入 CHGUSRPRF USRPRF(QUSER) 。
2. 將 狀態欄位變更為 *ENABLED，並按 ENTER 鍵。

QUSER 使用者設定檔現已備妥，可啟動 OS/400 主電腦伺服器。

與其它授權程式的相依關係

根據您使用 IBM Toolbox for Java 的方式，您可能必須安裝其它授權程式。 下列資訊說明這些這些相依關係。

排存檔檢視器

若要使用 IBM Toolbox for Java 的排存檔檢視器功能 (SpooledFileViewer 類別)，請確定您已經在伺服器中安裝好主電腦選項 8 (AFP 相容字型)。

附註：只有在連接到 V4R4 或更新的系統時，SpooledFileViewer、PrintObjectPageInputStream 及 PrintObjectTransformedInputStream 類別才能作用。

Secure Sockets Layer (SSL)

若要使用 Secure Sockets Layer (SSL)，請確定您已經安裝好下列項目：

- IBM HTTP Server for iSeries 授權程式，5722-DG1
- OS/400 選項 34 (數位憑證管理程式)
- iSeries 的 IBM 密碼存取提供者 (128 位元)，5722-AC3
- iSeries 從屬站加密 (128 位元)，5722-CE3

V5R2 版本的 IBM Toolbox for Java 需要使用 V5R1 或 V5R2 版的「iSeries 從屬站加密」。

有關 SSL 的詳細資訊，請參閱 Secure Sockets Layer 與 Java Secure Socket Extension。

HTTP 伺服器用來使用 Applets、Servlets、SSL 或 AS400ToolboxInstaller

如果您要在 iSeries 系統中使用 Applets、Servlets、SSL 或 AS400ToolboxInstaller 類別，您必須設定一部 HTTP 伺服器，並在 iSeries 系統中安裝類別檔。有關 IBM HTTP Server 的詳細資訊，請參閱 IBM HTTP Server for AS/400 Webmaster's Guide，書號 GC41-5434，此書可在下列 URL 找到：http://www.ibm.com/eserver/iseries/products/http/docs。Webmaster's Guide 有提供 HTML 與 PDF 兩種格式。

有關「數位憑證管理程式」及如何透過 IBM HTTP Server 建立和使用數位憑證的詳細資訊，請參閱 數位憑證管理。

與不同 OS/400 層次的相容性

因為伺服器及從屬站都可以使用 IBM Toolbox for Java,
所以相容性的問題就會影響到伺服器中的運作情形以及由從屬站連接回伺服器的情況。

在伺服器中執行 IBM Toolbox for Java, 版本 5 版次 2

若要在 iSeries 系統中安裝 IBM Toolbox for Java (授權程式 5722-JC1 V5R2M0), 該伺服器中必須執行下列其中一個版本：

- OS/400 版本 5 版次 2
- OS/400 版本 5 版次 1
- OS/400 版本 4 版次 5

每個系統上只能安裝一種 IBM Toolbox for Java 授權程式版本。若要安裝不同的版本, 請先移除現有的 IBM Toolbox for Java 授權程式。

使用 IBM Toolbox for Java, 由從屬站連接回伺服器

您可以在從屬站與所連接的伺服器中使用不同的 IBM Toolbox for Java 版本。 若要使用 IBM Toolbox for Java 版 5 版次 2 來存取 iSeries 系 統中的資料與資源, 您所連接的伺服器必須執行下列其中一個版本：

- OS/400 版本 5 版次 2
- OS/400 版本 5 版次 1
- OS/400 版本 4 版次 5

下表顯示在 OS/400 中安裝 IBM Toolbox for Java, 及連接回不同的 OS/400 版本時, 所需考慮的相容性基本要求。

修改工具箱	隨 OS/400 出貨	LPP	安裝在 OS/400 中	連接回 OS/400
Mod 0	V4R2	5763-JC1 V3R2M0	V3R2 及更新版	V3R2 及更新版
Mod 1	V4R3	5763-JC1 V3R2M1	V3R2 及更新版	V3R2 及更新版
Mod 2	V4R4	5769-JC1 V4R2M0	V4R2 及更新版	V4R2 及更新版
Mod 3	V4R5	5769-JC1 V4R5M0	V4R3 及更新版	V4R2 及更新版
Mod 4	V5R1	5722-JC1 V5R1M0	V4R4 及更新版	V4R3 及更新版
Mod 5	V5R2	5722-JC1 V5R2M0	V4R5 及更新版	V4R5 及更新版

在 iSeries JVM 中執行時的原有最佳化

原有最佳化是一組功能，能讓 IBM Toolbox for Java 類別在 OS/400 中的運作情形滿足使用者的預期。最佳化只有
iSeries JVM 中執行時，才會影響 IBM Toolbox for Java 的作業。

請記得，當您使用的 IBM Toolbox for Java 版本與伺服器中的 OS/400 版本相符時，您的 Java
程式才會使用原有最佳化。最佳化包括：

- 登入：AS400 物件中沒有指定使用者 ID 或密碼時，就會使用目前工作的使用者 ID 與密碼
- 直接呼叫 OS/400 API，而不對主電腦伺服器進行 Socket 呼叫：
  - 符合安全基本要求時，就呼叫記錄層次資料庫存取、資料佇列及使用者空間。
  - 符合安全基本要求與緒安全基本要求時，則進行程式呼叫以及指令呼叫。

»附註：當 Java 程式與資料庫檔案都位於相同的 iSeries 系統內時，為了取得最佳效能，請將
驅動程式內容設定為使用原生驅動程式«

您不需要為了取得最佳化而變更 Java 應用程式。IBM Toolbox for Java 會在適當時機自動啟用最佳化。

原有最佳化相容性基本要求

下表顯示您必須執行的 IBM Toolbox for Java 與 OS/400 版本，才能使用原有最佳化。
此表格僅引證會影響到原有最佳化的相容性問題。有關一般的相容性問題，請參閱 與不同 OS/400
層次的相容性。

OS/400 的層次	使用原有最佳化所需的工具箱修改				
V4R2	沒有可用的 Mod0 加強功能。				
V4R3	Mod1	Mod2			
V4R4	Mod1	Mod2			
V4R5			Mod3		
V5R1				Mod4	
»V5R2					Mod5 «

為了提高效能，您必須確定使用包含 OS/400 原有最佳化的 JAR 檔。如需詳細資訊，請參閱檔中的附註 1。

若 IBM Toolbox for Java 與 OS/400 的版本不相符時，將無法使用原有最佳化。 在此情況下，IBM Toolbox for J
的運作狀況會如同是在從屬站中執行。

ToolboxME for iSeries 基本要求

您的工作站、無線裝置及伺服器必須符合以下列出的特定基本要求， 才能開發與執行 ToolboxME for iSeries 應用程式。雖然 IBM Toolbox for Java 2 Micro Edition 被 視為 IBM Toolbox for Java 的一部份，但授權產品中並未將其包含在內。

ToolboxME for iSeries (jt400Micro.jar) 包含於 Toolbox for Java 的開放原始碼版本中， 稱為 JTOpen。您必須分別下載及設定 ToolboxME for iSeries (包含在 JTOpen 中)。

基本要求

若要使用 ToolboxME for iSeries，您的工作站、Tier0 無線裝置及 伺服器必須符合下列基本要求。

工作站基本要求

開發 ToolboxME for iSeries 應用程式的工作站基本要求：

- Java 2 Platform 標準版，版本 1.3 或更新的版本
- 用於無線裝置的 Java 虛擬機器 (Java VM)
- 無線裝置模擬器

無線裝置基本要求

在 Tier0 裝置中執行 ToolboxME for iSeries 應用程式的唯一基本要求為： 使用於無線裝置的 Java 虛擬機器 (Java VM)。

伺服器基本要求

使用 ToolboxME for iSeries 應用程式的伺服器基本要求：

- 內含於 IBM Toolbox for Java 或最新版本的 JTOpen MEServer
- 用於 IBM Toolbox for Java 的 OS/400 基本要求

IBM Toolbox for Java 的工作站基本要求

決定好要如何管理安裝系統後，請確定您的工作站符合下列基本要求：

- 執行 Java 應用程式的基本要求
- 執行 Java Applets 的基本要求
- 開發 ToolboxME for iSeries 應用程式的基本要求
- Swing 基本要求

附註：開始使用 Toolbox for Java 之前，請先備妥與您的環境相關的 iSeries 400 基本要求。

執行 Toolbox for Java 應用程式的工作站基本要求

若要開發與執行 Toolbox for Java 應用程式，請確定您的工作站符合下列基本要求：

- 我們建議使用 Java 虛擬機器 (JVM)，它支援 Java 2 Platform 標準版 (J2SE™) 或企業版 (J2EE™)，版本 1.3.x 或更新版。 許多新的 Toolbox for Java 功能需要使用這一個 JVM 版本。不過，您仍然可以使用完全支援 JDK 1.1.8 或任何更新版本 JDK 的 JVM，包括 Java 2 Platform。
- 如果您的程式使用到 Graphical Toolbox 或 Vaccess 套件中的類別，那麼您也需要 1.1。已經測試的環境如下：
  - Windows® 2000
  - Windows® XP
  - AIX 版本 4.3.3.1
  - Sun Solaris™ 版本 5.7
  - OS/400 版本 4 版次 5 或更新版
  - Linux (Red Hat 7.1)
- 已安裝及配置 TCP/IP

執行 IBM Toolbox for Java Applets 的工作站基本要求

若要開發與執行 Toolbox for Java 應用程式，請確定您的工作站符合下列基本要求：

- 瀏覽器具有相容的 Java 虛擬機器 (JVM)。已經測試的環境如下：
  - »Netscape Communicator 4.7，使用 Java 1.3 或更新版的外掛程式

  附註：IBM Toolbox for Java 不再支援在 Netscape Navigator 或 Microsoft Internet Explorer 中預設執行。若要在瀏覽器中執行使用 Toolbox for Java 類別的 Applet，您應該安裝外掛程式，例如：

  Java 2 Runtime Environment (JRE) 1.3.0 外掛程式«

- 已安裝及配置 TCP/IP
- »工作站必須連接到執行 OS/400 V4R5 或更新版的伺服器

IBM Toolbox for Java 的工作站 Swing 基本要求

IBM Toolbox for Java 從 V4R5 變更為支援 Swing 1.1，此一版次會繼續支援。 切換至 Swing 需要變更 IBM Toolbox for Java 類別中的程式設計。所以，如果您的程式使用 Graphical Toolbox 或 V4R5 版以前的 Vaccess 類別，您的程式也需要變更。

除了程式設計變更之外，當執行程式時，Swing 類別必須位於 CLASSPATH 中。Swing 類別是 Java 2 Platform 的一部份。如果您沒有 Java 2 Platform，您可以從 Sun Microsystems, Inc. 下載 Swing 1.1 類別。

在 iSeries 伺服器中安裝 IBM Toolbox for Java

當您將伺服器或伺服器分割區配置成從屬站時，才需要在 iSeries 伺服器中安裝 IBM Toolbox for Java。

附註：原版的 Toolbox for Java 會隨 OS/400 一同出貨。 所以，如果您只是要在 iSeries 伺服器中使用 Toolbox for Java 時，就不需要安裝授權產品。有關原版 Toolbox for Java 的詳細資訊，請參閱 檔：附註 1。

安裝 IBM Toolbox for Java 之前，您必須確定您的 OS/400 版本符合 Toolbox for Java 的基本要求 此外，有些伺服器會預先配置為已安裝 Toolbox for Java。您可能需要檢查伺服器中是否已安裝了 Toolbox for Java 授權產品。

安裝 Toolbox for Java

您可以使用「iSeries 領航員」或指令行來安裝 IBM Toolbox for Java 授權程式。

使用「iSeries 領航員」安裝 Toolbox for Java

若要使用「iSeries 領航員」安裝 Toolbox for Java，請完成下列步驟：

1. 請在「iSeries 領航員」中，登入您要使用的系統。
2. 在「功能樹」(左窗格) 中，展開我的連接。
3. 然後在我的連接底下，在您要安裝 Toolbox for Java 的系統上按一下滑鼠右鍵。
4. 選取執行指令。
5. 在復置授權程式 (RSTLICPGM) 對話框中，鍵入下列資訊，然後按一下 確定 ：
   - 產品：5722JC1
   - 裝置：裝置或儲存檔案名稱

   附註：如需詳細資訊，請按一下 復置授權程式 (RSTLICPGM) 對話框中的說明。

請完成下列步驟，就可以使用「iSeries 領航員」來檢視「管理中心指令」作業的狀態：

1. 展開管理中心。
2. 展開作業活動。
3. 在作業活動 底下，選取指令 。
4. 在「明細」窗格中，按一下適當的 執行指令 作業。

使用指令行安裝 Toolbox for Java

若要從 iSeries 指令行安裝 Toolbox for Java，請完成下列步驟：

1. 在 iSeries 指令行上，使用「跳至功能表」CL 指令。G鍵入ENU(LICPGM) 再按ENTER 。
2. 選取11.安裝授權程式。
3. 選取5722-JC1 IBM Toolbox for Java。

有關安裝授權程式的詳細資訊，請參閱管理軟體及授權程式。

在工作站中安裝 IBM Toolbox for Java

安裝 Toolbox for Java 之前，請先備妥與您的環境相關的工作站基本要求。 在工作站中安裝 IBM Toolbox for Java 的方式取決於您要使用哪種安裝系統：

- 若要在個別的從屬站中安裝 Toolbox for Java，請將 JAR 檔複製到您的工作站中，並配置工作站的 CLASSPATH。
- 若要使用安裝於伺服器中的 Toolbox for Java，您只需要將工作站的 CLASSPATH 指向伺服器中的安裝系統。若要將工作站的 CLASSPATH 指向伺服器，您的伺服器必須有安裝 iSeries Netserver。

本文件說明如何將類別檔案複製到工作站中。有關在工作站中設定 CLASSPATH 的詳細資訊，請參照工作站的作業系統文件，或位於 Sun Java 網站的資訊。

附註：您的系統還必須符合 i5/OS 的基本要求，才能在應用程式中使用 Toolbox for Java 類別。

Toolbox for Java 類別檔案封裝在數個 JAR 檔中，因此您必須複製一個或多個這些 JAR 檔到您的工作站中。有關特定 Toolbox for Java 功能需要哪些 JAR 檔的詳細資訊，請參閱 JAR 檔。

範例：複製 jt400.jar

下面的範例假設您要複製 jt400.jar，此檔包含基核 IBM Toolbox for Java 類別。

若要手動複製 JAR 檔，請完成下列步驟：

1. 在下面的目錄中找出 jt400.jar 檔：
   /QIBM/ProdData/HTTP/Public/jt400/lib
2. 將 jt400.jar 從伺服器複製到您的工作站。有不同的方法可用來完成此項步驟：
   - 使用 iSeries Access for Windows，將工作站中的網路磁碟機對映至伺服器，然後複製檔案。
   - 使用檔案轉送通信協定 (FTP) 將檔案傳送 (以二進位模式) 到工作站。
3. 更新工作站的 CLASSPATH 環境變數。
   - 例如，如果您是使用 Windows NT，且您已將 jt400.jar 複製到 C:\jt400\lib，請在 CLASSPATH 的末端新增下列字串：

     ```
     ;C:\jt400\lib\jt400.jar
     ```

您也可以選擇使用 Toolbox for Java 的開放原始碼版本，它稱為 JTOpen。 有關 JTOpen 的詳細資訊，請參閱 IBM Toolbox for Java 與 JTOpen 網站。

Jar 檔

IBM Toolbox for Java 以一組 jar 檔的形式銷售。每個 jar 檔包含提供特定功能的 Java 套件。您可以減少所需儲存體空間的容量，其方法為只使用需要的 jar 檔來啟用您要的特定功能。

若要使用 jar 檔，請確定您的 CLASSPATH 已包含了它的登錄。

下圖指出哪些 jar 檔必須在您的 CLASSPATH 中，方能使用所列出套件的類別。

表格中的部份登錄，具提供詳細資訊的附註。如果您的瀏覽器具有 Javascript 功能， 請點一下影像，在個別視窗中顯示資訊。 否則，您可以按一下文字鏈結，以鏈結到列於下表中的相同資訊。

Toolbox for Java 套件或功能	Jar 檔必須在您的 CLASSPATH 中
存取類別	在一般從屬站/伺服器環境中的 jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 或代理站環境中的 jt400Proxy.jar
CommandPrompter 附註 2	jt400.jar、jui400.jar、util400.jar 附註 3 及 x4j400.jar
HTML 類別	jt400.jar 附註 1 加上 jt400Servlet.jar (從屬站)、或 jt400Native.jar (伺服器) 附註 1
JDBC 資料來源 GUI 附註 4	jt400.jar (從屬站) 附註 1 及 jui400.jar
NLS 系統及錯誤訊息	jt400Mri_lang_cntry.jar 附註 5
PCML (開發) 附註 6	jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 附註 7 及 x4j400.jar
PCML (執行期間，已編序)	jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 附註 7
PDML (開發) 附註 2	uitools.jar、jui400.jar、util400.jar 附註 3 及 x4j400.jar
PDML (執行期間，已剖析) 附註 2	jui400.jar、util400.jar 附註 3 及 x4j400.jar
PDML (執行期間，已編序) 附註 2	jui400.jar 及 util400.jar 附註 3
ReportWriter 類別	jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 及 reportwriter.jar 附註 8
Resource 類別	jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1
RFML	jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 及 x4j400.jar
Security 類別	在一般從屬站/伺服器環境中的 jt400.jar (從屬站) 或 jt400Native.jar (伺服器) 附註 1 或代理站環境中的 jt400Proxy.jar

Servlet 類別	jt400.j 附註 1 加上 jt400Servlet.jar (從屬站)、或 jt400Native.jar (伺服器) 附註 1
iSeries 系統除 附註 2	jt400.jar (從屬 附註 1 及 tes jar
ToolboxME for iSeries	jt400Micro.jar (從屬站) 及 jt400.jar (伺服器) 或 jt400Native.jar (伺服器) 附註
Vaccess 類別	jt400.jar (從屬 附註 1

附註 1：有些 IBM Toolbox for Java 類別位於一個以上的 jar 檔之中：

- jt400.jar 存取、來源、vaccess、 安全、PCML、RFML、JDBC 支援及 MEServer。
- jt400.zip 使用 jt400.jar 代替 jt400.zip。出貨時會提供 jt400.zip, 以保有與前版次 IBM Toolbox fo 的相容性。
- jt400Access.zip 存取、資源、安全、PCML；亦即除 vaccess 類別外其它都是 與 jt400.jar 中相同的相同類別。出貨時會提供 jtAccess400.zip, 以保有與前版次 IBM Toolbox for Java 的相容性。請使用 jt400.jar 或 jt400Native.jar 代替 jt400Access.zip。
- jt400Native.jar 存取 、資源、安全、PCML、HTML、RFML、MEServer 及 原有的最佳化 原有的最佳化是一組當它在 iSeries JVM 上執行時, 會利用 iSeries 功能的類別 (小於 20 個)。 由於 jt400Native.jar 包含原有的最佳化, 所以在 iSeries JVM 上執行時, 請使用 jt400Native.jar 代替 jt400.jar。jt400Native.jar 隨 OS/400 一起出貨, 而且常駐在目錄 /QIBM/ProdData/OS400/jt400/lib 中。
- jt400Native11x.jar 只 原有的最佳化 如果您是在 iSeries JVM 上執行而且想使用 jt400.jar, 請在您的 CLASSPATH 中包 含 jt400Native11x.jar 而非 jt400Native.jar。jt400Native11x.jar 隨 OS/400 一起出貨, 而且常駐在目錄 /QIBM/ProdData/OS400/jt400/lib 中。

附註 2：使用 CommandPrompter、PDML 或「iSeries 系統除錯器」也需要有一個額外的 jar 檔, 其並不屬於 Toolbox for Java 的一部份：jhall.jar。關於下載 jhall.jar 的詳細資訊, Sun 請參閱 Help(TM) 網 。

附註 3：util400.jar 包含 iSeries 特有的類別, 用於格式化輸入及使用指令行 (CL) 提示器。使用 CommandPrompter 類別需要 util400.jar。 使用 PDML 並不需要 util400.jar, 但它是有用的。

附註 4：jui400.jar 包含 使用 JDBC DataSource GUI 介面所需的類別。jt4 附註 jar 包含其它所有 JDBC 功能需要的類別。

附註 5：jt400Mri_xx_yy.jar 包含轉譯 的訊息, 包括內含在異常情況訊息、對話框、以及其它正常處理程序輸出中的字串。在 jt400Mri_lang_cntry.jar 中 使用轉譯內含文字的 lang =「ISO 語言碼」, 而 cntry =「ISO 國家或地區碼」。在某些情況下, 不會使用「ISO 國家或地區碼」。在 iSeries 上安裝特定國家語言版本的 IBM Toolbox for Java 授 權程式, 會安裝適當的 jt400Mri_lang_cntry.jar 檔。如果語言不受支援, 則安裝會預設成英文版本, 這個版本內含在 IBM Toolbox for jar 檔中。

- 例如, 安裝德國語言版本的授權程式 5722-JC1 會安裝德文 jar 檔 jt400Mri_de.jar。

藉由新增一個以上的 jar 檔到 classpath 可以支援新增其它語言的支援。Java 會根據現行語言環境載入正確的字串。

附註 6：在發展期間編序 PCML 檔有兩個好處：

1. 您只有在發展期間而非執行期間才需要剖析 PCML 檔
2. 使用者在他們 CLASSPATH 中只要有較少的 jar 檔即可執行應用程式

若要在發展期間剖析 PCML 檔，您需要 data.jar 或 jt400.jar 中有 PCML 執行時間，以及在 x4j400.jar 中有 P
剖析器。若要執行編序的應用程式，使用者只需要 jt400.jar。詳細資訊，請參閱 PCML 建置 iSeries
程式呼叫。

附註 7：請使用 jt400.jar 及 jt400Native.jar 代替 data400.jar。data400.jar 包含 PCML
執行期間類別，而且這些類別現在也在 jt400.jar 及 jt400Native.jar 中了。出貨時會提供
data400.jar，以保有與前版次 IBM Toolbox for Java 的相容性。

附註 8：ReportWriter 類別的副本位於一個以上的 jar 檔之中：

- composer.jar
- outputwriter.jar
- reportwriters.jar
- xslparser.jar
- x4j400.jar

如果您的應用程式會將 PCL 資料串流到一個 iSeries 排存檔，則您必須使用適當的 jar 檔（
讓存取類別成為可用。建立一個排存檔來保留 PCL 資料需要 AS400、OutputQueue、PrintParameterList 及
SpooledFileOutputStream 類別。詳細資訊，請參閱 ReportWriter 類別。

»附註 9：jt400Micro.jar 不包含執行 MEServer 所需的類別，這些類別常駐在 jt400.jar 及 jt400Native.jar
(附註) 中。jt400Micro.jar 只能從 Toolbox for Java and JTOpen 網站 取得。«

系統內容

您可指定系統內容以配置 IBM Toolbox for Java 的不同面貌。 例如，您可使用系統內容來定義 PROXY 伺服器或追蹤層次。
系統內容對於簡便的執行時間配置非常有用，而不需重新編譯程式碼。 當您在執行時間變更系統內容，系統內容的作業就像環境變數，
變更通常要等到下次執行應用程式時才會反映。

您可使用下列數種方式設定系統內容：

- 使用 java.lang.System.setProperties() 方法

  您可使用 java.lang.System.setProperties() 方法，程式化地設定系統內容。

  例如，下列程式碼將 com.ibm.as400.access.AS400.proxyServer 內容設定為 hqoffice ：

  ```
  Properties systemProperties = System.getProperties();
   systemProperties.put ("com.ibm.as400.access.AS400.proxyServer",
  "hqoffice");
   System.setProperties (systemProperties);
  ```

- 使用 Java 指令的 -D 選項

  許多環境可讓您使用 Java 指令的 -D 選項，在從指令行執行應用程式時設定系統內容。

  例如，下列程式執行名為 Inventory 的應用程式，並設定 com.ibm.as400.access.AS400.proxyServer 內容設定為 hqoffice ：

  ```
  java -Dcom.ibm.as400.access.AS400.proxyServer=hqoffice Inventory
  ```

- 使用 jt400.properties 檔案

  在部份環境中，可能不適宜指示所有的使用者設定其自有的系統內容。 另一個選擇方案為在 jt400.properties 檔案中指定 IBM Toolbox for Java
  系統內容， 將該檔案視為 com.ibm.as400.access 資料包的一部份加以搜尋。換言之， 將 jt400.properties 檔案置於 classpath 所指向的
  com/ibm/as400/access 目錄。

  例如，在 jt400.properties 檔案中插入下列程式行，以設定 com.ibm.as400.access.AS400.proxyServer 內容設定為 hqoffice ：

  ```
  com.ibm.as400.access.AS400.proxyServer=hqoffice
  ```

  反斜線字元 (\) 的功能如同內容檔案中的跳出字元。 使用雙反斜線 (\\) 指定文字反斜線字元。

  針對您的環境修改這個 jt400.properties 檔案的

- 使用 Properties 類別

  部份瀏覽器不載入沒有明確變更安全性設定值的內容檔案。 然而，大部份的瀏覽器都接受 .class 檔案中的內容， 因此，也可使用延伸
  java.util.Properties 的 com.ibm.as400.access.Properties 類別 來指定 IBM Toolbox for Java 系統內容。

  例如，若要將 com.ibm.as400.access.AS400.proxyServer 內容設定為 hqoffice ，請使用下列 Java 程式碼：

  ```
  package com.ibm.as400.access;

  public class Properties
  extends java.util.Properties
    {
  public Properties ()
        {
  put ("com.ibm.as400.access.AS400.proxyServer", "hqoffice");
        }
    }
  ```

  針對您的環境，修改及編譯此 Properties.java 原始檔的

若使用上述說明的多個機制來設定 IBM Toolbox for Java 系統內容， 則其優先順序如下 (依遞減的優先順序)：

1. 使用 java.lang.System.setProperties() 程式化地設定系統內容

2. 使用Java 指令的-D 選項設定系統內容
3. 使用 Properties 類別設定系統內容
4. 使用 jt400.properties 檔案設定系統內容

IBM Toolbox for Java 支援下列的系統內容：

PROXY 伺服器內容

PROXY 伺服器內容	說明
com.ibm.as400.access.AS400.proxyServer	使用下列格式指定 PROXY 伺服器主電腦名稱及埠號： hostName:portNumber 埠號是可選用的。
com.ibm.as400.access.SecureAS400.proxyEncryptionMode	使用 SSL 指定要加密的虛擬資料流程部份。有效值為： • 1 = Proxy 從屬站到 PROXY 伺服器 • 2 = PROXY 伺服器到 iSeries • 3 = Proxy 從屬站到 PROXY 伺服器及 PROXY 伺服器到 iSeries
com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval	指定 PROXY 伺服器尋找閒置中連線的頻率（以秒為單位）。PROXY 伺服器啟動一個緒來尋找不再通信的從屬站。使用此內容來設定緒尋找閒置中連線的頻率。
com.ibm.as400.access.TunnelProxyServer.clientLifetime	指定在 PROXY 伺服器移除對物件的參照之前，從屬站可閒置的時間（以秒為單位），使 JVM 可進行垃圾收集。PROXY 伺服器啟動一個緒來尋找不再通信的從屬站。 使用此內容以設定在從屬站上執行垃圾收集之前的閒置時間。

追蹤內容

追蹤內容	說明
com.ibm.as400.access.Trace.category	指定要啟用的追蹤種類。這是一個包含任何追蹤種類組合的逗點分界式清單。 完整的追蹤種類清單定義於 Trace 類別中。
com.ibm.as400.access.Trace.file	指定追蹤輸出寫入的檔案。預設為將追蹤輸出寫入 System.out。
com.ibm.as400.access.ServerTrace.JDBC	指定要在 JDBC 伺服器工作上啟動的追蹤種類。 有關支援值的詳細資訊，請參閱 JDBC 伺服器追蹤內容。

CommandCall/ProgramCall 內容

CommandCall/ProgramCall 內容	說明
com.ibm.as400.access.CommandCall.threadSafe	指定是否應將 CommandCall 假設為安全緒。若為 true， 則假設所有的 CommandCall 為安全緒。若為 false， 則假設所有的 CommandCall 為非安全緒。如果已於所指定的 CommandCall 物件上 執行 CommandCall.setThreadSafe(true/false) 或 AS400.setMustUseSockets(true) ，則系統不處理此內容。
com.ibm.as400.access.ProgramCall.threadSafe	指定是否應將 ProgramCall 假設為安全緒。若為true， 則假設所有的 ProgramCall 為安全緒。若為 false， 則假設所有的 ProgramCall 為非安全緒。如果已於所指定的 ProgramCall 物件上執行 CommandCall.setThreadSafe(true/false) 或 AS400.setMustUseSockets(true) ，則系統不處理此內容。

簡式程式設計範例

這些範例顯示您可以使用 IBM Toolbox for Java 類別來開始撰寫您自己的 Java 程式的一些方式。旨在供開始使用 Toolbox for Java 類別的程式設計師參考， 這些範例中包括程式碼中的關鍵行的詳細說明。

您可以使用下列方式來檢視詳細說明：

- 按一下 
  影像，以便在蹦現視窗中顯示詳細說明。(如果要在蹦現視窗中查看附註，您的瀏覽器必須支援 JavaScript。)
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

如果您需要入門協助，請參閱 撰寫第一支 Toolbox for Java 程式

如需取得連到 Toolbox for Java 資訊中提供的許多其它範例鏈結,請參閱 程式範例 。

請使用下列清單來檢視簡式的程式設計範例：

- 呼叫指令
- 使用訊息佇列
- 使用記錄層次存取
- 使用 JDBC 類別建立及輸入表格資料
- 在 GUI 中顯示伺服器工作清單

以下不保聲明適用於所有的 IBM Toolbox for Java 範例：

程式碼不保事項聲明

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
您可以依據這些範例，產生類似的函數，來符合您的需要。

IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM 不擔保或默示保證這些程式的可靠性、可用性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

IBM Toolbox for Java 類別

IBM Toolbox for Java 類別已分成套件 (如同所有 Java 類別)。
每個套件提供某種功能。為了方便起見，此說明文件通常使用簡短名稱
以參照每個套件。例如，com.ibm.as400.access 套件簡稱為存取套件。

請使用下列清單中的鏈結來尋找不同的 Toolbox for Java 套件中各種類別之相關資訊：

- 存取類別可讓您存取及管理您 iSeries 上的資源
- HTML 類別 可讓您快速建立 HTML 套表及表格
- Micro 類別可讓您建立提供無線裝置直接存取 iSeries 伺服器資料及服務的 Java 程式
- ReportWriter 類別可讓您自 XML 資料來源建立格式化文件
- 資源類別使用一般組織架構來存取及管理 iSeries 資源
- Security 類別確保伺服器的連線安全，並驗證在 iSeries 系統中工作的使用者身份
- Servlet 類別協助擷取及格式化資料以供在 Java Servlet 中使用
- 公用程式類別可使您執行管理作業，如使用 AS400ToolboxInstaller 類別及 AS400JarMaker 類別
- Vaccess 類別可讓您以目視方式呈現及操作資料

IBM Toolbox for Java 存取類別

IBM Toolbox for Java 存取類別代表 iSeries 資料與資源，會使用 iSeries 及 AS/400e 伺服器來提供一具有上網能力的介面，使您能夠存取與更新伺服器資料及資源。

下列類別提供 iSeries 及 AS/400e 資源的存取：

- AS400 - 管理登入資訊、建立與維護 socket 連線、傳送與接收資料
- SecureAS400 - 可讓您在傳送或接收加密資料時使用 AS400 物件
- AS400JPing - 可讓 Java 程式查詢主電腦伺服器，瞭解哪些服務程式正在執行，以及哪幾個埠正在使用中
- BidiTransform - 可讓您任意轉換雙向文字
- 叢集雜湊表類別 - 可讓 Java 程式在高可用性叢集雜湊表中的節點之間共用及抄寫非永久性的資料
- 指令呼叫 - 執行 iSeries 批次指令
- 連線儲存區 - 管理 AS400 物件的儲存區，您可以使用此儲存區來共用連線及管理一位使用者所可以擁有一個 iSeries 伺服器的連線數量。
- 資料區 - 建立、存取與刪除資料區
- 資料轉換與說明 - 轉換與處理資料，以及描述資料緩衝區的記錄格式
- 資料佇列 - 建立、存取、變更及刪除資料佇列
- 數位認證 - 管理 iSeries 伺服器上的數位認證
- 環境變數 - 管理 iSeries 環境變數
- 事件日誌 - 提供一個方法來記錄異常情況及訊息，此與用來顯示這些異常情況及訊息的裝置無關
- 異常情況 - 例如，當發生裝置錯誤或程式設計錯誤時丟棄錯誤
- FTP - 提供您使用 FTP 功能的可程式介面
- 整合檔案系統 - 存取檔案、開啟檔案、開啟輸入與輸出串流，及列出目錄內容
- Java 應用程式呼叫 呼叫執行於 iSeries Java 虛擬機器 (Java VM) 上之 iSeries 伺服器的 Java 程式
- JDBC - 存取 iSeries 資料的 DB2 UDB
- 工作 - 存取 iSeries 工作與工作日誌
- 訊息 - 存取 iSeries 系統上的訊息與訊息佇列
- NetServer 配置 存取及修改 iSeries NetServer 的狀態及配置
- 許可權 - 顯示及變更 iSeries 伺服器上的物件權限
- 列印 - 操作 iSeries 列印資源
- 產品授權 - 管理 iSeries 產品的授權
- 程式呼叫 - 呼叫 iSeries 程式
- QSYS 物件路徑名稱 - 代表 iSeries 整合檔案系統中的物件
- 記錄層次存取 - 建立、讀取、更新與刪除 iSeries 檔案及成員
- 服務程式呼叫 - 呼叫 iSeries 服務程式
- 系統狀態 顯示系統狀態資訊及容許存取系統儲存區資訊
- 系統值 擷取及變更系統值與網路屬性
- 追蹤 (可服務性) - 記載追蹤點與診斷訊息
- 使用者與群組 存取 iSeries 使用者及群組
- 使用者空間 存取 iSeries 使用者空間

註：Toolbox for Java 提供第二組類別，稱為資源類別，可與 iSeries 物件及清單並用。資源類別呈現一個同屬組織架構及用來使用各種 iSeries 物件及清單的一致性可程式設計介面。讀取套件及套件 中的類別之後，您可以選擇最適合您應用程式的物件。

AS400 類別

AS400 類別會管理下列項目：

- 一組在 iSeries 伺服器上，連線至 iSeries 伺服器工作的 socket。
- 伺服器的登入行為。這包括提示使用者登入資訊、密碼快取以及預設使用者管理。

當 Java 程式使用某個存取 iSeries 的類別實例時，Java 程式必須提供一個 AS400 物件。 例如，CommandCall 物件也必須擁有 AS400 物件才能傳送指令給 iSeries。

當 AS400 物件在 iSeries Java 虛擬機器中執行時，會以不同的方式處理連線、 使用者 ID 及密碼。詳細資訊，請參閱 iSeries Java 虛擬機器 (Java VM)。

»AS400 物件現在可以支援 Kerberos 鑑別，方法是使用 Java 同屬安全服務應用程式設計介面 (JGSS API) 代替使用者 ID 及密碼來鑑別伺服器。

註： 使用 Kerberos 通行證需先安裝 J2SDK v1.4 並配置「Java 同屬 安全服務 (JGSS) 應用程式設計介面」。如需 JGSS 的相關資訊，請參閱 J2SDK v1.4 安全性文件。 «

請參閱 管理連線， 取得如何透過 AS/400 物件來管理 iSeries 的連線的資訊。AS請參閱 連線儲存區， 取得如何藉由向連線儲存區要求連線來減少起始連線時間的資訊。

AS400 類別會提供下列登入功能：

- 鑑別 使用者設定檔
- »取得設定檔記號 機密並驗證相關的使用者設定檔 «
- »設定設定檔記號 機密 «
- 管理預設使用者 ID
- 快取密碼
- 使用者 ID 的提示
- 變更 密碼
- 取得 iSeries 版本及版次

如需傳送或接收加密資料時如何使用 AS400 物件的資訊，請參閱 SecureAS400 類別。

SecureAS400 類別

當AS400 物件與伺服器通信時，使用者資料（使用者密碼除外）會以未加密型式傳送至伺服器。所以，與 AS400 物件相關的 IBM Toolbox for Java 物件，會以一般的連線與伺服器交換資料。

當您要使用 IBM Toolbox for Java 來和伺服器交換機密資料時，可以使用 Secure Sockets Layer (SSL) 將資料加密。請使用 SecureAS400 物件來指定要加密的資料。與 SecureAS400 物件相關的 IBM Toolbox for Java 物件，會以安全連線與伺服器交換資料。

»如 需詳細資訊，請參閱 Secure Sockets Layer 與 Java Secure Socket Extension«

SecureAS400 類別是 AS400 類別 的次類別。

您可以經由下列方式，透過建立 SecureAS400 物件的案例，來設定安全伺服器連線：

- SecureAS400(String systemName, String user)會提示登入資訊
- SecureAS400(String systemName, String userID, String password)並不會提示登入資訊

下面範例顯示如何使用安全連接，以 CommandCall 傳送指令給 iSeries 系統：

```
// Create a secure AS400 object.This is the only statement that changes
// from the non-SSL case.
 SecureAS400 sys = new SecureAS400("mySystem.myCompany.com");

 // Create a command call object
 CommandCall cmd = new CommandCall(sys, "myCommand");

 // Run the commands.A secure connection is made when the
 // command is run.All the information that passes between the
 // client and server is encrypted.
 cmd.run();
```

AS400JPing 類別

AS400JPing 可讓 java 程式查詢主電腦伺服器，瞭解哪些服務程式正在執行， 以及哪幾個埠正在使用中。若要從指令行查詢伺服器，請使用JPing 類別。

AS400JPing 類別提供許多方法：

- 連通測試（ping）伺服器
- 連通測試（ping）伺服器上的特定服務程式
- 設定 PrintWriter 物件，此物件為您所要記載之連通測試（ping）資訊
- 為連通測試（ping）作設定 逾時

範例： 於 Java 程式中使用 AS400JPing 來連通測試（ping）「iSeries 遠端指令服務程式」：

```
AS400JPing pingObj = new AS400JPing("myAS400", AS400.COMMAND, false);
  if (pingObj.ping())
    System.out.println("SUCCESS");
   else
    System.out.println("FAILED");
```

AS400BidiTransform 類別

AS400BidiTransform 類別提供佈置轉換功能，可使您將 iSeries 格式的雙向文 字（於首先將它轉換成 Unicode 之後）轉換成 Java 格式的雙向文字， 或由 Java 格式轉換成 iSeries 格式。

AS400BidiTransform 類別可讓您：

- 取得並設定系統 CCSID
- 取得並設定 iSeries 資料的字串類型
- 取得並設定 Java 資料的字串類型
- 由 Java 佈置轉換資料 為 iSeries
- 由 iSeries 佈置轉換資料 為 Java

範例：使用 AS400BidiTransform 類別來轉換雙向文字

下列範例顯示您如何使用 AS400BidiTransform 類型來轉換雙向文字：

```
// Java data to iSeries layout:
AS400BidiTransform abt;
abt = new AS400BidiTransform(424);
String dst = abt.toAS400Layout("some bidirectional string");
```

ClusteredHashTable 類別

ClusteredHashTable 類別可讓 Java 程式使用高可用性的叢集雜湊表 以共用及抄寫資料至叢集中之節點間的非永久性儲存體。若要使用ClusteredHashTable 類別， 請確定您可以讓資料使用非永久性儲存體。抄寫的資料不會加密。

註： 下列資訊假設您已瞭解 iSeries 形成叢集的一般概念及術語。 關於叢集及如何使用它們的相關資訊，請參閱叢集。

使用 ClusteredHashTable 類別需要您事先在 iSeries 系統上定義及啟動叢集。您還必須啟動一個有表格伺服器的叢集。 詳細資訊，請參閱 Configurecluster 及叢集雜湊表 API。

需要的參數是叢集雜湊表伺服器及 AS400 物件的名稱，此代表了包含叢集雜湊表伺服器的系統。

為了將資料存放在叢集雜湊表伺服器中，您需要一個連線控點及一個金鑰：

- 當您開啟連線時，叢集雜湊表伺服器會指派供您後續對叢集雜湊表伺服器提出需求時， 必須指定的連線控點。此連線控點僅對案例化的 AS400 物件才有用， 所以如果您使用的是不同的 AS400 物件，則需開啟另一個連線。
- 您必須指定金鑰來存取及變更叢集雜湊表格中的資料。不支援重複金鑰。

ClusteredHashTable 類別提供讓您執行下列動作的方法：

- 對叢集雜湊表伺服器工作開啟連線
- 產生唯一的金鑰將資料存入叢集雜湊表
- 對叢集雜湊表伺服器工作關閉使用中連線

有些 ClusteredHashTable 類別中的方法使用 ClusteredHashTableEntry 類別以執行下列動作：

- 自叢集雜湊表取得項目
- 儲存項目於叢集雜湊表中
- 由叢集雜湊表為所有使用者設定權取得項目清單

下列範例操作於名為 CHTSVR01 的叢集雜湊表伺服器上。
它假設有個叢集及叢集雜湊表伺服器已在作用中。它會開啟連線、產生金鑰、使用新金鑰將項目置於叢集雜湊表中、由叢集雜湊表取得項目，並關閉連線。

```
    ClusteredHashTableEntry myEntry = null;

    String myData = new String("This is my data.");
    System.out.println("Data to be stored: " + myData);

AS400 system = new AS400();

    ClusteredHashTable cht = new ClusteredHashTable(system,"CHTSVR01");

    // Open a connection.
    cht.open();

    // Get a key to the hash table.
    byte[] key = null;
    key = cht.generateKey();

    // Prepare some data that you want to store into the hash table.
    // ENTRY_AUTHORITY_ANY_USER means that any user can access the
    // entry in the clustered hash table.
    // DUPLICATE_KEY_FAIL means that if the specified key already exists,
    // the ClusteredHashTable.put() request will not succeed.
    int timeToLive = 500;
    myEntry = new ClusteredHashTableEntry(key,myData.getBytes(),timeToLive,
         ClusteredHashTableEntry.ENTRY_AUTHORITY_ANY_USER,
         ClusteredHashTableEntry.DUPLICATE_KEY_FAIL);

    // Store (or put) the entry into the hash table.
    cht.put(myEntry);

    // Get an entry from the hash table.
    ClusteredHashTableEntry output = cht.get(key);

    // Close the connection.
```

```
        cht.close();
```

使用 ClusteredHashTable 類別會導致 AS400 物件連接伺服器。 詳細資訊，請參閱連線 《

指令呼叫

CommandCall 類別 可讓 Java 程式呼叫非交談式 iSeries 指令。指令的結果列於 AS400Message 物件清單中取得。

CommandCall 的輸入如下：

- 要執行的指令字串
- 代表將執行指令之系統的 AS400 物件

經由 setCommand() 方法， 或 run() 方法可以在建構元上設定指令字串。 執行指令之後，Java 程式可使用 getMessageList() 方法來擷取由指令產生的任何 iSeries 訊息。

使用 CommandCall 類別會導致 AS400 物件連線至 iSeries。

下列範例將告訴您如何使用 CommandCall 類別，執行 iSeries 系統中的指令：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a command call object. This
                    // program sets the command to run
                    // later. It could set it here on the
                    // constructor.
CommandCall cmd = new CommandCall(sys);

                    // Run the CRTLIB command
cmd.run("CRTLIB MYLIB");

                    // Get the message list which
                    // contains the result of the
                    // command.
AS400Message[] messageList = cmd.getMessageList();

                    // ... process the message list.

                    // Disconnect since I am done sending
                    // commands to the server
sys.disconnectService(AS400.COMMAND);
```

使用 CommandCall 類別會導致 AS400 物件連線至 iSeries。請參閱 管理連線 取得如何管理連線的資訊。

»當 Java 程式 及 iSeries 伺服器指令是在同一個伺服器上時，預設的 Toolbox for Java 行為將為指令檢查系統上的緒安全« 如果安全緒是安全的，則會在緒中執行指令。您可以制止於執行期間進行檢查， 方法是使用 setThreadSafe() 方法明確地指定指令的安全緒。

範例

執行使用者所指定的指令。

連線儲存區

使用連線儲存區可以共用連線及管理 iSeries 伺服器連線集合 (儲存區)。
例如應用程式可以從儲存區擷取連線、使用該連線，再將其傳回儲存區，以供重複使用。

AS400ConnectionPool 類別會管理 AS400 物件的儲存區。AS400JDBCConnectionPool 類別代表的
AS400JDBCConnections 儲存區，可以供 Java 程式用為 JDBC 2.0 Optional Package API 工具箱支援的≫部份。
JDBC ConnectionPool 介面在 JDBC 3.0 API 中也受支援， 該 API 與 Java 2 Platform 標準版，版本 1.4 相連結
≪

不管是哪一種類型的連線儲存區都會保持追蹤它建立的連線數量。 使用繼承自 ConnectionPool
的方法可以設定各種連線儲存區內容，包括：

- 一個儲存區可以提供的最大連線數量
- 連線的最大存留時間
- 連線的最大非作用時間

從效能方面來看，連接伺服器是高價的操作。使用連線儲存區可以藉由避免重複的連線時間來提升效能。
例如，當您藉由將作用中（已預先連線）的連線填滿儲存區來建立連線儲存區時建立連線。
您可以使用一個連線儲存區輕鬆地擷取、使用、傳回、及重新使用連線物件，而非建立新的連線。

≫使 用 AS400ConnectionPool 來擷取連線，方法是指定系統名稱、使用者 ID、 密碼、及 (選用性地) 服務程式。
≪ 若要指定您要連線的服務程式，請使用 AS400 類別中的常數 (FILE、PRINT, COMMAND 等)。

在擷取及使用連線之後，應用程式會將連線傳回儲存區。 將連線傳回儲存區以利重新使用是每一個應用程式的責任。
如果沒有將連線傳回儲存區，則連線儲存區的大小會繼續成長且無法重新使用連線。

請參閱 管理連線，取得當使用 AS400ConnectionPool 類別 來開啟 iSeries 連線時，管理連線的詳細資訊。

範例： 使用 AS400ConnectionPool 來重新使用 AS400 物件

資料區

DataArea 類別是一個摘要基礎類別，它代表 iSeries 資料區物件。
這個基礎類別具有四種次類別來支援：字元資料、小數資料、邏輯資料與含有字元資料的區域資料區。

使用 DataArea 類別，您可以執行下列：

- 取得資料區的大小
- 取得資料區的名稱
- 傳回資料區的 AS400 系統物件
- 重新整理資料區屬性
- 設定資料區所在的系統

使用 DataArea 類別會導致 AS400 物件連接伺服器。請參閱管理連線，取得如何管理連線的資訊。

CharacterDataArea

CharacterDataArea 類別代表伺服器上含有字元資料的資料區。 字元資料沒有以適當的 CCSID 標示資料的機能；因此資料區物件假設資料在使用者的 CCSID。
寫入時，於資料寫入到伺服器之前，資料區物件會從字串 (Unicode) 轉換為使用者的 CCSID。讀取時，於傳回字串給程式之前，資料區物件會假設資料是使用者的 CCSID，並從 CCSID 轉換為 Unicode。
當讀取資料區的資料時，讀取的資料量是以字元數來計算，而不是以位元組數目來計算。

使用 CharacterDataArea 類別，您可以執行下列：

- 清除資料區，以使它含有所有空白。
- 使用預設內容值，在系統建立一個字元資料區
- 以特定的屬性建立字元資料區
- 從資料區所在的系統中,刪除資料區
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 從資料區中位移 0 或您指定的位移讀取資料指定數量
- 設定資料區的完整整合檔案系統路徑名稱
- 將資料 寫入至資料區的開頭
- 寫入資料的指定數量至從位移 0 或您指定之位移開始的資料區

DecimalDataArea

DecimalDataArea 類別代表伺服器上含有小數資料的資料區。

使用 DecimalDataArea 類別，您可以執行下列：

- 清除資料區，使它含有 0.0
- 使用預設內容值，在系統建立一個小數資料區

- 以特定的屬性 建立小數資料區
- 從資料區所在的伺服器中，刪除資料區
- 傳回數字位數 至資料區中小數點的右邊
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 設定 資料區的完整整合檔案系統路徑名稱
- 將資料 寫入 至資料區的開頭

下列範例將告訴您如何建立及寫入到小數資料區：

```
      // Establish a connection to the server "My400".
    AS400 system = new AS400("MyServer");
      // Create a DecimalDataArea object.
    QSYSObjectPathName path = new QSYSObjectPathName("MYLIB", "MYDATA",
"DTAARA");
    DecimalDataArea dataArea = new DecimalDataArea(system, path.getPath());
      // Create the decimal data area on the server using default values.
    dataArea.create();
      // Clear the data area.
    dataArea.clear();
      // Write to the data area.
    dataArea.write(new BigDecimal("1.2"));
      // Read from the data area.
    BigDecimal data = dataArea.read();
      // Delete the data area from the server.
    dataArea.delete();
```

LocalDataArea

LocalDataArea類別代表伺服器上的區域資料區。
區域資料區雖是以字元資料資料區的形式存在於伺服器上，但是區域資料區具有一些您需要注意的限制。

區域資料區與伺服器工作相連，所以無法從一另工作中存取它。因此，您無法建立或刪除區域資料區。
當伺服器工作結束時，與伺服器工作相連的區域資料區將自動刪除，而且正在參照該工作的 LocalDataArea 物件不再有效。 您也應該注意區域資料區在伺服器上具有固定大小：1024 個字元。

使用 LocalDataArea 類別，您可以執行下列：

- 清除資料區，以使它含有所有空白
- 讀取資料區中含有的全部資料
- 從您指定之位移開始的資料區，讀取資料的指定數量
- 將資料 寫入 至資料區的開頭
- 寫入資料的 指定數量 至要寫入第一個字元來位移的資料區中

LogicalDataArea

LogicalDataArea類別代表伺服器上含有邏輯資料的資料區

使用 LogicalDataArea 類別，您可以執行下列：

- 清除資料區，以使之包含 false
- 使用預設內容值，在伺服器建立一個字元資料區
- 以 指定的屬性 建立字元資料區
- 從資料區所在的伺服器中，刪除資料區
- 傳回資料區所代表的物件整合檔案系統路徑名稱。
- 讀取資料區中含有的全部資料
- 設定資料區的完整整合檔案系統路徑名稱
- 將資料 寫入至資料區的開頭

DataAreaEvent

DataAreaEvent 類別代表資料區事件。

DataAreaEvent 類別可以與任一 DataArea 類別搭配使用。使用 DataAreaEvent 類別，您可以執行下列：

- 取得事件的ID

DataAreaListener

DataAreaListener類別提供接收資料區事件的介面。

DataAreaListener 類別可以與任一 DataArea 類別搭配使用。當執行下列任一項時， 您可以呼叫 DataAreaListener 類別：

- 清除
- 建立
- 刪除
- 讀取
- 寫入

資料轉換與說明

資料轉換 類別會提供在 iSeries 與 Java 格式之間轉換數值與字元的能力。 從 Java 程式存取 iSeries 資料時，可能需要轉換。資料轉換類別支援不同數值格式的轉換，及不同 EBCDIC 字碼頁與 Unicode 之間的轉換。

資料說明 類別會建置在資料轉換類別中，以便能夠透過單一方法呼叫，來轉換記錄中的所有欄位。RecordFormat 容許程式描述構成下列的資料：DataQueueEntry、ProgramCall 參數、透過記錄層次存取類別存取的資料庫檔案中的記錄， 或 iSeries 資料的任何緩衝區。「記錄」類別容許程式轉換記錄的內容並依欄位名稱或索引存取資料。

資料類型

AS400DataType 是一個介面，定義資料轉換時所需要的方法。 當需要轉換個別片段的資料時，Java 程式將使用資料類型。下列資料類型有轉換類別：

- 數字
- 文字 (字元)
- 組合 (數字與文字)

指定記錄格式的轉換

IBM Toolbox for Java 會提供依據資料類型類別而建置的類別，以一次一個記錄而非一次一個欄位方式，來處理資料的轉換。 例如，假設 Java 程式從資料佇列中讀取資料。資料佇列物件即會 傳回 iSeries 資料的位元組陣列給 Java 程式。此陣列可能含有多種類型的 iSeries 資料。應用程式可以使用資料類型類別，從位元組陣列中一次轉換一個欄位，或是程式可以建立一個記錄格式，指出以位元組陣列表示的欄位。然後，該記錄即會執行轉換。

當您使用來自程式呼叫、資料佇列與記錄層次存取類別的資料時，記錄格式轉換將非常有用。來自這些類別的輸入與輸出即是可含有多種不同類型的欄位的位元組陣列。透過記錄格式轉換器，可更輕易地在 iSeries 格式與 Java 格式之間轉換此資料。

透過記錄格式的轉換將使用下列三種類別：

- FieldDescription 類別以資料類型與名稱來識別欄位或參數。
- RecordFormat 類別描述一組欄位。
- Record 類別結合了記錄說明 (在 RecordFormat 類別中) 與實際資料。
- LineDataRecordWriter 類別會以行式資料格式將記錄寫入 OutputStream

範例

以下兩個範例說明使用含有資料佇列的記錄格式轉換：

- 範例： 使用 Record 與 RecordFormat 類別將資料置於佇列上
- 範例： 使用 FieldDescription、RecordFormat 與 Record 類別

數值資料轉換類別

數值資料轉換類別就只是將數值資料從 iSeries 或 AS/400e 伺服器格式 (在下列表格中稱作 伺服器格式) 轉換成 Java 格式。 支援的類型顯示在下表中：

數值類型	說明
AS400Bin2	在伺服器格式中的帶正負號之二位元組數字與 Java Short 物件之間的轉換。
AS400Bin4	在伺服器格式中的帶正負號之四位元組數字與 Java Integer 物件之間的轉換。
AS400ByteArray	在兩個位元組陣列之間轉換。這是很有用的, 因為轉換器會 正確地以零填入並填補目標緩衝區。
AS400Float4	在伺服器格式中的帶正負號之四位元組浮點數字與 Java Float 物件之間的轉換。
AS400Float8	在伺服器格式中的八位元組浮點數字與 Java Double 物件之間的轉換。
AS400PackedDecimal	在伺服器格式中的壓縮十進位數數字與 Java BigDecimal 物件之間的轉換。
AS400UnsignedBin2	在伺服器格式中的無正負號二位元組數字與 Java Integer 物件之間的轉換。
AS400UnsignedBin4	在伺服器格式中的無正負號四位元組數字與 Java Long 物件之間的轉換。
AS400ZonedDecimal	在伺服器格式中的區化十進位數數字與 Java BigDecimal 物件之間的轉換。

下面範例將顯示從伺服器格式內數字類型至 Java int 的轉換：

```
                    // Create a buffer to hold the server data
                    // type. Assume the buffer is filled with
                    // numeric data in the server format by data
                    // queues, program call, etc.
    byte[] data = new byte[100];

                    // Create a converter for this
                    // server data type.
    AS400Bin4 bin4Converter = new AS400Bin4();

                    // Convert from server type to Java
                    // object. The number starts at the
                    // beginning of the buffer.
    Integer intObject = (Integer) bin4Converter.toObject(data,0);

                    // Extract the simple Java type from
                    // the Java object.
    int i = intObject.intValue();
```

下面範例將顯示從 Java int 至伺服器格式內數字資料類型的轉換：

```
                    // Create a Java object that contains
                    // the value to convert.
    Integer intObject = new Integer(22);

                    // Create a converter for the server
```

```
                    // data type.
AS400Bin4 bin4Converter = new AS400Bin4();

                    // Convert from Java object to
                    // server data type.
byte[] data = bin4Converter.toBytes(intObject);

                    // Find out how many bytes of the
                    // buffer were filled with the
                    // server value.
int length = bin4Converter.getByteLength();
```

文字轉換

字元資料會經由AS400Text 類別轉換。此類別可在 EBCDIC 字碼頁及字集 (CCSID) 與 Unicode 之間轉換字元。當建構 AS400Text 物件時，Java 程式會指定要被轉換的字串長度及伺服器 CCSID 或編碼方式。 Java 程式的 CCSID 會假設為 13488 Unicode。toBytes() 方法會從 Java 套表轉換成 iSeries 格式的位元組陣列。toObject() 方法則從 iSeries 格式的位元組陣列轉換為 Java 格式。

AS400BidiTransform 類別提供佈置轉換功能，可讓您將 iSeries 格式的雙向文字 (在將它轉換成 Unicode 之後) 轉換成 Java 格式的雙向文字，或由 Java 格式轉換成 iSeries 格式。 預設的轉換會因工作的 CCSID 而定。若要改變文字的方向及形狀， 請指定 BidiStringType 請注意 IBM Toolbox for Java 物件執行內部地轉換位置 (如在 DataArea 類別中)， 它們會有一個置換字串類型的方法。例如，DataArea 類別具有 addVetoableChangeListener() 方法，可讓您以指定來接收特定內容的拒絕變更，包括字串類型。

例如，假定 DataQueueEntry 物件將以 EBCDIC 方式傳回 iSeries 文字。 下列範例會將這個資料轉換為 unicode，以便 Java 程式可以使用它：

```
        // ... Assume the data queue work has already been done to
        // retrieve the text from the iSeries and the data has been
        // put in the following buffer.
    int textLength = 100;
    byte[] data = new byte[textLength];

        // Create a converter for the iSeries data type. Note a default
        // converter is being built.  This converter assumes the iSeries
        // EBCDIC code page matches the client's locale.  If this is not
        // true the Java program can explicitly specify the EBCDIC
        // CCSID to use. However, it is recommended that you specify a
        // CCSID whenever possible (see the Notes: below).
    AS400Text textConverter = new AS400Text(textLength)

        // Note: Optionally, you can create a converter for a specific
        // CCSID. Use an AS400 object in case the program is running
        // as a Toolbox for Java proxy client.
    int ccsid = 37;
    AS400 system = ...; // AS400 object
    AS400Text textConverter = new AS400Text(textLength, ccsid, system);

        // Note: You can also create a converter with just the AS400
object.
        // This converter assumes the iSeries code page matches
        // the CCSID returned by the AS400 object.
    AS400Text textConverter = new AS400Text(textLength, system);

        // Convert the data from EBCDIC to Unicode. If the length of
        // the AS400Text object is longer than the number of
        // converted characters, the resulting String will be
        // blank-padded out to the specified length.
    String javaText = (String) textConverter.toObject(data);
```

複合類型轉換類別

組合類型的轉換類別如下：

- AS400Array - 容許 Java 程式使用資料類型的陣列
- AS400Structure- 可使 Java 程式使用其元素為資料類型的結構。

下面範例將顯示從 Java 結構至位元組陣列的轉換,
以及從後者至前者的轉換。範例中假定對傳送資料與接收資料使用相同的資料格式。

```
                    // Create a structure of data types
                    // that corresponds to a structure
                    // that contains:
                    //    - a four-byte number
                    //    - four bytes of pad
                    //    - an eight-byte number
                    //    - 40 characters
    AS400DataType[] myStruct =
    {
       new AS400Bin4(),
       new AS400ByteArray(4),
       new AS400Float8(),
       new AS400Text(40)
    };

                    // Create a conversion object using
                    // the structure.
    AS400Structure myConverter = new AS400Structure(myStruct);

                    // Create the Java object that holds
                    // the data to send to the server.
    Object[] myData =
    {
         new Integer(88),        // the four-byte number
         new byte[0],            // the pad (let the conversion object 0
pad)
         new Double(23.45),      // the eight-byte floating point number
         "This is my structure"  // the character string
    };

                    // Convert from Java object to byte array.
    byte[] myAS400Data = myConverter.toBytes(myData);


                    // ... send the byte array to the
                    // server. Get data back from the
                    // server.  The returned data will
                    // also be a byte array.


                    // Convert the returned data from
                    // iSeries to Java format.
    Object[] myRoundTripData =
```

```
            (Object[])myConverter.toObject(myAS400Data,0);

                  // Pull the third object out of the
                  // structure. This is the double.
Double doubleObject = (Double) myRoundTripData[2];

                  // Extract the simple Java type from
                  // the Java object.
double d = doubleObject.doubleValue();
```

欄位說明類別

欄位說明類別可使 Java 程式以資料類型及含有欄位名稱的字串，來說明欄位及參數的內容。
如果程式使用來自記錄層次存取的資料，它同時也可指定任何 iSeries 或 AS/400e 資料定義規格 (DDS) 關鍵字， 來進一步描述欄位。

下列是欄位說明類別：

- BinaryFieldDescription
- CharacterFieldDescription
- DateFieldDescription
- DBCSEitherFieldDescription
- DBCSGraphicFieldDescription
- DBCSOnlyFieldDescription
- DBCSOpenFieldDescription
- FloatFieldDescription
- HexFieldDescription
- PackedDecimalFieldDescription
- TimeFieldDescription
- TimestampFieldDescription
- ZonedDecimalFieldDescription

例如，假定資料佇列中的登錄具有相同格式。每一個登錄均具有訊息碼 (AS400Bin4)、 時間戳記 (8 個字元) 與訊息文字 (50 個字元)。這些可以透過如下的欄位說明來加以描述：

```
                        // Create a field description for
                        // the numeric data. Note it uses
                        // the AS400Bin4 data type. It also
                        // names the field so it can be
                        // accessed by name in the record
                        // class.
     BinaryFieldDescription bfd = new BinaryFieldDescription(new
AS400Bin4(),
                                                                "msgNumber");

                        // Create a field description for
                        // the character data. Note it uses
                        // the AS400Text data type. It also
                        // names the field so it can be
                        // accessed by name by the record
                        // class.
     CharacterFieldDescription cfd1 = new CharacterFieldDescription(new
AS400Text(8),

"msgTime");

                        // Create a field description for
                        // the character data. Note it uses
                        // the AS400Text data type. It also
                        // names the field so it can be
```

```
                        // accessed by name by the record
                        // class.
    CharacterFieldDescription cfd2 = new CharacterFieldDescription(new
AS400Text(50),

"msgText");
```

現在，欄位說明已可以群組在一個記錄格式類別中。範例將在 記錄格式段落中繼續。

RecordFormat 類別

RecordFormat 類別可讓 Java 程式說明欄位或參數的群組。記錄物件含有 RecordFormat 物件所描述的資料。如果程式將使用記錄層次存取類別，則 RecordFormat 類別亦會容許程式指定索引欄位的說明。

RecordFormat 物件含有一組欄位說明。欄位說明可透過索引或名稱來存取。 方法將存在於 RecordFormat 類別中，以進行下列動作：

- 將欄位說明 新增 至記錄格式。
- 新增鍵值 欄位說明至記錄格式。
- 依照索引或名稱，自記錄格式中擷取欄位說明。
- 依索引或依名稱，自記錄格式中擷取鍵值 欄位說明。
- 擷取組成記錄格式的欄位名稱
- 擷取組成記錄格式之鍵值欄位的名稱
- 於記錄格式中 擷取欄位數 。
- 於記錄格式中 擷取鍵值欄位數 。
- 依照此記錄格式,建立記錄物件。

例如，若要將欄位說明範例中所建立的欄位說明新增到記錄格式中：

```
                    // Create a record format object,
                    // then fill it with field
                    // descriptions.
    RecordFormat rf = new RecordFormat();
    rf.addFieldDescription(bfd);
    rf.addFieldDescription(cfd1);
    rf.addFieldDescription(cfd2);
```

程式現在準備依據記錄格式建立記錄。範例將在記錄段落中繼續。

Record 類別

record類別可讓 Java 程式處理記錄格式類別所述的處理資料。 資料會在含有伺服器資料及 Java 物件的位元組陣列間轉換。 將在記錄類別中提供方法，來執行下列動作：

- 依據索引或名稱 擷取欄位的內容，作為 Java 物件
- 記錄中擷取欄位數 。
- 依據索引或名稱，透過 Java 物件設定欄位的內容。
- 將記錄內容當成伺服器資料，擷取到位元組陣列或輸出串流
- 從位元組陣列或輸入串流設定記錄的內容。
- 將記錄內容 轉換為字串。

例如，使用記錄格式範例中建立的記錄格式：

```
                    // Assume data queue setup work has
                    // already been done. Now read a
                    // record from the data queue.
DataQueueEntry dqe = dq.read();

                    // The data from the data queue is
                    // now in a data queue entry. Get
                    // the data out of the data queue
                    // entry and put it in the record.
                    // We obtain a default record from
                    // the record format object and
                    // initialize it with the data from the
                    // data queue entry.
Record dqRecord = rf.getNewRecord(dqe.getData());

                    // Now that the data is in the
                    // record, pull the data out one
                    // field at a time, converting the
                    // data as it is removed. The result
                    // is data in a Java object that the
                    // program can now process.
Integer msgNumber = (Integer) dqRecord.getField("msgNumber");
String  msgTime   = (String)  dqRecord.getField("msgTime");
String  msgText   = (String)  dqRecord.getField("msgText");
```

LineDataRecordWriter 類別

LineDataRecordWriter類別會以行式資料格式將記錄資料寫入 OutputStream。 此類別會使用指定的 CCSID 將資料轉譯成位元組。與記錄關聯的記錄格式 會決定資料的格式。

使用 LineDataRecordWriter 需要您設定下列記錄格式屬性：

- 記錄格式 ID
- 記錄格式類型

與記錄或 RecordFormat 類別結合使用時，LineDataRecordWriter 會將一筆記錄做為writeRecord() 方法的輸入。(當您將記錄案例化時，記錄會以 RecordFormat 為輸入。)

LineDataRecordWriter 類別提供可讓您執行下列動作的方法：

- 取得 CCSID
- 取得編碼的名稱
- 以行式資料格式將記錄資料寫入 OutputStream

範例：使用 LineDataRecordWriter 類別

```
    // Example using the LineDataRecordWriter class.
    try
       {
         // create a ccsid
         ccsid_ = system_.getCcsid();

         // create output queue and specify spooled file data to be *LINE
         OutputQueue outQ = new OutputQueue(system_,
"/QSYS.LIB/RLPLIB.LIB/LDRW.OUTQ");
         PrintParameterList parms = new PrintParameterList();
         parms.setParameter(PrintObject.ATTR_PRTDEVTYPE, "*LINE");

         // initialize the record format for writing data
         RecordFormat recfmt = initializeRecordFormat();

         // create a record and assign data to be printed...
         Record record = new Record(recfmt);
         createRecord(record);

         SpooledFileOutputStream os = null;
     try {
             // create the output spooled file to hold the record data
             os = new SpooledFileOutputStream(system_, parms, null, outQ);
         }
   catch (Exception e) {
             System.out.println("Error occurred creating spooled file");
           e.printStackTrace();
         }
```

```
        // create the line data record writer
        LineDataRecordWriter ldw;
        ldw = new LineDataRecordWriter(os, ccsid_, system_);

        // write the record of data
        ldw.writeRecord(record);

        // close the outputstream
  os.close();
      }

catch (Exception e)
      {
        failed(e, "Exception occurred.");
      }
```

資料佇列

DataQueue 類別容許 Java 程式與伺服器資料佇列彼此互動。iSeries 及 AS/400e 伺服器上的資料佇列有下列特性：

- 資料佇列容許工作間的快速通信。 因此，它是一種在工作間進行資料同步與傳遞的非常好方式。
- 許多工作可同時存取資料佇列。
- 資料佇列上的訊息是沒有格式限制。不同於資料庫檔案，它不需要欄位。
- 資料佇列可用於同步與非同步處理。
- 資料佇列上的訊息可以按下列方式排序：
  - 後進先出（LIFO）。資料佇列的最後一個（最新）訊息是第一個離開佇列的訊息。
  - 先進先出（FIFO）。資料佇列的第一個（最舊）訊息是離開佇列的第一個訊息。
  - 索引式。資料佇列中的每一個訊息都有一個相關的索引。只能藉由指定與某訊息相關的索引才能從佇列中取出該訊息。

資料佇列類別提供一組完整介面，從 Java 程式存取伺服器資料佇列。它是一種在 Java 程式與以任何程式設計語言撰寫的伺服器程式之間進行通信的好方法。

每一個資料佇列物件的必要參數是 AS400 物件， 代表具有資料佇列或要建立資料佇列的伺服器系統。

使用資料佇列類別會導致 AS400 物件連接伺服器。請參閱管理連線， 取得關於管理連線的資訊。

每一個資料佇列物件都需要資料佇列的整合檔案系統路徑名稱。資料佇列的類型是 DTAQ。請參閱 整合檔案系統路徑名稱，以取得其餘相關資訊。

循序與索引資料佇列

資料佇列類別支援下列資料佇列：

- 循序資料佇列
- 索引資料佇列

這兩種佇列類型的共同方法是在 BaseDataQueue 類別中。 DataQueue 類別會擴充 BaseDataQueue 類別，以便完成循序資料佇列的實施。BaseDataQueue 類別是由KeyedDataQueue 類別加以擴充，以完成密碼鎖資料佇列的實施。

從某資料佇列讀取資料時，資料會存放在 DataQueueEntry 物件。此物件保留索引式與循序資料佇列的資料。從密碼鎖資料佇列中讀取資料時，其它可用的資料是放 在擴充 DataQueueEntry 類別的 KeyedDataQueueEntry 物件中。

資料佇列類別不變更寫入至或讀取自伺服器資料佇列的資料。Java 程式必須正確製作此資料格式。 資料轉換類別 會提供一些方法來轉換資料。

下列範例建立 DataQueue 物件，從 DataQueueEntry 物件讀取資料，然後切斷系統。

```
// Create an AS400 object
```

```
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create the DataQueue object
DataQueue dq = new DataQueue(sys, "/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

                    // read data from the queue
DataQueueEntry dqData = dq.read();

                    // get the data out of the DataQueueEntry object.
byte[] data = dqData.getData();

                    // ... process the data

                    // Disconnect since I am done using data queues
sys.disconnectService(AS400.DATAQUEUE);
```

循序資料佇列

依先進先出（FIFO）或後進先出（LIFO）順序來除去伺服器上循序資料佇列中的登錄。BaseDataQueue 及 DataQueue 類別提供下列方法，以使用循序資料佇列：

- 在伺服器上建立一個資料佇列。Java 程式必須指定此資料佇列中登錄的最大大小。建立此佇列時 Java 程式可選擇性地指定其它資料佇列參數（FIFO 對 LIFO、儲存傳送者資訊、指定權限資訊、強迫至磁碟以及提供佇列說明）。
- 察看資料佇列中的登錄，但不從佇列中除去登錄。如果目前在佇列中沒有登錄，則 Java 程式可等待或立即返回。
- 從佇列中讀出登錄。如果佇列中沒有登錄，則 Java 程式可等待或立即返回。
- 將登錄寫入到佇列。
- 從佇列清除所有登錄。
- 刪除佇列。

BaseDataQueue 類別會提供其它方法來擷取資料佇列的屬性。

範例

循序資料佇列範例，在這個範例中，生產者將項目置於資料佇列，消費者則自佇列中取出項目，然後處理它們：

- 循序資料佇列生產者範例
- 循序資料佇列消費者範例

金鑰資料佇列

BaseDataQueue 及KeyedDataQueue 類別提供下列方法，以使用金鑰資料佇列：

- 在伺服器上建 立一個金鑰資料佇列。Java 程式必須指定索引長度與佇列中的最大登錄大小。Java 程式可選擇性地指定權限資訊、儲存傳送者資訊、強迫至磁碟以及提供佇列說明。
- 依照指定的金鑰察 看某一登錄，而不將它從佇列中移除。如果目前在佇列中沒有符合索引準則的登錄，則 Java 程式可等待或立即返回。
- 依照指定的金鑰，讀 取佇列中的登錄。如果在佇列中沒有符合索引準則的登錄，則 Java 程式可等待或立即返回。
- 寫入索引 登錄到佇列中。
- 清除所有登錄或所有符合指定金鑰的登錄。
- 刪除佇列。

BaseDataQueue 和 KeyedDataQueue 類別也提供其它方法來擷取資料佇列的屬性。

範例

在下列索引式資料佇列範例中，生產者將項目置於資料佇列，消費者則自佇列中取出項目，然後處理它們：

- 索引資料佇列生產者範例
- 索引資料佇列消費者範例

數位認證

數位認證是以數字簽署的聲明，用於保護透過網際網路的交易。(數位認證可在執行 OS/400 版本 4 版次 3 (V4R3) 及更新版本的伺服器中使用)。 若要使用 Secure Sockets Layer (SSL) 以使連線安全，將需要數位認證。

數位認證是由下列所構成：

- 使用者的公用暗碼鍵
- 使用者的名稱與位址
- 協力廠商資格認定權限 (CA) 的數位簽名。 權限的簽名表示使用者是可靠的實體。
- 認證的發出日期
- 認證的有效日期

作為安全伺服器的管理員，您可以對伺服器新增資格認定權限的「授信單位鍵」。
這表示您的伺服器將信任已透過該特殊資格認定權限所認證的任何人員。

數位認證也會提供暗碼，透過專用暗碼鍵確保資料轉送的安全。

您可以透過 javakey 工具來建立數位認證。(若需 javakey 與 Java 安全的其餘相關資訊，請參閱 Sun Microsystems, Inc., Java Security 頁面。) IBM Toolbox for Java 授權程式具有管理 iSeries 或 AS/400e 伺服器中的數位認證的類別。

AS400Certificate 類別會提供不同方式來管理 X.509 ASN.1 編碼認證。提供的類別會執行下列事項：

- 取得與設定認證資料。
- 依驗證清單或使用者設定檔列示認證。
- 管理認證，例如，對使用者設定檔新增認證，或從驗證清單中刪除認證。

使用認證類別將導致 AS400 物件連線至伺服器。請參閱 管理連線，取得關於管理連線的資訊

在伺服器上，認證屬於驗證清單或使用者設定檔。

- AS400CertificateUserProfileUtil 類別具有管理使用者設定檔中認證的方法。
- AS400CertificateVldlUtil 類別具有管理驗證清單中認證的方法。

這兩種類別擴充了 AS400CertificateUtil，這是一個摘要基礎類別，定義這兩種次類別共用的方法。

AS400Certificate 類別提供讀取及寫入認證資料的方法。 資料是以位元組陣列方式存取。Java 虛擬機器 1.2 中的 Java.Security 資料包提供可以用來取得及設定認證個別欄位的類別。

列示認證

若要取得認證的清單，Java 程式必須執行下列事項：

1. 建立 AS400 物件。
2. 建構正確的認證物件。不同物件係針對使用者設定檔中的列示驗證 (AS400CertificateUserProfileUtil) 及驗證清單中的列示認證 (AS400CertificateVldlUtil) 而使用。
3. 依據認證屬性建立選取準則。 AS400CertificateAttribute 類別包含做為選取準則使用的屬性。
   一或多個屬性物件會定義在認證新增到清單中之前必須符合的準則。例如，清單可能僅含有某個使用者或組織的認證。
4. 在伺服器上建立一個 使用者空間，並將認證置於該使用者空間中。 大量資料可經由列示作業來產生。在 Java 程式可以存取資料之前，該資料將置於使用者空間中。 使用 listCertificates() 方法，將認證置於使用者空間中。
5. 使用 getCertificate() 方法，自使用者空間擷取認證。

下面範例會列出驗證清單中的認證。 它僅會列示那些屬於某人的認證。

```
                    // Create an AS400 object.  The
                    // certificates are on this system.
    AS400 sys = new AS400("mySystem.myCompany.com");


                    // Create the certificate object.
    AS400CertificateVldlUtil certificateList =
            new AS400CertificateVldlUtil(sys,
"/QSYS.LIB/MYLIB.LIB/CERTLIST.VLDL");


                    // Create the certificate attribute
                    // list. We only want certificates
                    // for a single person so the list
                    // consists of only one element.
    AS400CertificateAttribute[] attributeList = new
AS400CertificateAttribute[1];
    attributeList[0] = new
AS400CertificateAttribute(AS400CertificateAttribute.SUBJECT_COMMON_NAME,
"Jane Doe");


                    // Retrieve the list that matches
                    // the criteria. User space "myspace"
                    // in library "mylib" will be used
                    // for storage of the certificates.
                    // The user space must exist before
                    // calling this API.
    int count = certificateList.listCertificates(attributeList,

"/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC");


                    // Retrieve the certificates from
                    // the user space.
    AS400Certificates[] certificates =
certificateList.getCertificates("/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC", 0, 8);


                    // ... process the certificates
```

EnvironmentVariable 類別

EnvironmentVariable 類別及EnvironmentVariableList 類別讓您存取及設定 iSer系統層次的環境變數。

每個變數都有唯一的 ID：系統名稱及環境變數名稱。每個環境變數都會連結一個 CCSID，而且是預設成現行作業的 CCSID，用來說明存放變數內容的位置。

註： 環境變數不同於系統值，雖然它們的用途經常相同。 請參閱 SystemValues，以取得如何存取系統值的詳細資訊。

使用 EnvironmentVariable 物件以對環境變數執行下列動作：

- 取得及設定名稱
- 取得及設定系統
- 取得及設定值 (可讓您變更 CCSID)
- 重新整理值

範例：建立、設定及取得環境變數

下列範例會建立兩個 EnvironmentVariables，並且設定及取得它們的值。

```
// Create the iSeries system object.
AS400 system = new AS400("mySystem");
// Create the foreground color environment variable and set it to red.
EnvironmentVariable fg = new EnvironmentVariable(system, "FOREGROUND");
fg.setValue("RED");
// Create the background color environment variable and get its value.
EnvironmentVariable bg = new EnvironmentVariable(system, "BACKGROUND");
String background = bg.getValue();
```

異常情況

發生裝置錯誤、實體限制、程式設計錯誤或使用者輸入錯誤時，IBM Toolbox for Java 存取類別會擲出異常。根據發生的錯誤類型而不是出現錯誤的位置來決定異常情況類別。

大部份的異常含有三種資訊：

- 錯誤類型 -
  擲出的異常情況物件會指出發生的錯誤類型。同一類型的錯誤會在某異常情況類別中組成群組。
- 錯誤明細 -
  此異常情況包含一回覆碼來進一步識別發生的錯誤的原因。回覆碼值是異常情況類別中的常數。
- 錯誤文字 - 此異常情況包含說明發生的錯誤的文字字串。以從屬站 Java 虛擬機器的語言環境來轉換該字串

下面範例會顯示如何攔截擲出的異常、擷取回覆碼及顯示異常文字：

```
                    // ... all the setup work to delete
                    // a file on the server through the
                    // IFSFile class is done. Now try
                    // deleting the file.
    try
    {
       aFile.delete();

    }

                    // The delete failed.
    catch (ExtendedIOException e)
    {
                    // Display the translated string
                    // containing the reason that the
                    // delete failed.
       System.out.println(e);

                    // Get the return code out of the
                    // exception and display additional
                    // information based on the return
                    // code.

int rc = e.getReturnCode()

       switch (rc)
       {
          case ExtendedIOException.FILE_IN_USE:
             System.out.println("Delete failed, file is in use "):
             break;

          case ExtendedIOException.PATH_NOT_FOUND:
             System.out.println("Delete failed, path not found ");
             break;

                    // ... for every specific error you
                    // want to track
```

```
            default:
               System.out.println("Delete failed, rc = ");
               System.out.println(rc);
         }
      }
```

FTP 類別

FTP 類別 提供您使用 FTP 功能的可程式介面。您不再需要使用
java.runtime.exec()，或告訴您的使用者在個別的應用程式 中執行 FTP 指令。亦即，您可以直接對您的應用程式
FTP 功能的程式設計。 所以，在您的程式中，您可以執行下列：

- 連接 FTP 伺服器
- 傳送指令到伺服器
- 列出目錄中 的檔案
- 取得伺服器中的檔案及
- 放入檔案到伺服器內

例如，利用 FTP 類別，您可以從伺服器中的某一目 錄複製 一組檔案：

```
FTP client = new FTP("myServer", "myUID", "myPWD");
client.cd("/myDir");
client.setDataTransferType(FTP.BINARY);
String [] entries = client.ls();

for (int i = 0; i < entries.length; i++)
{
  System.out.println("Copying " + entries[i]);
try
  {
    client.get(entries[i], "c:\\ftptest\\" + entries[i]);
  }
catch (Exception e)
  {
    System.out.println("  copy failed, likely this is a directory");
  }
}

client.disconnect();
```

FTP 是使用許多不同 FTP 伺服器的同屬介面。 因此，程式設計師可以決定伺服器的語意。

FTP 次類別

當 FTP 類別是同屬 FPT 介面時，則會明確地寫入伺服器上 FTP 伺服器的AS400FTP subclass。
亦即，它瞭解 iSeries 或 AS/400e 伺服器上 FTP 伺服器的語意，
所以程式設計師不需要撰寫。例如，此類別瞭解將儲存檔轉送到伺服器時所需要的各種步驟，
必會自動執行這些步驟。AS400FTP 也結合了 IBM Toolbox for Java 的安全機能。在使用其 它 IBM Toolbox for J
類別時，AS400FTP 會視 AS400 物件的不同而要求系統名稱、使用者 ID 及密碼。

下列範例會將儲存檔放入伺服器。 請注意應用程式並沒有將資料轉送類型設定為二進位，或使 用 Toolbox
CommandCall 來建立儲存檔。既然副檔名是 .savf，AS400FTP 類別
會偵測到要放入的檔案是一個儲存檔，所以它會自動執 行這些步驟。

```
AS400 system = new AS400();
 AS400FTP ftp = new AS400FTP(system);
 ftp.put("myData.savf", "/QSYS.LIB/MYLIB.LIB/MYDATA.SAVF");
```

整合檔案系統

整合檔案系統類別可讓 Java 程式存取 iSeries 或 AS/400e 伺服器整合檔案系統中的檔案，作 為位元組串流或字元串流。因為 java 資料包沒有提供檔案重新導向以及其它 iSeries 功 能，所以建立整合檔案系統類別。

IFSFile 類別提供的功能是由 java.io 資料包中的檔案 IO 類別提供的功能超集。java.io FileInputStream、FileOutputStream 以及 RandomAccessFile 內的 所有方法都是在整合檔案系統類別之內。

除了這些方法以外，這些類別還包含一些方法來執行下列動作：

- 指定檔案共用模式來拒絕存取使用中的檔案
- 指定檔案建立模式來開啟、建立或置換此檔案
- 鎖定檔案的某區段並在使用該檔案時拒絕該部份的存取
- 更有效率地列示目錄的內容
- 藉由限制呼叫伺服器來快取目錄的內容，以增進效能
- 決定伺服器檔案系統中可用的位元組數
- 讓 Java applet 存取伺服器檔案系統中的檔案
- 以文字而非二進位資料來讀取與寫入資料
- 決定當物件於 QSYS.LIB 檔案系統中時，該檔案物件的類型（邏輯、實體、儲存等）。

透過整合檔案系統類別，Java 程式可直接存取 iSeries 中的串流檔。Java 程 式仍可使用 java.io 資料包，但從屬站作業系統必須提供重新導向的方法。 例如，如果在 Windows 95 或 Windows NT 作業系統中執行 Java 程式， 則需 iSeries Access for Windows 的「網路磁碟機」功能， 才能將 java.io 呼叫重新導向至 iSeries。透過整合檔案系統類別， 您就不 iSeries Access for Windows。

整合檔案系統類別的必要參數之一是 AS400 物件， 物件代表包含此檔案的 iSeries 系統。使用整合檔案系統類別會使 AS400 物件連線至 iSeries。 請參閱管理連線 取得關於管理連線的資訊

整合檔案系統類別需要此物件在整合檔案系統中的階層式名稱。使用正斜線作為路徑分隔字元。下面範例將告訴您如何存取目錄路徑 DIR1/DIR2 中的 FILE1：

```
/DIR1/DIR2/FILE1
```

整合檔案系統類別如下。

整合檔案系統類別	說明
IFSFile	代表在整合檔案系統中的檔案
IFSJavaFile	代表在整合檔案系統中的檔案 (擴充 java.io.File)
IFSFileInputStream	代表從 iSeries 檔案中讀取資料的輸入串流
IFSTextFileInputStream	代表從檔案中讀取的字元資料的串流
IFSFileOutputStream	代表將資料寫入到 iSeries 檔的輸出串流
IFSTextFileOutputStream	代表將寫入到檔案中的字元資料的串流
IFSRandomAccessFile	代表 iSeries 中用來讀取及寫入資料的檔案
IFSFileDialog	容許使用者在檔案系統內移動，以及在檔案系統內選取檔案

範例

IFSCopyFile 範例將告訴您如何使用整合檔案系統類別， 將 iSeries 中的某個目錄內的檔案複製到另一個目錄中。

檔案清單範例 將告訴您如何使用整合檔案系統類別， 列出 iSeries 中某個目錄的內容。

IFSFile 類別

IFSFile類別代表在 iSeries 整合檔案系統中的物件。IFSFile 的方法代表對整個物件所執行的一些作業。您可使用
IFSFileInputStream、IFSFileOutputStream 以及 IFSRandomAccessFile 來讀取和寫入此檔案。IFSFile
類別可讓 Java 程式執行下列動作：

- 確定物件是否 存在 而且 是一個 目錄 或檔案
- 確定 Java 程式是否可以從檔案 讀取或寫入 檔案
- 確定檔案的 長度
- 確定物件的 許可 權並設 定物件的許可權
- 建立 目錄
- 刪除檔案或目錄
- 更名檔案或目錄
- 取得或設定檔案的前次修改日期
- 列示目錄內容
- 列出目錄內容並將屬性資訊存入本端快取
- 決定系統上的可用空間量
- 決定當檔案物件位於 QSYS.LIB 檔案系統中時， 該 檔案物件的類型

您可以使用list() 方法或 listFiles() 來取得目錄中的檔案清單：

- listFiles() 方法會透過初始呼叫快取每個檔案的資訊。在呼叫 listFiles() 後，
  使用其它方法來查詢檔案明細會有較佳的效能，因為資料是擷取自快取。 例如，呼叫 isDirectory() 取得
  listFiles() 傳回的 IFSFile 物件不需要 呼叫伺服器。
- list() 方法會透過向伺服器發出個別要求來擷取每個檔案的相關資訊，
  使它的執行速度較慢，而且需要更多伺服器資源。

註： 使用 listFiles() 表示在快取中的資料已變舊， 您必須藉由再次呼叫 listFiles() 來重新整理資料。

範例

下列範例說明如何使用 IFSFile 類別：

- 範例： 建立目錄
- 範例： 使用異常來追蹤錯誤
- 範例： 列出具有 .txt 副檔名的檔案
- 範例： 使用 listFiles() 列出目錄內容

IFSJavaFile 類別

IFSJavaFile類別代表 iSeries 整合檔案系統中的檔案，並延伸 java.io.File 類別。IFSJavaFile 可讓您寫入 java.io.File 介面的檔案，以存取 iSeries 整合檔案系統。

IFSJavaFile 將產生與 java.io.File 相容的可攜性介面，並僅使用 java.io.File 使用的錯誤與異常情況。IFSJav 會使用來自 java.io.File 的 安全管理程式特性，但不像 java.io.File，IFSJavaFile 會不斷地使用安全特性。

IFSJavaFile 可以搭配 IFSFileInputStream 及 IFSFileOutputStream 一起使用。它不支援 java.io.FileInputStr java.io.FileOutputStream。

IFSJavaFile 是以 IFSFile 為基礎；它的介面比 IFSFile 更像 java.io.File。IFSFile 是 IFSJavaFile 的替代類

您可以使用 list() 方法或 listFiles() 方法，以取得目錄中的檔案清單：

- listFiles() 方法會執行得比較好， 因為它會在起始呼叫中擷取及快取每一個檔案的資訊。之後，則會從快取記憶體中擷取每一個檔案的相關資訊。
- list() 方法會在個別要求中擷取每一個檔案的相關資訊，這使得它的執行速度較慢且需要更多伺服器資源。

註： 使用 listFiles() 表示在快取中的資訊最後會變舊， 所以您可能需要重新整理資料。

下面是如何使用 IFSJavaFile 類別的範例。

```
// Work with /Dir/File.txt on the system flash.
AS400 as400 = new AS400("flash");
IFSJavaFile file = new IFSJavaFile(as400, "/Dir/File.txt");


// Determine the parent directory of the file.
String directory = file.getParent();


// Determine the name of the file.
String name = file.getName();


// Determine the file size.
long length = file.length();


// Determine when the file was last modified.
Date date = new Date(file.lastModified());


// Delete the file.
if (file.delete() == false)
{
  // Display the error code.
  System.err.println("Unable to delete file.");
}
```

```
   try
{
  IFSFileOutputStream os = new IFSFileOutputStream(file.getSystem(),
                               file,
                               IFSFileOutputStream.SHARE_ALL,
                                        false);
  byte[] data = new byte[256];
  int i = 0;
  for (; i < data.length; i++)
  {
    data[i] = (byte) i;
    os.write(data[i]);
  }
  os.close();
}
catch (Exception e)
{
  System.err.println ("Exception: " + e.getMessage());
}
```

IFSFileInputStream

IFSFileInputStrea類別代表一種輸入串流，可以在伺服器上讀取檔案中的資料。 如同在 IFSFile 類別中，IFSFileInputStream 中的方法從 java.io 資料包 中複製 FileInputStream 的方法。除了這些方法外，IFSFileInputStream 有其它專供 iSeries 及 AS/400e 伺服器使用的方法。IFSFileInputStream 類別可讓 Jav 程式執行下列動作：

- 開啟 檔案以讀取。因為此類別不會在伺服器上建立檔案，所以檔案必須存在。您可以使用建構元，它可讓您指定檔案共用模式。
- 決定串流中的位元組數。
- 讀取串流中的位元組。
- 略過串流中的位元組。
- 鎖定 或解除 鎖定 串流中的位元組。
- 關閉檔案。

如同在 java.io 中的 FileInputStream 中，此類別可讓 Java 程式讀取此檔 案的位元組串流。Java 程式以略過串流中位元組的唯一附加選項來循序讀取位元組。

下面範例說明如何使用 IFSFileInputStream 類別。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file.
IFSFileInputStream aFile =
              new IFSFileInputStream(sys,"/mydir1/mydir2/myfile");

                    // Determine the number of bytes in
                    // the file.
int available = aFile.available();

                    // Allocate a buffer to hold the data
byte[] data = new byte[10240];

                    // Read the entire file 10K at a time
for (int i = 0; i < available; i += 10240)
{
aFile.read(data);
}

                    // Close the file.
aFile.close();
```

除了 FileInputStream 中的方法以外，IFSFileInputStream 提供 Java 程式 下列選項：

- 鎖定和解除鎖定串流的位元組。如需相關資訊，請參閱 IFSKey。
- 開啟檔案之後指定共用模式。如需相關資訊，請參閱 共用模式。

IFSTextFileInputStream 類別

IFSTextFileInputStream類別代表從檔案讀取的字元資料串流。 讀取自 IFSTextFileInputStream 物件的資料會提供給「Java 字串」物件中的 Java 程式，所以它一定是 Unicode。開啟檔案之後，IFSTextFileInputStream 物件會判斷檔案中的資料的 CCSID。如果資料是以 Unicode 以外的編碼儲存， 則 IFSTextFileInputStream 物件會在提供資料給 Java 程式之前，先將資料從檔案編碼轉換成 Unicode。 如果無法轉換資料，則會丟出 UnsupportedEncodingException。

下面範例說明如何使用 IFSTextFileInputStream：

```
                    // Work with /File on the system
                    // mySystem.
    AS400 as400 = new AS400("mySystem");
    IFSTextFileInputStream file = new IFSTextFileInputStream(as400,
"/File");

                    // Read the first four characters of
                    // the file.
    String s = file.read(4);

                    // Display the characters read. Read
                    // the first four characters of the
                    // file. If necessary, the data is
                    // converted to Unicode by the
                    // IFSTextFileInputStream object.
    System.out.println(s);

                    // Close the file.
    file.close();
```

IFSFileOutputStream

IFSFileOutputStream類別代表寫入資料到伺服器中的檔案的輸出串流。如 同在 IFSFile 類別中，IFSFileOutputStream 中的方法從 java.io 資料包中複製 FileOutputStream 的方法。IFSFileOutputStream 也有專供伺服器使用的其它方法。IFSFileOutputStream 類別可讓 Java 動態式執行下列

- 開啟檔案以寫入。如果此檔案已存在，則會置換該檔案。您可使用建置器來指定檔案共用模式，以及是否已經附加現有檔案的內容。
- 寫入位元組到串流。
- 確定 位元組寫入串流的磁碟。
- 鎖定 或解除 鎖定 串流中的位元組。
- 關閉檔案。

如同在 java.io 中的 FileOutputStream，此類別可讓 Java 程式循序地 寫入位元組串流到檔案。

下面範例說明如何使用 IFSFileOutputStream 類別。

```
                    // Create an AS400 object
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file.
IFSFileOutputStream aFile =
             new IFSFileOutputStream(sys,"/mydir1/mydir2/myfile");

                    // Write to the file
byte i = 123;
   aFile.write(i);

                    // Close the file.
aFile.close();
```

除了 FileOutputStream 中的方法以外，IFSFileOutputStream 還提供 Java 程式下列選項：

- 鎖定和解除鎖定串流的位元組。請參閱 IFSKey 以取得其餘相關資訊。
- 開啟檔案之後指定共用模式。請參閱 共用模式以取得其餘相關資訊。

IFSTextFileOutputStream 類別

IFSTextFileOutputStream類別代表寫入檔案的字元資料串流。 提供給 IFSTextFileOutputStream 物件的資料是在「Java 字串」物件中， 所以輸入一律是 Unicode。不過當資料寫入檔案時，IFSTextFileOutputStream 物件可轉換此資料為 另一個 CCSID。預設行為是將 Unicode 字元寫入檔案， 但 Java 程式可以在開啟檔案之前，先設定目標 CCSID。在此情況下，IFSTextFileOutputStream 物件會先將字元從 Unicode 轉換成指定的 CCSID，然後才將它們寫入檔案。如果無法轉換資料，則會丟出 UnsupportedEncodingException。

下面範例說明如何使用 IFSTextFileOutputStream：

```
                        // Work with /File on the system
                        // mySystem.
    AS400 as400 = new AS400("mySystem");
    IFSTextFileOutputStream file = new IFSTextFileOutputStream(as400,
"/File");

                        // Write a String to the file.
                        // Because no CCSID was specified
                        // before writing to the file,
                        // Unicode characters will be
                        // written to the file. The file
                        // will be tagged as having Unicode
                        // data.
    file.write("Hello world");

                        // Close the file.
file.close();
```

IFSRandomAccessFile

IFSRandomAccessFile 類別代表在伺服器中用來讀取和寫入資料的檔案。Java 程式可循序或隨機地讀取及寫入資料。如 同在 IFSFile 中, IFSRandomAccessFile 中的方法從 java.io 資料包中複製 RandomAccessFile 的方法。除了這些方 法以外, IFSRandomAccessFile 尚有其它專供 iSeries 或 AS/400e 伺服器使用的方法。 透過 IFSRandomAccessFile, Java 程式可執行下列動作：

- 開啟 檔案供讀取、寫入或讀取/寫入存取。Java 程式可選擇性地指定檔案共用模式與存在選項。
- 從現行位移讀檔案中的資料。
- 從現行位移將資料寫入檔案。
- 取得或設 定檔案的現行位移。
- 關閉檔案。

下面範例將告訴您如何使用 IFSRandomAccessFile 類別, 以 1K 四個位元組寫入到檔案中。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that represents
                    // the file.
IFSRandomAccessFile aFile =
              new IFSRandomAccessFile(sys,"/mydir1/myfile", "rw");

                    // Establish the data to write.
byte i = 123;

                    // Write to the file 10 times at 1K
                    // intervals.
for (int j=0; j<10; j++)
{
                    // Move the current offset.
   aFile.seek(j * 1024);

                    // Write to the file. The current
                    // offset advances by the size of
                    // the write.
   aFile.write(i);
}

                    // Close the file.
aFile.close();
```

除了 java.io RandomAccessFile 中的方法以外, IFSRandomAccessFile 還提 供 Java 程式下列選項：

- 確定 已寫入的磁碟位元組。
- 鎖定 或解除 鎖定 檔案中的位元組。
- 鎖定和解除鎖定串流的位元組。請參閱 IFSKey 以取得其餘相關資訊。
- 開啟檔案之後指定共用模式。請參閱 共用模式以取得其餘相關資訊。
- 開啟檔案之後指定存在選項。Java 程式可選取下列其中一項：
    - 如果檔案存在, 則開啟此檔案；如果檔案不存在, 則建立此檔案。
    - 如果檔案存在, 則置換此檔案；如果檔案不存在, 則建立此檔案。

- 如果檔案存在，則無法開啟；如果檔案不存在，則建立此檔案。
- 如果檔案存在，則開啟此檔案；如果檔案不存在，則無法開啟。
- 如果檔案存在，則置換此檔案；如果檔案不存在，則無法開啟。

IFSFileDialog

IFSFileDialog類別可讓您瀏覽檔案系統並選取檔案。 此類別使用 IFSFile 類別，以在 iSeries 或 AS/400e 伺服器的整合檔案系統中瀏覽目錄及檔案清單。此類別中的方法可讓 Java 程式設定關於對話按鈕的文字以及設定過濾。 請注意，也可以使用依據 Swing 的 IFSFileDialog類別。

您可以透過FileFilter類別設定過濾程式。如果使用者在對話框中選取檔案， 可以使用getFileName() 方法來取得已選取的檔案名稱。 可以使用getAbsolutePath()方法以取得已選取的檔案路徑與名稱。

下面範例說明如何設定具有兩個過濾程式的對話，以及設定關於對話按鈕的文字。

```
                    // Create an AS400 object.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a dialog object setting
                    // the text of the dialog's title
                    // bar and the server to traverse.
    IFSFileDialog dialog = new IFSFileDialog(this, "Title Bar Text", sys);

                    // Create a list of filters then set
                    // the filters in the dialog. The
                    // first filter will be used when
                    // the dialog is first displayed.
    FileFilter[] filterList = {new FileFilter("All files (*.*)", "*.*"),
                               new FileFilter("HTML files (*.HTML",
"*.HTM")};

    dialog.setFileFilter(filterList, 0);

                    // Set the text on the buttons of
                    // the dialog.
    dialog.setOkButtonText("Open");
    dialog.setCancelButtonText("Cancel");

                    // Show the dialog. If the user
                    // selected a file by pressing the
                    // Open button, get the file the
                    // user selected and display it.
    if (dialog.showDialog() == IFSFileDialog.OK)
       System.out.println(dialog.getAbsolutePath());
```

IFSKey 類別

如果 Java 程式容許其它程式同時存取某檔案，則 Java 程式可在某期間內鎖定此檔案的位元組。在該期間，此程式專用此檔案的該區段。當鎖定順利完成時， 整合檔案系統類別會傳回 IFSKey 物件。 提供此物件給 unlock() 方法來指出要解除鎖定的位元組。關閉檔案之後，系統會解除鎖定仍在檔案中的全部鎖定（系統會對程式沒有解除鎖定的每一個鎖定執行解除鎖定）。

下面範例說明如何使用 IFSKey 類別。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open an input stream. This
                    // constructor opens with share_all
                    // so other programs can open this
                    // file.
IFSFileInputStream aFile =
              new IFSFileInputStream(sys,"/mydir1/mydir2/myfile");

                    // Lock the first 1K bytes in the
                    // file. Now no other instance can
                    // read these bytes.
IFSKey key = aFile.lock(1024);

                    // Read the first 1K of the file.
byte data[] = new byte[1024];
aFile.read(data);

                    // Unlock the bytes of the file.
aFile.unlock(key);

                    // Close the file.
aFile.close();
```

檔案共用模式

開啟檔案之後 Java
程式可指定共用模式。此程式可讓其它程式同時開啟此檔案，或者讓此程式對此檔案具有專用存取權。

下面範例說明如何指定檔案共用模式。

```
                    // Create an AS400 object.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Open a file object that
                    // represents the file. Since this
                    // program specifies share-none, all
                    // other open attempts fail until
                    // this instance is closed.
    IFSFileOutputStream aFile =
                 new IFSFileOutputStream(sys,
                                         "/mydir1/mydir2/myfile",
                                         IFSFileOutputStream.SHARE_NONE,
                                         false);

                    // ... perform operations on the
                    // file.

                    // Close the file. Now other open
                    // requests succeed.
    aFile.close();
```

JavaApplicationCall

JavaApplicationCall 類別讓您有能力從您的從屬站， 使用伺服器 JVM 來執行常駐在伺服器上的 Java 程式。

在建立自從屬站到伺服器的連線後，JavaApplicationCall 類別可讓您配置下列：

1. 使用 setClassPath() 方法，在伺服器中設定 CLASSPATH 環境變數
2. 使用 setParameters() 方法，定義您程式的參數
3. 使用 run() 執行程式
4. 自從屬站傳送輸入到 Java 程式。Java 程式會經由以 setStandardInString() 方法設定的標準輸入來讀取輸入。您可以經由 getStandardOutString() 及 getStandardErrorString() 將標準輸出及標準錯誤從 Java 程式重新導向從屬站

JavaApplicationCall 是一個您可以從 Java 程式呼叫的類別。 然而，IBM Toolbox for Java 也提供了公用程式以 呼叫常駐在伺服器上的 Java 程式。這些公用程式是完整的 Java 程式，您可以在您的工作站中執行它們。請參閱 RunJavaApplication 類別以取得其餘相關資訊。

範例

此範 例會告訴您如何在從屬站上執行伺服器中的程式，輸出 "Hello World!".

JDBC

JDBC (TM) 是 Java 平台中的一個應用程式設計介面 (API)， 啟用 Java 程式以連接廣大的資料庫。

IBM Toolbox for Java JDBC 驅動程式可讓您使用 JDBC API 介面， 對伺服器上的資料發出結構化查詢語言 (SQL) 陳述式，並處理來自伺服器上資料庫的結果。您也可 使用IBM Developer Kit for Java JDBC 驅動程式，又稱為「原有的」JDBC 驅動程式：

- 如同在主從架構的環境下，Java 程式在一個系統上且資料庫檔案在另一個系統上， 則使用 ToolboxJDBC 驅動程式
- 當 Java 程式及資料庫檔案都在相同的 iSeries 系統上時則使用原有的 JDBC 驅動程式

有關目前進行中的改進事項，請參閱V5R2 的新增功能及Toolbox for Java JDBC 支援的加強功能。

JDBC 的不同版本

JDBC API 有不同的版本，IBM Toolbox for Java JDBC 驅動程式支援下列的版本：

- 包括在 Java Platform 1.1 core API 及 JDK 1.1 的 JDBC 1.2 API (java.sql 資料包)。
- JDBC 2.1 基核 API (java.sql 資料包) 同時包括於 Java 2 Platform 標準版 (J2SE) 及 Java 2 Platfor 企業版 (J2EE) 中。
- JDBC 2.0 Optional Package API (javax.sql 資料包) 包括於 J2EE 中，並當做個別下載來使用。這些延伸程式以前稱為 JDBC 2.0 Standard Extension API。
- JDBC 3.0 API (java.sql 及 javax.sql 資料包) 包括於 J2SE, Version 1.4 中。

支援的介面

下表列出所支援的 JDBC 介面，以及使用該介面必要的 API：

支援的 JDBC 介面	必要的 API
Blob 提供對二進位大型物件 (BOLB) 的存取	JDBC 2.1 基核
CallableStatement 執行 SQL 儲存程序	JDK 1.1
Clob 提供對字元大型物件 (CLOB) 的存取	JDBC 2.1 基核
Connection 代表對特定資料庫的連線	JDK 1.1
ConnectionPool 代表 Connection 物件的儲存區。	JDBC 2.0 選用性套件
ConnectionPoolDataSource 代表儲存 AS400JDBCPooledConnection 物件的工廠。	JDBC 2.0 選用性套件
DatabaseMetaData 提供有關資料庫的整體資訊。	JDK 1.1
DataSource 代表資料庫連線的工廠。	JDBC 2.0 選用性套件
Driver 建立連線並傳回關於驅動程式版本的資訊。	JDK 1.1

≫ParameterMetaData提供取得 PreparedStatement 物件中之參數類型及內容資訊的功能	JDBC 3.0 API≪
PreparedStatement執行已編譯的 SQL 陳述式	JDK 1.1
ResultSet提供對資料表的存取，這些資料是藉由執行 SQL 查詢或 DatabaseMetaData 編目方法所產生的	JDK 1.1
ResultSetMetaData提供有關特定的 ResultSet 之資訊	JDK 1.1
RowSet 是一個封裝 ResultSet 的已連接列集	JDBC 2.0 選用性套件
≫Savepoint提供異動內部小範圍資料的控制	JDBC 3.0 API≪
Statement執行 SQL 陳述式並得到結果	JDK 1.1
XAConnection 是指在廣域 XA 異動中使用的資料庫連線	JDBC 2.0 選用性套件
XAResource 是在 XA 異動中使用的資源管理程式	JDBC 2.0 選用性套件

我們已併入了列出 JDBC 內容的表格，以便於參照。

範例

下列範例說明使用 IBM Toolbox for Java JDBC 驅動程式的方法。

- 使用 JDBC 驅動程式建立表格並大量輸入資料
- 使用 JDBC 驅動程式查詢表格並輸出其內容

»

Toolbox for Java JDBC 支援的加強功能

OS/400 版本 5 版次 2 中 JDBC 的強化功能包括：

FOR UPDATE 限制的移除

您不再需要於 SELECT 陳述式中指定 FOR UPDATE 以保證游標可更新。連接至 V5R1 及 更新版本的 OS/400 時，Toolbox for Java 提供建立陳述式時傳入的並行性。
如果不指定並行性，則預設值仍為唯讀游標。

資料截斷僅於截斷字元 資料寫入資料庫時，才會發出異常訊息

Toolbox for Java 的資料截斷規則現在與 IBM Developer Kit for Java JDBC driver 的資料截斷規則相同。
其它資訊，請參閱 IBM Toolbox for Java JDBC properties。

依名稱取得與修改直欄及參數

新的方法可讓您依 ResultSet 中的直欄名稱取得與更新資訊， 及依 CallableStatement 中的參數名稱取得與設定資訊。 例如，在 ResultSet 中，先前您是使用下列項目：

```
ResultSet rs = statement.executeQuery( SELECT * FROM
MYCOLLECTION/MYTABLE );
rs.getString(1);
```

現在您可使用：

```
ResultSet rs = statement.executeQuery( SELECT * FROM
MYCOLLECTION/MYTABLE );
rs.getString( 'STUDENTS' );
```

請注意，依參數的索引存取參數之效能較依參數名稱存取參數的效能為佳。您也可指定 在 CallableStatement 中設定的參數名稱。先前在 CallableStatement 中您可能使用：

```
CallableStatement cs = connection.prepareCall( CALL MYPGM (?) );
cs.setString( 1 );
```

現在您可使用：

```
CallableStatement cs = connection.prepareCall( CALL MYPGM (?) );
cs.setString( 'PARAM_1' );
```

若要使用這些新的方法，需要 JDBC 3.0 或其更新版本及 Java 2 Platform 版本 1.4 (標準版或企業版)。

擷取自動產生的鍵值

AS400JDBCStatement 中 的 getGeneratedKeys() 方法會擷取執行該陳述式物件所建立的所有自動產生之鍵值。陳述式物件不產生任何鍵值時，會傳回空的 ResultSet 物件。 目前伺服器僅支援傳回一個自動產生的鍵值 (最後插入的列之鍵值)。 下列範例告訴您如何將值插入表格後取得自動產生的鍵值：

```
Statement s = statement.executeQuery
     ("INSERT INTO MYSCHOOL/MYSTUDENTS (FIRSTNAME) VALUES ('JOHN'");
ResultSet rs = s.getGeneratedKeys();
     // Currently the iSeries server supports returning only one auto-
generated
     // key -- the key for the last inserted row.
       rs.next ();
String autoGeneratedKey = rs.getString(1);
     // Use the auto-generated key, for example, as the primary key in
another table
```

若要擷取自動產生的鍵值，需要 JDBC 3.0 或其更新版本，及 Java 2 Platform 版本 1.4 (標準版或企業版)。擷取自動產生的鍵值也需與 V5R2 或 OS/400 更新版本連線。

改進於批次中執行 SQL insert 陳述式的效能

於批次中執行 SQL insert 陳述式的效能已改進。於批次中執行 SQL 陳述式的方法為： 使用下列項目中不同的 addBatch() 方法AS400JDBCStatement、AS400JDBCPreparedStatement 及 AS400JDBCCallableStatement。 強化的批次支援僅會影響 insert 要求。例如，使用批次支援來處理數個僅牽涉對伺服器一次傳送的 insert。 不過，使用批次支援來處理一個 insert、update 及一個 delete，則會分別傳送各要求。

若要使用批次支援，需要 JDBC 2.0 或其更新版本及 Java 2 Platform 版本 1.2 (標準版或企業版)。

強化對 ResultSet.getRow() 的支援

之前，IBM Toolbox for Java JDBC 驅動程式受限於ResultSet中 getRow() 方法的支援。 特別是使用具負值的 ResultSet.last()、ResultSet.afterLast() 及 ResultSet.absolute() 會使目前的列號無法使用。現在已將此限制移除，因而使此方法的功能更全面性。

對直欄名稱使用大小寫

Toolbox for Java 方法必須使使用者提供的直欄名稱，或應用程式提供的直欄名稱，與資料庫表格上的名稱相符。不論是哪一種情況，當直欄名稱未以雙引號括住 時，Toolbox for Java 會於名稱與伺服器上的名稱相符合之前，將名稱變更為大寫字元。而當直欄名稱以雙引號括住時，其必須完全與伺服器上的名稱相符，否則 Toolbox for Java 會發出異常訊息。

指定建立的 Statements、CallableStatements 及 PreparedStatements 之保留能力

AS400JDBCConnection 中的 新方法可讓您指定您所建立的 Statements、CallableStatements 及 PreparedStatements 之保留能力。當確定異動時，保留能力決定游標是否維持在開啟或是關閉。現在您會具有與其連線物件不同保留能力的陳述式。此外，連線物件可具有多個開啟的陳述式物件，每一個具有不同的指定保留能力。 呼 commit 會使系統根據指定給該陳述式的保留能力來處理每一個陳述式。

保留能力以下列順序衍生：

1. 使用 Connection 類別方法 createStatement()、prepareCall() 或 prepareStatement() 建立陳述式時指定的保留能力。
2. 使用 Connection.setHoldability(int) 指定的保留能力。
3. 由 Toolbox for Java JDBC 游標保留特性 指定的保留能力 (當不使用 1. 或 2. 中的方法時)

若要使用這些方法，需要 JDBC 3.0 或其更新版本，及 Java 2 Platform 版本 1.4 (標準版或企業版)。此外，執行 V5R1 或 OS/400 較早版本之伺服器僅可使用由 JDBC 游標保留特性指定的保留能力。

強化異動隔離支援

IBM Toolbox for Java JDBC 驅動程式現在支援連接後切換至異動隔離層次 *NONE。 先前在 V5R2，Toolbox for Java JDBC 驅動程式在連接後切換至 *NONE 時會發出異常訊息。

IBM Toolbox for Java JDBC 內容

使用 JDBC 連接伺服器資料庫時可指定許多內容。所有的內容都是可選用的， 且可將之指定為 URL 的一部份，或是在 java.util.Properties 物件中指定。 若同時在 URL 及 Properties 物件中設定內容，將會使用 URL 中的值。

註： 下列清單中不包括 DataSource 內容。

下列表格列出由此驅動程式所辨識之不同的連線內容。這些內容中的 部份內容會影響效能，其它的則是伺服器工作屬性。表格將內容組織為下列 媜 G

一般內容

一般內容是指定使用者、密碼及連接伺服器時是否需要提示的系統屬性。

一般內容	說明	必要的	選項	預設
"密碼"	指定連接伺服器的密碼。除非將 "提示" 內容設定為 "false" (某些情況下會造成連線失敗)，否則在未指定密碼時，將會提示使用者。	否	伺服器密碼	(將會提示使用者)
"提示"	如果在連接伺服器時需要使用者名稱及密碼，則指定是否應提示使用者。 如果未提示使用者便無法連線，且此內容已設定為 "false"， 則嘗試連線會失敗。	否	"true" "false"	"true"
"使用者"	指定用於連接伺服器的使用者名稱。除非將 "提示" 內容設定為 "false" (某些情況下會造成連線失敗)，否則在未指定密碼時，將會提示使用者	否	伺服器使用者	(將會提示使用者)

伺服器內容

伺服器內容指定支配交易、檔案庫及資料庫的屬性。

伺服器內容	說明	必要的	選項	預設
"游標保留"	指定是否在異動內保留游標。 若設定此內容為 "true"，當已確定或回轉異動時， 不關閉游標。在工作單元期間所擷取的所有資源都將保留， 但會釋放工作單元期間所隱含擷取之特定列及物件的鎖定。	否	"true" "false"	"true"
»"游標靈敏度"	指定資料庫所要求的游標靈敏度。此行為取決於 resultSetType： • ResultSet.TYPE_FORWARD_ONLY 或 ResultSet.TYPE_SCROLL_SENSITIVE 表示此內容值可控制 Java 程式從資料庫中所要求的游標靈敏度。 • ResultSet.TYPE_SCROLL_INSENSITIVE 則會忽略此內容。 連接至執行 OS/400 V5R1 及更早版本的系統時，忽略此內容。	否	"低靈敏" "不靈敏" "靈敏"	"低靈敏" «
»"資料庫名稱"	指定連線使用的資料庫，其中包括儲存於 獨立輔助 儲存體儲存區 (ASP)的資料庫。此內容僅適用於連接 V5R2 或 OS/400 的 更新版本時。指定資料庫名稱時，該名稱必須存在於伺服器上的關聯式資料庫目錄中。 下列的基準判定所存取的資料庫： • 當使用此內容指定資料庫時，所指定的資料庫已被使用。 指定的資料庫不存在時則連線失敗。 • 當使用此內容指定 *SYSBAS 為資料庫名稱時，則使用系統預設資料庫。 • 當省略此內容時，則使用在使用者設定檔之工作說明中指定的資料庫名稱。 當工作說明未指定資料庫名稱時，則使用系統預設資料庫。	否	資料庫名稱 "*SYSBAS"	使用在使用者設定檔之工作說明中指定的資料庫名稱。 工作說明未指定資料庫名稱時，則使用系統預設資料庫。 «

"檔案庫"	指定您要在伺服器工作的檔案庫清單中新增或置換的一或多個檔案庫，並選擇性地設定預設檔案庫（預設綱目）。 檔案庫清單 伺服器會使用指定的檔案庫來解析未限定的儲存程序名稱，且儲存程序會使用它們來解析未限定的名稱。若要指定多個檔案庫，請使用逗點或空格來分隔個別登錄。 您可以使用 *LIBL 作為伺服器工作之現行檔案庫清單的位置保留符號： 第一個登錄是 *LIBL 時，即會在伺服器工作的現行檔案庫清單中新增指定的檔案庫。 不使用 *LIBL 時，指定的檔案庫即會置換伺服器工作的現行檔案庫清單 預設綱目 伺服器會使用預設綱目來解析 SQL 陳述式中未限定的名稱。 例如，在陳述式 "SELECT * FROM MYTABLE" 中，伺服器只會在預設綱目中尋找 MYTABLE。 您可以指定連線 URL 中的預設綱目。 當您沒有指定連線 URL 中的預設綱目時，則會套用下列條件，視您是否使用「SQL 命名」或「系統命名」而定。 • SQL 命名 當您沒有指定連線 URL 中的預設綱目時： ◦ 第一個登錄（除非它是 *LIBL）會變成預設綱目 ◦ 當第一個登錄是 *LIBL 時，第二個登錄會變成預設綱目 ◦ 當您沒有設定此內容或當它只有 *LIBL 時，使用者設定檔會變成預設綱目 • 系統命名 當您沒有指定連線 URL 中的預設綱目時： ◦ 沒有設定任何預設綱目，且伺服器會使用指定的檔案庫來搜尋未限定的名稱 ◦ 當您沒有設定此內容或當它只有 *LIBL 時，伺服器會使用伺服器工作的現行檔案庫清單來搜尋未限定的名稱	否	伺服器檔案庫清單，以逗點或空格區隔	"*LIBL"
"異動隔離"	指定預設異動隔離。	否	"無" "讀取未確定" "讀取已確定" "可重複讀取" "序列化"	"讀取未確定"

格式內容

格式內容指定日期及時間格式、日期及小數點符號， 以及 SQL 陳述式內用到的表格命名慣例。

格式內容	說明	必要的	選項	預設
"日期格式"	指定 SQL 陳述式內日期文字使用的日期格式。	否	"mdy" "dmy" "ymd" "usa" "iso" "eur" "jis" "julian"	(伺服器工作)
"日期分隔字元"	指定 SQL 陳述式內日期文字使用的日期分隔字元。除非將 "日期格式" 內容 設定為 "julian"、"mdy"、"dmy" 或 "ymd"，否則此內容將無作用。	否	"/" (斜線) "-" (破折號) "." (句點) "," (逗點) "b" (空格)	(伺服器工作)
"小數點符號"	指定 SQL 陳述式內數字文字使用的小數點符號。	否	"." (句點) "," (逗點)	(伺服器工作)
"命名"	指定參照表格時使用的命名慣例。	否	"sql" (如此：schema .table 中) "系統" (如此：schema /table 中)	"sql"

"時間格式"	指定 SQL 陳述式內時間文字使用的時間格式。	否	"hms" "usa" "iso" "eur" "jis"	(伺服器工作)
"時間分隔符號"	指定 SQL 陳述式內時間文字使用的時間分隔符號。 除非將 "時間格式" 內容設定為 "hms"， 否則此內容將無作用。	否	":" (冒號) "." (句點) "," (逗點) "b" (空格)	(伺服器工作)

效能內容

效能內容是指影響效能的屬性，其中包括快取、資料轉換、資料壓縮及預先提取。

效能內容	說明	必要的	選項	預設
"big decimal"	指定是否在壓縮及劃分區域十進位轉換使用中階的 java.math.BigDecimal 物件。如果此內容設定為 "true"， 則如同 JDBC 規格的說明，會在壓縮及劃分區域十進位轉換中使用中階的 java.math.BigDecimal 物件。如果此內容設定為 "false"，壓縮及劃分區域十進位轉換將不使用中階物件。而這類的值將直接轉換為 Java 倍整數，或從 Java 倍整數轉換。這類的轉換會比較快， 但是不會遵循 JDBC 規格所記載的所有轉換及資料截斷規則。	否	"true" "false"	"true"
"區塊基準"	指定自伺服器擷取區塊記錄資料的基準。 對此內容指定非零的值將減少與伺服器通信的頻率，因而可增加效能。 如果游標將用於後續的更新，請確定已關閉區塊標示記錄， 否則所更新的列不需為目前的列。	否	"0" (未以區塊標示記錄) "1" (如果指定 FOR FETCH ONLY 則以區塊標示) "2" (除非指定 FOR UPDATE，否則將以區塊標示)	"2"
"區塊大小"	指定自伺服器及由從屬站上快取擷取的區塊大小 (以千位元組 (KB) 為單位)。除非 "區塊基準" 內容為非零的值，否則此內容將無作用。較大的區塊大小將減少與伺服器 通信的頻率，因而可增加效能。	否	"0" "8" "16" "32" "64" "128" "256" "512"	"32"
"資料壓縮"	指定是否壓縮結果集資料。若將此內容設定為 "true"，則會壓縮結果集資料。若將此內容設定為 "false"，則不壓縮結果集資料。資料壓縮在擷取大量的結果集時， 可增進效能。	否	"true" "false"	"true"
"延伸動態"	指定是否要使用延伸的動態支援。延伸的動態支援 提供伺服器上快取動態 SQL 陳述式的機制。第一次 所準備的特殊 SQL 陳述式會儲存於伺服器上的 SQL 資料包。若資料包不存在， 將會自動建立之。在相同 SQL 陳述式的後續準備上，伺服器可使用儲存於 SQL 資料包的資訊，以跳過部份 處理邏輯。 若設定為 "true"，則資料包名稱必須使用 "資料包" 內容來設定。	否	"true" "false"	"false"
"延遲關閉"	指定是否要等到後續的要求才延遲關閉游標。 這可藉由減少總要求數來增加整體的效能。	否	"true" "false"	"false"
"lob 臨界值"	指定可以擷取作為部份結果集的 LOB (大型物件) 大小上限 (以位元組為單位)。當 LOB 大於這個臨界值時，將使用伺服器的額外通信，以片斷方式擷取它們。較大的 LOB 臨界值雖可減少伺服器通信的頻率， 但會下載更多的 LOB 資料，即使是一些用不到資料。較小的 LOB 臨界值可能增加伺服器通信的頻率，但它們僅下載需要的 LOB 資料。	否	"0" - "16777216"	"0"
"資料包"	指定 SQL 資料包的基本名稱。 請注意， 僅使用前 7 個字元來產生伺服器上的 SQL 資料包名稱。除非將 "延伸動態" 內容設定為 "true"，否則此內容將無作用。此外，如果將 "延伸動態" 內容設定為 "true"，則必須設定此內容。	否	SQL 資料包	""
"資料包新增"	指定 是否要將新準備的陳述式新增到 "資料包" 內容中所指定的 SQL 資料包。 除非將 "延伸動態" 內容設定為 "true"，否則此內容將無作用。	否	"true" "false"	"true"
"資料包快取"	指定 是否要快取從屬站記憶體中 SQL 資料包資訊的子集。在本端快取 SQL 資料包 減少伺服器在準備及說明上的通信量。除非將 "延伸動態" 內容設定為 "true"，否則此內容將無作用。	否	"true" "false"	"false"
"資料包基準"	指定要儲存在 SQL 資料包中的 SQL 陳述式類型。 對增進複雜之結合條件的效能而言十分有用。除非將 "延伸動態" 內容 設定為 "true"，否則此內容將無作用。	否	"預設" (僅在資料包中儲存具有參數標記的 SQL 陳述式) "選取" (在資料包中儲存所有的 SQL SELECT 陳述式)	"預設"
"資料包錯誤"	指定當發生 SQL 資料包錯誤時，要採取的動作。 發生 SQL 資料包錯誤時，驅動程式將依據此內容的值，選用性地 擲出 SQLException 或對「連線」發佈警告。 除非將 "延伸動態" 內容設定為 "true"， 否則此內容將無作用。	否	"異常" "警告" "無"	"警告"
"資料包檔案庫"	指定 SQL 資料包的檔案庫。除非將 "延伸動態" 內容 設定為 "true"，否則此內容將無作用。	否	SQL 資料包的檔案庫	"QGPL"
"預先提取"	指定是否要在執行 SELECT 陳述式時預先提取資料。 如此在存取 ResultSet 中的起始列時將會增加效能。	否	"true" "false"	"true"

排序內容

排序內容指定伺服器執行儲存及排序的方法。

排序內容	說明	必要的	選項	預設
"排序"	指定在伺服器傳送記錄到從屬站之前，伺服器排序記錄的方法。	否	"十六進位" (依據十六進位值排序) "工作" (依據伺服器工作的設定值排序) "語言" (依據 "排序語言" 內容中的語言排序) "表格" (依據 "排序表格" 內容中的排序順序表排序)	"工作"
"排序語言"	指定在進行排序順序的選項時，所要使用的三個字元的語言 ID。 除非將 "排序" 內容設定為 "語言"，否則此內容將無作用。	否	語言 ID	ENU
"排序表"	指定在伺服器中儲存排序順序表的檔案庫及檔名。 除非將 "排序" 內容設定為 "表格"，否則此內容將無作用。	否	限定的排序表格名稱	""
"排序加權"	指定伺服器在排序記錄時，如何處理情況。除非將 "排序" 內容 設定為 "語言"，否則此內容將無作用。	否	"共用" (大小寫字體在排序時視為相同的字元) "唯一" (大小寫字體在排序時視為不同的字元)	"共用"

其它內容

其它內容是指不易分類的內容。這些內容判定所使用的 JDBC 驅動程式， 並指定與資料庫存取、雙向字串類型及資料截斷等相關的選項。

其它內容	說明	必要的	選項	預設
"存取"	指定連線的資料庫存取層次。	否	"全部" (接受所有的 SQL 陳述式) "讀取呼叫" (接受 SELECT 及 CALL 陳述式) "唯讀" (僅接受 SELECT 陳述式)	"全部"
»"行為置換"	指定要置換的 IBM Toolbox for Java JDBC 驅動程式行為為何。 您可以在此內容中新增常數並傳送該總和，以變更組合中的多個行為。 請確定您的應用程式會正確地處理變更的行為。	否	"" (不置換任何行為) "1" (不排除異常，但如果 Statement.executeQuery() 或 PreparedStatement.executeQuery() 沒有傳回結果集，則會傳回空值作為結果集。)	""«
"雙向字串類型"	指定雙向資料的輸出字串類型。若需取得更多資訊， 請參閱 BidiStringType	否	"" (使用 CCSID 以判定雙向字串類型) "0" (非雙向類型資料 (LTR) 的預設字串類型) "4" "5" "6" "7" "8" "9" "10" "11"	""
"資料截斷"	»指定 字元資料的截斷是否產生警告及異常。 此內容為 "true" 時，適用於下列情況： • 將截斷字元資料寫入資料庫時擲出異常 • 使用查詢中的截斷字元資料時發佈警告。 當此內容為「false」時，將截斷的資料寫入資料庫或使用查詢中的這類資料而不產生異常或警告。 預設值為 "true"。 此內容不會影響數值資料 。將截斷數值資料 寫入資料庫時一律擲出錯誤，使用查詢中的截斷數值資料時一律發佈警告。 «	否	"true" "false"	"true"
"驅動程式"	指定 JDBC 驅動程式施行。IBM Toolbox for Java JDBC 驅動程式可依據環境使用不同的 JDBC 驅動程式施行。 如果環境是與程式連接的資料庫在相同伺服器上的 iSeries JVM， 則可使用原有的 IBM Developer Kit for Java JDBC 驅動程式。 在其它的環境下，則使用 IBM Toolbox for Java JDBC 驅動程式。 如果設定了 "次要 URL" 內容，則此內容將沒有作用。	否	"工具箱" (僅適用於 IBM Toolbox for Java JDBC 驅動程式)。 "原有的" (如果在伺服器上執行，則使用 IBM Developer Kit for Java JDBC 驅動程式， 否則使用 Toolbox for Java JDBC 驅動程式)。	"工具箱"

"錯誤"	指定伺服器上發生錯誤時，於訊息中所傳回的明細量。	否	"基本" "完整"	"基本"
»"延伸的 meta 資料"	指定驅動程式是否向伺服器要求延伸的 meta 資料。 將此內容設定為 TRUE，可增加從下列 ResultSetMetaData 方法傳回之資訊的精確度： • getColumnLabel(int) • isReadOnly(int) • isSearchable(int) • isWriteable(int) 此外，設定此內容為 TRUE 會啟用 ResultSetMetaData.getSchemaName(int) 方法的支援。 設定此內容為 TRUE 可能會降低效能，這是因為它需要從伺服器擷取較多的資訊。 除非您需要由所列出的方法中得到更多特定的資訊，否則請保留此內容的預設值 (false)。 例如，當關閉此內容時 (false)，ResultSetMetaData.isSearchable(int) 會一律傳回「true」， 因為驅動程式並未從伺服器取得可供判斷的足夠資訊。開啟此內容 (true) 會強制驅動程式由伺服器取得正確的資料。 您僅可在連接執行 OS/400 V5R2 或更新版本的伺服器時使用延伸的 meta 資料。	否	"true" "false"	"false«"
"完整開啟"	指定伺服器是否完整開啟每一個查詢的檔案。 依據預設值，伺服器會以最佳化開啟要求。這種最佳化會增進效能， 但是，當查詢的執行超過一次時，如果資料庫監督程式為作用中則可能會失敗。 因此，僅可在發出相同的查詢且監督程式為作用中時，將此內容設定為 TRUE。	否	"true" "false"	"false"
"金鑰環名稱"	指定伺服器用於 SSL 連線的金鑰環類別名稱。 除非將 "安全" 設定為 true，且使用 "金鑰環密碼" 內容設定金鑰環密碼，否則此內容將無作用。	否	"金鑰環名稱"	""
"金鑰環密碼"	指定伺服器用於 SSL 通信的金鑰環類別密碼。 除非將 "安全" 設定為 true，且使用 "金鑰環名稱" 內容設定金鑰環名稱，否則此內容將無作用。	否	"金鑰環密碼"	""
"PROXY 伺服器"	指定正在執行 PROXY 伺服器的主電腦名稱及中間層機器的埠。 其格式為 hostname[:port]，其中 port 部份是可選用的。 若未設定此項，則會從 com.ibm.as400.access.AS400.proxyServer 內容擷取主電腦名稱及埠。預設的埠為 3470 (如果連線使用 SSL， 則預設的埠為 3471)。必須在中間層機器上執行 ProxyServer。 在二層式環境下，會省略中間層機器的名稱。	否	PROXY 伺服器主電腦名稱及埠	(proxyServer 內容的值，若未設定則沒有值)
"備註"	指定 DatabaseMetaData 方法所傳回 ResultSets 中之 REMARKS 直欄的文字來源。	否	"sql" (SQL 物件註解) "系統" (OS/400 物件說明)	"系統"
»"序列化時儲存密碼"	指定在序列化此資料來源物件時，是否要在本端儲存密碼與其餘內容。 儲存密碼表示應用程式必須保護物件的序列化形式，因為該物件含有連接伺服器的所有必要資訊。 將此內容設定為 "true" 之前，請考慮儲存此密碼與其餘內容所造成的安全風險。	否	"true" "false"	"false«"
"次要 URL"	如果所指定之多層式環境內中間層 DriverManager 上連線所使用的 URL 與原指定的不同，則在此指定之。此內容可讓您使用此驅動程式連線到不是 iSeries 或 AS/400e 伺服器的資料庫。在 URL 的反斜線及分號之前，使用反斜線當成跳出字元。	否	JDBC URL	(目前的 JDBC URL)
"安全"	指定是否使用 Secure Sockets Layer (SSL) 連線與伺服器通信。SSL 連線僅可用於連接 V4R4 或更新版本的伺服器。		"true" (加密所有主從架構通信) "false" (僅加密密碼)	"false"
»"伺服器追蹤"	指定 JDBC 伺服器工作的追蹤層次。當啟用追蹤時， 追蹤會由從屬站連線到伺服器時開始，而在切斷連線時結束。 您必須於連線到伺服器之前啟動追蹤，因為從屬站僅在連線時才啟用伺服器追蹤。	否	"0" (追蹤不在作用中) "2" (啟動 JDBC 伺服器工作上的資料庫監督程式) "4" (啟動 JDBC 伺服器工作上的除錯) "8" (在 JDBC 伺服器工作結束時儲存工作日誌) "16" (啟動 JDBC 伺服器工作上的工作追蹤) "32" (儲存 SQL 資訊) 將這些值加在一起可啟動多種追蹤類型。例如，"6" 即表示啟動資料庫監督程式及除錯。	"0"«
"已使用的緒"	指定是否應使用緒與主電腦伺服器通信。	否	"true" "false"	"true"

"追蹤"	指定是否應記錄追蹤訊息。追蹤訊息在呼叫 JDBC 之程式的除錯上十分有用。 然而，追蹤訊息的記錄對效能有極大的影響，因此，僅應在除錯才將此內容設定為 "true"。追蹤訊息會記錄於 System.out。	否	"true" "false"	"false"
"轉換二進位"	指定是否要轉換二進位資料。如果此內容設定為 "true"，則會將 BINARY 及VARBINARY 欄位視為 CHAR 及 VARCHAR 欄位。	否	"true" "false"	"false"

AS400JDBCBlob 類別

您可使用AS400JDBCBlob 物件 存取二進位大型物件 (BLOB)，如聲音位元組檔 (.wav) 或影像檔 (.gif)。

AS400JDBCBlob 類別與 AS400JDBCBlobLocator 類別的最大不同之處在於 blob 的儲存處。透過 AS400JDBCBlob 類別，blob 會儲存在資料庫中，進而影響資料庫檔案的大小。AS400JDBCBlobLocator 類別會在資料庫檔案中儲存一個定位碼 (可視為一個指標)，指向 blob 所在之處。

透過 AS400JDBCBlob 類別，可使用 lob 臨界特性。這個特性指定可擷取作為部份結果集的最大大型物件 (LOB) 大小 (千位元組)。 當 LOB 大於這個臨界時，將使用伺服器的額外通信，以片斷方式擷取它們。較大的 LOB 臨界雖可減少伺服器通信的頻率，但它們會下載更多的 LOB 資料，即使用不到它們。較小的 lob 臨界可能增加伺服器通信的頻率，但它們僅下載需要的 LOB 資料。請參閱 JDBC 內容，以取得其它可用內容的資訊。

您可使用 AS400JDBCBlob 類別來執行下列事項：

- 傳回完整的二進位大型物件以作為解譯位元組的串流
- 傳回二進位大型物件部份的內容
- 傳回二進位大型物件的長度
- 建立二進位串流以寫入二進位大型物件
- 將位元組陣列寫入二進位大型物件
- 將全部或部份的位元組陣列寫入二進位大型物件
- 截斷二進位大型物件

範例

範例： 使用 AS400JDBCBlob 類別自二進位大型物件讀取資料：

```
Blob blob = resultSet.getBlob (1);
long length = blob.length ();
     byte[] bytes = blob.getBytes(1, (int) length);
```

範例： 使用 AS400JDBCBlob 類別更新二進位大型物件：

```
     ResultSet rs = statement.executeQuery ("SELECT BLOB FROM MYTABLE");
     rs.absolute(5);
     Blob blob = rs.getBlob(1);
          // Change the bytes in the blob, starting at the seventh byte
          // of the blob
     blob.setBytes (7, new byte[] { (byte) 57, (byte) 58, (byte) 98});
          //Update the blob in the result set, changing the blob starting
          // at the seventh byte of the blob (1-based) and truncating the
          // blob at the end of the updated bytes (the blob now has 9
bytes).
     rs.updateBlob(1, blob);
          // Update the database with the change.  This will change the blob
          // in the database starting at the seventh byte of the blob, and
```

```
        // truncating at the end of the updated bytes.
        rs.updateRow ();
    rs.close();
```

«

AS400JDBCBlobLocator 類別

您可使用AS400JDBCBlobLocator 物件來存取二進位大型物件。

您可使用 AS400JDBCBlobLocator 類別來執行下列作業：

- 傳回完整的二進位大型物件以作為解譯位元組的串流
- 傳回二進位大型物件部份的副本
- 傳回二進位大型物件長度
- 建立二進位串流以寫入二進位大型物件
- 將位元組陣列寫入二進位大型物件
- 將全部或部份的位元組陣列寫入二進位大型物件
- 截斷二進位大型物件

CallableStatement 介面

您可以使用CallableStatement物件來執行 SQL
儲存程序。呼叫的儲存程序必須已儲存在資料庫。CallableStatement 沒有儲存程序，它僅會呼叫儲存程序。

儲存程序可傳回一或多個 ResultSet 物件，而且可使用 IN 參數、OUT 參 數以及 INOUT 參數。使用 Connection.prepareCall() 來建立新的 CallableStatement 物件。

CallableStatement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。
您可以使用批次支援來獲得較佳的效能，因為處理一組作業通常比一次處理一個作業更快。
關於使用批次支援的詳細資訊，請參閱JDBC 支援的加強功能

»CallableStatement 可 讓您依名稱取得及設定參數和直欄 ，不過使用直欄索引的效能較佳。«

下面範例說明如何使用 CallableStatement 介面。

```
                    // Connect to the server.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create the CallableStatement
                    // object. It precompiles the
                    // specified call to a stored
                    // procedure. The question marks
                    // indicate where input parameters
                    // must be set and where output
                    // parameters can be retrieved.
                    // The first two parameters are
                    // input parameters, and the third
                    // parameter is an output parameter.
    CallableStatement cs = c.prepareCall("CALL MYLIBRARY.ADD (?, ?, ?)");

                    // Set input parameters.
    cs.setInt (1, 123);
    cs.setInt (2, 234);

                    // Register the type of the output
                    // parameter.
    cs.registerOutParameter (3, Types.INTEGER);

                    // Run the stored procedure.
    cs.execute ();

                    // Get the value of the output
                    // parameter.
    int sum = cs.getInt (3);

                    // Close the CallableStatement and

// Connection。
    cs.close();
    c.close();
```

AS400JDBCClob 類別

您可使用AS400JDBCClob 物件來 存取字元大型物件 (CLOB)，如大型文件。

AS400JDBCClob 類別與 AS400JDBCClobLocator 類別的最大不同之處在於 blob 的儲存處。 透過 AS400JDBCClob 類別，clob 是儲存在資料庫中，進而擴增資料庫檔案的大小。AS400JDBCClobLocator 類別會在資料庫檔案中儲存一個定位碼 (可視為一個指標)，指向 clob 所在之處。

透過 AS400JDBCClob 類別，可使用 lob 臨界特性。這個特性指定可擷取作為部份結果集的最大大型物件 (LOB) 大小 (千位元組)。 當 LOB 大於這個臨界時，將使用伺服器的額外通信，以片斷方式擷取它們。較大的 LOB 臨界雖可減少伺服器通信的頻率，但它們會下載更多的 LOB 資料，即使用不到它們。較小的 lob 臨界可能增加伺服器通信的頻率，但它們僅下載需要的 LOB 資料。 請參閱 JDBC 內容，以取得其它附加的內容的相關資訊。

使用 AS400JDBCClob 類別，您可進行下列作業：

- 傳回完整的 clob 作為ASCII 字元串流
- 傳回 clob內容作為字元串流
- 傳回 clob部份內容
- 傳回 clob長度
- 建立 Unicode 字元串流或ASCII 字元串流以寫入 clob
- 將字串寫入clob
- 截斷 clob

範例

範例： 使用 AS400JDBCClob 類別以從 clob 讀取：

```
Clob clob = rs.getClob (1);
     int length = clob.length();
     String s = clob.getSubString(1, (int) length);
```

範例： 使用 AS400JDBCClob 類別以更新 clob：

```
     ResultSet rs = statement.executeQuery ("SELECT CLOB FROM MYTABLE");
     rs.absolute(4);
Clob clob = rs.getClob (1);
          // Change the characters in the clob, starting at the third
character
          // of the clob
     clob.setString (3, "小");
          // Update the clob in the result set, starting at the third
character
          // of the clob and truncating the clob at the end of the update
string
          // (the clob now has 7 characters).
```

```
    rs.updateClob(1, clob);
         // Update the database with the updated clob.  This will change
the
         // clob in the database starting at the third character of the
clob,
         // and truncating at the end of the update string.
        rs.updateRow ();
    rs.close();
```

«

AS400JDBCClobLocator 類別

您可使用AS400JDBCClobLocator 物件來存取字元大型物件 (CLOB)。

使用 AS400JDBCClobLocator 類別, 您可進行下列作業：

- 傳回完整的 clob 作為ASCII 字元串流
- 傳回完整的 clob作為字元串流
- 傳回 clob部份內容
- 傳回 clob長度
- »建立 Unicode 字元串流或ASCII 字元串流以寫入 clob«
- »將字串寫入clob«
- »截斷 clob«

AS400JDBCConnection 類別

AS400JDBCConnection 類別針對特定的 DB2 UDB for iSeries 資料庫提供 JDBC 連線。使用 DriverManager.getConnection() 來建立新的 AS400JDBCConnection 物件。詳細資訊，請參閱 AS400JDBCDriver 。

在建立連線時可以指定許多可選用的內容。可將內容指定為 URL 的一部份，或是在 java.util.Properties 物件中指定。請參閱 JDBC 內容，以取得 AS400JDBCDriver 所支援之內容的完整清單。

註：一個連線最多可有 9999 個 open 陳述式。

AS400JDBCConnection 包括儲存點及陳述式層次保留功能的支援，
以及傳回自動產生索引的限制支援。如需更詳細的資訊及其它加強功能的資訊，請參閱 Toolbox for Java JDBC 支援的加強功能

若要使用 Kerberos通行證，僅須在 JDBC URL 物件上設定系統名稱（而非密碼）。 使用者 ID 是透過 Java Generic Security Services (JGSS) 組織架構所擷取的， 因此您也並不需在 JDBC URL 上指定使用者。在 AS400JDBCConnection 物件中一次 僅可設定一種鑑別方法。設定密碼會清除 Kerberos 通行證或設定檔記號。 若要取得更多的資訊，請參閱AS400 類別 及J2SDK，v1.4 安全性文件。

使用 AS400JDBCConnection 類別，您可以執行下列作業：

- 建立陳 述式(Statement、PreparedStatement 或 CallableStatement 物件)
- 建立 陳述式，該陳述式具有特定的結果設定類型及並行 (Statement、PreparedStatement 或 CallableStater 物件)
- 對資料庫的確 定 及回轉變更， 以及釋放目前已保留的資料庫鎖定
- 關閉連線， 立即關閉伺服器資源，而不是等到它們自動釋放
- 針對連線的設定 保留功能及取得 保留功能
- 連線的設定 異動隔離及取得 異動隔離
- 連線的取得 meta 資料
- 開啟或關閉 設定 自動確定
- 取得 主電腦伺服器工作的工作 ID 該工作對應於連線

若您使用的是 JDBC 3.0 並連接到執行 OS/400 V5R2 的伺服器，您可使用 AS400JDBCConnection 以執行下列動作：

- 建立 具有特定結果設定保留功能的陳述式(Statement、PreparedStatement 或 CallableStatement 物件)
- 建立 傳回任何自動產生索引的預備陳述式(Statement 物件上呼叫 getGeneratedKeys() 時)
- 使用儲存點，它透過交易提供更多的顆粒性控制：
  - 設定儲存點
  - 回轉儲存點
  - 釋放儲存點 ≪

AS400JDBCConnectionPool

AS400JDBCConnectionPool 類別代表 AS400JDBCConnection 物件的儲存區，Java 程式可利用這些物件，將之當成 JDBC 2.0 Optional Package API 的 「工具箱」支援的一部份。

您可使用AS400JDBCConnectionPoolDataSource 來指定在儲存區中所建立之連線的內容，如下列範例 所示。

您無法在已要求連線之後及正在使用儲存區時變更連線儲存區資料來源。若要重設
連線儲存區資料來源，首先呼叫儲存區上的lose()。

使用 AS400JDBCConnection 物件上的lose()，將連線傳回 AS400JDBCConnectionPool。

註： 未將連線傳回儲存區時，連線儲存區的大小會持續成長，且無法重覆使用連線。

使用由ConnectionPool所繼承的方法來設定儲存區上的內容。您可設定的部份內容如下：

- 儲存區中可允許的最大連線數
- 連線的最大生命週期
- 連線的最大休止時間

您還可使用 Java Naming 及 Directory Interface(TM) (JNDI) 服務 提供者來登記
AS400JDBCConnectionPoolDataSource 物件。 若要取得有關 JNDI 服務提供者的詳細資訊，請參閱 Toolbox for Java 參照鏈結

範例：使用連線儲存區

下列範例從 JNDI 取得連線儲存區資料來源，並用它來建立一個有 10 個連線的連線儲存區：

```
        // Obtain an AS400JDBCConnectionPoolDataSource object from JNDI
        // (assumes JNDI environment is set).
    Context context = new InitialContext(environment);
    AS400JDBCConnectionPoolDataSource datasource =

(AS400JDBCConnectionPoolDataSource)context.lookup("jdbc/myDatabase");

        // Create an AS400JDBCConnectionPool object.
    AS400JDBCConnectionPool pool = new AS400JDBCConnectionPool(datasource);

        // Adds 10 connections to the pool that can be used by the
        // application (creates the physical database connections based on
        // the data source).
    pool.fill(10);

        // Get a handle to a database connection from the pool.
    Connection connection = pool.getConnection();

    ... 於資料庫上執行雜項查詢/更新。
```

```
    // Close the connection handle to return it to the pool.
connection.close();

... 應用程式由儲存區使用其它更多連線。

// Close the pool to release all resources.
pool.close();
```

DatabaseMetaData 介面

您可以使用DatabaseMetaData物件取得關於整個資料庫的資訊以及型錄資訊。

下面範例說明如何傳回表格清單，它是一個分目功能：

```
                    // Connect to the server.
Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Get the database metadata from
                    // the connection.
DatabaseMetaData dbMeta = c.getMetaData();

                    // Get a list of tables matching the
                    // following criteria.
String catalog = "myCatalog";
String schema  = "mySchema";
String table   = "myTable%"; // % indicates search pattern
String types[]  = {"TABLE", "VIEW", "SYSTEM TABLE"};
ResultSet rs = dbMeta.getTables(catalog, schema, table, types);

                    // ... iterate through the ResultSet
                    // to get the values

                    // Close the Connection.
c.close();
```

AS400JDBCDataSource 類別

AS400JDBCDataSource 類別 代表 iSeries 資料庫連線的工廠AS400JDBCConnectionPoolDataSource 類別代表 AS400JDBCPooledConnection 物件的工廠。

您可使用 Java Naming 及 Directory Interface (JNDI) 服務提供者來登記任一種類的資料來源物件。若要取得有關 JNDI 服務提供者的詳細資訊，請參閱IBM Toolbox for Java 參照鏈結

範例

下列範例說明建立及使用 AS400JDBCDataSource 物件的方法。最後的兩個範例顯示如何以 JNDI 登記一個 AS400JDBCDataSource 物件，然後使用 JNDI 傳回的物件取得資料庫連線。 請注意，即使使用不同的 JNDI 服務提供者，其程式碼都十分類似。

範例：建立 AS400JDBCDataSource 物件

下列範例顯示如何建立 AS400JDBCDataSource 物件並將之與資料庫連接：

```
        // Create a data source for making the connection.
        AS400JDBCDataSource datasource = new AS400JDBCDataSource("myAS400");
        datasource.setUser("myUser");
        datasource.setPassword("MYPWD");

        // Create a database connection to the iSeries.
        Connection connection = datasource.getConnection();
```

範例：建立可用於快取 JDBC 連線的 AS400JDBCConnectionPoolDataSource 物件

下列範例將告訴您如何使用 AS400JDBCConnectionPoolDataSource 快取 JDBC 連線。

```
        // Create a data source for making the connection.
        AS400JDBCConnectionPoolDataSource dataSource = new
AS400JDBCConnectionPoolDataSource("myAS400");
        datasource.setUser("myUser");
        datasource.setPassword("MYPWD");

        // Get the PooledConnection.
        PooledConnection pooledConnection = datasource.getPooledConnection();
```

範例：使用 JNDI 服務提供者類別儲存 AS400JDBCDataSource 物件

下列範例將說明如何使用 JNDI 服務提供者類別，直接將 DataSource 物件儲存到伺服器上的 IFS 檔案系統：

```
        // Create a data source to the iSeries database.
        AS400JDBCDataSource dataSource = new AS400JDBCDataSource();
        dataSource.setServerName("myAS400");
        dataSource.setDatabaseName("myAS400 資料庫");
```

```
        // Register the datasource with the Java Naming and Directory
Interface (JNDI).
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.fscontext.RefFSContextFactory");
        Context context = new InitialContext(env);
        context.bind("jdbc/customer", dataSource);

        // Return an AS400JDBCDataSource object from JNDI and get a
connection.
        AS400JDBCDataSource datasource = (AS400JDBCDataSource)
context.lookup("jdbc/customer");
        Connection connection = datasource.getConnection("myUser", "MYPWD");
```

範例：使用具有輕裝備目錄存取通信協定 (LDAP) 目錄伺服器的 AS400JDBCDataSource 物件 及 IBM SecureWay Directory 類別

下列範例說明如何使用 IBM SecureWay Directory 類別將物件儲存到 輕裝備目錄存取通信協定 (LDAP) 目錄伺服器：

```
        // Create a data source to the iSeries database.
        AS400JDBCDataSource dataSource = new AS400JDBCDataSource();
        dataSource.setServerName("myAS400");
        dataSource.setDatabaseName("myAS400 資料庫");

        // Register the datasource with the Java Naming and Directory
Interface (JNDI).
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.jndi.LDAPCtxFactory");
        Context context = new InitialContext(env);
        context.bind("cn=myDatasource, cn=myUsers,
ou=myLocation,o=myCompany,c=myCountryRegion", dataSource);

        // Return an AS400JDBCDataSource object from JNDI and get a
connection.
        AS400JDBCDataSource datasource = (AS400JDBCDataSource)
context.lookup("cn=myDatasource,
             cn=myUsers, ou=myLocation,o=myCompany,c=myCountryRegion");
        Connection connection = datasource.getConnection("myUser", "MYPWD");
```

登記 JDBC 驅動程式

使用 JDBC 存取伺服器資料庫檔案中的資料之前，必須以 DriveManager 登 記 IBM Toolbox for Java 授權程式的JDBC 驅動程式。 您可以使用 Java 系統特性來登記驅動程式，或由 Java 程式登記該驅動程式：

- 使用系統特性來登記

  每一個虛擬機器都有自己設定系統特性的方法。例如，來自 JDK 的 Java 指令使用 -D 選項來設定系統特性。若要使用系統特性設定驅動程式，請指定下列：

  ```
  "-Djdbc.drivers=com.ibm.as400.access.AS400JDBCDriver"
  ```

- ≫使用 Java 程式登記

  若要載入 Toolbox for Java JDBC 驅動程式，則在第一個 JDBC 呼叫之前， 於 Java 程式中新增下列指令：

  ```
  Class.forName("com.ibm.as400.access.AS400JDBCDriver");
  ```

  Toolbox for Java JDBC 驅動程式在載入時會自我登記，這也是登記驅動程式較喜歡使用的方法。您也可使用下列方法明確地登記 Toolbox JDBC 驅動程式：

  ```
  java.sql.DriverManager.registerDriver (new
  com.ibm.as400.access.AS400JDBCDriver ());
  ```

  ≪

IBM Toolbox for Java JDBC 驅動程式和其它從伺服器取得資料的 IBM Toolbox for Java 類別 不同，它不需要 AS400 物件作為輸入參數。不過， 內部使用 AS400 物件來管理預設使用者和密碼快取。第一次連接伺服器時， 可能會提示使用者輸入使用者 ID 與密碼。使用者可選擇儲存此使用者 ID 作為預設使用者 ID，然後新增密碼到密碼快 速記憶體。與其它 IBM Toolbox for Java 功能一樣，如果由 Java 程式提 供使用者 ID 與密碼，則不設定預設使用者也不快取密碼。請參閱 管理連線 取得如何管理連線的資訊。

使用 JDBC 驅動程式連接伺服器上的資料庫

您可以使用 DriverManager.getConnection() 方法來連接伺服器資料庫。DriverManager.getConnection() 採用一致資源定位器 (URL) 字串作為引數。JDBC 驅動程式管理程式會試圖尋找一驅動程式，因為該驅動程式可連接至由 此 URL 代表的資料庫。使用 IBM Toolbox for Java 驅動程式時， 請使用下列的 URL 語法：

```
"jdbc:as400://systemName/defaultSchema;listOfProperties"
```

備註： URL 可省略 systemName 或 defaultSchema。

≫若要使用 Kerberos 通行證，僅須在 JDBC URL 物件上設定系統名稱 (而非密碼)。 使用者 ID 是透過 Java

Generic Security Services (JGSS) 組織架構所擷取的，因此您也不需在 JDBC URL 上指定使用者。 在 AS400JDBCConnection 物件中一次僅可設定一種鑑別方法。設定密碼會清除 Kerberos 通行證或設定檔記號。 若要取得更多的資訊，請參閱AS400 類別 及J2SDK，v1.4 安全性文件。《

範例：使用 JDBC 驅動程式連接伺服器

範例 1： 使用未指定系統名稱的 URL。這將提示使用者輸入想要連接的系統的名稱。

```
"jdbc:as400:"
```

範例 2： 連接到伺服器資料庫；未指定預設綱目或內容。

```
                    // Connect to system 'mySystem'. No
                    // default schema or properties are
                    // specified.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem");
```

範例 3： 連接到伺服器資料庫；已指定預設綱目。

```
                    // Connect to system 'mySys2'. The
                    // default schema 'myschema' is
                    // specified.
   Connection c2 =
DriverManager.getConnection("jdbc:as400://mySys2/mySchema");
```

範例 4： 連接伺服器資料庫；指定使用 java.util.Properties 的內容。Java 程式可指定一組 JDBC 特性，其方法是使用 java.util.Properties 介面或指定特性作為此 URL 的一部份。 請參閱特性，取得支援的特性的清單。

例如，若要使用 Properties 介面指定特性，請使用下列程式作為範例：

```
                    // Create a properties object.
    Properties p = new Properties();

                    // Set the properties for the
                    // connection.
    p.put("naming", "sql");
    p.put("errors", "full");

                    // Connect using the properties
                    // object.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem",p);
```

範例 5： 連接伺服器資料庫；使用全球資源定位器 (URL) 指定內容。

```
                    // Connect using properties. The
                    // properties are set on the URL
                    // instead of through a properties
```

```
                    // object.
    Connection c = DriverManager.getConnection(
                     "jdbc:as400://mySystem;naming=sql;errors=full");
```

範例 6： 連接伺服器資料庫；已指定使用者 ID 和密碼。

```
                    // Connect using properties on the
                    // URL and specifying a user ID and
                    // password
    Connection c = DriverManager.getConnection(
                     "jdbc:as400://mySystem;naming=sql;errors=full",
                     "auser",
                     "apassword");
```

範例 7： 中斷資料庫連接。使用 Connecting 物件上 的 close() 方法切斷與伺服器的連線。若要關閉上例建立的連接, 請使用下列陳述式：

```
    c.close();
```

»

AS400JDBCParameterMetaData

AS400JDBCParameterMetaData 類別可讓您的程式擷取 PreparedStatement 及 CallableStatement 物件中參數內容的相關資訊。

AS400JDBCParameterMetaData 提供的方法，可讓您執行下列作業：

- 取得參數的類別名稱
- 取得 PreparedStatement 參數的數目
- 取得參數的 SQL 類型
- 取得參數的資料庫專屬類型名稱
- 取得參數的精準度或參數的 小數位數

範例：使用 AS400JDBCParameterMetaData

下列範例告訴您一種使用 AS400JDBCParameterMetaData 從動態產生的 PreparedStatement 物件擷取參數之方法：

```
// Get a connection from the driver.
Class.forName("com.ibm.as400.access.AS400JDBCDriver");
Connection connection =
   DriverManager.getConnection
      ("jdbc:as400://myAS400",
       "myUserId",
       "myPassword");
// Create a prepared statement object.
PreparedStatement ps =
   connection.prepareStatement
      ("SELECT STUDENTS FROM STUDENTTABLE WHERE STUDENT_ID= ?");
// Set a student ID into parameter 1.
ps.setInt(1, 123456);
// Retrieve the parameter meta data for the prepared statement.
ParameterMetaData pMetaData = ps.getParameterMetaData();
// Retrieve the number of parameters in the prepared statement.
// Returns 1.
int parameterCount = pMetaData.getParameterCount();
// Find out what the parameter type name of parameter 1 is.
// Returns INTEGER.
String getParameterTypeName = pMetaData.getParameterTypeName(1);
```

«

PreparedStatement 介面

當 SQL 陳述式將執行許多次時，您可以使用 PreparedStatement 物件。 可前置編譯 SQL 陳述式。"prepared." 陳述式是已經過前置編譯的 SQL 陳述式。 此方法比使用某 Statement 物件執行同一個陳述式數次來得更有效率，因為後者 在每次執行此陳述式時都要編譯陳述式。此外，包含在 PreparedStatement 物件的 SQL 陳述式可能有一個或多個 IN 參數。使用 Connection.prepareStatement() 來建立 PreparedStatement 物件。

PreparedStatement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。 您可以使用批次支援來獲得較佳的效能，因為處理一組作業通常比一次處理一個作業更快。 關於使用批次支援的詳細資訊，請參閱 JDBC 支援的加強功能

下面範例說明如何使用 PreparedStatement 介面。

```
                    // Connect to the server.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create the PreparedStatement
                    // object. It precompiles the
                    // specified SQL statement. The
                    // question marks indicate where
                    // parameters must be set before the
                    // statement is run.
    PreparedStatement ps = c.prepareStatement("INSERT INTO
MYLIBRARY.MYTABLE (NAME, ID) VALUES (?, ?)");

                    // Set parameters and run the
                    // statement.
    ps.setString(1, "JOSH");
    ps.setInt(2, 789);
    ps.executeUpdate();

                    // Set parameters and run the
                    // statement again.
    ps.setString(1, "DAVE");
    ps.setInt(2, 456);
    ps.executeUpdate();

                    // Close PreparedStatement and the
                    // Connection.
    ps.close();
    c.close();
```

ResultSet 介面

您可以使用ResultSet物件來存取一個因執行查詢而產生的資料表格。 會依順序擷取表格列。
在一列中，可依任何次序來存取直欄值。

儲存在 ResultSet 中的資料利用各種不同的方法 加以擷取，視擷取的資料類型而定。next()
方法是用來移到下一列。

»ResultSet 可 讓您名稱取得和更新直欄 ，不過使用直欄索引的效能較佳。«

游標移動

游標（內部指標）係被結果用來指向結果集中正被 Java 程式存取的橫列。

»getRow() 方法的效能已經改進。在 V5R2 之前，使用含有負值的 ResultSet.last()、ResultSet.afterLast() 和
ResultSet.absolute() 會使現行列號變成無法使用。 前項限制已移除，這使得 getRow() 方法的«功能更全面性。

JDBC 2.0 與以上的版本的 JDBC 規格提供其它方法來存取資料庫內的特定位置：

可捲動的游標位置	
absolute afterLast beforeFirst first getRow isAfterLast isBeforeFirst	isFirst isLast last moveToCurrentRow moveToInsertRow previous relative

捲動功能

如果透過執行一個陳述式來建立結果集，您可以向後（最後一個到第一個）或向前（第一個到最後一個）移動
(捲動）表格中的橫列。

支援這個移動的結果集稱為可捲動的結果集。可捲動的結果集也支援相對與絕對定位。
相對位置可讓您經由指定相對於現行列的位置，來移至結果集中的某一列。絕對位置可讓您經由在
結果集中指定一個位置，直接移到該橫列。

透過 JDBC 2.0 與以上的版本的 JDBC 規格，您有兩個額外捲動功能可在使用 ResultSet
類別時使用：不可捲動及可捲動結果集。

不可捲動結果集通常不會感應到當開啟它時所做的變更，而可捲動結果則可感應到變更。»IBM Toolbox for Java
JDBC 驅動程式不支援不可捲動的結果集。«

可更新的結果集

在您的應用程式中，您可以使用使用唯讀並行處理（不對資料做任何變更）的結果集，或 可更新並行處理(容許更新資料並使用資料寫入鎖定來控制不同異動對同一資料的存取權）的結果集。
在可更新結果集中，橫列可被更新、插入及刪除。有不少更新方法可供您在程式中使用，例如：

- 更新 ASCII 串流
- 更新 Big Decimal
- 更新二進位串流

請參閱 方法總結 以取得透過 ResultSet 介面可用的更新方法的完整報表。

範例：可更新的結果集

下面範例將告訴您如何使用當開啟時容許變更資料（更新並行處理），以及容許變更結果集（可捲動）的結果集。

```
                    // Connect to the server.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create a Statement object.  Set the result set
            // concurrency to updatable.
    Statement s = c.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_UPDATABLE);

                    // Run a query. The result is placed
                    // in a ResultSet object.
    ResultSet rs = s.executeQuery ("SELECT NAME,ID FROM MYLIBRARY.MYTABLE
FOR UPDATE");

                    // Iterate through the rows of the ResultSet.
                // As we read the row, we will update it with
                    // a new ID.
    int newId = 0;
    while (rs.next ())
    {

                    // Get the values from the ResultSet.
                    // The first value is a string, and
                    // the second value is an integer.
        String name = rs.getString("NAME");
        int id = rs.getInt("ID");

        System.out.println("Name = " + name);
        System.out.println("Old id = " + id);

              // Update the id with a new integer.
        rs.updateInt("ID", ++newId);

                    // Send the updates to the server.
        rs.updateRow ();
```

```
        System.out.println("New id = " + newId);
    }

                    // Close the Statement and the

// Connection。
    s.close();
    c.close();
```

ResultSetMetaData

ResultSetMetaData介面決定 ResultSet 中的直欄類型和內容。

»連接到執行 OS/400 V5R2 或以上的版本的伺服器時，使用延伸的 meta 資料內容可讓您增加下列 ResultSetMetaData 方法的精確度：

- getColumnLabel(int)
- isReadOnly(int)
- isSearchable(int)
- isWriteable(int)

此外，設定此內容為 TRUE 會啟用 ResultSetMetaData.getSchemaName(int) 方法的支援。 注意，使用延伸的 meta 資料內容可能會降低效能，這是因為它需要從伺服器擷取較多的資訊。«

AS400JDBCRowSet

AS400JDBCRowSet 類別 代表封裝一個 JDBC 結果集的已連接列集。AS400JDBCRowSet 的方法非常類似 AS400JDBCResultSet 的方法。 使用時，資料庫會與之保持連線。

您可使用AS400JDBCDataSource 物件 或 AS400JDBCConnectionPoolDataSource物件來建立與資料庫的連線 (您會想要使用此連線來存取 AS400JDBCRowSet 的資料)。

範例

下列範例告訴您如何使用 AS400JDBCRowSet 類別。

範例：建立、植入及更新 AS400JDBCRowSet 物件：

```
            DriverManager.registerDriver(new AS400JDBCDriver());
        // Establish connection by using a URL.
        AS400JDBCRowSet rowset = new
AS400JDBCRowSet("jdbc:as400://mySystem","myUser", "myPassword");

        // Set the command used to populate the list.
        rowset.setCommand("SELECT * FROM MYLIB.DATABASE");

        // Populate the rowset.
        rowset.execute();

        // Update the customer balances.
        while (rowset.next())
        {
           double newBalance = rowset.getDouble("BALANCE") +
july_statements.getPurchases(rowset.getString("CUSTNUM"));
           rowset.updateDouble("BALANCE", newBalance);
           rowset.updateRow();
        }
```

範例：從 JNDI 取得資料來源時，建立及植入 AS400JDBCRowSet 物件

```
        // Get the data source that is registered in JNDI (assumes JNDI
environment is set).
        Context context = new InitialContext();
        AS400JDBCDataSource dataSource = (AS400JDBCDataSource)
context.lookup("jdbc/customer");

        AS400JDBCRowSet rowset = new AS400JDBCRowSet();
        // Establish connection by setting the data source name.
        rowset.setDataSourceName("jdbc/customer");
        rowset.setUsername("myuser");
        rowset.setPassword("myPasswd");

        // Set the prepared statement and initialize the parameters.
        rowset.setCommand("SELECT * FROM MYLIBRARY.MYTABLE WHERE STATE = ?
```

```
AND BALANCE > ?");
        rowset.setString(1, "MINNESOTA");
        rowset.setDouble(2, MAXIMUM_LIMIT);

        // Populate the rowset.
        rowset.execute();
```

»

AS400JDBCSavepoint 類別

AS400JDBCSavepoint 類別代表異動中的邏輯岔斷點。使用 savepoints 可讓您對回轉異動時影響的變更進行更精確的控制。

圖 1：使用儲存點來控制異動中的回轉

例如，「圖 1」顯示一筆包含兩個儲存點 (A 及 B) 的異動。將異動回轉至其中一個儲存點僅會還原 (或反轉) 回轉呼叫點至儲存點的變更。 這可防止還原整個異動中之所有的變更。請注意，一旦回轉至儲存點 A，便無法於稍後回轉至儲存點 B。對儲存點 B 進行回轉作業後，便無法存取儲存點 B。

範例：使用儲存點

在此案例中，讓我們假設應用程式要更新學生成績。在更新每筆學生成績之特定欄位的最後，您執行確定。您的程式碼偵測到與更新此欄位相關聯的特定錯誤，並在錯誤發生時回轉已進行的作業。而您知道此特定錯誤只會影響對目前成績進行的作業。

因此，您在每一次更新學生成績之間設定一個儲存點。現在，當此錯誤發生時，您僅須回轉學生表格中的前次更新。毋需回轉大量的作業，現在您只需回轉少量的作業。

以下程式碼範例說明如何使用儲存點。範例假設 John 的學生 ID 為 123456， 而 Jane 的學生 ID 為 987654。

```
      // Get a connection from the driver
      Class.forName("com.ibm.as400.access.AS400JDBCDriver");
      // Get a statement object
          Statement statement = connection.createStatement();
      // Update John's record with his 'B' grade in gym.
      int rows = statement.executeUpdate("UPDATE STUDENTTABLE SET
GRADE_SECOND_PERIOD = 'B'
            WHERE STUDENT_ID= '123456'");
```

```
    // Set a savepoint marking an intermediate point in the transaction
    Savepoint savepoint1 = connection.setSavepoint("SAVEPOINT_1");
    // Update Jane's record with her 'C' grade in biochemistry.
    int rows = statement.executeUpdate("UPDATE STUDENTTABLE SET
GRADE_SECOND_PERIOD = 'C'
           WHERE STUDENT_ID= '987654'");
    // An error is detected, so we need to roll back Jane's record, but not
John's.
    // Rollback the transaction to savepoint 1. The change to Jane's record
is
    // removed while the change to John's record remains.
    connection.rollback(savepoint1);
    // Commit the transaction; only John's 'B' grade is committed to the
database.
    connection.commit();
```

注意事項與限制

使用儲存點的要求：您必須注意到下列注意事項與限制：

注意事項

Toolbox for Java 會遵循關於回轉如何影響游標及其餘鎖定的資料庫規則。
例如，將連線選項設定為在一般回轉後保持游標開啟時，則在回轉至儲存點後游標仍會開啟。
換言之，當回轉要求牽涉到儲存點時，Toolbox for Java 不會在基礎資料庫不進行此項支援時移動或關閉游標。

使用儲存點來回轉異動僅會還原從開始回轉點至儲存點執行的作業。
於回轉至該儲存點之前執行的作業仍會保留。如同在先前的範例中一樣，
請注意您可確定的異動包括回轉至特定的儲存點之前執行的作業，　但不包括回轉至儲存點之後執行的作業。

異動確定時或整個異動已回轉時，會解除所有的儲存點，且這些儲存點也會變成無效。
您也可以下列方法解除儲存點：呼叫Connection.releaseSavepoint()。

限制

使用儲存點時有下列限制：

- 儲存點的名稱必須唯一。
- 直到解除、確定或回轉儲存點之後才可重覆使用儲存點的名稱。
- 自動確定必須設定為「關閉」，儲存點才會有效。將自動確定設定為 「關閉」的方法為：使用 Connection.setAutoCommit(false)　　。 使用儲存點時啟用自動確定系統會發出異常訊息。
- 儲存點在整個 XA 連線過程中均無效。在 XA 連線時使用儲存點系統會發出異常訊息。
- 伺服器必須執行 OS/400 版本 5 版次 2 或更新的版本。與執行 V5R1 或更早版本的 OS/400 之伺服器連接
(或已連接時) 使用儲存點，系統會發出異常訊息。

以 Statement 物件執行 SQL 陳述式

使用Statement物件執行 SQL 陳述式和選用性地取得它產生的 ResultSet。

PreparedStatement 繼承 Statement，Callable Statement 繼 承 PreparedStatement。使用下列 Statement 物件來執行不同的 SQL 陳述式

- Statement- 執行一個沒有參數的簡單 SQL 陳述式。
- PreparedStatemen執行一個經過前置編譯但不一定有 IN 參數的 SQL 陳述式
- CallableStatement 執行資料庫儲存程序的呼叫 CallableStatement 不一定有 IN、OUT 與 INOUT 參數。

Statement 物件可讓您以單一群組的方式，透過批次支援的使用，對資料庫提出多重 SQL 指令。
您可以使用批次支援來獲得較佳的效能，因為處理一組作業通常比一次處理一個作業更快。 關於使用批次支援的詳細資訊，請參閱JDBC 支援的加強功能

當使用批次更新時，通常您應該關閉自動確定。關閉自動確定可讓您決定若發生錯誤及並未執行所有指令時，是否要確定異動。在 JDBC 2.0 與以上的版本 的 JDBC 規格中，Statement 物件記錄可順利以一個群組來提出及執行的指令清單。 當 executeBatch() 方法執行批次指令的這個清單時，將依指令新增到清單的次序， 來執行它們。

AS400JDBCStatement 提供的方法可讓您執行許多動作，它們包括：

- 執行不同種類的陳述式
- 擷取 Statement 物件不同的參數值，包括：
  - 連線
  - 執行 Statement 所建立的任何自動產生之鍵值
  - 提取大小和 提取方向
  - 最大欄位大和 最大列數限制
  - 現行結果集 下一個結果集 、結果集類型、結果集並行以及結果集游標保留性
- 新增 SQL 陳述到現行批次中
- 執行 SQL 陳述式的現行批次

Statement 介面

使用 Connection.createStatement() 來 建立新的 Statement 物件。

下面範例說明如何使用 Statement 物件。

```
                    // Connect to the server.
    Connection c = DriverManager.getConnection("jdbc:as400://mySystem");

                    // Create a Statement object.
    Statement s = c.createStatement();

                    // Run an SQL statement that creates
                    // a table in the database.
    s.executeUpdate("CREATE TABLE MYLIBRARY.MYTABLE (NAME VARCHAR(20), ID
INTEGER)");

                    // Run an SQL statement that inserts
                    // a record into the table.
    s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES
('DAVE', 123)");

                    // Run an SQL statement that inserts
                    // a record into the table.
    s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES
('CINDY', 456)");

                    // Run an SQL query on the table.
    ResultSet rs = s.executeQuery("SELECT * FROM MYLIBRARY.MYTABLE");
```

```
                // Close the Statement and the
                // Connection.
s.close();
c.close();
```

JDBC XA 分散式異動管理

JDBC XA 分散式異動管理類別可讓您在分散式異動中使用 IBM Toolbox for Java JDBC 驅動程式。 使用 XA 類別可使 IBM Toolbox for Java JDBC 驅動程式參與跨越多重資料來源的異動。

通常, XA 分散式異動管理類別是由交易管理程式 (與 JDBC 驅動程式分離) 所直接使用與控制。
分散式異動管理介面定義為 JDBC 2.0 Optional Package 及 Java Transaction API (JTA) 的一部份。 兩者皆由 S
jar 檔案的形式提供。分散式異動 管理介面也於 JDBC 3.0 API (與 Java 2 Platform 標準版, 版本 1.4 連結)
中受支援。

其它資訊, 請參閱 Sun 網站的JDBC 及JTA。

您可使用下列物件來啟用 IBM Toolbox for Java JDBC 驅動程式以參與 XA 分散式異動：

- AS400JDBCXADataSource - AS400JDBCXAConnection 物件的工廠。 此為AS400JDBCDataSource 的次類別。
- AS400JDBCXACConnection - 已置於儲存區的連線物件, 提供連線儲存區管理及 XA 資源管理的連結鉤。
- AS400JDBCXAResource - 用於 XA 異動管理的資源管理程式。

範例：使用 XA 類別

下列範例說明 XA 類別的簡單用法。請記住明細將由使用其它資料來源的作業填入。
此類型的程式碼通常出現在異動管理程式之內。

```
    // Create an XA data source for making the XA connection.
    AS400JDBCXADataSource xaDataSource = new
AS400JDBCXADataSource("myAS400");
    xaDataSource.setUser("myUser");
    xaDataSource.setPassword("myPasswd");

    // Get an XAConnection and get the associated XAResource.
    // This provides access to the resource manager.
    XAConnection xaConnection = xaDataSource.getXAConnection();
    XAResource xaResource = xaConnection.getXAResource();

    // Generate a new Xid (this is up to the transaction manager).
    Xid xid = ...;

    // Start the transaction.
    xaResource.start(xid, XAResource.TMNOFLAGS);

    // ...Do some work with the database...

    // End the transaction.
    xaResource.end(xid, XAResource.TMSUCCESS);

    // Prepare for a commit.
    xaResource.prepare(xid);

    // Commit the transaction.
```

```
xaResource.commit(xid, false);

// Close the XA connection when done.  This implicitly
// closes the XA resource.
xaConnection.close();
```

Jobs 類別

IBM Toolbox for Java Jobs 類別 (在 access 套件中) 可讓 Java 程式擷取與變更工作資訊。

註：Toolbox for Java 也提供 resource 類別，其提出同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。讀取 access 套件與 resource 套件中的類別後，
您便可選擇最適用於您應用程式的物件。與 Jobs 搭配使用的 Resource 類別為 RJob、RJobList 及 RJobLog。

請將 Jobs 類別與下列類型的工作資訊搭配使用：

- 日期及時間資訊
- 工作佇列
- 語言識別字
- 記載的訊息
- 輸出佇列
- 印表機資訊

access 套件中的 Job 類別如下：

- Job - 擷取及變更 iSeries 工作資訊
- JobList- 擷取 iSeries 工作清單
- JobLog - 代表 iSeries 的工作日誌

範例

列出屬於 特定使用者的工作，並列出具有工作狀態資訊的工作。

顯示 工作日誌 中的訊息

當設定某一值及取得某一值時，請使用快取 記憶體：

```
        try {
      // Creates AS400 object.
      AS400 as400 = new AS400("systemName");
      // Constructs a Job object
      Job job = new Job(as400,"QDEV002");
      // Gets job information
      System.out.println("User of this job :" + job.getUser());
      System.out.println("CPU used :" + job.getCPUUsed();
      System.out.println("Job enter system date : " +
job.getJobEnterSystemDate());
      // Sets cache mode
      job.setCacheChanges(true);
      // Changes will be store in the cache.
      job.setRunPriority(66);
      job.setDateFormat("*YMD");
      // Commit changes. This will change the value on the iSeries.
```

```
    job.commitChanges();
    // Set job information to system directly(without cache).
    job.setCacheChanges(false);
    job.setRunPriority(60);
} catch (Exception e)
{
    System.out.println(quot;error :" + e)
}
```

Job

Job 類別（在 access 套件中）可 讓 Java 程式擷取與變更伺服器工作資訊。

註： Toolbox for Java 也提供 Resource 類別，其提出同屬組織架構及一致的程式設計介面， 用以處理不同的 iSeries 物件與清單。讀取 access 套件與 resource 套件中的類別後，
您便可選擇最適用於您應用程式的物件。與 Jobs 搭配使用的 Resource 類別為 RJobList 及 RJobLog。

您可以透過 Job 類別擷取及變更下列類型的工作資訊：

- 工作佇列
- 輸出佇列
- 訊息日誌
- 印表機裝置
- 國家或地區 ID
- 日期格式

Job 類別也能使您一次變更一個單一值，或使用 setCacheChanges(true) 方法快取 數個變更，並使用 commitChanges() 方法 確定變更。如果未開啟快取，您將需要執行確定。

使用範例，瞭解如何設定及取得快取記憶體中的值，以便透 過 setRunPriority()
方法設定執行優先順序，以及透過 setDateFormat() 方法， 設定日期格式。

JobList 類別

您可以使用JobList類 別（在存取套件中）來列出 iSeries工作 。

註： Toolbox for Java 也提供resource 類別， 其提出同屬組織架構及一致的程式設計介面， 用以處理不同的 iSeries
物件與清單。閱讀過access 套 件和 resource 套
件中的類別相關資訊後， 您可以選擇最適合您的應用程式使用的物件。處理工作的 Resource 類別 為
RJob、RJobList 及RJobLog。

透過 JobList 類別， 您可以擷取下列：

- 所有工作
- 依名稱、工作號碼 或使用者的工作

使用getJobs()方法 傳回 iSeries 工作清單， 或getLength()方法傳回以上一個 getJobs() 擷取的工作數。

下面範例會列出系統中所有作用中的工作：

```
                    // Create an AS400 object. List the
                    // jobs on this iSeries.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create the job list object.
JobList jobList = new JobList(sys);

                    // Get the list of active jobs.
Enumeration list = jobList.getJobs();

                    // For each active job on the system
                    // print job information.
while (list.hasMoreElements())
{
    Job j = (Job) list.nextElement();

    System.out.println(j.getName() + "." +
                       j.getUser() + "." +
                       j.getNumber());
}
```

JobLog

JobLog 類別（在 access 套件中）以呼叫getMessages()的方式來擷取伺服器工作的工作日誌中的訊息。

註：Toolbox for Java 也提供resource 類別，其提出同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。閱讀 access 套件和 resource 套件中的類別相關資訊後，您可以選擇最適合您的應用程式使用的物件。 處理工作的 Resource 類別 為 RJob、RJobList 及RJobLog。

下列範例為指定的使用者列印工作日誌中的所有訊息：

```
                    // ... Setup work to create an AS400
                    // object and a jobList object has
                    // already been done

                    // Get the list of active jobs on
                    // the iSeries
    Enumeration list = jobList.getJobs();

                    // Look through the list to find a
                    // job for the specified user.
    while (list.hasMoreElements())
    {
       Job j = (Job) list.nextElement();

       if (j.getUser().trim().equalsIgnoreCase(userID))
       {
                    // A job matching the current user
                    // was found. Create a job log
                    // object for this job.
          JobLog jlog = new JobLog(system,
                                   j.getName(),
                                   j.getUser(),
                                   j.getNumber());

                    // Enumerate the messages in the job
                    // log then print them.
          Enumeration messageList = jlog.getMessages();

          while (messageList.hasMoreElements())
          {
              AS400Message message = (AS400Message)
messageList.nextElement();
              System.out.println(message.getText());
          }

       }
    }
```

Message 類別

AS400Message

AS400Message 物件 可讓 Java 程式擷取一個從前作業 (例如, 從指令呼叫) 產生的 iSeries 訊息。從訊息物件中, Java 程式可擷取下列項目：

- 包含訊息的 iSeries 檔案 庫 和 訊息檔
- 訊息 ID
- 訊息 類型
- 訊息 嚴重性
- 訊息 文字
- 訊息 解說文字

下列範例說明如何使用 AS400Message 物件：

```
                    // Create a command call object.
CommandCall cmd = new CommandCall(sys, "myCommand");

                    // Run the command
cmd.run();

                    // Get the list of messages that are
                    // the result of the command that I
                    // just ran
AS400Message[] messageList = cmd.getMessageList();

                   // Iterate through the list
                   // displaying the messages
for (int i = 0; i < messageList.length; i++)
{
   System.out.println(messageList[i].getText());
}
```

範例

下列範例告訴您如何利用 CommandCall 和 ProgramCall 來使用訊息清單。

- 範例： 以 CommandCall 使用訊息清單
- 範例： 以 ProgramCall 使用訊息清單

QueuedMessage

QueuedMessage 類別延伸 AS400Message 類別。

註：Toolbox for Java 也提供 resource 類別, 其提出同屬組織架構及一致的程式設計介面, 用以處理不同的 iSeries 物件與清單。閱讀 access 套 件和 resource 套

件中的類別相關資訊後，您可以選擇最適合您的應用程式使用的物件。 處理佇列訊息的 Resource 類別為 RQueuedMessage 。

QueuedMessage 類別存取關於 iSeries 訊息佇列中的訊息的資訊。透過此類別，Java 程式可以擷取：

- 關於訊息源起的資訊，如程式 工作名稱 、工作號碼 和 使用者
- 訊息 佇列
- 訊息 金鑰
- 訊息 回答狀態

下列範例列印現行（已登入）使用者的訊息佇列中的所有訊息：

```
                    // The message queue is on this iSeries.
    AS400 sys = new AS400(mySystem.myCompany.com);

                    // Create the message queue object.
                    // This object will represent the
                    // queue for the current user.
    MessageQueue queue = new MessageQueue(sys, MessageQueue.CURRENT);

                    // Get the list of messages currently
                    // in this user's queue.
    Enumeration e = queue.getMessages();

                    // Print each message in the queue.
    while (e.hasMoreElements())
    {
       QueuedMessage msg = e.getNextElement();
       System.out.println(msg.getText());
    }
```

MessageFile

MessageFile類別 可讓您從 iSeries 訊息檔接收訊息。MessageFile 類別會傳回一個含有訊息的 AS400Message 物件。使用 MessageFile 類別，您可以執行下列：

- 傳回包含訊息的訊息物件
- 傳回包含訊息中的替代文字的訊息物件

下列範例說明如何擷取及列印訊息：

```
    AS400 system = new AS400("mysystem.mycompany.com");
    MessageFile messageFile = new MessageFile(system);
    messageFile.setPath("/QSYS.LIB/QCPFMSG.MSGF");
    AS400Message message = messageFile.getMessage("CPD0170");

System.out.println(message.getText());
```

MessageQueue

MessageQueue 類別 可讓 Java 程式與 iSeries 訊息佇列相互作用。

註： Toolbox for Java 也提供 Resource 類別， 其提出同屬組織架構及一致的程式設計介面，用以處理不同的 iSeries 物件與清單。閱讀 access 套件和 resource 套件中的類別相關資訊後，您可以選擇最適合您的應用程式使用的物件。 處理訊息佇列的 Resource 類別為 RMessageQueue 。

MessageQueue 類別作為 QueuedMessage 類別的儲存區。 尤其，getMessages()方法傳回 QueuedMessage 物件清單。MessageQueue 類別可執行下列：

- 設定訊息佇列屬性
- 取得關於訊息佇列的資訊
- 接收訊息佇列中的訊息
- 傳送訊息到訊息佇列
- 回答訊息

下列範例列出現行使用者的訊息佇列中的訊息：

```
                // The message queue is on this iSeries.
AS400 sys = new AS400(mySystem.myCompany.com);

                // Create the message queue object.
                // This object will represent the
                // queue for the current user.
MessageQueue queue = new MessageQueue(sys, MessageQueue.CURRENT);

                // Get the list of messages currently
                // in this user's queue.
Enumeration e = queue.getMessages();

                // Print each message in the queue.
while (e.hasMoreElements())
{
   QueuedMessage msg = e.getNextElement();
   System.out.println(msg.getText());
}
```

NetServer

NetServer 類別代表 iSeries 伺服器中的 NetServer 服務。NetServer 物件可讓您查詢及修改 NetServer 服務的狀態與配置。

例如，您可使用 NetServer 類別來：

- 啟動 或停止 NetServer
- 取得所有目前檔案共用 及列印共用的清單
- 取得所有現行階段作業的清單
- 查詢及變更 屬性值 (使用繼承自 ChangeableResource 的方法)

註：若要使用 NetServer 類別，您需要具有 *IOSYSCFG 權限的伺服器使用者設定檔。

NetServer 類別為ChangeableResource及Resource 的延伸， 因此其提供了代表不同 NetServer 值及設定值的「屬性」集合。您 可查詢或變更 屬性，來存取或變更 NetServer 的配置。部份 NetServer 的屬性有：

- NAME
- NAME_PENDING
- DOMAIN
- ALLOW_SYSTEM_NAME
- AUTOSTART
- CCSID
- WINS_PRIMARY_ADDRESS

擱置中屬性

許多的 NetServer 屬性為擱置中屬性 (例如,NAME_PENDING)。 擱置中屬性代表下一次您在伺服器啟動 (或重新啟動) NetServer 時，生效的 NetServer 的值。

當您有一對相關的屬性，其中一個屬性為擱置中屬性，而另一個屬性為非擱置中屬性時：

- 擱置中屬性的狀態會為讀取/寫入，因此可加以變更
- 非擱置中屬性的狀態會為唯讀，因此可加以查詢但不能進行變更

其它 NetServer 類別

相關的 NetServer 類別可讓您取得與設定特定連線、階段作業、檔案共用及列印共用的相關詳細資訊：

- NetServerConnection 代表 NetServer 連線
- NetServerFileShare 代表 NetServer 檔案伺服器共用
- NetServerPrintShare 代表 NetServer 列印伺服器共用
- NetServerSession 代表 NetServer 階段作業

- NetServerShare - 代表 NetServer 共用

範例：使用 NetServer 物件變更 NetServer 的名稱

```
     // Create a system object to represent the iSeries server.
     AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWD");
     // Create an object with which to query and modify the NetServer.
     NetServer nServer = new NetServer(system);
     // Set the "pending name" to NEWNAME.
     nServer.setAttributeValue(NetServer.NAME_PENDING, "NEWNAME");
     // Commit the changes.  This sends the changes to the server.
     nServer.commitAttributeChanges();
     // The NetServer name will get set to NEWNAME the next time the
NetServer
     // is ended and started.
```

Permission 類別

permission 類別可讓您取得及設定物件權限資訊。 物件權限資訊也稱為許可權。Permission 類別代表許多使用者對特定物件的權限的集合。UserPermission 類別代表單一使用者對特定物件的權限。

Permission 類別

Permission類別 可讓您擷取及變更物件權限資訊。它包括許多有權使用物件的使用者的集合。Permission 物件容許 Java program 快取權限變更，直到呼叫 commit() 方法為止。 一旦呼叫了 commit() 方法，則到那時候所做的變更都會傳送到伺服器。Permission 類別提供的一些功能如下：

- addAuthorizedUser(): 新增授權使用者。
- commit(): 向伺服器確定許可權變更。
- getAuthorizationList(): 傳回物件的授權清單。
- getAuthorizedUsers(): 傳回授權使用者的列舉。
- getOwner(): 傳回物件擁有者的名稱。
- getSensitivityLevel(): 傳回物件的靈敏度層次。
- getType(): 傳回物件權限類型 (QDLO、QSYS 或 Root)。
- getUserPermission(): 傳回特定使用者的許可權給物件。
- getUserPermissions(): 傳回使用者的許可權列舉給物件。
- setAuthorizationList(): 設定物件的授權清單。
- setSensitivityLevel(): 設定物件的靈敏度層次。

範例

這個範例將告訴您如何建立一個許可權，新增授權使用者到物件。

```
// Create AS400 object
AS400 as400 = new AS400();

// Create Permission passing in the AS400 and object
Permission myPermission = new Permission(as400, "QSYS.LIB/myLib.LIB");

// Add a user to be authorized to the object
myPermission.addAuthorizedUser("User1");
```

UserPermission 類別

UserPermission類別代表單一特定使用者的權限。UserPermission 具有三種次類別，它們依據物件類型來處理權限：

UserPermission 類別	說明
DLOPermission	代表使用者對儲存在 QDLS 中的「文件檔案庫物件 (DLO)」的權限。

QSYSPermission	代表使用者對儲存在 QSYS.LIB 及包含在伺服器中的物件的權限。
RootPermission	代表使用對包含在主目錄結構中的物件的權限。RootPermissions 物件即是那些未包含在 QSYS.LIB 或 QDLS 的物件。

UserPermission 類別可讓您執行下列：

- 判斷使用者設定檔是否為群組設定檔
- 傳回使用者設定檔名稱
- 指出使用者是否具有權限
- 為授權清單管理 設定權限

範例

這個範例告訴您如何擷取對物件具有許可權的使用者和群組，並一次列印他們中的一位。

```
      // Create a system object.
      AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");

      // Represent the permissions to an object on the system, such as a
library.
      Permission objectInQSYS = new Permission(sys, "/QSYS.LIB/FRED.LIB");

      // Retrieve the various users/groups that have permissions set on that
object.
      Enumeration enum = objectInQSYS.getUserPermissions();
     while
(enum.hasMoreElements ())
      {
        // Print out the user/group profile names one at a time.
        UserPermission userPerm = (UserPermission)enum.nextElement();
        System.out.println(userPerm.getUserID());
      }
```

DLOPermission 類別

DLOPermission UserPermission 的次類別。DLOPermission 可讓您顯示及設定使用者對文件檔案庫物件 (DLO) 具有的權限 (稱為許可權)。

下列其中一個權限將指定給每一使用者。

權限值	說明
*ALL	使用者可以執行所有作業，但那些由授權清單管理所控制的作業 除外。
*AUTL	授權清單係用來決定文件的權限。
*CHANGE	使用者可以對物件變更及執行基本的功能。
*EXCLUDE	使用者無法存取物件。
*USE	使用者具有物件的作業權限、讀取權限及執行權限。

您必須使用下列方法之一，方能變更或決定使用者的權限：

- 使用getDataAuthority()來顯示使用者的權限值
- 使用setDataAuthority()來設定使用者的權限值

於設定許可權後，您必須由許可權類別使用 commit() 方法來傳送變更至伺服器。

關於許可權及權限的詳細資訊，請參閱第五章 iSeries 安全參照中的資源安全。

範例

這個範例告訴您如何擷取及列印 DLO 許可權，包括每一許可權的使用者設定檔。

```
      // Create a system object.

      AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");
      // Represent the permissions to a DLO object.
      Permission objectInQDLS = new Permission(sys, "/QDLS/MyFolder");

      // Print the object pathname and retrieve its permissions.
      System.out.println("Permissions on "+objectInQDLS.getObjectPath()+"
are as follows:");
      Enumeration enum = objectInQDLS.getUserPermissions();
     while
(enum.hasMoreElements ())
      {
        // For each of the permissions, print out the user profile name
        // and that user's authorities to the object.
        DLOPermission dloPerm = (DLOPermission)enum.nextElement();
        System.out.println(dloPerm.getUserID()+":
"+dloPerm.getDataAuthority());
```

}

QSYSPermission

QSYSPermission 為UserPermission類別的次類別。QSYSPermission 可讓您顯示及設定使用者對儲存在 QSYS.LIB 的傳統 iSeries 或 AS/400e 檔案庫結構中的 物件具有的許可權。儲存在 QSYS.LIB 中的物件，可經由設定系統定義的權限值、
或設定個別物件權限、或設定個別物件及資料權限，來設定它的權限。

下表列出並說明系統定義的有效權限值：

系統定義的權限值	說明
*ALL	使用者可以執行所有作業，但那些由授權清單管理所控制的作業除外。
*AUTL	授權清單係用來決定文件的權限。
*CHANGE	使用者可以對物件變更及執行基本的功能。
*EXCLUDE	使用者無法存取物件。
*USE	使用者具有物件的作業權限、讀取權限及執行權限。

每個系統定義的權限值實際上代表個別物件權限與資料權限的組合。
下表說明個別物件的系統定義權限與資料權限之間的關係：

系統定義的權限	物件權限					資料權限				
	Opr	Mgt	Exist	Alter	Ref	Read	Add	Upd	Dlt	Exe
All	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Change	Y	n	n	n	n	Y	Y	Y	Y	Y
Exclude	n	n	n	n	n	n	n	n	n	n
Use	Y	n	n	n	n	Y	n	n	n	Y
Autl	限定於使用者 (*PUBLIC) 和決定個別物件 和資料權限的指定授權清單才有效。									

Y指出可以指定的權限。
n 指出不能指定的權限。

指定系統定義的權限時，即自動指派適當的個別權限。同樣地，
指定各種個別權限時，即變更適當的個別權限值。當個別的物件權限與資料權限的
組合不對映單一的系統定義權限值時，該單一值則變成「使用者定義」。

請使用 getObjectAuthority()方法來顯示現行的系統定義權限。 請使用setObjectAuthority()
方法，透過單一值設定現行的系統定義權限。

使用適當的 set 方法，將個別物件權限值設定為 on 或 off：

- setAlter()
- setExistence()
- setManagement()
- setOperational()

- setReference()

使用適當的 set 方法，將個別資料權限值設定為 on 或 off：

- setAdd()
- setDelete()
- setExecute()
- setRead()
- setUpdate()

關於不同的權限如需更多資訊，請參閱第五章：資源安全，在iSeries 安全性 參照  。 有關使用 iSeries CL 指令來授予和編輯物件權限，如需詳細資訊， 請參閱 iSeries CL 指令 授予物件權限 (GRTOBJAUT) 和 編輯物件權限 (EDTOBJAUT)。

範例

這個範例告訴您如何擷取及列印 QSYS 物件的許可權。

```
      // Create a system object.
      AS400 sys = new AS400("MYAS400", "USERID", "PASSWORD");

      // Represent the permissions to a QSYS object.
      Permission objectInQSYS = new Permission(sys, "/QSYS.LIB/FRED.LIB");

      // Print the object pathname and retrieve its permissions.
      System.out.println("Permissions on "+objectInQSYS.getObjectPath()+"
are as follows:");
      Enumeration enum = objectInQSYS.getUserPermissions();
     while
(enum.hasMoreElements ())
      {
        // For each of the permissions, print out the user profile name
        // and that user's authorities to the object.
        QSYSPermission qsysPerm = (QSYSPermission)enum.nextElement();
        System.out.println(qsysPerm.getUserID()+":
"+qsysPerm.getObjectAuthority());
      }
```

RootPermission

RootPermission為UserPermission類別的次類別。RootPermission
類別可讓您顯示及設定使用者對包含在主目錄結構中的物件的許可權。

主目錄結構上的物件可以設定資料權限或物件權限。您可以把資料權限設定為下表所列的值。 請使用
getDataAuthority()方法來顯示現行值，使用setDataAuthority()方法來設定資料權限。

下表列出並說明有效資料權限值：

資料權限值	說明
*none	使用者沒有物件的權限。
*RWX	使用者具有讀取、新增、更新、刪除與執行權限。
*RW	使用者具有讀取、新增與刪除權限。
*RX	使用者具有讀取及執行權限。
*WX	使用者具有新增、更新、刪除與執行權限。
*R	使用者具有讀取權限。
*W	使用者具有新增、更新與刪除權限。
*X	使用者具有執行權限。
*EXCLUDE	使用者無法存取物件。
*AUTL	這個物件上的公用權限來自於授權清單。

這些物件權限可設定為下列一個或多個值：更新、存在、管理或參照。您可以使用
setAlter()、setExistence()、setManagement()或setReference()方法來設定值為開啟或關閉。

設定了物件的資料權限或物件權限之後，您必須從Permissions類別 使用commit() 方法來傳送變更到伺服器。

如需不同權限的相關資訊，請參閱 iSeries 安全性參照中第五章：資源安全。

範例

這個範例告訴您如何擷取及列印主物件的的許可權。

```
        // Create a system object.
        AS400 sys = new AS400("MYAS400", "使用者 ID", "密碼");

        // Represent the permissions to an object in the root file system.
        Permission objectInRoot = new Permission(sys, "/fred");

        // Print the object pathname and retrieve its permissions.
        System.out.println("許可權於 "+objectInRoot.getObjectPath()+"
如下：");
        Enumeration enum = objectInRoot.getUserPermissions();
```

```
    while
(enum.hasMoreElements ())
      {
        // For each of the permissions, print out the user profile name
        // and that user's authorities to the object.
        RootPermission rootPerm = (RootPermission)enum.nextElement();

System.out.println(rootPerm.getUserID()+" : "+rootPerm.getDataAuthority());
      }
```

Print 類別

列印物件包括排存檔、輸出佇列、印表機、印表機檔案、寫出器工作及 「高階功能列印 (AFP)」資源，這些包括字形組、格式定義、套印格式、頁面定義與頁面區段。 只有在「版本 3 版次 7」(V3R7) 和更新版本的 OS/400 系統中才可存取 AFP 資源。(嘗試對執行比 V3R7 還早的版本的系統開啟 AFPResourceList 將產生 RequestNotSupportedException 異常情況。)

IBM Toolbox for Java 的列印物件類別是依據基礎 類別PrintObject 及六個列印物件類型每一個的次類別所組織而成的。
基礎類別包含所有伺服器列印物件共用的方法與屬性。次類別包含專屬每一個次類型使用的方法與屬性。

print 類別使用於下列情形：

- 使用伺服器列印物件：
  - PrintObjectLis類別 - 用於列示及使用伺服器列印物件。(列印物件包括排存檔、輸出佇列、印表機、「 Advanced Function Printing」(AFP) 資源、印表機檔案及寫出器工作)
  - PrintObject基礎類別 - 供使用列印物件使用
- 擷取 PrintObject 屬性
- 使用 SpooledFileOutputStream 類別建立 新的 伺服器排存檔 (用於 EBCDIC 為主的印表機資料)
- 產生SNA 字元串流 (SCS) 印表機資料串流
- 使用 PrintObjectInputStream讀取排存檔與 AFP 資源
- 讀取排存檔，利用 PrintObjectPageInputStream 及 PrintObjectTransformedInputStream
- 檢視 Advanced Function Printing (AFP) 與 SNA Character Stream (SCS) 排存檔

範例

- 建立排存檔範例 說明 如何從輸入串流中建立伺服器的排存檔。
- 建立 SCS 排存檔範例 說明 如何使用 SCS3812Writer 類別產生 SCS 資料串流，以及如何將串流寫入到伺服器的排存檔中。
- 讀取排存檔範例 說明 如何讀取現存的伺服器排存檔。
- 第一個 非同步列示範例 說明如何以非同步方式列出系統中的所有排存檔，以及當建置清單時，如何使用 PrintObjectListListener 介面取得回饋
- 第二個 非同步列示範例 說明如何以非同步方式列出系統中的所有 排存檔，使用 PrintObjectListListener 介面
- 同步列示範例說明如何以同步方式列出系統中的所有排存檔

列出 Print 物件

您可以使用PrintObjectList類別及其次類別來使用列印物件的清單。 每一個次類別均具有一些方法，以便容許依據對特殊類型的 print 物件有意義的 條件，來過濾清單。例如，SpooledFileList容許您依據排存檔的建立者、排存檔所在的輸出佇列、紙張規格或排存檔的使用者資料，來過濾排存檔的清單。 僅有那些符合過濾條件的排存檔才會列示出來。如果未設定任何過濾條件，將使用每一個過濾條件的預設值。

若要從伺服器實際擷取列印物件的清單，請使用openSynchronously()或openAsynchronously()方法。除非已從伺服器擷取清單中的所有物件， 否則不會傳回 openSynchronously() 方法。openAsynchronously() 方法將會立即傳回，即使清單的建立時，呼叫程式可以在前景中執行其它工作。非同步開啟的清單同時也容許呼叫程式在物件傳回時，對使用者顯示這些物件。因為當物件傳回時，您可以看到它們，所以在使用者看來，回應時間似乎更快。事實上，就整體而言，回應時間可能更長，因為會對清單中的每一個物件做額外的處理。

如果清單是以非同步方式開啟，則呼叫程式可在清單建立時取得回應。例如，isCompleted()和 size()等方法會指出清單是否已經建立完成，或是傳回清單目前的大小。 其它方法例如waitForListToComplete()和 waitForItem()則容許呼叫程式等待清單完成，或等待一個特殊項目。除了呼叫這些 PrintObjectList 方法之外， 呼叫程式也可以透過清單登錄為聆聽者的方式，將通知呼叫程式在清單中所發生的事件。 若要登記事件或取消事件的登記，呼叫程式會使用PrintObjectListListener，然後再呼叫addPrintObjectListListener()來登記或呼叫 removePrintObjectListListener()來取消登記。下表將顯示從 PrintObjectList 傳遞的事件。

PrintObjectList 事件	事件傳遞時
listClosed	清單結束時。
listCompleted	清單完成時。
listErrorOccurred	如果擷取清單時有丟出任何異常情況時。
listOpened	清單開啟時。
listObjectAdded	在清單中新增物件時。

在清單開啟並且清單中的物件已處理過之後，請使用close()方法來關閉清單。這將釋出在開啟期間，已配置給垃圾收集器的任何資源。關閉清單之後， 可修改它的過濾條件，然後可再重新開啟該清單。

列出列印物件後，伺服器會傳送所列出的每一個列印物件的屬性， 並隨著列印物件一起儲存。更新這些屬性可以使用 PrintObject 類別中的update()方法。伺服器傳回哪些屬性，可根據列出的列印物件類型來決定。有關各類型之列印物件的屬性有一份預設清單存在，想要置換時可使用 PrintObjectList 中的setAttributesToRetrieve()方法。請參閱擷取 PrintObject 屬性章節，取得各類型 print 物件支援的屬性清單。

僅在版本 3 版次 7 與更新版的 OS/400 中，才可列出 AFP 資源。 對 V3R7 之前的舊版系統開啟ResourceList會產生RequestNotSupportedException的異常。

範例

非同步清單範例 1

非同步清單範例 2

同步清單範例

使用 Print 物件

PrintObject是一種抽象類別。(抽象類別不容許您建立類別的實例。
相反地，您必須建立其次類別之一的實例。）使用任何下列方法來建立次類別的物件：

- 如果知道系統和物件的識別屬性，請呼叫物件的公用建構元來明確地建構物件。
- 您可以使用PrintObjectList次類別來建置物件清單，然後從清單中取得個別物件。
- 由於所呼叫的某方法或 set 方法，結果會建立並傳回某物件。 例如,WriterJob類別中的 靜態方法start(會傳回 WriterJob 物件。

請使用基本類別、 PrintObject及其次類別來使用伺服器列印物件：

- OutputQueue
- Printer
- PrinterFile
- SpooledFile
- WriterJob

擷取 PrintObject 屬性

您可以使用屬性 ID 以及來自基礎 PrintObject 類別的其中一個方法來擷取列印物件屬性：

- 請使用 getIntegerAttribute(int attrib) 來擷取整數類型屬性。
- 請使用 getFloatAttribute(int attrib) 來擷取浮點類型屬性。
- 請使用 getStringAttribute(int attrib) 來擷取字串類型屬性。

attributeID 參數是一個識別要擷取哪個屬性的整數。在基礎 PrintObject 類別中全部 ID 都定義成公用常數。PrintAttributes 檔包含每一個屬性 ID 的登錄。 登錄包括屬性說明和它的類型 (整數、浮點或字串)。若要列示可能使用這些方法所擷取的屬性，請選取下列鏈結：

- 用於 AFP 的 AFPResourceAttrs
- 用於輸出佇列的 OutputQueueAttrs
- 用於印表機的 PrinterAttrs
- 用於印表機檔案的 PrinterFileAttrs
- 用於排存檔的 SpooledFileAttrs
- 用於寫出器工作的 WriterJobAttrs

若要達到可接受的效能，這些屬性會複製到從屬站中。如果隱含地建立物件，則在列示物件時或第一次需要物件時會複製這些屬性。此動作可在為應用程式需要擷取屬性時，防止將物件傳送到主電腦。此動作也有可能使 Java 列印物件實例包含關於伺服器物件的過期資訊。 物件的使用者可對物件呼叫 update() 來復新全部屬性。此外，如果應用程式對物件呼叫任何會變更物件屬性的方法，則會自動更新屬性。例如，若輸出佇列有 RELEASED 的狀態屬性 (getStringAttribute(ATTR_OUTQSTS)；傳回字串 RELEASED)，hold() 方法 在輸出佇列中被呼叫，則取得狀態屬性，之後傳回 HELD。

setAttributes 方法

您可以使用 setAttributes 方法來變更排存檔及印表機檔案物件的屬性。選取下列鏈結取得清單，或選取可設定的屬性：

- 用於印表機檔案的 PrinterFileAttrs
- 用於排存檔的 SpooledFileAttrs

setAttributes 方法取用 PrintParameterList 參數，這是用來保留屬性 ID 及其值的集合的類別。 清單原本是空的，呼叫程式可以對它使用不同的 setParameter() 方法，新增屬性到清單中。

PrintParameterList 類別

您可以使用 PrintParameterList 類別來傳送某屬性群組到某個使用許多屬性作為參數的方法。 例如，您可以使用 SpooledFile.send() 方法，傳送使用使用 TCP (LPR) 的排存檔。PrintParameterList 物件包含 send 指令的必要參數 (如遠程系統與佇列) 以及想要的任何選用參數， 如傳送排存檔之後是否要刪除排存檔。在這些情況中， 方法文件提供屬性的清單。PrintParameterList setParameter() 方法不會檢查設定哪些屬性及對屬性設定哪些值。PrintParameterList setParameter() 方法僅包含要傳送至此方法的值。一般而言，Print 系統忽略 PrintParameterList 中的額外屬性，而且對屬性使用的不合法值會在伺服器中加以診斷。

印表機檔案屬性

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法，可為印表機檔案擷取下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面邊距橫向位移
- ATTR_BKMGN_DWN - 底面邊距向下位移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下位移
- ATTR_BKOVL_ACR - 底面套板橫向位移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字形組檔案庫名稱
- ATTR_CODEPAGE - 字碼頁
- ATTR_CODEDFNT - 字碼字型名稱
- ATTR_CONTROLCHAR - 控制字元
- ATTR_CONVERT_LINEDATA - 轉換行式資料
- ATTR_COPIES - 份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS (雙位元組字集) 延伸字元
- ATTR_DBCSROTATE - DBCS (雙位元組字集) 字元旋轉
- ATTR_DBCSCPI - DBCS (雙位元組字集) 每吋字元數
- ATTR_DBCSSISO - DBCS (雙位元組字集) SO/SI 間距
- ATTR_DFR_WRITE - 延遲寫入
- ATTR_PAGRTT - 頁旋轉度
- ATTR_EDGESTITCH_NUMSTAPLES - 裝訂邊縫線數
- ATTR_EDGESTITCH_REF - 邊縫線參照
- ATTR_EDGESTITCH_REFOFF - 邊縫線參照
- ATTR_ENDPAGE - 終止頁
- ATTR_FILESEP - 檔案分隔頁
- ATTR_FOLDREC - 摺疊記錄
- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向位移
- ATTR_FTMGN_DWN - 正面邊距向下位移
- ATTR_FRONT_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向位移
- ATTR_FTOVL_DWN - 正面套板向下位移
- ATTR_CHAR_ID - 圖形字集
- ATTR_JUSTIFY - 硬體調整
- ATTR_HOLD - 保留排存檔

- ATTR_LPI - 每吋行數
- ATTR_MAXRCDS - 最大排存輸出記錄
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERFLOW - 溢位行號
- ATTR_PAGE_DEFINITION - 頁面定義整合檔案系統
- ATTR_PAGELEN - 頁長度
- ATTR_MEASMETHOD - 測量方法
- ATTR_PAGEWIDTH - 頁寬度
- ATTR_MULTIUP - 每邊的頁數
- ATTR_POINTSIZE - 字體大小
- ATTR_FIDELITY - 列印清晰度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTTEXT - 列印文字
- ATTR_PRINTER - 印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型
- ATTR_RPLUNPRT - 置換不可列印的字元
- ATTR_RPLCHAR - 置換字元
- ATTR_SADDLESTITCH_NUMSTAPLES - 馬鞍型裝訂縫針數
- ATTR_SADDLESTITCH_REF - 馬鞍型縫針參照
- ATTR_SAVE - 儲存排存檔
- ATTR_SRCDRWR - 來源匣
- ATTR_SPOOL - 排存資料
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_DESCRIPTION - 本文說明
- ATTR_UNITOFMEAS - 測量單位
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

設定屬性

使用 setAttributes() 方法設定印表機檔案的下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面邊距橫向位移
- ATTR_BKMGN_DWN - 底面邊距向下位移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下位移
- ATTR_BKOVL_ACR - 底面套板橫向位移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字形組檔案庫名稱
- ATTR_CODEPAGE - 字碼頁

- ATTR_CODEDFNT - 字碼字型名稱
- ATTR_CONTROLCHAR - 控制字元
- ATTR_CONVERT_LINEDATA - 轉換行式資料
- ATTR_COPIES - 份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS (雙位元組字集) 延伸字元
- ATTR_DBCSROTATE - DBCS (雙位元組字集) 字元旋轉
- ATTR_DBCSCPI - DBCS (雙位元組字集) 每吋字元數
- ATTR_DBCSSISO - DBCS (雙位元組字集) SO/SI 間距
- ATTR_DFR_WRITE - 延遲寫入
- ATTR_PAGRTT - 頁旋轉度
- ATTR_EDGESTITCH_NUMSTAPLES - 裝訂邊縫線數
- ATTR_EDGESTITCH_REF - 邊縫線參照
- ATTR_EDGESTITCH_REFOFF - 邊縫線參照
- ATTR_ENDPAGE - 終止頁
- ATTR_FILESEP - 檔案分隔頁
- ATTR_FOLDREC - 摺疊記錄
- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向位移
- ATTR_FTMGN_DWN - 正面邊距向下位移
- ATTR_FRONT_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向位移
- ATTR_FTOVL_DWN - 正面套板向下位移
- ATTR_CHAR_ID - 圖形字集
- ATTR_JUSTIFY - 硬體調整
- ATTR_HOLD - 保留排存檔
- ATTR_LPI - 每吋行數
- ATTR_MAXRCDS - 最大排存輸出記錄
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE - 輸出佇列整合檔案系統名稱
- ATTR_OVERFLOW - 溢位行號
- ATTR_PAGE_DEFINITION - 頁面定義整合檔案系統
- ATTR_PAGELEN - 頁長度
- ATTR_MEASMETHOD - 測量方法
- ATTR_PAGEWIDTH - 頁寬度
- ATTR_MULTIUP - 每邊的頁數
- ATTR_POINTSIZE - 字體大小
- ATTR_FIDELITY - 列印清晰度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTTEXT - 列印文字
- ATTR_PRINTER - 印表機
- ATTR_PRTDEVTYPE - 印表機裝置類型

- ATTR_RPLUNPRT - 置換不可列印的字元
- ATTR_RPLCHAR - 置換字元
- ATTR_SADDLESTITCH_NUMSTAPLES - 馬鞍型裝訂縫針數
- ATTR_SADDLESTITCH_REF - 馬鞍型縫針參照
- ATTR_SAVE - 儲存排存檔
- ATTR_SRCDRWR - 來源匣
- ATTR_SPOOL - 排存資料
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_DESCRIPTION - 本文說明
- ATTR_UNITOFMEAS - 測量單位
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

排存檔屬性

擷取屬性

使用適當的 getIntegerAttribute()、getStringAttribute() 或 getFloatAttribute() 方法， 可為排存檔擷取下

- ATTR_AFP - 進階功能列印
- ATTR_ALIGN - 調整頁面
- ATTR_BKMGN_ACR - 底面邊距橫向位移
- ATTR_BKMGN_DWN - 底面邊距向下位移
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下位移
- ATTR_BKOVL_ACR - 底面套板橫向位移
- ATTR_CPI - 每吋字元數
- ATTR_CODEDFNTLIB - 編碼字形組檔案庫名稱
- ATTR_CODEDFNT - 字碼字型名稱
- ATTR_CODEPAGE - 字碼頁
- ATTR_CONTROLCHAR - 控制字元
- ATTR_COPIES - 份數
- ATTR_COPIESLEFT - 待產生份數
- ATTR_CORNER_STAPLE - 裝訂角
- ATTR_CURPAGE - 現行頁
- ATTR_DATE - 物件建立日期
- ATTR_DATE_WTR_BEGAN_FILE - 寫出器開始處理排存檔日期
- ATTR_DATE_WTR_CMPL_FILE - 寫出器完成處理排存檔日期
- ATTR_DBCSDATA - 使用者指定的 DBCS 資料
- ATTR_DBCSEXTENSN - DBCS (雙位元組字集) 延伸字元
- ATTR_DBCSROTATE - DBCS (雙位元組字集) 字元旋轉
- ATTR_DBCSCPI - DBCS (雙位元組字集) 每吋字元數
- ATTR_DBCSSISO - DBCS (雙位元組字集) SO/SI 間距
- ATTR_PAGRTT - 頁旋轉度
- ATTR_EDGESTITCH_NUMSTAPLES - 裝訂邊縫線數
- ATTR_EDGESTITCH_REF - 邊縫線參照
- ATTR_EDGESTITCH_REFOFF - 邊縫線參照位移
- ATTR_ENDPAGE - 終止頁
- ATTR_FILESEP - 檔案分隔頁
- ATTR_FOLDREC - 摺疊記錄
- ATTR_FONTID - 字型 ID
- ATTR_FORM_DEFINITION - 套表定義的物件整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FTMGN_ACR - 正面邊距橫向位移
- ATTR_FTMGN_DWN - 正面邊距向下位移

- ATTR_SAVE - 儲存排存的檔案
- ATTR_SRCDRWR - 來源匣
- ATTR_SPOOLFILE - 排存檔名稱
- ATTR_SPLFNUM - 排存檔號碼
- ATTR_SPLFSTATUS - 排存檔狀態
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_SYSTEM - 系統建立位置
- ATTR_TIME - 物件建立時間
- ATTR_TIME_WTR_BEGAN_FILE - 寫出器開始處理排存檔時間
- ATTR_TIME_WTR_CMPL_FILE - 寫出器完成處理排存檔時間
- ATTR_PAGES - 總頁數
- ATTR_UNITOFMEAS - 測量單位
- ATTR_USERCMT - 使用者註解
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFDATA - 使用者定義資料
- ATTR_USRDEFFILE - 使用者定義檔
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義物件整合檔案系統名稱

設定屬性

使用 setAttributes() 方法設定排存檔的下列屬性：

- ATTR_ALIGN - 調整頁面
- ATTR_BACK_OVERLAY - 底面套板整合檔案系統名稱
- ATTR_BKOVL_DWN - 底面套板向下位移
- ATTR_BKOVL_ACR - 底面套板橫向位移
- ATTR_COPIES - 份數
- ATTR_ENDPAGE - 終止頁
- ATTR_FILESEP - 檔案分隔頁
- ATTR_FORM_DEFINITION - 套表定義整合檔案系統名稱
- ATTR_FORMFEED - 換頁
- ATTR_FORMTYPE - 紙張規格
- ATTR_FRONTSIDE_OVERLAY - 正面套板整合檔案系統名稱
- ATTR_FTOVL_ACR - 正面套板橫向位移
- ATTR_FTOVL_DWN - 正面套板向下位移
- ATTR_OUTPTY - 輸出優先順序
- ATTR_OUTPUT_QUEUE -輸出佇列整合檔案系統名稱
- ATTR_MULTIUP - 每邊的頁數
- ATTR_FIDELITY - 列印清晰度
- ATTR_DUPLEX - 雙面列印
- ATTR_PRTQUALITY - 列印品質
- ATTR_PRTSEQUENCE - 列印順序

- ATTR_PRINTER - 印表機
- ATTR_RESTART - 重新啟動列印
- ATTR_SAVE - 儲存排存的檔案
- ATTR_SCHEDULE - 排存的輸出時程表
- ATTR_STARTPAGE - 起始頁
- ATTR_USERDATA - 使用者資料
- ATTR_USRDEFOPT - 使用者定義選項
- ATTR_USER_DEFINED_OBJECT - 使用者定義整合檔案系統名稱

建立新的排存檔

您可以使用SpooledFileOutputStrea類別來建立新的伺服器排存檔。從標準 JDK java.io.OutputStream 類別中衍生此類別；建構此類別之後， 可在有使用 OutputStre方使的用此類別。

建立新 SpooledFileOutputStream 時，呼叫程式可能指定下列項目：

- 要使用的印表機檔案
- 要存放排存檔的輸出佇列
- PrintParameterList 物件，可包含參數來置換印表機檔案中的欄位

這些參數全部都是選用的（呼叫程式可能對任何一個或全部參數傳送空值）。如果未指定印表機檔案，則網路列印伺服器會使用預設網路列印佇NPSPRTF。 此處使用輸出佇列參數是為了方便起見；也可在 PrintParameterList 中指定它。如果這兩處都指定輸出佇列參數，則 PrintParameterList 欄位會置換輸出佇列參數。Sp請參閱FileOutputStream 建構的文件，取得完整的清單，當中有列出哪些屬性可設定 在 PrintParameterList 中，來建立新的排存檔。

請使用一個 write()方法將資料寫入排存檔。SpooledFileOutputStream 物件會緩衝資料， 然後等到結束輸出串流或緩衝區已滿之後再傳送資料。基於兩個理由需要 執行緩衝：

- 容許自動資料鍵入（請參閱 排存檔中的資料串流類型 ）來分析滿緩衝區資料，以判斷資料類型。
- 因為不是每一個寫入要求都傳送至伺服器，所以可使輸出串流更快輸出。

請使用 flush()方法以強制資料寫入伺服器。

呼叫程式將資料寫入到新的排存檔之後，會呼叫lose()方法來關閉該排存檔。排存檔關閉之後，便無法對此檔案寫入資料。 一旦關閉排存檔後，藉由立刻呼叫 getSpooledFile() 方法，呼叫程式即可取得代表排存檔的 SpooledFile 物件的參照。

排存檔中的資料串流類型

使用排存檔的「印表機資料類型」屬性來設定要存放在排存檔的資料類型。如果呼叫程式未指定印表機資料類型，則預設值會使用自動資料鍵入。本方法查看排存檔資料的最前面幾千個位元組，來判斷它是適合「SNA 字元串流 (SCS)」或「高階功能列印資料串流 (AFPDS)」資料串流架構， 然後適當地設定屬性。如果排存檔的位元組不符合這些架構ii 顯示成 *USERASCII。
大部份時間都可使用自動資料鍵入。除非呼叫程式發生無法使用自動資料鍵入的特殊情況， 否則呼叫程式通常會使用自動資料鍵入。在那種情程式可設定「印表機資料類型」屬性為特定值 (例如，*SCS)。如果呼叫程式要使用印表機檔案中的印表機資料，則呼叫程式必須使用特殊值 *PRTF。 建立排存檔時如果呼叫程式置換預設資料
請確定存放於排存檔的資料符合此資料類型屬性。若將非 SCS 資料存放於某個標示為接收 SCS 資料的排存檔， 則會從主電腦發出錯誤訊息並失 h排存檔。

一般而言，本屬性可擁有三個值：

- *SCS - EBCDIC，以文字為主的印表機資料串流。
- *AFPDS （進階功能呈現資料串流） - 伺服器支援的另一種資料串流。*AFPDS 可包含文字、影像以及圖形，而且可在頁面區段中使用外部資源格式，如頁面與影像。
- *USERASCII - 指任何非 SCS 和非 AFPDS 印表機資料，伺服器只負責傳送。Postscript 和 HP-PCL 資料串流是要存放在 *USERASCII 排存檔的資料串流範例

範例

建立排存檔範例

建立 SCS 排存檔範例

產生 SCS 資料串流

若要產生要在某些連接至伺服器的印表機上列印的排存檔，則可能必須建立「SNA 字元串流 (SCS)」資料串流。(SCS 是以文字為基礎的 EBCDIC 資料串流，可在 SCS 印表機、IPDS 印表機或 PC 印表機上列印。) 在伺服器中使用模擬程式或主電腦列印轉換來轉換之後可列印出 SCS。

您可以使用 SCS 寫出器類別來產生這樣的 SCS 資料串流。SCS 寫出器類別轉換 Java Unicode 字元和格式化選項成為 SCS 資料串流。五個 SCS 寫出器類別產生不同層次的 SCS 資料串流。呼叫程式應選擇符合呼叫程式或一般使用者要列印的最終目的印表機的寫出器。

使用下列 SCS 寫出器類別來產生 SCS 印表機資料串流：

SCS 寫出器類別	說明
SCS5256Writer	最簡單的 SCS 寫出器類別。支援文字、回車、換行、新行、換頁、絕對水平及垂直定位、相對水平及垂直定位以及設定垂直格式。
SCS5224Writer	擴充 5256 寫出器，並新增方法來設定每吋字元數 (CPI) 以及每吋行數 (LPI)。
SCS5219Writer	擴充 5224 寫出器，並新增對下列項目的支援：左邊距、底線、紙張規格 (紙張或信封)、紙張大小、列印品質、字碼頁、字集、來源紙匣號碼以及目的紙匣號碼。
SCS5553Writer	擴充 5219 寫出器，並新增對字元旋轉、格線以及字型比例的支援。5553 是雙位元組字集 (DBCS) 資料串流。
SCS3812Writer	擴充 5219 寫出器，並新增對粗體、雙面列印、文字方向以及字型的支援。

若要建構 SCS 寫出器，則呼叫程式需要輸出串流和編碼 (可選用的)。資料串流會寫入輸出串流。若要建立 SCS 排存檔，則呼叫程式必須先建構 SpooledFileOutputStream，然後使用 SpooledFileOutputStream 來建構 SCS 寫出器物件。此編碼參數提供目標 EBCDIC 編碼字集識別字 (CCSID) 來轉換字元。

一旦建構寫出器後，請使用 write() 方法來輸出文字。請使用 carriageReturn()、lineFeed() 和 newLine() 等方法將寫入游標定位在頁中。請使用 endPage() 方法來結束現行頁並起始新頁。

寫入所有資料之後，請使用 close() 方法結束資料串流並關閉輸出串流。

讀取排存檔與 AFP 資源

您可以使用PrintObjectInputStream類別從伺服器中讀取排存檔的原始內容或 「進階功能列印」(Advanced Function Printing, AFP) 資源。 此類別擴充標準 JDK java.io.InputStream 類別， 因此可在有使用 InputStream 的任何地方使用它。

要取得 PrintObjectInputStream 物件，可呼叫 SpooledFile 類別的實例中的getInputStream()方法或呼叫 AFPResource 類別的實例中的getInputStream()方法。版本 3 版次 2 (V3R2)、V3R7 以及 OS/400 程式之更新版本都支援取得排存檔的輸入串流。V3R7 和更新版本都支援取得 AFP 資源的輸入串流。

請使用一個 read()方法自輸入串流中讀取資料。
這些方法會傳回實際讀取的位元組數，如果沒有讀取到位元組或到達檔案結尾，則傳回 -1。

您可以使用 PrintObjectInputStream 的available()方法來傳回排存檔或 AFP 資源中的總位元組數。PrintObjectInputStream 類別支援標示輸入串流，以便 PrintObjectInputStream 一律以markSupported()方法傳回 true。呼叫程式可以使用mark() 和 reset()方法，將輸入串流中的現行讀取位置往後移。請使用skip()方法在輸入串流中向前移動讀取位置但不讀取資料。

範例

讀取排存檔範例

使用 PrintObjectPageInputStream 與 PrintObjectTransformedInputStream 讀取排存檔

您可以使用PrintObjectPageInputStre類別一次一頁地讀取 server AFP 與 SCS 排存檔的資料。

使用getPageInputStream()方法可以取得 PrintObjectPageInputStream 物件。

使用一個 read()方法從輸入串流中讀取資料。所有這些方法會傳回實際讀取的位元組數，
如果沒有讀取到位元組或到達頁結尾，則傳回 -1。

您可以使用 PrintObjectPageInputStream 的available()
方法來傳回現行頁的總位元組數目。PrintObjectPageInputStream 類別支援標示輸入串流，以便使
PrintObjectPageInputStream 一律從markSupported()方法傳回 true。呼叫程式可以使用mark() 和 reset()
方法將輸入串流中的現行讀取位置向後移， 以便後續讀取可以重新讀取同一位元組。呼叫程式可以使用skip()
方法在輸入串流中向前移動讀取位置但不讀取資料。

然而，當想要轉換整個排存檔資料串流時，請使用 PrintObjectTransformedInputStream類別。

產品授權

ProductLicense 類別讓您能夠要求 iSeries 上所安裝產品的授權。 為了與其它 iSeries 授權使用者相容，類別在要求或釋放授權時， 是透過 iSeries 產品授權支援執行。

類別並不強制實施授權原則，但會傳回足夠的資訊， 使得應用程式能夠強制實施原則。當要求授權時，ProductLice 類別 會傳回要求的狀態 -- 授權授予或者拒絕。如果要求遭到拒絕， 應用程式必須停用需要授權的行為，因為 IBM Toolbox for Java 不知道 應停用哪個功能。

請使用 ProductLicense 類別與 iSeries 授權支援來強制您的應用程式的授權：

- 您的應用程式的伺服端使用 iSeries 授權支援來註冊您的產品和授權條款。
- 您的應用程式的從屬端使用 ProductLicense 物件來要求和釋出授權。

範例：ProductLicense 實務

例如，假定您的客戶購買了您產品的 15 套並行使用授權。並行使用表示 可同時由 15 位使用者使用產品，但不限定 15 位特定的使用者。可以為 組織內的任何 15 位使用者。此資訊是以 iSeries 授權支援登記註冊的。當使用者連接您的應用程式時， 會使用 ProductLicense 類別來要求授權。

- 當並行使用者人數少於 15，要求會順利完成，您的應用程式也就執行。
- 當有第 16 位使用者連線時，ProductLicense 要求便告失敗。您的應用程式就會 顯示錯誤訊息，並且終止

當有使用者停止執行應用程式時，您的應用程式就會經由 ProductLicense 類別釋出授權。授權即可供他人使用。

如需更多資訊和程式碼範例，請參閱 ProductLicense javadoc

ProgramCall 類別

ProgramCall類別 可讓 Java 程式呼叫 iSeries 程式。您可以使用ProgramParameter 類別指定輸入、輸出和輸入/輸出參數。當執行程式時， 輸出及輸入/輸出參數會含有 iSeries 程式所傳回的資料。如果 iSeries 程式無法順利 執行，Java 程式可以擷取產生的 iSeries 訊息，成為一個AS400Message 物件清單。

必要的參數如下：

- 要執行的程式與參數
- 代表具有該程式的 iSeries AS400系統的物件。

經由 setProgram()方法， 程式名稱和參數清單可以設定在建構元上，或設定在 run()方法上。run() 會方法呼叫該程式。

ProgramCall 物件類別使 AS400 物件連接到 iSeries。

下面範例說明如何使用 ProgramCall 類別：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program object. I choose
                    // to set the program to run later.
ProgramCall pgm = new ProgramCall(sys);

                    // Set the name of the program.
                    // Because the program does not take
                    // any parameters, pass null for the
                    // ProgramParameter[] argument.
pgm.setProgram(QSYSObjectPathName.toPath("MYLIB",
                                         "MYPROG",
                                         "PGM"));

                    // Run the program. My program has
                    // no parms. If it fails to run, the failure
                    // is returned as a set of messages
                    // in the message list.
if (pgm.run() != true)
{
                    // If you get here, the program
                    // failed to run. Get the list of
                    // messages to determine why the
                    // program didn't run.
   AS400Message[] messageList = pgm.getMessageList();

                    // ... Process the message list.
}

                    // Disconnect since I am done
                    // running programs
```

```
sys.disconnectService(AS400.COMMAND);
```

ProgramCall 物件需要程式的整合檔案系統路徑名稱。

使用 ProgramCall 類別會使 AS400 物件連接到 iSeries。管理連線 取得關於管理連線的資訊。

預設行為是即使當 Java 程式與 iSeries 程式位於同一個伺服器上， 也可以讓 iSeries 程式在不同的伺服器工作中執行。您可以置換預設行為，並使用setThreadSafe()方法讓 iSeries 程式在 Java 工作中執行。

使用 ProgramParameter 物件

您可以使用ProgramParameter物件，在 Java 程式與 iSeries 程式之間傳遞參數資料。setInputData() 方法設定輸入資料。在執行程式後，可 以用getOutputData() 方法擷取輸出資料。每一個參數都是一個位元組陣列。Java 程式必須在 Java 與 iSeries 格式之間轉換位元組陣列資料轉換 類別提供轉換資料的方法。 參數會新增到 ProgramCall 物件作為清單。

下列範例說明如何使用 ProgramParameter 物件來傳送參數資料。

```
                    // Create an AS400 object
AS400 sys = new AS400("mySystem.myCompany.com");

                    // My program has two parameters.
                    // Create a list to hold these
                    // parameters.
ProgramParameter[] parmList = new ProgramParameter[2];

                    // First parameter is an input
                    // parameter
byte[] key = {1, 2, 3};
parmList[0] = new ProgramParameter(key);

                    // Second parameter is an output
                    // parameter. A four-byte number
                    // is returned.
parmList[1] = new ProgramParameter(4);

                    // Create a program object
                    // specifying the name of the
                    // program and the parameter list.
ProgramCall pgm = new ProgramCall(sys,
                                  "/QSYS.LIB/MYLIB.LIB/MYPROG.PGM",
                                  parmList);

                    // Run the program.
if (pgm.run() != true)
{

                    // If the iSeries cannot run the
                    // program, look at the message list
                    // to find out why it didn't run.
```

```
    AS400Message[] messageList = pgm.getMessageList();

}
else
{
                    // Else the program ran. Process the
                    // second parameter, which contains
                    // the returned data.

                    // Create a converter for this
                    // iSeries data type
    AS400Bin4 bin4Converter = new AS400Bin4();

                    // Convert from iSeries type to Java
                    // object. The number starts at the
                    // beginning of the buffer.
    byte[] data = parmList[1].getOutputData();
    int i = bin4Converter.toInt(data);
}

                    // Disconnect since I am done
                    // running programs
sys.disconnectService(AS400.COMMAND);
```

QSYSObjectPathName 類別

您可以使用QSYSObjectPathName 類別來代表整合檔案系統中的物件。
使用這個類別來建立一個整合檔案系統名稱, 或將整合檔案系統名稱剖析為它的元件。

數個 IBM Toolbox for Java 類別需要一個整合檔案系統路徑名稱才能加以使用。 請使用 QSYSObjectPathName 物件, 來建置名稱。

下面範例說明如何使用 QSYSObjectPathName 類別：

範例 1： ProgramCall 物件需要要呼叫的伺服器程式的整合檔案系統名稱。 QSYSObjectPathName 物件是用來建置名稱。若要使用 QSYSObjectPathName, 呼叫檔案庫 REPORTS 中的程式 PRINT_IT：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program call object.
ProgramCall pgm = new ProgramCall(sys);

                    // Create a path name object that
                    // represents program PRINT_IT in
                    // library REPORTS.
QSYSObjectPathName pgmName = new QSYSObjectPathName("REPORTS",
                                                    "PRINT_IT",
                                                    "PGM");

                    // Use the path name object to set
                    // the name on the program call
                    // object.
pgm.setProgram(pgmName.getPath());

                    // ... run the program, process the
                    // results
```

範例 2： 如果 AS400 物件的名稱只用一次, 則 Java 程式可使用toPath()方法建置該名稱。這個方法比建立 QSYSObjectPathName 物件更有效。

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a program call object.
ProgramCall pgm = new ProgramCall(sys);

                    // Use the toPath method to create
                    // the name that represents program
                    // PRINT_IT in library REPORTS.
pgm.setProgram(QSYSObjectPathName.toPath("REPORTS",
                                                    "PRINT_IT",
                                          "PGM"));

                    // ... run the program, process the
```

```
                    // results
```

範例 3：在此範例中，有提供一個整合檔案系統路徑給 Java 程式。QSYSObjectPathName **類別可用來將這個名稱剖析為它的元件：**

```
                    // Create a path name object from
                    // the fully qualified integrated
                    // file system name.
    QSYSObjectPathName ifsName = new QSYSObjectPathName(pathName);

                    // Use the path name object to get
                    // the library, name and type of
                    // server object.
    String library = ifsName.getLibraryName();
    String name    = ifsName.getObjectName();
    String type    = ifsName.getObjectType();
```

記錄層次存取類別

記錄層次存取類別可提供執行下列的能力：

- 指定下列其中一項來建立 iSeries 實體檔：
  - 記錄長度
  - 現存的資料說明規格 (DDS) 原始檔
  - RecordFormat 物件
- 從 iSeries 實體檔或邏輯檔案中擷取記錄格式， 或從 iSeries 多重格式邏輯檔案中擷取記錄格式。

  備註： 不會完整地擷取檔案的記錄格式。 表示當設定 AS400File 物件的記錄格式時，將使用所擷取的記錄格式。僅在擷取足夠資訊時，才能描述檔案記錄的內容。不會擷取記錄格式資訊，如直欄標頭與別名。

- 根據記錄號碼或索引來依序存取 iSeries 檔中的記錄。
- 寫入記錄到 iSeries 檔。
- 根據記錄號碼或索引來依序更新 iSeries 檔中的記錄。
- 根據記錄號碼或索引來依序刪除 iSeries 檔中的記錄。
- 針對不同存取類型來鎖定 iSeries 檔。
- 使用確定控制來讓 Java 程式執行下列動作：
  - 啟動此連接的確定控制。
  - 針對不同檔案來指定不同確定控制鎖定層次。
  - 確定並回轉異動。
- 刪除 iSeries 檔。
- 從 iSeries 檔刪除成員。

備註： 記錄層次存取類別不支援邏輯結合檔或空字元索引欄位。

下列類別執行這些功能：

- AS400File
  類別是記錄層次存取類別的抽象基礎類別。它提供循序記錄存取、建立及刪除檔案與成員以及確定控制活動的方法。
- KeyedFile 類別代表按照索引存取的 iSeries 檔。
- SequentialFile 類別代表按照記錄號碼存取的 iSeries 檔。
- AS400FileRecordDescription 類別會提供擷取 iSeries 檔的記錄格式的方法。

記錄層次存取類別需要一個 AS400 物件 來代表有資料庫檔案的系統。使用記錄層次存取類別可使 AS400 物件連接 iSeries。請參閱 管理連線 取得關於管理連線的資訊。

記錄層次存取類別需要資料庫檔案的整合檔案系統路徑名稱。請參閱 整合檔案系統路徑名稱，取得詳細資訊。關資訊。

記錄層次存取類別會使用下列項目：

- 用來說明資料庫檔案記錄的 RecordFormat 類別
- 提供資料庫檔案記錄的存取權的 Record 類別
- LineDataRecordWriter 類別以行式資料格式寫入記錄

這些類別均會在資料轉換 區段中加以描述。

範例

- 循序存取範例說明如何循序存取 iSeries 檔。
- 讀取檔案範例 說明如何使用記錄層次存取類別，讀取 iSeries 檔。

- 索引檔範例 說明如何使用記錄層次存取類別，依索引從 iSeries 檔中讀取記錄。

AS400File

AS400File 類別會提供執行下列事項的方法：

- 建立與刪除伺服器實體檔案及成員
- 在伺服器檔案中讀取及寫入記錄
- 鎖定檔案，進行不同類型的存取
- 使用記錄區塊改善效能
- 在已開啟的伺服器檔案內設定游標位置
- 管理確定控制活動

KeyedFile

KeyedFile 類別給予 Java 程式在伺服器上索引存取檔案的權限。 索引存取表示 Java 程式可指定索引來存取檔案的記錄。按照索引來定位游標、讀取、更新以及刪除記錄的方法。

若要定位游標，請使用下列方法：

- positionCursor(Object[]) 將游標設定到具有指定之索引的第一筆記錄。
- positionCursorAfter(Object[]) 將游標設定到具有指定之索引的第一筆記錄之後。
- positionCursorBefore(Object[]) 將游標設定到具有指定之索引的第一筆記錄之前。

若要刪除記錄，請使用下面方法：

- deleteRecord(Object[]) 刪除具有指定之索引的的第一筆記錄。

讀取方法如下：

- read(Object[]) 讀取具有指定索引的第一筆記錄
- readAfter(Object[]) 讀取具有指定之索引的第一筆記錄之後的記錄。
- readBefore(Object[]) 讀取具有指定之索引的第一筆記錄之前的記錄。
- readNextEqual() 讀取索引符合指定之索引的下一筆記錄。 將從現行游標位置後的記錄開始搜尋。
- readPreviousEqual() 讀取索引符合指定之索引的前一筆記錄。
  將從現行游標位置前的記錄開始搜尋。

若要更新記錄，請使用下面方法：

- update(Object[]) 更新具有指定之索引的記錄。

按照索引來定位、讀取以及更新時，也提供方法來指定搜尋準則。有效搜尋標準值如下：

- Equal - 尋找索引符合指定之索引的第一筆記錄。
- Less than - 尋找最後一筆記錄，其索引在檔案之索引次序的指定索引之前。
- Less than or equal 尋找索引符合指定之索引的第一筆記錄。如果沒有記錄符合指定索引，
  請按照檔案的索引順序來尋找其索引位於指定索引前面的最後一筆記錄。
- Greater than 尋找第一筆記錄，其索引在檔案之索引次序的指定索引之後。
- Greater than or equal 尋找索引符合指定索引的第一筆記錄。如果沒有記錄符合指定索引，
  請按照檔案的索引順序來尋找其索引在指定索引後面的第一筆記錄。

KeyedFile 是 AS400File 的次類別；AS400File 中的全部方法皆可用於 KeyedFile。

指定索引

KeyedFile 物件的索引以「Java 物件」陣列表示，其類型與次序對應到索引欄位的類型與次序（由檔案的 RecordFormat 物件指定）。

下面範例說明如何指定 KeyedFile 物件的索引。

```
                    // Specify the key for a file whose key fields, in
order,
                    // are:
                    //    CUSTNAME    CHAR(10)
                    //    CUSTNUM     BINARY(9)
                    //    CUSTADDR    CHAR(100)VARLEN()
                    // Note that the last field is a variable-length
field.
     Object[] theKey = new Object[3];
     theKey[0] = "John Doe";
     theKey[1] = new Integer(445123);
     theKey[2] = "2227 John Doe Lane, ANYTOWN, NY 11199";
```

KeyedFile 物件接受部份索引以及完整索引。然而，必須依照順序指定索引欄位值。

例如：

```
                    // Specify a partial key for a file whose key fields,
                    // in order, are:
                    //    CUSTNAME    CHAR(10)
                    //    CUSTNUM     BINARY(9)
                    //    CUSTADDR    CHAR(100)VARLEN()
     Object[] partialKey = new Object[2];
     partialKey[0] = "John Doe";
     partialKey[1] = new Integer(445123);

                    // Example of an INVALID partial key
     Object[] INVALIDPartialKey = new Object[2];
     INVALIDPartialKey[0] = new Integer(445123);
     INVALIDPartialKey[1] = "2227 John Doe Lane, ANYTOWN, NY 11199";
```

不支援空字元索引和空字元索引欄位。

記錄的索引欄位值可透過getKeyFields()方法，從檔案的記錄物件取得。

下面範例說明如何按照索引從檔案讀取。

```
                    // Create an AS400 object, the file exists on this
          // server.
     AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that represents the file
     KeyedFile myFile = new KeyedFile(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                    // Assume that the AS400FileRecordDescription class
                    // was used to generate the code for a subclass of
                    // RecordFormat that represents the record format
```

```
                    // of file MYFILE in library MYLIB.  The code was
                    // compiled and is available for use by the Java
program.
     RecordFormat recordFormat = new MYKEYEDFILEFormat();

                    // Set the record format for myFile.  This must
                    // be done prior to invoking open()
     myFile.setRecordFormat(recordFormat);

                    // Open the file.
     myFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // The record format for the file contains
                    // four key fields, CUSTNUM, CUSTNAME, PARTNUM
                    // and ORDNUM in that order.
                    // The partialKey will contain 2 key field
                    // values.  Because the key field values must be
                    // in order, the partialKey will consist of values
for
                    // CUSTNUM and CUSTNAME.
     Object[] partialKey = new Object[2];
     partialKey[0] = new Integer(1);
     partialKey[1] = "John Doe";

                    // Read the first record matching partialKey
     Record keyedRecord = myFile.read(partialKey);

                    // If the record was not found, null is returned.
     if (keyedRecord != null)
     { // Found the record for John Doe, print out the info.
       System.out.println("Information for customer " +
(String)partialKey[1] + ":");
       System.out.println(keyedRecord);
     }

                  ....

                    // Close the file since I am done using it
     myFile.close();

                    // Disconnect since I am done using record-level
access

sys.disconnectService(AS400.RECORDACCESS);
```

SequentialFile

SequentialFile 類別給予 Java 程式依據記錄編號存取伺服器上檔案的權限。
按照記錄號碼來定位游標、讀取、更新以及刪除記錄的方法。

若要定位游標，請使用下列方法：

- positionCursor(int) 將游標設定到具有指定之記錄編號的記錄處。
- positionCursorAfter(int) 將游標設定到指定之記錄編號的記錄之後。
- positionCursorBefore(int) 將游標設定指定之記錄編號的記錄之前。

若要刪除記錄，請使用下面方法：

- deleteRecord(int) 刪除具有指定之記錄編號的記錄。

若要讀取記錄，請使用下列方法：

- read(int) 讀取具有指定之記錄編號的記錄。
- readAfter(int) 讀取指定之記錄編號之後的記錄。
- readBefore(int) 讀取指定之記錄編號之前的記錄。

若要更新記錄，請使用下面方法：

- update(int) 更新具有指定之記錄編號的記錄。

SequentialFile 是 AS400File 的次類別；AS400File 中的全部方法皆可用於 SequentialFile。

下面範例說明如何使用 SequentialFile 類別：

```
                        // Create an AS400 object, the file exists on this
                        // server.
     AS400 sys = new AS400("mySystem.myCompany.com");

                        // Create a file object that represents the file
     SequentialFile myFile = new SequentialFile(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/%FILE%.MBR");

                        // Assume that the AS400FileRecordDescription class
                        // was used to generate the code for a subclass of
                        // RecordFormat that represents the record format
                        // of file MYFILE in library MYLIB.  The code was
                        // compiled and is available for use by the Java
program.
     RecordFormat recordFormat = new MYFILEFormat();

                        // Set the record format for myFile.  This must
                        // be done prior to invoking open()
     myFile.setRecordFormat(recordFormat);
```

```
                    // Open the file.
    myFile.open(AS400File.READ_WRITE, 0, AS400File.COMMIT_LOCK_LEVEL_NONE);

                    // Delete record number 2.
    myFile.delete(2);

                    // Read record number 5 and update it
    Record updateRec = myFile.read(5);
    updateRec.setField("CUSTNAME", newName);

                    // Use the base class' update() method since I am
                    // already positioned on the record.
    myFile.update(updateRec);

                    // Update record number 7
    updateRec.setField("CUSTNAME", nextNewName);
    updateRec.setField("CUSTNUM", new Integer(7));
    myFile.update(7, updateRec);

                ....

                    // Close the file since I am done using it
    myFile.close();

                    // Disconnect since I am done using record-level
access

sys.disconnectService(AS400.RECORDACCESS);
```

AS400FileRecordDescription

AS400FileRecordDescription 類別提供方法來擷取伺服器上某檔案的記錄格式。 此類別提供方法，為 RecordFormat 的次類別建立 Java 原始程式碼以及傳回 RecordFormat 物件，這些物件說明伺服器上使用者指定的實體或邏輯檔案的記錄格式。 設定記錄格式時，這些方法的輸出可作為 AS400File 物件的輸入。

當此檔案已存在於伺服器時，建議一律使用 AS400FileRecordDescription 類別來產生 RecordFormat 物件。

備註： AS400FileRecordDescription 類別不會擷取檔案的整個記錄格式。僅擷取足夠資訊來描述構成檔案的記錄的內容。不會擷取如直欄標頭、別名及參照欄位等資訊。因此，所擷取的記錄格式不足以用來建立一個檔案，因為其記錄格式同於從其中擷取格式的檔案。

為 RecordFormat 次類別建立 Java 原始程式碼以代表伺服器上的檔案記錄格式

createRecordFormatSource() 方法為 RecordFormat 類別的次類別建立 Java 原始檔。應用程式或 applet 可以編譯和使用這些檔案作為 AS400File.setRecordFormat() 方法的輸入。

createRecordFormatSource() 方法應該作為一種開發時間工具使用，來擷取伺服器上現有的檔案的記錄格式。此方法可讓 RecordFormat 類別的次類別的原始程式建立一次，必要時加以修改和編譯，然後供許多 Java 程式使用，這些 Java 程式存取伺服器上的相同檔案。 因為本方法在本端系統中建立檔案，所以只有 Java 應用程式能夠使用本方法。 然而，您可編譯此輸出 (Java 原始程式碼)，然後同樣地由 Java 應用程式和 Applet 使用。

備註： 本方法會使用將建立的 JAVA 原始程式的名稱來改寫檔案名稱。

範例 1：下面範例說明如何使用 createRecordFormatSource() 方法：

```
                        // Create an AS400 object, the file exists on this
                        // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                        // Create an AS400FileRecordDescription object that
represents the file
    AS400FileRecordDescription myFile = new AS400FileRecordDescription(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");

                        // Create the Java source file in the current working
directory.
                        // Specify "package com.myCompany.myProduct;" for the
                        // package statement in the source since I will ship
the class
                        // as part of my product.
    myFile.createRecordFormatSource(null, "com.myCompany.myProduct");

                        // Assuming that the format name for file MYFILE is
FILE1, the
                        // file FILE1Format.java will be created in the
current working directory.
```

```
                    // It will overwrite any file by the same name.  The
name of the class
                    // will be FILE1Format.  The class will extend from
RecordFormat.
```

範例 2：編譯上述建立的檔案 FILE1Format.java，然後以下列方式使用此檔案：

```
                    // Create an AS400 object, the file exists on this
                    // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create an AS400File object that represents the
file
    SequentialFile myFile = new SequentialFile(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");

                    // Set the record format
                    // This assumes that
import.com.myCompany.myProduct.FILE1Format;
                    // has been done.

    myFile.setRecordFormat(new FILE1Format());

                    // Open the file and read from it
                    ....

                    // Close the file since I am done using it
    myFile.close();

                    // Disconnect since I am done using record-level
access
    sys.disconnectService(AS400.RECORDACCESS);
```

建立 RecordFormat 物件以代表伺服器上的檔案的記錄格式

retrieveRecordFormat() 方法傳回 RecordFormat 物件的陣列，這些物件代表伺服器上現有的檔案的記錄格式。一般而言，在陣列中只傳回一個 RecordFormat 物件。
當被擷取記錄格式的檔案是多重格式邏輯檔案時，會傳回一個以上 的 RecordFormat 物件。使用此方法，在執行時間動態擷取伺服器上現有的檔案的記錄格式。 然後可以使用 RecordFormat 物件作為 AS400File.setRecordFormat() 方法的輸入。

下面範例說明如何使用 retrieveRecordFormat() 方法：

```
                    // Create an AS400 object, the file exists on this
                    // server.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create an AS400FileRecordDescription object that
represents the file
    AS400FileRecordDescription myFile = new AS400FileRecordDescription(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");
```

```
                    // Retrieve the record format for the file
    RecordFormat[] format = myFile.retrieveRecordFormat();

                    // Create an AS400File object that represents the
file
    SequentialFile myFile = new SequentialFile(sys,
"/QSYS.LIB/MYLIB.LIB/MYFILE.FILE");

                    // Set the record format
    myFile.setRecordFormat(format[0]);

                    // Open the file and read from it
                    ....

                    // Close the file since I am done using it
    myFile.close();

                    // Disconnect since I am done using record-level
access
    sys.disconnectService(AS400.RECORDACCESS);
```

ServiceProgramCall 類別

ServiceProgramCall類別可讓您呼叫 iSeries 服務程式。ServiceProgramCall是ProgramCall類別的次類別，該類別可呼叫 iSeries 程式。如果您要呼叫 iSeries 程式，請使用 ProgramCall 類別。

ServiceProgramCall 類別可讓您呼叫 iSeries 服務程式、 經由輸入參數傳送資料到 iSeries 服務程式，並經由輸出參數存取 iSeries 服務程式傳回的資料。使用 ServiceProgramCall 會使 AS400 物件連接到 iSeries。請參閱管理連線，取得關於管理連線的資訊。

預設行為可以讓服務程式即使當 Java 程式與服務程式位於同一伺服器上時，也能以不同的伺服器工作執行。您可以置換預設行為，並使用繼承的（來自 ProgramCall）setThreadSafe() 方法，使服務程式在 Java 工作中執行。

使用 ServiceProgramCall 類別

要使用 ServiceProgramCall 類別，您必須確定遵循下列基本要求：

- 服務程式必須位於執行 OS/400 V4R4 或更新版本的 iSeries 或 AS/400 伺服器上
- 您最多只能傳送七個參數到服務程式
- 服務程式的回傳值為取消或是數字。

使用 ProgramParameter 物件

ProgramParameter類別使用 ServiceProgramCall 類別，以傳遞參數資料至 iSeries 服務程式，或自其中取得參數資料。輸入資料可以使用 setInputData()傳送到 iSeries 服務程式。

您可以使用setOutputDataLength()要求要傳回的輸出資料數量。使用getOutputData()可以在服務程式結束執行之後，擷取輸出資料。除了資料本身以外，ServiceProgramCall 也需要知道如何傳送參數資料到服務程式。ProgramParameter 的setParameterType()方法可用來提供這個資訊。該類型可以指出以值傳送參數或以參照傳送參數。在任何一種情形下，資料都是自從屬站傳送到伺服器。一旦資料到達 iSeries，伺服器便會使用參數類型來正確地呼叫服務程式。

所有參數都是在位元組陣列的套表中。因此，若要轉換 iSeries 和 Java 之間的格式，您可以使用資料轉換及說明類別。

SystemStatus 類別

SystemStatus類別可讓您擷取系統狀態資訊，以及擷取及變更系統儲存區資訊。SystemStatus 物件可讓您擷取包含於下列的系統狀態資訊：

- getUsersCurrentSignedOn()傳回目前登入系統的使用者數
- getUsersTemporarilySignedOff()傳回斷線的交談式作業數
- getDateAndTimeStatusGathered()傳回收集系統狀態資訊時的日期與時間
- getJobsInSystem()傳回目前正在執行的使用者與系統工作總數
- getBatchJobsRunning()傳回目前在系統上執行的批次作業數
- getBatchJobsEnding()傳回正在結束程序的批次作業數
- getSystemPools()為每個系統儲存區傳回包含 SystemPool 物件的細目

除了 SystemStatus 類別內的方法外，您也可以透過 SystemStatus 存取 SystemPool。SystemPool 可讓您取得系統儲存區的相關資訊及變更系統儲存區資訊。

範例

此範例顯示您如何以 SystemStatus 類別來使用快取：

```
AS400 system = new AS400("MyAS400");
SystemStatus status = new SystemStatus(system);

// Turn on caching. It is off by default.
status.setCaching(true);

// This will retrieve the value from the system.
// Every subsequent call will use the cached value
// instead of retrieving it from the system.
int jobs = status.getJobsInSystem();


// ... Perform other operations here ...


// This determines if caching is still enabled.
if (status.isCaching())
{
// This will retrieve the value from the cache.
jobs = status.getJobsInSystem();
}

// Go to the system next time, regardless if caching is enabled.
status.refreshCache();

// This will retrieve the value from the system.
jobs = status.getJobsInSystem();

// Turn off caching. Every subsequent call will go to the system.
status.setCaching(false);
```

```
// This will retrieve the value from the system.
jobs = status.getJobsInSystem();
```

系統值

系統值類別可讓 Java 程式擷取並變更系統值和網路屬性。您也可以定義您自己的群組以包含您要的系統值。

SystemValue 物件主要包含下列資訊：

- 名稱
- 說明
- 版次
- 值

利用 SystemValue 類別，使用getValue()方法擷取單一系統值，並使用setValue()方式變更系統值。

您也可以擷取關於特定系統值的群組資訊：

- 若要擷取系統值所屬的系統定義之群組，請使用getGroup()方法。
- 若要擷取 SystemValue 物件所屬的使用者定義之群組（如果有的話），請使用getGroupName() 以及getGroupDescription()方法。

每當第一次擷取系統值時，都會從 iSeries 擷取該值並儲存於快取記憶體中。
在後續的擷取中，將傳回快取的值。如果您要的是現有的 iSeries 值而非快取值，必須先執行clear()清除目前的快取。

系統值清單

SystemValueList代表指定的 iSeries 上的系統值清單。這個清單將分成數個系統定義的群組，可讓 Java 程式一次存取某一部份的系統值。

系統值群組

SystemValueGroup
代表使用者定義的系統值與網路屬性集合。與其說是儲存區，不如說它是產生及維護唯一系統值集合的工廠。

若要建立 SystemValueGroup，可以指定其中一個系統定義的群組（SystemValueList 類別中的其中一個常數）或指定一個系統值名稱陣列。

您可以使用add()方法，單獨新增系統值名稱於群組中。也可以使用remove() 方法來移除系統值名稱。

在以想要的系統值名稱植入 SystemValueGroup 後，可以呼叫getSystemValues()方法，從群組取得實際的 SystemValue 物件。 這樣的話，SystemValueGroup 物件會取得一組的系統值名稱並且產生一個 SystemValue 物件的「向量」，而它們都具有 SystemValueGroup 的系統、群組名稱以及群組說明。

若要同時重新整理所有 SystemValue 物件的「向量」，請使用refresh()方法。

使用 SystemValue 及 SystemValueList 類別的範例

下列範例將告訴您如何建立及擷取系統值：

```
//Create an AS400 object
 AS400 sys = new AS400("mySystem.myCompany.com");

//Create a system value representing the current second on the system.
SystemValue sysval = new SystemValue(sys, "QSECOND");


//Retrieve the value.
String second = (String)sysval.getValue();


//At this point QSECOND is cached. Clear the cache to retrieve the most
//up-to-date value from the system.
sysval.clear();
second = (String)sysval.getValue();

//Create a system value list.
SystemValueList list = new SystemValueList(sys);


//Retrieve all the of the date/time system values.
Vector vec = list.getGroup(SystemValueList.GROUP_DATTIM);


//Disconnect from the system.
sys.disconnectAllServices();
```

使用 SystemValueGroup 類別的範例

下列範例顯示如何建置一組系統值名稱以及維護它們的方法：

```
//Create an AS400 object
 AS400 sys = new AS400("mySystem.myCompany.com");

//Create a system value group initially representing all of the network
attributes on the system.
String name = "我的群組";
String description = "這是我的其中一個系統值。";
SystemValueGroup svGroup = new SystemValueGroup(sys, name, description,
SystemValueList.GROUP_NET);

//Add some more system value names to the group and remove some we do not
want.
svGroup.add("QDATE");
svGroup.add("QTIME");
svGroup.remove("NETSERVER");
svGroup.remove("SYSNAME");

//Obtain the actual SystemValue objects. They are returned inside a Vector.
Vector sysvals = svGroup.getSystemValues();
```

```
//You will notice that this is one of my system values.
SystemValue mySystemValue = (SystemValue)sysvals.elementAt(0);
System.out.println(mySystemValue.getName()+" -
"+mySystemValue.getGroupDescription());

//We can add another SystemValue object from another system into the group.
AS400 sys2 = new AS400("otherSystem.myCompany.com");
SystemValue sv = new SystemValue(sys2, "QDATE");
sysvals.addElement(sv);

//Now refresh the entire group of system values all at once.
//It does not matter if some system values are from different iSeries
systems.
//It does not matter if some system values were generated using
SystemValueGroup and some were not.
SystemValueGroup.refresh(sysvals);

//Disconnect from the systems.
sys.disconnectAllServices();
sys2.disconnectAllServices();
```

Trace 類別

Trace 物件可讓 Java 程式記載追蹤點與診斷訊息。這個資訊可協助重新產生與診斷問題。

註： 您也可以利用追蹤系統內容來設定追蹤。

Trace 類別會記載下列種類的資訊：

資訊種類	說明
轉換	記載 Unicode 與字碼頁之間的字集轉換。此種類應僅為 IBM Toolbox for Java 類別所使用。
資料串流	記載 iSeries 與 Java 程式之間流動的資料。此種類應僅為 IBM Toolbox for Java 類別所使用。
診斷	記載狀態資訊。
錯誤	記載其它引起異常情況的錯誤。
參考	追蹤程式的流程。
≫PCML	此種類是用來決定 PCML 如何解譯在伺服器之間往返傳送的資料。≪
Proxy	此種類為 IBM Toolbox for Java 類別用來記錄從屬站與 PROXY 伺服器間的資料流。
警告	記載程式可從其中恢復的錯誤的資訊。
全部	此種類可用來同時啟動或關閉以上所有種類的追蹤。 伴讀可直接將追蹤資訊直接記入到種類中。

IBM Toolbox for Java 類別也會使用追蹤種類。當 Java 程式啟用記載功能 時，IBM Toolbox for Java 資訊會包含在應用程式所記錄的資訊中

您可以對單一種類或一組種類啟用追蹤功能 。一旦選取了種類，請使用 setTraceOn 方法， 來開啟及關閉追蹤。並使用log 方法將資料寫入日誌中。

您可以將不同元件的追蹤資料傳送到個別的日誌。在預設的情況下，追蹤資料會寫入預設日誌中。請使用元件追蹤，將特定應用程式追蹤資料寫入個別的日誌或標準輸出。藉由使用元件追蹤，您可以輕鬆地將特定應用程式的追蹤資料與其它資料分開。

過多的記載可能會影響效能。使用isTraceOn 方法，來查詢追蹤的現行狀態。您的 Java 程式可以使用這種方法， 來決定在呼叫 log 方法之前，是否應該先建立追蹤記錄。在關閉記載功能時呼叫 log 方法不是一種錯誤，但會花費較長的時間。

預設值為將日誌資訊寫入至標準輸出中。 若要將日誌重新導向至檔案，請從 Java 應用程式中呼叫setFileName()方法。一般說來，這僅對 Java 應用程式有作用， 因為大多數瀏覽器並不會給與小型程式本端檔案系統的寫入權。

就預設值而言，記載功能是關閉的。Java 程式應該提供一種方法給使用者， 來啟動記載功能，以便能夠輕鬆地啟用記載功能。例如，應用程式可以剖析命令行參數，來指出哪一類的資料應被記載。 當需要日誌資訊時，使用者可以設定這個參數。

下列範例說明如何使用 Trace 類別。

範例 1： 下面範例將示範如何使用 setTraceOn 方法，以及如何透過 Log 方法，將資料寫入日誌中。

```
// Enable diagnostic, information, and warning logging.
Trace.setTraceDiagnosticOn(true);
Trace.setTraceInformationOn(true);
Trace.setTraceWarningOn(true);

// Turn tracing on.
Trace.setTraceOn(true);

// ... At this point in the Java program,
Trace.log(Trace.INFORMATION, "Just entered class xxx, method xxx");

// Turning tracing off.
Trace.setTraceOn(false);
```

範例 2： 下列範例顯示如何使用追蹤。方法 2 是較常用來撰寫會使用 trace 的程式碼的方法。

```
 // Method 1 - build a trace record
 // then call the log method and let the trace class determine if the
 // data should be logged. This will work but will be slower than the
 // following code.
 String traceData = new String("Just entered class xxx, data = ");
traceData = traceData + data + "state = " + state;
Trace.log(Trace.INFORMATION, traceData);

 // Method 2 - check the log status before building the information to
 // log. This is faster when tracing
 if (Trace.isTraceOn() && Trace.isTraceInformationOn())
     {
String traceData = new String("just entered class xxx, data = ");
traceData = traceData + data + "state = " + state;
Trace.log(Trace.INFORMATION, traceData);
     }
```

範例 3：下列範例顯示如何使用元件追蹤。

```
 // Create a component string. It is more efficient to create an
 // object than many String literals.
 String myComponent1 = "com.myCompany.xyzComponent";
 String myComponent2 = "com.myCompany.abcComponent";

 // Send Toolbox and the component trace data each to separate files.
 // The Toolbox trace will contain all trace information, while each
 // component log file will only contain trace information specific to
 // that component.If a Trace file is not specified, all trace data
 // will go to standard out with the component specified in front of
 // each trace message.

 // Trace.setFileName("c:\\bit.bucket");
 // Trace.setFileName(myComponent1, "c:\\Component1.log");
 // Trace.setFileName(myComponent2, "c:\\Component2.log");

 Trace.setTraceOn(true); // Turn trace on.
 Trace.setTraceInformationOn(true);// Enable information messages.

 // Log component specific trace data or general toolbox
 // trace data.

 Trace.setFileName("c:\\bit.bucket");
 Trace.setFileName(myComponent1, "c:\\Component1.log");
```

如果您不指定任何追蹤檔案，範例的結果如下：

```
 Toolbox for Java - Version 5 Release 1 Modification level 0
 [com.myCompany.xyzComponent] Tue Oct 24 16:02:44 CDT 2000 I am here
 [com.myCompany.abcComponent] Tue Oct 24 16:02:44 CDT 2000 I am there
 Tue Oct 24 16:02:44 CDT 2000 I am everywhere
```

使用者和群組

使用者與群組類別容許您透過 Java 程式，取得 iSeries 系統上的使用者與使用者群組清單，以及每一位使用者的相關資訊。

註：Toolbox for Java 也提供資源類別，它們呈現一種同屬組織架構和一致的程式設計介面，可以用來處理各種 iSeries 物件和清單。閱讀過 Access 套件和 resource 套件 中的類別相關資訊後，您可以選擇最適合您的應用程式使用的物件。 用來處理與使用者相關的 resource 類別包括 RUser 和 RUserList。

您可以擷取某些使用者資訊，包括前次登入日期、狀態、上次變更密碼的日期、密碼到期日及使用者類別。當您存取 User 物件時，您應該使用 setSystem() 方法來設定系統名稱，並使用 setName() 方法來設定使用者名稱。執行那些步驟之後，您可以使用 loadUserInformation() 方法，從 iSeries 取得資訊。

UserGroup 物件代表一個特殊使用者，其使用者設定檔是群組設定檔。藉由使用 getMembers() 方法，即可傳回該群組的成員使用者清單。

Java 程式可以使用列舉方式重複傳回清單。所有列舉的元素皆為 Java 物件；例如：

```
// Create an AS400 object.
AS400 system = new AS400 ("mySystem.myCompany.com");

// Create the UserList object.
UserList userList = new UserList (system);

// Get the list of all users and groups.
Enumeration enum = userList.getUsers ();

// Iterate through the list.
 while
(enum.hasMoreElements ())
    {
User u = (User) enum.nextElement ();
System.out.println(u);
    }
```

擷取使用者和群組的相關資訊

您可以使用 UserList 來取得下列各項的清單：

- 全部使用者和群組
- 僅群組
- 為群組成員的所有使用者
- 非群組成員的所有使用者

UserList 物件中必須設定的唯一內容即是 AS400 物件，它代表將從其中擷取使用者清單的系統。

依據預設值，將傳回所有使用者。合併使用 setUserInfo() 和 setGroupInfo()，可以明確指定必須傳回的使用者。

範例： 使用 UserList 列出給定群組中的所有使用者。

UserSpace 類別

UserSpace 類別代表伺服器上的使用者空間。 必要的參數為使用者空間的名稱，和代表擁有使用者空間的伺服器之 AS400 物件。 存在於使用者空間中的方法將執行下列：

- 建立使用者空間
- 刪除使用者空間
- 從使用者空間讀取資料
- 寫入使用者空間。
- 取得使用者空間的屬性。Java 程式可以取得使用者空間的 起始值、長度值和 自動延伸屬性。
- 設定使用者空間的屬性。Java 程式可以設定使用者空間的 起始值、長度值和 自動延伸屬性。

UserSpace 物件需要程式的整合檔案系統路徑名稱。請參閱整合檔案系統路徑名稱，取得詳細資訊。

使用 UserSpace 類別會使得 AS400 物件連接到伺服器。請參閱連線，取得有關管理連線的資訊。

下面範例會建立一個使用者空間，然後將資料寫入其中。

```
 // Create an AS400 object.
 AS400 sys = new AS400("mySystem.myCompany.com");

 // Create a user space object.
 UserSpace US
= new UserSpace(sys,
 "/QSYS.LIB/MYLIB.LIB/MYSPACE.USRSPC");

 // Use the create method to create the user space on
 // the server.
 US.create(10240, // The initial size is 10K
 true,// Replace if the user space already exists
 " ", // No extended attribute
 (byte) 0x00, // The initial value is a null
 "由 Java 程式建立",// The description of the user space
 "*USE"); // Public has use authority to the user space


 // Use the write method to write bytes to the user space.
 US.write("寫此字串寫入使用者空間。", 0);
```

HTML 類別

IBM Toolbox for Java HTML 類別可協助您：

- 設定 HTML 頁的套表及表格
- 對齊文字
- 使用各種 HTML 標示
- 改變語言及文字方向
- 建立排序及未排序的清單
- 建立檔案清單及 HTML 階層樹 (以及在其中的元素)
- 新增尚未定義在 HTML 類別中的標示屬性 (如 bgcolor 及 style 屬性)

HTML 類別會執行 HTMLTagElement 介面。每一個類別會產生特定元素類型的 HTML 標示。 使用 getTag() 方法可以擷取標示， 然後將它們嵌入任何 HTML 文件中。您以 HTML 類別所產生的標示，與 HTML 3.2 規格一致。

HTML 類別可以使用 Servlet 類別，自 iSeries 伺服器取出資料。然而，如果您提供表格或套表資料，則可以單獨使用該類別。

HTML 類別可以讓製作 HTML 套表、表格及其它元素的工作容易些：

- BidiOrdering 類別可讓您改變語言及文字方向。
- DirFilter 類別可讓您確定檔案物件是否為目錄。
- HTMLAlign 類別可讓您對齊 HTML 輸出的區塊。
- HTMLFileFilter 類別可讓您確定檔案物件是否為檔案。
- HTMLForm 套表類別 可協助您比使用「通用閘道介面 SCRIPT」更輕易地製作套表。
- HTMLHeading 類別可讓您建立 HTML 頁的標頭標示。
- HTMLHyperlink 類別可協助您在 HTML 頁中建立鏈結。
- HTMLImage 類別可讓您建立 HTML 頁的影像標示。
- HTMLList 類別 可協助您建立 HTML 頁的清單。
- HTMLMeta 類別可讓您建立 HTML 頁的 meta 標示。
- HTMLParameter 類別指定可供 HTMLServlet 使用的參數。
- HTMLServlet 類別可讓您建立一個伺服器端的併入項目。
- HTMLTable 類別 可協助您製作 HTML 頁的表格。
- HTMLText 類別可讓您在您的 HTML 頁中存取字型內容。
- HTMLTree 類別 可讓您顯示 HTML 元素的 HTML 階層樹。
- URLEncoder 類別可將要在 URL 字串中使用的定界符號編碼。
- URLParser 類別可讓您為 URI、內容及參照剖析 URL 字串。

備註 ：jt400Servlet.jar 檔包含 HTMLServlet 及 Servlet 兩種類別。如果您想要使用 在 com.ibm.as400.util.html 資料包中的類別，則必須更 新 CLASSPATH 並指向 jt400Servlet.jar 檔。

BidiOrdering 類別

BidiOrdering類別代表一個可改變語言及文字方向的 HTML 標示。HTML <BDO> 字串需要兩個屬性，一個用於語言，另一個用於文字的方向。

BidiOrdering 類別可讓您：

- 取得及設定語言屬性
- 取得及設定文字的方向

如需使用 <BDO> HTML 標示的詳細資訊，請參閱 W3C 網站。

範例：使用 BidiOrdering

下列範例會建立一個 BidiOrdering 物件並設定它的語言及方向：

```
// Create a BidiOrdering object and set the language and direction.
BidiOrdering bdo = new BidiOrdering();
bdo.setDirection(HTMLConstants.RTL);
bdo.setLanguage("AR");


// Create some text.
HTMLText text = new HTMLText("Some Arabic Text.");
    text.setBold(true);

// Add the text to the BidiOrdering and get the HTML tag.
bdo.addItem(text);
bdo.getTag();
```

列印陳述式會產生下列標示：

```
<bdo lang="AR" dir="rtl">
  <b>Some Arabic Text.</b>
</bdo>
```

當您在 HTML 網頁中使用此標示時，能瞭解 <BDO> 標示的瀏覽器會將範例顯示成這個樣子：

.txeT cibarA emoS

HTMLAlign 類別

HTMLAlign 類別可讓您對齊 HTML 文件的區段，而非僅對齊個別的項目，例如段落或標頭。

HTMLAlign 類別代表 <DIV> 標示及其相關的對齊屬性。 您可以使用向右、向左、或置中對齊。

您可以使用此類別來執行各種動作，包括：

- 新增或從您要對齊標示的清單移除項目
- 取得及設定對齊
- 取得及設定文字解譯的方向
- 取得及設定輸入元素的語言
- 取得 HTMLAlign 物件的字串表示

範例：建立 HTMLAlign 物件

下列範例會建立一個未排序的清單，然後建立一個 HTMLAlign 物件來對齊整份清單：

```
// Create an unordered list.
UnorderedList uList = new UnorderedList();
uList.setType(HTMLConstants.DISC);
UnorderedListItem uListItem1 = new UnorderedListItem();
uListItem1.setItemData(new HTMLText("Centered unordered list"));
uList.addListItem(uListItem1);
UnorderedListItem uListItem2 = new UnorderedListItem();
uListItem2.setItemData(new HTMLText("Another item"));
uList.addListItem(uListItem2);

// Align the list.
HTMLAlign align = new HTMLAlign(uList, HTMLConstants.CENTER);
System.out.println(align);
```

前面的範例會產生下列標示：

```
<div align="center">
<ul type="disc">
  <li>Centered unordered list</li>
  <li>Another item</li>
</ul>
```

當您在 HTML 網頁中使用此標示時，它看起來像：

- 置中對齊的未排序清單
- 其它項目

HTML 套表類別

HTMLForm 類別代表一個 HTML 套表。此類別可讓您：

- 新增元素，如按鈕、超鏈結或 HTML 表到套表中
- 從套表中移除元素
- 設定其它的套表屬性，如使用哪一種方法來傳送套表內容到伺服器、隱藏 式參數清單或動作 URL 位址

HTMLForm 物件的建構元會採用 URL 位址。 此位址會被參照為動作 URL。它是伺服器中，處理套表輸入的應用程式位置。 您可以在建構元中指定動作 URL，或使用setURL() 方法來設定位址。使用不同的設定方法可以設定套表屬性，或使用不同的取得方法擷取它們。

使用addElement()可以將任何 HTML 標示元素新增到 HTMLForm 物件，或使用 removeElement() 將它移除。在您的 HTMLForms 中使用下列 HTML 標示元素類別：

- FormInput 類別：代表 HTML 套表的輸入元素
- LayoutFormPanel 類別：代表 HTML 套表的套表元素佈置
- TextAreaFormElement：代表在 HTML 套表中的本文區域元素
- LabelFormElement：代表 HTML 套表元素的 標示
- SelectFormElement：代表 HTML 套表的選項 輸入類型
- SelectOption：代表 HTML 套表中 SelectFormElement 物件的選項
- RadioFormInputGroup：代表圓鈕輸入物件 的群組，可讓使用者從群組中選取一個選項

當然，您也可以新增其它標示元素到套表，包括下列元素：

- HTMLText
- HTMLHyperlink
- HTMLTable

若需更多有關使用 HTMLForm 類別建立套表的資訊，請參閱本 範例 及結果輸出。

FormInput 類別

FormInput 類別可讓您：

- 取得及設定輸入元素的名稱
- 取得及設定輸入元素的大小
- 取得及設定輸入元素的起始值

FormInput 類別是由下列清單中的類別加以擴充的。這些類別提供了建立特定類型的套表輸入元素的方法，並可讓您取得及設定輸入元素的各種屬性，或擷取 輸入元素的 HTML 標示：

- ButtonFormInput：代表 HTML 套表的按鈕元素
- FileFormInput：代表 HTML 套表的檔案輸入類型
- HiddenFormInput：代表 HTML 套表的隱藏式輸入類型
- ImageFormInput：代表 HTML 套表的影像輸入類型
- ResetFormInput：代表 HTML 套表的重設鈕輸入
- SubmitFormInput：代表 HTML 套表的提出按鈕輸入
- TextFormInput：代表 HTML 套表的單一行文字輸入，您可在其中定義每一行的最大字元數。對於密碼輸入類型，則請您使用 PasswordFormInput，它會擴充 TextFormInput，並代表 HTML 套表的密碼輸入類型
- ToggleFormInput：代表 HTML 套表的輪換輸入類型。使用者可以設定或取出文字標示，並指定是否要勾選或選取輪換。輪換輸入類型可以是兩者之一：
  - RadioFormInput：代表 HTML 套表的圓鈕輸入類型。 圓鈕可以放在具有 RadioFormInputGroup 類別的群組中；這會建立一個圓鈕群組，使用者僅可在其中選取 其所呈現的選項之一。
  - CheckboxFormInput：代表 HTML 套表的勾選框輸入類型，使用者可在其中選取其所呈現的一個以上的選項，且其中勾選框的起始設定可以是已勾選或未勾選。

ButtonFormInput 類別

ButtonFormInput類別代表 HTML 套表的按鈕元素。

下列範例會告訴您如何建立 ButtonFormInput 物件：

```
    ButtonFormInput button = new ButtonFormInput("button1", "Press Me",
"test()");
System.out.println(button.getTag());
```

此範例會產生下列標籤：

```
    <input type="button" name="button1" value="Press Me" onclick="test()" />
```

FileFormInput 類別

FileFormInput類別代表 HTML 套表中的檔案輸入類型。

下列程式碼範例會告訴您如何建立新的 FileFormInput 物件

```
FileFormInput file = new FileFormInput("myFile");
System.out.println(file.getTag());
```

上述程式會建立下列輸出：

```
<input type="file" name="myFile" />
```

HiddenFormInput 類別

HiddenFormInput 類別代表 HTML 套表中的隱藏式輸入類型。

下列程式範例會告訴您如何建立 HiddenFormInput 物件：

```
    HiddenFormInput hidden = new HiddenFormInput("account", "123456");
    System.out.println(hidden.getTag()):
```

上述程式會產生下列標示：

```
<input type="hidden" name="account" value="123456" />
```

在 HTML 網頁中，HiddenInputType 不會顯示。它只會將資訊（在此例中為帳戶號碼）傳送回伺服器。

ImageFormInput 類別

ImageFormInput 類別代表 HTML 套表中的影像輸入類型。

您可以使用提供的方法，擷取及更新 ImageFormInput 類別的許多屬性。

- 取得或設 定來源
- 取得或設 定對齊
- 取得或設 定高度
- 取得或設 定寬度

下面程式範例會告訴您如何建立 ImageFormInput 物件：

```
    ImageFormInput image = new ImageFormInput("myPicture", "myPicture.gif");
    image.setAlignment(HTMLConstants.TOP);
    image.setHeight(81);
    image.setWidth(100);
```

上面的程式範例會產生下列標示：

```
    <input type="image" name="MyPicture" src="myPicture.gif" align="top"
height="81" width="100" />
```

ResetFormInput 類別

ResetFormInput類別代表 HTML 套表中的重設鈕輸入類型。

下列程式碼範例會告訴您如何建立 ResetFormInput 物件：

```
    ResetFormInput reset = new ResetFormInput();
reset.setValue("重設");
System.out.println(reset.getTag());
```

上述的程式碼範例會產生下列 HTML 標記：

```
    <input type="重設" value="重設" />
```

SubmitFormInput 類別

SubmitFormInput 類別以 HTML 形式，表示提出按鈕輸入類型。

下列程式碼範例顯示如何建立 SubmitFormInput 物件：

```
SubmitFormInput submit = new SubmitFormInput();
submit.setValue("Send");
System.out.println(submit.getTag());
```

以上的程式碼範例會產生下列輸出：

```
    <input type="submit" value="Send" />
```

TextFormInput 類別

TextFormInput 類別代表 HTML 套表中的單行文字輸入類型。TextFormInput 提供的方法可讓您取得和 設定使用者能在文字欄位中輸入的字元數上限。

下列範例會顯示建立新 TextFormInput 物件的方法：

```
    TextFormInput text = new TextFormInput("userID");
text.setSize(40);
System.out.println(text.getTag());
```

以上的程式碼範例會產生下列標示語言：

```
    <input type="text" name="userID" size="40" />
```

PasswordFormInput 類別

PasswordFormInput類別代表 HTML 套表中的密碼輸入欄位類型。

下列程式碼範例會告訴您如何建立新的 PasswordFormInput 物件：

```
PasswordFormInput pwd = new PasswordFormInput("密碼");
pwd.setSize(12);
System.out.println(pwd.getTag());
```

上述的程式碼範例會產生下列標記：

```
<input type="密碼" name="密碼" size="12" />
```

RadioFormInput 類別

RadioFormInput 類別代表在 HTML 套表中的圓鈕輸入類型。 圓鈕可以在建構時選取並起始設定。

具有相同控制項名稱的一組圓鈕可組成圓鈕群組。RadioFormInputGroup
類別建立圓鈕群組。一次只能選取一個群組內的圓鈕。 在建構群組時，也可以選取並起始設定特定的按鈕。

下列程式碼範例會告訴您如何建立一個 RadioFormInput 物件：

```
    RadioFormInput radio = new RadioFormInput("年齡", "二十幾歲", "20 - 29
歲", true);
    System.out.println(radio.getTag());
```

上述的程式碼範例會產生下列標記：

```
    <input type="圓鈕" name="年齡" value="二十幾歲" checked="已選取" />
```

CheckboxFormInput 類別

CheckboxFormInput 類別代表 HTML 套表中的勾選框輸入類型。
使用者可以在套表中選取一個以上以勾選框方式呈現的選項。

下列範例會告訴您如何建立新的 CheckboxFormInput 物件：

```
    CheckboxFormInput checkbox = new CheckboxFormInput("uscitizen", "yes",
"textLabel", true);
System.out.println(checkbox.getTag());
```

上面的程式碼會產生下列輸出：

```
    <input type="checkbox" name="uscitizen" value="yes" checked="checked" />
textLabel
```

LayoutFormPanel 類別

LayoutFormPanel 類別代表 HTML 套表的套表元素佈置。 您可以使用 LayoutFormPanel 提供的方法，在畫面中新增或移除元素，或取 得佈置中的元素數。 您可以選擇使用兩種佈置之一：

- GridLayoutFormPanel: 代表 HTML 套表的套表元素格線佈置。
- LineLayoutFormPanel: 代表 HTML 套表的套表元素行佈置。

GridLayoutFormPanel

GridLayoutFormPanel類別代表套表元素的格線佈置。 您可以指定格線的直欄數，並對 HTML 套表使用此佈置。

下列範例會建立具有兩個直欄的 GridLayoutFormPanel 物件：

```
        // Create a text form input element for the system.
  LabelFormElement sysPrompt = new LabelFormElement("System:");
  TextFormInput system = new TextFormInput("System");

        // Create a text form input element for the userId.
  LabelFormElement userPrompt = new LabelFormElement("User:");
  TextFormInput user = new TextFormInput("User");

        // Create a password form input element for the password.
  LabelFormElement passwordPrompt = new LabelFormElement("Password:");
  PasswordFormInput password = new PasswordFormInput("Password");

        // Create the GridLayoutFormPanel object with two columns and add
the form elements.
  GridLayoutFormPanel panel = new GridLayoutFormPanel(2);
  panel.addElement(sysPrompt);
  panel.addElement(system);
  panel.addElement(userPrompt);
  panel.addElement(user);
  panel.addElement(passwordPrompt);
  panel.addElement(password);

        // Create the submit button to the form.
  SubmitFormInput logonButton = new SubmitFormInput("logon", "Logon");

        // Create HTMLForm object and add the panel to it.
  HTMLForm form = new HTMLForm(servletURI);
  form.addElement(panel);
  form.addElement(logonButton);
```

此範例會產生下列 HTML 程式碼：

```
<form action=servletURI method="get">
<table border="0">
<tr>
<td>System:</td>
<td><input type="text" name="System" /></td>
</tr>
<tr>
<td>User:</td>
<td><input type="text" name="User" /></td>
</tr>
```

```
<tr>
<td>Password:</td>
<td><input type="password" name="Password" /></td>
</tr>
</table>
<input type="submit" name="logon" value="Logon" />
</form>
```

LineLayoutFormPanel 類別

LineLayoutFormPanel類別 代表 HTML 套表的套表元素之行佈置。套表元素會在畫面中以單一列的方式排列。

此範例會建立 LineLayoutFormPanel 物件, 並新增兩個套表元素。

```
    CheckboxFormInput privacyCheckbox = new CheckboxFormInput("機密", "是",
"機密", true);
    CheckboxFormInput mailCheckbox = new CheckboxFormInput("郵寄清單", "是",
"加入郵寄清單", false);
  LineLayoutFormPanel panel = new LineLayoutFormPanel();
  panel.addElement(privacyCheckbox);
  panel.addElement(mailCheckbox);
  String tag = panel.getTag();
```

上面的程式範例會產生下列 HTML 程式碼：

```
    <input type="勾選框" name="機密" value="是"
    checked="已選取" /> Confidential <input type="勾選框"
    name="郵寄清單" value="是" /> Join our mailing list <br/>
```

TextAreaFormElement 類別

TextAreaFormElement 類別代表 HTML 套表中的一個本文區域元素。您可以透過設定列和 直欄 的數目，來決定本文區域的大小。您可以設定getRows() 和 getColumns()方法來決定本文區域元素的大小。

您可以使用setText()方法來設定本文區域中的起始文字。您可以使用getText()方法來查看已設定的起始文字。

下列範例會顯示建立新 TextAreaFormElement 的方法：

```
    TextAreaFormElement textArea = new TextAreaFormElement("foo", 3, 40);
textArea.setText("預設 TEXTAREA 值位於此處");
System.out.println(textArea.getTag());
```

以上的程式碼範例會產生下列 HTML 程式碼：

```
    <form>
<textarea name="foo" rows="3" cols="40">
預設 TEXTAREA 值位於此處
</textarea>
</form>
```

LabelFormElement 類別

LabelFormElement 類別代表 HTML 套表元素的標籤。 您可使用 LabelFormElement 類別在 HTML 套表的元素上加上標籤，如 本文區域或密碼表單輸入。標籤是您使用 setLabel() 方法設定的一行文字。此文字不會回應使用者輸入，且有了 它，可以讓使用者更易於瞭解套表。

下面程式會告訴您如何建立 LabelFormElement 物件：

```
LabelFormElement label = new LabelFormElement("Account Balance");
System.out.println(label.getTag());
```

此範例會產生下列輸出：

```
Account Balance
```

SelectFormElement 類別

SelectFormElement類別代表 HTML 套表的選取輸入型態。 您可以在選取的元素中增及移 除多種選項。

SelectFormElement 提供一些方法，可讓您用來檢視和變更選取元素的屬性：

- 使用setMultiple()可以設定使用者是否能選取一個以上的選項
- 使用getOptionCount()可以決定選項佈置中有幾項元素
- 使用setSize()可以設定選取元素內可見的選項數目，而使用getSize()可以決定可見的選項數目。

下列範例會建立一個具有三個選項的 SelectFormElement 物件。名為list1的 SelectFormElement 物件會以高亮度顯示。新增的前兩個選項可以指定選項文字、名稱以及選取屬性。 新增的第三個選項是由SelectOption物件定義。

```
    SelectFormElement list = new SelectFormElement("list1");
SelectOption option1 =     list.addOption("Option1", "opt1")      ;
SelectOption option2 =     list.addOption("Option2", "opt2", false)       ;
SelectOption option3 = new SelectOption("Option3", "opt3", true);
list.addOption(option3);
System.out.println(    list.getTag()   );
```

上述的程式碼範例會產生下列 HTML 程式碼：

```
    <select name="list1">
<option value="opt1">Option1</option>
<option value="opt2">Option2</option>
<option value="opt3" selected="selected">Option3</option>
</select>
```

SelectOption 類別

SelectOption類別代表 HTML 選項套表元素中的選項。您可以選取套表中使用選項套表元素。

您可以使用所提供的方法在 SelectOption 中擷取並設定屬性。例如，您可以設定選項預設值是否選取。
您也可以設定在提出套表時，它所要使用的輸入值。

下列範例會在一個選取套表中建立三個 SelectOption 物件。 下列的每一個 SelectOption
物件都是以高亮度顯示。他們的名稱是 option1、option2以及option3。option3物件會在起始時設定成已選取。

```
    SelectFormElement list = new SelectFormElement("list1");
SelectOption option1     = list.addOption("Option1", "opt1");
SelectOption option2     = list.addOption("Option2", "opt2", false);
SelectOption option3 = new SelectOption("Option3", "opt3", true);
list.addOption(   option3 );
System.out.println(list.getTag());
```

上述的程式碼範例會產生下列 HTML 標示：

```
    <select name="list1">
<option value="opt1">Option1</option>
<option value="opt2">Option2</option>
<option value="opt3" selected="selected">Option3</option>
</select>
```

RadioFormInputGroup 類別

RadioFormInputGroup 類別代表一群組的RadioFormInput 物件。使用者只能從 RadioFormInputGroup 選取一個 RadioFormInput 物件。

RadioFormInputGroup 類別方法可讓您使用圓鈕群組的各種屬性。有了這些方法, 您可以：

- 新增 圓鈕
- 移除圓鈕
- 取得或設定 圓鈕群組的名稱

下面範例會建立圓鈕群組：

```
      // Create some radio buttons.
    RadioFormInput radio0 = new RadioFormInput("年齡", "小孩", "0-12",
true);
    RadioFormInput radio1 = new RadioFormInput("年齡", "青少年", "13-19",
false);
    RadioFormInput radio2 = new RadioFormInput("年齡", "二十幾歲", "20-29",
false);
    RadioFormInput radio3 = new RadioFormInput("年齡", "三十幾歲", "30-39",
false);
  // Create a radio button group and add the radio buttons.
    RadioFormInputGroup ageGroup = new RadioFormInputGroup("年齡");
  ageGroup.add(radio0);
  ageGroup.add(radio1);
  ageGroup.add(radio2);
  ageGroup.add(radio3);
  System.out.println(ageGroup.getTag());
```

上述的程式碼範例會產生下列 HTML 程式碼：

```
    <input type="radio" name="age" value="kid" checked="checked"
/> 0-12    <input type="radio" name="age" value="teen" /> 13-19
    <input type="radio" name="age" value="twentysomething" /> 20-29
    <input type="radio" name="age" value="thirtysomething" /> 30-39
```

HTMLHeading 類別

HTMLHeading 類別代表一個 HTML 標頭。每一個標頭可以有 自己的對齊方式及 1（最大字型、最重要）到 6 的層次。

HTMLHeading 類別的方法包括：

- 取得及設定標頭的文字
- 取得及設定標頭的層次
- 取得及設定標頭的對齊方式
- 取得及設定文字說明的使用方向
- 取得及設定輸入元素的語言
- 取得 HTMLHeader 物件的字串表示

範例：建立 HTMLHeading 物件

下列範例會建立三個 HTMLHeading 物件：

```
// Create and display three HTMLHeading objects.
HTMLHeading h1 = new HTMLHeading(1, "Heading", HTMLConstants.LEFT);
HTMLHeading h2 = new HTMLHeading(2, "Subheading",
HTMLConstants.CENTER);
HTMLHeading h3 = new HTMLHeading(3, "Item", HTMLConstants.RIGHT);
System.out.print(h1 + "\r\n" + h2 + "\r\n" + h3);
```

前面的範例會產生下列標示：

```
<h1 align="left">Heading</h1>
<h2 align="center">Subheading</h2>
<h3 align="right">Item</h3>
```

HTMLHyperlink 類別

HTMLHyperlink 類別代表 HTML 超鏈結標示。您可使用 HTMLHyperlink 類別，在您的 HTML 頁中建立鏈結。您可以使用此類別取得及 設定超鏈結的許多屬性，包括：

- 取得或設定鏈結的「一致資源 ID」
- 取得或設定鏈結的標題
- 取得或設定鏈結的目標框架

HTMLHyperlink 類別可以使用已定義的特性來列印完整的超鏈結， 所以您可以在您的 HTML 頁中使用輸出。

下面是 HTMLHyperlink 的範例：

```
// Create an HTML hyperlink to the IBM Toolbox for Java home page.
HTMLHyperlink toolbox = new
HTMLHyperlink("http://www.ibm.com/as400/toolbox", "IBM Toolbox for Java home
page");

// Display the toolbox link tag.
System.out.println(toolbox.toString());
```

上面的程式會產生下列標示：

```
<a href="http://www.ibm.com/as400/toolbox">IBM Toolbox for Java home page</a>
```

當您在 HTML 網頁中使用此標示時，它看起來像：

IBM Toolbox for Java 首頁

»HTMLImage 類別

HTMLImage 類別 可讓您建立 HTML 頁的影像標示。HTMLImage 類別提供幾種方法讓您取得及設定影像屬性, 包括：

- 取得或設定影像的高度
- 取得或設定影像的寬度
- 取得或設定影像的名稱
- 取得或設定影像的替代文字
- 取得或設定影像周圍的水平空間
- 取得或設定影像周圍的垂直空間
- 取得或設定影像的絕對或相對參照
- 擷取 HTMLImage 物件的「字串」表示

下列範例顯示一個建立 HTMLImage 物件的方法：

```
// Create an HTMLImage.
HTMLImage image = new HTMLImage("http://myWebSite/picture.gif",
                                "Alternate text for this graphic");
image.setHeight(94);
image.setWidth(105);
System.out.println(image);
```

列印陳述式會在單行上產生下列標示。覆蓋文字僅供顯示用。

```
<img src="http://myWebSite/picture.gif"
     alt="Alternate text for this graphic"
     height="94" width="105" />
```

«

HTMLList 類別

HTMLList 可讓您輕鬆地在您的 HTML 頁內建立清單。這些類別提供一些方法，用來取得及設定各種清單屬性和清單內的項目。

尤其是上層類別 HTMLList 提供一個產生壓縮清單的方法，這類清單會儘可能以較小的垂直空間顯示項目。

- HTMLList 的方法包括：
  - 壓縮 清單
  - 新增 及從 清單移除項目
  - 新增 及從 清單移除清單（使它儘可能成為巢狀清單）

- HTMLListItem 的方法包括：
  - 取得及設定 項目的內容
  - 取得及設定 文字解譯的方向
  - 取得及設定 輸入元素的語言

使用 HTMLList 及 HTMLListItem 的次類別來建立您的 HTML 清單：

- OrderedList 及 OrderedListItem
- UnorderedList 及 UnorderedListItem

有關編碼片段，請參閱下列範例：

- 範例： 建立排序的清單
- 範例： 建立未排序的清單
- 範例： 建立巢狀清單

OrderedList 及 OrderedListItem

使用OrderedList及OrderedListItem類別在您的 HTML 頁中建立排序的清單。

- OrderedList 的方法包括：
  - 取得及設定 清單中第一個項目的起始號碼
  - 取得及設定 項目號碼的類型（或樣式）

- OrderedListItem 的方法包括：
  - 取得及設定 項目的號碼
  - 取得及設定 項目號碼的類型（或樣式）

經由使用 OrderedListItem 中的方法，您可以置換清單中特定項目的編號及類型。

請參閱 建立排序的清單 範例。

UnorderedList 及 UnorderedListItem

使用UnorderedList及UnorderedListItem類別在您的 HTML 頁中建立未排序的清單。

- UnorderedList 的方法包括：
  - 取得及設定項目的類型（或樣式）

- UnorderedListItem 的方法包括：
  - 取得及設定項目的類型（或樣式）

請參閱 建立未排序的清單 範例。

範例

下列範例告訴您如何使用 HTMLList 類別來建立排序的清單、未排序的清單及巢狀清單。

範例：建立排序的清單

下列範例會建立一個排序的清單：

```
     // Create an OrderedList.
OrderedList oList = new OrderedList(HTMLConstants.SMALL_ROMAN);
     // Create the OrderedListItems.
OrderedListItem listItem1 = new OrderedListItem();
OrderedListItem listItem2 = new OrderedListItem();
     // Set the data in the OrderedListItems.
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
     // Add the list items to the OrderedList.
oList.addListItem(listItem1);
oList.addListItem(listItem2);
System.out.println(oList.getTag());
```

前面的範例會產生下列標示：

```
<ol type="i">
<li>First item</li>
<li>Second item</li>
</ol>
```

當您在 HTML 頁中使用這些標示時，它看起來像：

i.第一個項目
ii.第二個項目

範例：建立未排序的清單

下列範例會建立一個未排序的清單：

```
        // Create an UnorderedList.
    UnorderedList uList = new UnorderedList(HTMLConstants.SQUARE);
        // Create the UnorderedListItems.
    UnorderedListItem listItem1 = new UnorderedListItem();
    UnorderedListItem listItem2 = new UnorderedListItem();
        // Set the data in the UnorderedListItems.
    listItem1.setItemData(new HTMLText("First item"));
    listItem2.setItemData(new HTMLText("Second item"));
        // Add the list items to the UnorderedList.
    uList.addListItem(listItem1);
    uList.addListItem(listItem2);
    System.out.println(uList.getTag());
```

前面的範例會產生下列標示：

```
<ul type="square">
<li>First item</li>
<li>Second item</li>
</ul>
```

當您在 HTML 頁中使用這些標示時，它看起來像：

- 第一個項目
- 第二個項目

範例：建立巢狀清單

下列範例會建立一個巢狀清單：

```
        // Create an UnorderedList.
    UnorderedList uList = new UnorderedList(HTMLConstants.SQUARE);
        // Create and set the data for UnorderedListItems.
    UnorderedListItem listItem1 = new UnorderedListItem();
    UnorderedListItem listItem2 = new UnorderedListItem();
    listItem1.setItemData(new HTMLText("First item"));
    listItem2.setItemData(new HTMLText("Second item"));
        // Add the list items to the UnorderedList.
    uList.addListItem(listItem1);
    uList.addListItem(listItem2);

        // Create an OrderedList.
    OrderedList oList = new OrderedList(HTMLConstants.SMALL_ROMAN);
        // Create the OrderedListItems.
    OrderedListItem listItem1 = new OrderedListItem();
    OrderedListItem listItem2 = new OrderedListItem();
    OrderedListItem listItem3 = new OrderedListItem();
        // Set the data in the OrderedListItems.
```

```
listItem1.setItemData(new HTMLText("First item"));
listItem2.setItemData(new HTMLText("Second item"));
listItem3.setItemData(new HTMLText("Third item"));
     // Add the list items to the OrderedList.
oList.addListItem(listItem1);
oList.addListItem(listItem2);
     // Add (nest) the unordered list to OrderedListItem2
oList.addList(uList);
     // Add another OrderedListItem to the OrderedList
     // after the nested UnorderedList.
oList.addListItem(listItem3);
System.out.println(oList.getTag());
```

前面的範例會產生下列標示：

```
<ol type="i">
<li>First item</li>
<li>Second item</li>
<ul type="square">
<li>First item</li>
<li>Second item</li>
</ul>
<li>Third item</li>
</ol>
```

HTMLMeta 類別

HTMLMeta 類別代表在 HTMLHead 標示內使用的 meta 資訊。在 HTML 文件內識別、編索及定義資訊時會使用 META 標示中的屬性。

META 標示的屬性包括：

- NAME - 與 META 標示內容相關的名稱
- CONTENT - 與 NAME 屬性相關的值
- HTTP-EQUIV - 由 HTTP 伺服器針對回應訊息標頭收集的資訊
- LANG - 語言
- URL - 用來將使用者從現行頁重新導向另一個 URL

例如，若要協助搜尋引擎判定頁面的內容，您可以使用下列 META 標示：

```
<META name="keywords" lang="en-us" content="games, cards, bridge">
```

您也可以使用 HTMLMeta 將使用者從某頁重新導向另一頁。

HTMLMeta 類別的方法包括：

- 取得及設定 NAME 屬性
- 取得及設定 CONTENT 屬性
- 取得及設定 HTTP-EQUIV 屬性
- 取得及設定 LANG 屬性
- 取得及設定 URL 屬性

範例：建立 META 標示

下列範例會建立兩個 META 標示：

```
// Create a META tag to help search engines determine page content.
HTMLMeta meta1 = new HTMLMeta();
meta1.setName("keywords");
meta1.setLang("en-us");
meta1.setContent("games, cards, bridge");
// Create a META tag used by caches to determine when to refresh the
page.
HTMLMeta meta2 = new HTMLMeta("Expires", "Mon, 01 Jun 2000 12:00:00
GMT");
System.out.print(meta1 + "\r\n" + meta2);
```

前面的範例會產生下列標示：

```
<meta name="keywords" content="games, cards, bridge">
<meta http-equiv="Expires" content="Mon, 01 Jun 2000 12:00:00 GMT">
```

HTMLParameter 類別

HTMLParameter 類別代表可以與 HTMLServlet 類別一起使用的參數。每個參數都有它自己的名稱及值。

HTMLParameter 類別的方法包括：

- 取得並設 定參數名稱
- 取得並設 定參數值

範例： 建立 HTMLParameter 標籤

下面範例會建立一個 HTMLParameter 標籤：

```
// Create an HTMLServletParameter.
 HTMLParameter parm = new HTMLParameter ("年齡", "21");
 System.out.println(parm);
```

以上的範例會產生下列標記：

```
<param name="年齡" value="21">
```

HTMLServlet 類別

HTMLServlet 類別代表 一個伺服器端的併入項目。servlet 物件會指定 servlet 的名稱及位置 (後者為選擇性)。您也可以選擇使用本端系統上的預設位置。

HTMLServlet 類別與 HTMLParameter 類別一起使用， 後者會指定可供 servlet 使用的參數。

HTMLServlet 類別的方法包括：

- 新增及 從 servlet 標移除 HTMLParameters
- 取得及設 定 servlet 的位置
- 取得及設 定 servlet 的名稱
- 取得及設 定 servlet 的替代文字

範例：建立 HTMLServlet 標示

下列為 HTMLServlet 標示的範例：

```
          // Create an HTMLServlet.
     HTMLServlet servlet = new HTMLServlet("myServlet",
"http://server:port/dir");
          // Create a parameter, then add it to the servlet.
     HTMLParameter param = new HTMLParameter("parm1", "value1");
     servlet.addParameter(param);
          // Create and add second parameter
     HTMLParameter param2 = servlet.add("parm2", "value2");
          // Create the alternate text if the Web server does not support
the servlet tag.
     servlet.setText("The Web server providing this page does not support
the SERVLET tag.")
     System.out.println(servlet);
```

前面的範例會產生下列標示：

```
     <servlet name="myServlet" codebase="http://server:port/dir">
     <param name="parm1" value="value1">
     <param name="parm2" value="value2">
     The Web server providing this page does not support the SERVLET tag.
     </servlet>
```

HTML 表格類別

HTMLTable 類別可讓您輕易地設定您可使用於 HTML 頁中的表格。此類別提供一些方法以取得及設定表格的各種屬性，包括：

- 取得及設定邊框的寬度
- 取得表格中的列數
- 新增 欄 或列到表格的末端
- 移除特定欄或列位置上的 欄 或列

HTMLTable 類別會使用其它的 HTML 類別來讓製作表格的工作更容易些。 用來建立表格的其它 HTML 類別是：

- HTMLTableCell：建立表格的資料格
- HTMLTableRow：建立表格的列
- HTMLTableHeader：建立表格的表頭資料格
- HTMLTableCaption：建立表格標題

範例

範例： 使用 HTMLTable 類別。

HTMLTableCell 類別

HTMLTableCell 類別會將任何的HTMLTagElement 物件當作輸入，並以指定的元素建立表格資料格標籤。 您可以在建構元上設定元素，或者透過兩種 setElement()方法的其中之一來設定之。

您可使用 HTMLTableCell 類別中所提供的方法來擷取或更新許多資料格屬性。
有些動作您可以使用下列這些方法執行：

- 取得或設 定列距
- 取得或設 定資料格高度
- 設定資料格資料是否要使用一般 HTML 行分段慣例

下列範例會建立 HTMLTableCell 物件並顯示標示語言：

```
//Create an HTMLHyperlink object.
HTMLHyperlink link = new HTMLHyperlink("http://www.ibm.com",
"IBM 首頁");
HTMLTableCell cell = new HTMLTableCell(link);
cell.setHorizontalAlignment(HTMLConstants.CENTER);
System.out.println(cell.getTag());
```

上述 getTag()方法會提供範例輸出：

```
<td align="center"><a href="http://www.ibm.com">IBM 首頁</a></td>
```

HTMLTableRow 類別

HTMLTableRow 類別會在表格中建立一列。此類別提供許多取得和設定列屬性的方法。您可使用這些方法執行下列動作：

- 新增或移除列中的直欄
- 取得直欄資料（從指定的直欄索引）
- 取得直欄索引（具有指定資料格的直欄）。
- 取得列中的直欄數
- 設定水平和垂直對齊

以下是 HTMLTableRow 的範例：

```
// Create a row and set the alignment.
HTMLTableRow row = new HTMLTableRow();
row.setHorizontalAlignment(HTMLTableRow.CENTER);

// Create and add the column information to the row.
HTMLText account = new HTMLText(customers_[rowIndex].getAccount());
HTMLText name = new HTMLText(customers_[rowIndex].getName());
HTMLText balance = new HTMLText(customers_[rowIndex].getBalance());

row.addColumn(new HTMLTableCell(account));
row.addColumn(new HTMLTableCell(name));
row.addColumn(new HTMLTableCell(balance));

// Add the row to an HTMLTable object (assume that the table already
exists).
table.addRow(row);
```

HTMLTableHeader 類別

HTMLTableHeader 類別繼承了 HTMLTableCell 類別。它會建立特定的資料格類型，表頭資料料格，給您 <th> 資料格而不是 <td> 資料格。如同 HTMLTableCell 類別，你可呼叫多種方法以更新或擷取標頭資料格的屬性。

以下為 HTMLTableHeader 的範例：

```
// Create the table headers.
HTMLTableHeader account_header = new HTMLTableHeader(new HTMLText("帳戶"));
HTMLTableHeader name_header = new HTMLTableHeader(new HTMLText("名稱"));
HTMLTableHeader balance_header = new HTMLTableHeader();
HTMLText balance = new HTMLText("餘額");
balance_header.setElement(balance);

// Add the table headers to an HTMLTable object (assume that the table
already exists).
table.addColumnHeader(account_header);
table.addColumnHeader(name_header);
table.addColumnHeader(balance_header);
```

HTMLTableCaption 類別

HTMLTableCaption 類別會建立 HTML 表格的標題。此類別提供更新與擷取標題屬性的方法。例如，您可以使用setAlignment()方法來指定表格的某部份標題要對齊。以下是 HTMLTableCaption 的範例：

```
// Create a default HTMLTableCaption object and set the caption text.
HTMLTableCaption caption = new HTMLTableCaption();
caption.setElement("客戶帳戶餘額 - 2000 年 1 月 1 日");

// Add the table caption to an HTMLTable object (assume that the table
already exists).
table.setCaption(caption);
```

HTMLText 類別

HTMLText 類別可讓您存取 HTML 頁的文字內容。利用 HTMLText 類別，您可以取得、設定並檢查許多文字屬性的狀態，包括：

- 取得或設定字型的大小
- 設定粗體屬性開啟（true）或關閉（false），或確定它是否已開啟
- 設定底線屬性開啟（true）或關閉（false），或確定它是否已 開啟
- 取得或設定文字的水平對齊。

下面範例會告訴您如何建立 HTMLText 物件，以及如何啟用其粗體屬性，並將 其字型大小設定為 5。

```
HTMLText text = new HTMLText("IBM");
     text.setBold(true);
text.setSize(5);
System.out.println(text.getTag());
```

列印陳述式會產生下列標示：

```
<font size="5"><b>IBM</b></font>
```

當您在 HTML 網頁中使用此標示時，它看起來像：

IBM

HTMLTree 類別

HTMLTree 類別可讓您輕易地設定您可在 HTML 頁中所使用的 HTML 元件階層樹。
此類別除了提供可讓您執行下列動作的方法外，還提供一些方法以取得及設定樹的各種屬性：

- 取得及設定 HTTP Servlet 需求
- 對樹新增 HTMLTreeElement 或 FileTreeElement
- 從樹移除 HTMLTreeElement 或 FileTreeElement

HTMLTree 類別使用其它 HTML 類別，使建立階層樹變得更容易：

- HTMLTreeElement：建立樹元素
- FileTreeElement 建立檔案樹元素
- FileListElement 建立檔案清單元素
- FileListRenderer 提供檔案及目錄清單

範例

下列範例顯示使用 HTMLTree 類別的不同方式。

- 範例： 使用 HTMLTree 類別
- 範例： 建立可瀏覽的整合檔案系統樹

HTMLTreeElement 類別

HTMLTreeElement 類別代表 HTMLTree 或其它 HTMLTreeElements 中的階層性元素。

您可使用 HTMLTreeElement 類別中所提供的方法來擷取或更新許多樹元素屬性。
有些動作您可以使用下列這些方法執行：

- 取得或設 定樹狀元素的可見文字
- 取得或設 定已展開及隱藏的圖示之 URL
- 設定是否要展開樹狀元素

下列範例會建立一個 HTMLTreeElement 物件並顯示其標籤：

```
 // Create an HTMLTree.
 HTMLTree tree = new HTMLTree();

 // Create parent HTMLTreeElement.
 HTMLTreeElement parentElement = new HTMLTreeElement();
 parentElement.setTextUrl(new HTMLHyperlink("http://myWebPage",
"我的網頁"));


 // Create HTMLTreeElement Child.
 HTMLTreeElement childElement = new HTMLTreeElement();
 childElement.setTextUrl(new HTMLHyperlink("http://anotherWebPage",
"另一個網頁"));
 parentElement.addElement(childElement);

 // Add the tree element to the tree.
 tree.addElement(parentElement);
 System.out.println(tree.getTag());
```

上述範例中的 getTag()方法會產生類似如下的 HTML 標籤：

```
<table cellpadding="0" cellspacing="3">
<tr>
<td><font color="#0000FF"><u>-</u></font> </td>
<td><font color="#0000FF"><u>我的網頁</u></font></td>
</tr>

<tr>
<td> </td>
<td>
<table cellpadding="0" cellspacing="3">
<tr>
<td><font color="#0000FF"><u>-</u></font> </td>
<td><font color="#0000FF"><u>另一個網頁</u></font> </td>
</tr>
</table>
</td>
```

</tr>
</table>

FileTreeElement 類別

FileTreeElement類別代表 HTMLTree 檢視畫面內的整合檔案系統。

您可使用 HTMLTreeElement 類別中所提供的方法來擷取或更新許多樹元素屬性。您也可以 取得及設定 NetServer 共用磁碟機的名稱及路徑。

這些方法能讓您執行的一些動作有：

- 取得或設定展開與隱藏圖示的 URL (繼承的方法)
- 設定是否將展開樹元素 (繼承的方法)
- 取得或設定 NetServer 共用磁碟機的名稱
- 取得或設定 NetServer 共用磁碟機的路徑

下列範例會建立 FileTreeElement 物件並顯示標示：

```
        // Create an HTMLTree.
        HTMLTree tree = new HTMLTree();

        // Create a URLParser object.
        URLParser urlParser = new
URLParser(httpServletRequest.getRequestURI());

    // Create an AS400 object.
       AS400 system = new AS400(mySystem, myUserId, myPassword);

        // Create an IFSJavaFile object.
        IFSJavaFile root = new IFSJavaFile(system, "/QIBM");

        // Create a DirFilter object and get the directories.
        DirFilter filter = new DirFilter();
        File[] dirList = root.listFiles(filter);

        for (int i=0; i < dirList.length; i++)
        {

            // Create a FileTreeElement.
            FileTreeElement node = new FileTreeElement(dirList[i]);

            // Set the Icon URL.
            ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
            sl.setHttpServletResponse(resp);
            element.setIconUrl(sl);

            // Add the FileTreeElement to the tree.
            tree.addElement(element);
        }

        System.out.println(tree.getTag());
```

前述getTag()方法提供了範例的輸出。

FileListElement 類別

FileListElement類別 可讓您建立一個檔案清單元素，它代表整合檔案系統目錄的內容。

»您可以 藉由取得及設定 NetServer 共用磁碟機的名稱及路徑 使用 FileListElement 物件來代表 NetServer 共用磁碟機的內容。«

FileListElement 類別提供可讓您執行下列動作的方法：

- 列出及排序 檔案清單的元素
- 取得及設定 HTTP Servlet 需求
- 取得及設定 FileListRenderer
- 取得及設定要用來顯示檔案清單的 HTMLTable
- »取得或 設定 NetServer 共用磁碟機的名稱«
- »取得或 設定 NetServer 共用磁碟機的路徑«

您可以與 html 套件中的其它類別一起使用 FileListElement 類別：

- 使用FileListRenderer，您可以指定您要如何顯示檔案的清單
- 使用FileTreeElement類別，您可以建立整合檔案系統 檔案 »或 NetServer 共用檔案的可瀏覽清單«

FileListElement javadoc會告訴您如何建立及顯示 FileListElement 物件。

範例

下列範例告訴您如何與HTMLTree 類別 一起使用 FileListElement 類 別 (FileTreeElement 及 HTMLTreeElement) 來建立可瀏覽的整合檔案系統樹。»此範例也包含用來設定 NetServer 共用磁碟機路徑的程式碼。«

- 範例： 建立可瀏覽的整合檔案系統樹

FileListRenderer 類別

FileListRenderer 類別 會提供任何欄位給 FileListElement 中的「檔案」物件 (目錄及檔案)。

FileListRenderer 類別提供可讓您執行下列動作的方法：

- 取得目錄的名稱
- 取得檔案的名稱
- 取得上層目錄的名稱
- 傳回您要顯示在 FileListElement 中的列資料

此範例會使用提供器建立 FileListElement 物件：

```
// Create a FileListElement.
FileListElement fileList = new FileListElement(sys, httpservletrequest);

// Set the renderer specific to this servlet, which extends
// FileListRenderer and overrides applicable methods.
fileList.setRenderer(new myFileListRenderer(request));
```

如果您不想使用預設的提供器，您可以擴充 FileListRenderer 並置換新方法或建立
新方法。例如，您可能想要確定您防止了傳送特定目錄的名稱，或有特定副檔名的檔案到 FileListElement。
藉由擴充此類別並置換適當的方法，您可以傳回這些檔案及目錄的空值，以確定沒有將它們顯示出來。

若要在 FileListElement 內完全自訂列， 請使用 getRowData() 方法 有一個使用 getRowData()
來自訂列資料的範例是新增直欄到列資料或重排直欄。 當您滿意 FileListRenderer
的預設行為時，便不需要額外的程式設計， 因為 FileListElement 類別會建立一個預設的 FileListRenderer。

ReportWriter 類別

com.ibm.as400.util.reportwriter 套件所提供的類別可讓您於使用 iSeries 時, 能夠更容易地存取和格式化 XML 原始檔中的資料 或 servlet 或 JavaServer Pages(TM)產生的資料。reportwriter 套件可以很便利地為三種不同但相關的套件命名：

- com.ibm.as400.util.reportwriter.pclwriter
- com.ibm.as400.util.reportwriter.pdfwriter
- com.ibm.as400.util.reportwriter.processor

»這些套件中包含各種類別, 可供您格式化 XML 資料串流和產生這些格式的報告。請確定您的 CLASSPATH 中有必需的 jar 檔案存在。 有關 reportwriter jar 檔案之相關資訊, 請參閱檔案。«

Context 類別 (在 pclwriter 和 pdfwriter 套件中) 定義 出 ReportProcessor 類別以選定的格式轉譯 XML 和 JSP 資料時所需的方法：

- 請使用 PCLContext 搭配上 ReportWriter 類別來產生 Hewlett Packard Printer Control Language (PCL) 格式的報告。
- 請使用 PDFContext 搭配上 ReportWriter 類別來產生 Adobe Portable Document Format (PDF) 格式的報告。

ReportProcessor 類別 (在 processor 套件中) 可使您由應用程式 自 XML 來源資料、Java servlet 和 JavaServer Pages (JSP) 所收集的 資訊中產生格式化的報告。

- 請使用 JSPReportProcessor 類別, 從 servlet 和 JSP 網頁擷取資料, 產生可用格式 (環境) 的報告。
- 請使用 XSLReportProcessor 類別, 使用 XSL 樣式表處理 XML 資料, 以產生可用格式 (環境) 的報告。

環境類別

「環境」類別支援特定的資料格式，此類別與 OutputQueue 及SpooledFileOutputStream 類別結合使用之後，可讓ReportWriter類別產生 該格式的報告，並將這些報告置於排存檔中。

您的應用程式只需建立一個「環境」類別的案例，ReportWriter 類別就會使用它來產生報告。您的應用程式絕對不能直接呼叫「環境」類別中的任何一種方法。PCLContext 及 PDFContext 方法原本就是設計來供 ReportWriter 類別於內部使用。

建構「環境」類別的案例需要 OutputStream (取自於 java.io 套件) 及 PageFormat (取自於 java.awt.print 套件)。 下列範例告訴您如何建構並與其它 ReportWriter 類別使用「環境」類別來產生報告：

範例：使用 XSLReportProcessor 與 PCLContext

範例：使用 JSPReportProcessor 與 PDFContext

JSPReportProcessor 類別

JSPReportProcessor 類別可讓您使用 JavaServer Page™ (JSP) 或 Java servlet 的內容 來產生文件或報告。

您可使用此類別從給定的 URL 取得 JSP 或 servlet，並使用其內容產生文件。JSP 或 servlet 必須提供文件資料，包括 XSL 格式化物件。您必須先指定輸出環境 及 JSP 輸入資料來源，才能產生文件的頁面。然後便可將報告資料轉換為指定的輸出資料串流格式。

JSPReportProcessor 類別可讓您：

- 處理報告
- 將 URL 設定為範本

下列範例告訴您如何使用 JSPReportProcessor 及 PDFContext 類別來產生報告。範例包含 Java 與 JSP 程式碼，您可使用下列鏈結來加以檢視。您也可下載包含下列項目的 zip 檔案：用於 JSPReportProcessor 與 XSLReportProcessor 範例的範例 JSP、XML 及 XSL 原始檔。

- 範例：使用 JSPReportProcessor 與 PDFContext
- 範例：JSPReportProcessor 範例 JSP 檔案

JSP 的其它相關資訊，請參閱 Java Server Pages 技術（位於 Sun 的 Java 網站）。

XSLReportProcessor 類別

XSLReportProcessor類別可讓您使用 XSL 樣式表來轉換及格式化您的 XML 來源資料，進而建立文件或報告。您可以此類別，使用包含 XSL 格式化物件 (FO) 的 XSL 樣式表 (必須符合 XSL 規格)，來建立報告。您接著可使用 Context 類別，將報告資料轉換成指定的輸出資料串流格式。

XSLReportProcessor 類別可讓您：

- 設定 XSL 樣式表
- 設定 XML 資料來源
- 設定 XSL FO 來源
- 處理報告

下列範例顯示如何使用 XSLReportProcessor 和 PCLContext 類別來產生報告。這些範例包括 Java、XML 和 XSL 程式碼，您可以使用下列鏈結來檢視它。您也可以針對 XSLReportProcessor 和 JSPReportProcessor 範例兩者， 下載包含 XML、XSL 和 JSP 原始檔範例的zip 檔：

- 範例：使用 XSLReportProcessor 含 PCLContext
- 範例：XSLReportProcessorXML 檔案範例
- 範例：XSLReportProcessorXSL 檔案範例

如需 XML 和 XSL 的詳細資訊，請參閱「資訊中心」的XML 主題。

Resource 類別

com.ibm.as400.resource 套件作為使用 AS400 各種物件和清單的同屬組織架構。
此組織架構能為所有這類物件和清單提供一致的程式設計介面。

此資源套件包含下列類別：

- Resource - 這是一個物件，它 代表 iSeries
  資源，例如使用者、印表機、工作、訊息或檔案。資源的具體次類別包括如下：
  - RIFSFile
  - RJavaProgram
  - RJob
  - RPrinter
  - RQueuedMessage
  - RSoftwareResource
  - RUser

  附註：access 套件中的NetServer類別也是 Resource 的具體次類別。

- ResourceList- 這是代表 iSeries 資源清單的物件，例如使用者、
  印表機、工作、訊息或檔案的清單。資源的具體次類別包括如下：
  - RIFSFileList
  - RJobList
  - RJobLog
  - RMessageQueue
  - RPrinterList
  - RUserList
- Presentation 這是一個物件，可讓您對一般使用者顯示出資源物件、
  資源清單、屬性、選項和排序用文字格式的相關資訊。

Resource 與 ChangeableResource 類別

com.ibm.as400.resource.Resource 和 com.ibm.as400.resource.ChangeableResource 抽象類別代表 iSeries 資源。

Resource

Resource 是提供任何資源屬性之同屬存取的抽象類別。 每個屬性都使用屬性 ID 來加以識別，任何給定的 Resource 次類別 一般都會把所支援的屬性 ID 記錄在文件中。

Resource 只提供屬性值的讀取權。

IBM Toolbox for Java 提供的資源物件如下：

- RIFSFile - 代表 iSeries 整合檔案系統中的檔案或目錄
- RJavaProgram - 代表 iSeries 中的 Java 程式
- RJob - 代表 iSeries 工作
- RPrinter- 代表 iSeries 印表機
- RQueuedMessage - 代表 iSeries 訊息佇列或工作日誌中的訊息
- RSoftwareResource- 代表 iSeries 中的授權程式
- RUser - 代表 iSeries 使用者

ChangeableResource

ChangeableResource 抽象類別是 Resource 的次類別， 新增了 iSeries 資源的屬性值的變更能力。 屬性變更為內部快取，直到已確定或已取消為止。 因此可讓您一次變更許多個屬性值。

附註：在存取套件 中的NetServer類別也是 Resource 和 ChangeableResource 的具體次類別。

範例

以下範例說明如何直接使用 Resource 和 ChangeableResource 的具體次類別， 以及同屬程式碼如何使用於任何 Resource 或 ChangeableResource 次類別。

- 從 RUser 擷取屬性值 , RUser 是 Resource 的具體次類別
- 設定 RJob 的屬性值, RJob 是 ChangeableResource 的具體次類別
- 使用同屬程式碼來存取資源

資源清單

com.ibm.as400.resource.ResourceList 類別代表 iSeries 資源清單。這是提供清單內容的同屬存取的抽象類別。

IBM Toolbox for Java 提供的資源清單如下：

- RIFSFileList - 代表 iSeries 整合檔案系統中的檔案和目錄清單
- RJobList - 代表 iSeries 工作清單
- RJobLog - 代表 iSeries 工作日誌中的訊息清單
- RMessageQueue - 代表 iSeries 訊息佇列中的訊息清單
- RPrinterList 代表 iSeries 的印表機清單
- RUserList - 代表 iSeries 的使用者清單

資源清單一律不是開啟就是已關閉。資源清單必須開啟以便存取當中的內容。
為能提供清單內容的立即存取和高效管理記憶體，資源清單大多以遞增方式載入。

資源清單可讓您：

- 開啟清單
- 關閉清單
- 從清單存取特定資源
- 等待特定資源載入
- 等待完整的資源清單載入

您也可以使用選項值來篩選資源清單。 每個選項值都使用選項 ID 來加以識別。
同樣地，資源清單還可以用排序值來排序。 每個排序值都使用排序 ID 來加以識別。ResourceList 的任何給定次類別一般都會把支援的選項 ID 和排序 ID 記錄在文件中。

範例

以下範例說明使用資源清單的各種方式：

- 範例：取得和列印 ResourceList 的內容
- 範例：使用同屬程式碼來存取 ResourceList
- 範例：在 servlet (HTML 表格) 中顯示出資源清單

Presentation 類別

每個資源物件、資源清單和 meta 資料物件都有相關的com.ibm.as400.resource.Presentation 物件可提供已轉換的資訊，例如名稱、完整名稱和圖示。

範例：把資源清單和清單的排序值使用它們的 Presentation 列印出來

您可使用 Presentation 資訊來把資源物件、資源清單、屬性、選項和排序用文字格式對一般使用者顯示出來。

```
void printCurrentSort(ResourceList resourceList) throws ResourceException
  {
// Get the presentation for the ResourceList and print its full name.
Presentation resourceListPresentation = resourceList.getPresentation();
System.out.println(resourceListPresentation.getFullName());

// Get the current sort value.
Object[] sortIDs = resourceList.getSortValue();

// Print each sort ID.
for(int i = 0; i < sortIDs.length; ++i)
      {
ResourceMetaData sortMetaData = resourceList.getSortMetaData(sortIDs[i]);
System.out.println("Sorting by " + sortMetaData.getName());
      }
  }
```

Security 類別

使用 IBM Toolbox for Java Security 類別可以提供和伺服器之間的安全連線、 驗證使用者 ID，以及在作業系統緒於本端伺服器上執行時，將其與使用者連結。包括的安全服務有：

- Java Secure Socket Extension (JSSE)
  透過將從屬站與伺服器階段作業之間交換的資料加密，以及透過執行伺服器鑑別，來提供安全連線。

  註： 關於使用 Secure Sockets Layer (SSL) 的資訊僅併入做為向後相容性。

- 鑑別服務 提供下列功能：
  - 將使用者識別及密碼與 OS/400 使用者登錄鑑別。
  - 可將識別指定到目前的 OS/400 緒。

Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) 透過下列方式提供安全的連線：

- 將從屬站與伺服器階段作業之間的交換資料加密
- 執行伺服器鑑別

»註： 請 考慮使用Java Secure Socket Extension (JSSE)來取代下列提供安全連線的方法。下列關於使用 SSL 的資訊僅內含做為向後相容性。«

使用 SSL 對於效能會有負面的影響，因為 SSL 連線比沒有加密的連線執行速度慢。
當轉送資料的機密性遠比因效能而增加的成本（例如，轉送信用卡或銀行對帳單資訊）重要時，請使用 SSL 連線。

在 IBM Toolbox for Java 開始使用 SSL 前，您必須瞭解您的任

»SSL 演算法

IBM Toolbox for Java 沒有加密及解密資料所需的演算法。V5R2 中，這些演算法會和 iSeries Client Encryptio
(128 位元) 授權程式，5722-CE3 一併出貨。

註：Toolbox for Java 也與 iSeries Client Encryption (56-bit) 授權程式，5722-CE2
相容，所以它不需再更新，也沒有 V5R2 版本。因為「從屬站加密」(56 位元)
包含強度低於「從屬站加密」(128 位元) 的演算法，您應考慮升級為 128 位元的加密。

請聯絡 IBM 業務代表，取得詳細資訊或如何訂購「從屬站加密」(128 位元)，5722-CE3 的資訊。«

設置 SSL 環境

IBM Toolbox for Java 提供兩種環境使用 SSL 來加密資料，所以您必須適當設置。

- 在 IBM Toolbox for Java 類別與 OS/400 伺服器之間使用加密
- 在虛擬從屬站與 PROXY 伺服器之間使用加密

»和先前版本的 IBM Toolbox for Java 的相容性

IBM Toolbox for Java V5R2 版、加密演算法以及 KeyRing 類別檔案需要您使用 V5R1 或 V5R2
版本的「從屬站加密」授權程式。

註： 如果您從 OS/400 的 V4R5 版本或更早版本升級，您也必須更新 KeyRing.class 檔案。

在從屬站中使用 IBM Toolbox for Java V5R2 版及相容版本的「從屬站加密」時，您可以連接到 V4R4
以及更新版本的 OS/400。有關「從屬站加密」相容版本的詳細資訊，請參閱SSL 演算法«

SSL 法律責任

IBM iSeries Client Encryption (128 位元) 授權產品使用 128 位元加密演算法，來提供 SSL 版本 3.0 加密支援。

本程式包含資料加密技術，該技術應遵守美國商業部的特殊出口授權之規定。其他國家或地區可能也有進出口授權之特別規定。

我們在此聲明，將同樣的程式移轉予同一國家，或不同國家的使用者使用，可能是受到限制的，或者應受到下列規範：

- 使用者所在國政府的特殊輸入法規或政策
- 貴國政府的特殊輸出法規或政策

從即刻起，以及授權期間屆滿之後，您都必須承擔所有責任，以確定保本程式是依據所有適用的進出口法規及政策使用或轉送。

您及您的使用者必須遵守其他國家或地區的進出口法律。

使用 SSL 將 IBM Toolbox for Java 與 OS/400 伺服器之間的資料加密

您可以使用 SSL 將 IBM Toolbox for Java 類別與 OS/400 伺服器之間的交換資料加密。在從屬站端，您可以使用 IBM iSeries Client Encryption 授權程式 (5722-CE2 或 5722-CE3) 隨附的檔案將資料加密。但在伺服器端，您就必須使用 OS/400 數位憑證管理程式來配置 OS/400 伺服器交換已加密的資料。

設置從屬站與伺服器使用 SSL

若要將 IBM Toolbox for Java 類別與 OS/400 伺服器之間流通的資料加密，請完成下列作業：

1. 設置您的伺服器交換已加密的資料。
2. 設置從屬站（IBM Toolbox for Java 類別）交換已加密的資料。 此步驟的程序取決於您在伺服器中設置 SSL 時，所使用的憑證種類而定：
   - 使用自授信中心取得的伺服器憑證
   - 使用自行簽署的憑證

   註： 使用自授信中心取得的憑證來設置從屬站，會比使用自行簽署的憑證來得更簡單與快速。
3. 使用SecureAS400 物件促使 IBM Toolbox for Java 將資料加密。

   註： 完成以上兩項步驟只能建立從屬站與伺服器之間的安全路徑。 您的應用程式必須使用 SecureAS400 物件來告知 IBM Toolbox for Java 所要加密的資料。唯有流經 SecureAS400 物件的資料才會進行加密。如果您使用的是 AS400 物件，就不會將資料加密，只會使用一般路徑通往伺服器。

設置 iSeries 伺服器使用 SSL

若要設置 iSeries 伺服器使其在 IBM Toolbox for Java 中使用 SSL，請完成下列步驟：

1. »安裝下列項目到您的 iSeries 伺服器中：
   - iSeries 的「IBM 密碼存取提供者 128 位元，5722-AC3」，可提供伺服器端的加密。
   - iSeries 從屬站加密 (128 位元)，5722-CE3，可提供 IBM Toolbox for Java 類別在從屬站端所使用的 Java 類別及公用程式。

   註：Toolbox for Java 也和「iSeries 密碼存取提供者 56 位元，5722-AC2」V5R1 版及「從屬站加密 (56 位元)，5722-CE2」 V5R1 版相容。«
2. 變更內含從屬站加密檔案的 目錄權限。
3. 取得並配置伺服器憑證。
4. 將認證套用到 IBM Toolbox for Java 使用的下列 iSeries 伺服器上：
   - QIBM_OS400_QZBS_SVR_CENTRAL
   - QIBM_OS400_QZBS_SVR_DATABASE
   - QIBM_OS400_QZBS_SVR_DTAQ
   - QIBM_OS400_QZBS_SVR_NETPRT
   - QIBM_OS400_QZBS_SVR_RMTCMD
   - QIBM_OS400_QZBS_SVR_SIGNON
   - QIBM_OS400_QZBS_SVR_FILE
   - QIBM_OS400_QRW_SVR_DDM_DRDA

變更內含從屬站加密檔案的目錄權限

為了協助您在使用加密演算法時，符合必要的 法律責任，內含加密檔案的目錄出貨時即設定公用權限為 *EXCLUDE。 您必須變更目錄的權限，只允許經過授權的使用者使用加密演算法。

請完成下列步驟，使用 OS/400 物件安全性來控制存取從屬站加密檔案：

1. 在伺服器中輸入下列指令：

   ```
   wrklnk '/QIBM/ProdData/HTTP/Public/jt400/*'
   ```

2. 在 SSL56 或 SSL128 目錄中選取選項 9。
3. 確定 *PUBLIC 具有 *EXCLUDE 權限。
4. 將目錄的 *RX 權限授予需要存取 SSL 檔的個人或使用者群組。

   註：對具有 *ALLOBJ 特殊權限的使用者，您無法拒絕其存取 SSL 檔案。

取得以及配置伺服器憑證

取得以及配置伺服器憑證之前，您必須安裝下列產品：

- IBM HTTP Server for iSeri (5722-DG1) 授權程式

- 基本作業系統選項 34 (數位憑證管理程式)

取得以及配置伺服器憑證所需遵循的程序，是根據您所使用的憑證種類而定：

- 如果您從授信中心 (像是 VeriSign、Inc. 或 RSA Data Security, Inc.) 取得憑證，請將憑證安裝在 iSeries 中，然後將它套用到主電腦伺服器上。
- 如果您選擇不使用來自授信中心的憑證，您可以建置自己的憑證，以在 iSeries 上使用。使用「數位憑證管理程式」建置憑證。
  1. 在 iSeries 伺服器上建立認證中心。請參閱「資訊中心」主題，做為您自己的 CA。
  2. 從您建立的認證中心建立系統憑證。
  3. 指定哪些主電腦伺服器要使用您建立的系統憑證。

使用自授信中心取得的認證

IBM Toolbox for Java 附有一個金鑰環檔案，它可支援來自一些授信中心的伺服器憑證，下列為代表的公司：

- IBM World Registry
- Integrion Financial Network
- RSA Data Security, Inc.
- Thawte Consulting
- VeriSign, Inc.

金鑰環檔案已可支援您自其中一家授信中心取得的憑證。您只需要取得包含加密演算法的 Zip 檔，並將它新增到 CLASSPATH 陳述式中。

若要使用憑證，請完成下列步驟：

1. 選取您要放置 Zip 檔的目錄。
2. 複製 Zip 檔到所選取的目錄下，以便下載您想要使用的 SSL 版本：
   - 若是 56 位元的加密 (搭配授權程式 5722-CE2)，請複製 /QIBM/ProdData/HTTP/Public/jt400/SSL56/sslightx.zip。
   - 若是 128 位元的加密 (搭配授權程式 5722-CE3)，請複製 /QIBM/ProdData/HTTP/Public/jt400/SSL128/sslightu.zip。
3. 新增 Zip 檔到 CLASSPATH 陳述式。

建置您自己的憑證 (SSL)

»當您選擇不使用 憑信中心 取得的憑證時，您必須下載自行簽署的「認證中心 (CA)」憑證 (從每部具有自行簽署 CA 憑證的伺服器下載)，如此，IBM Toolbox for Java 類別便可以使用該憑證。«您還必須取得內含加密演算法的 Zip 檔，並將它新增到 CLASSPATH 陳述式。

若要使用自行簽署的憑證，請完成下列步驟：

1. 選取您要放置 Zip 檔的目錄。
2. 複製要和自行簽署憑證一起使用的加密演算法與公用程式，以便下載您想要使用的 SSL 版本：
   - 針對 56 位元的加密 (配合授權程式 5722-CE2 使用)，請複製 /QIBM/ProdData/HTTP/Public/jt400/SSL56/sslightx.zip、cfwk.zip 及 ssltools.jar
   - 針對 128 位元的加密 (配合授權程式 5722-CE3 使用)，請複製 /QIBM/ProdData/HTTP/Public/jt400/SSL128/sslightu.zip、cfwk.zip 及 ssltools.jar。
3. 新增 ssltools.jar 與 Zip 檔到 CLASSPATH 陳述式。
4. 在您的從屬站中建立名為 <SSL>\com\ibm\as400\access 的目錄，其中的 <SSL> 是您複製 JAR 與 Zip 檔所在的目錄。
5. 由從屬站的 <SSL> 目錄的指令提示中，執行下列指令：

   ```
   java utilities.KeyringDB com.ibm.as400.access.KeyRing -connect
   <systemname>:<port>
   ```

   其中的 <port> 可以是任何一個主電腦伺服器的伺服器埠。例如，您可以使用 9476，這是 iSeries 上安全登入伺服器的預設埠。

6. »鍵 入您想要新增到 KeyRing 的「認證中心 (CA)」憑證的數目。«請確定新增 CA 憑證，而不要新增網站憑證。
7. 當系統提示您輸入憑證名稱時，您可以鍵入任何的英數字串。

   註：您必須對每個有自行簽署憑證的伺服器執行 KeyringDB，以新增每個憑證到 KeyRing 類別。請在您想要使用 SSL 連線的每部 iSeries 中執行下列指令，以新增憑證：

   ```
   java utilities.KeyringDB com.ibm.as400.access.KeyRing -connect
   <systemname>:<port>
   ```

完成以上步驟後，您就完成了自行認證的設置。確定您的 CLASSPATH 陳述式中包含下列項目後，您便可以開始執行應用程式：

- 包含 com\ibm\as400\access\KeyRing.class 的目錄
- jt400.jar 檔
- sslightx.zip 或 sslightu.zip (視您下載的檔案而定)

由於 jt400.jar 檔包含 KeyRing.class 的預設副本，因此包含 com\ibm\as400\access\KeyRing.class 的目錄在 CLASSPATH 中必須位於 jt400.jar 前面。

註：如不新增內含 KeyRing.class 檔的目錄到 CLASSPATH 陳述式，您可以利用新的 KeyRing.class 來置換 jt400 中的既有類別。

使用 SSL 將 PROXY 從屬站與 PROXY 伺服器之間的資料加密

您可以使用 SSL 將 PROXY 從屬站與 PROXY 伺服器之間的交換資料加密。IBM iSeries Client Encryption 授權程式 (5722-CE2 或 5722-CE3) 隨附的檔案就是用來將資料加密。如同 IBM Toolbox for Java, 這些檔案是與平台無關的 Java 類別, 可以讓 PROXY 從屬站及 PROXY 伺服器在具有 Java 虛擬機器的任一平台上執行。

請執行下列作業, 將 PROXY 從屬站及 PROXY 伺服器之間的資料流加密：

1. 設置PROXY 伺服器 來處理已加密的資料。
2. 設置 PROXY 從屬站 來處理已加密的資料。
3. 使用SecureAS400 物件促使 IBM Toolbox for Java 將資料加密。

註： 前兩個步驟只是在 PROXY 從屬站與 PROXY 伺服器之間建立一個安全路徑。應用程式必須使用 SecureAS400 物件來告知 IBM Toolbox for Java 讓資料流經安全路徑。使用 AS400 物件並不會將資料加密, 只會使用一般的路徑通往伺服器。

如果您要將 PROXY 從屬站與 PROXY 伺服器之間的資料流加密, 您只需要 Client Encryption 授權程式 (5722-CE2 或 5722-CE3) 隨附的 Java 類別即可。 如果您要將 PROXY 伺服器與 iSeries 伺服器之間的資料流加密, 您還必須設置 PROXY 伺服器與 iSeries 伺服器之間的加密。

在 PROXY 伺服器中設置 SSL

若要啟用 SSL，PROXY 伺服器必須有伺服器憑證。請使用 IKeyman 圖形式使用者介面 (GUI) 來建立 PROXY 伺服器所要使用的伺服器憑證。IKeyman 是一個 GUI 工具，因此您必須在用戶端機器上執行。
建立憑證之後，如果您的 PROXY 伺服器是在 iSeries 中執行，您就可以將它複製到 iSeries 上。

若要設置 PROXY 伺服器來處理已加密資料，請執行下列作業：

1. 設置從屬站執行 IKeyman GUI。
2. 建立 PROXY 伺服器的伺服器憑證。
3. 使用您剛剛建立的憑證來啟動 PROXY 伺服器。

設置從屬站執行 IKeyman GUI

IKeyman GUI 是以 Java Swing 1.1 介面為基準的 Java 程式。若要使用 IKeyman，您的從屬站必須執行 Java 1.1 JVM (以及 Swing 1.1 外掛程式) 或 Java 2 JVM。

IKeyman GUI 是 IBM iSeries Client Encryption 授權程式 (5722-CE2 或 5722-CE3) ssltools.jar 檔的一部份。用來設置從屬站使用 SSL (以及執行 IKeyman) 的程序，取決於所執行的授權程式版本。

請完成下列步驟，設置從屬站使用 SSL：

1. 在工作站中選取要放置必要的 JAR 與 Zip 檔案的目錄。
2. 將必要的檔案複製到所選取的目錄：
   - 若使用 56 位元加密，請在載入授權程式 5722-CE2 至 iSeries 後，複製下列檔案至您的工作站：
     - /QIBM/ProdData/http/public/jt400/ssl56/sslightx.zip
     - /QIBM/ProdData/http/public/jt400/ssl56/ssltools.jar
     - /QIBM/ProdData/http/public/jt400/ssl56/cfwk.zip
     - /QIBM/ProdData/http/public/jt400/ssl56/cfwk.sec
   - 若使用 128 位元加密，請在載入授權程式 5722-CE3 至 iSeries 後，複製下列檔案至您的工作站：
     - /QIBM/ProdData/http/public/jt400/ssl128/sslightu.zip
     - /QIBM/ProdData/http/public/jt400/ssl128/ssltools.jar
     - /QIBM/ProdData/http/public/jt400/ssl128/cfwk.zip
     - /QIBM/ProdData/http/public/jt400/ssl128/cfwk.sec
3. 新增 JAR 檔與 Zip 檔到 CLASSPATH 陳述式。請勿新增 .sec 檔到 CLASSPATH。

   註：cfwk.zip 必須是 CLASSPATH 中的第一個項目。

建立伺服器憑證

您可以使用 IKeyman GUI 建立自行簽署的憑證。

註：如果 IKeyman GUI 停止執行，請檢查 cfwk.zip 是否為 CLASSPATH 中的第一個項目，且 cfwk.sec 和 cfwk.zip 位於同一個目錄中。

請完成下列步驟，建立 PROXY 伺服器的伺服器憑證：

1. 使用下列指令啟動 IKeyman GUI：

   ```
   java -Dkeyman.javaOnly=true com.ibm.gsk.ikeyman.Ikeyman
   ```

2. 從 IKeyman 金鑰資料庫檔案 功能表，選取新增。
3. 在新增 對話框中，不要改變 金鑰資料庫類型，它應該是 SSLight 金鑰資料庫類別。
4. 鍵入檔名 (例如,ProxyServerKeyring.class)，或按一下 瀏覽，找出 keyring 要使用的類別檔案。

   註：請記得這個 Keyring 檔名，因為您需要用它來啟動安全 PROXY 伺服器。

5. 鍵入位置 (路徑) 或接受預設的位置，也就是現行工作目錄，然後按一下確定。
6. 在密碼提示 對話框中，鍵入 密碼 以及確認密碼，然後按一下 確定。
   (設定到期時間是可選用的，您不需要選取它。)

   註：請記得您的密碼，在啟動安全 PROXY 伺服器時需要用到。
   此對話框中的金鑰圖示代表密碼的相對強度。強勢密碼需要混合大寫與小寫的英數字元。

7. 從 IKeyman 建立 檔案功能表，選取新的自行簽署憑證。
8. 在建立新的自行簽署憑證 對話框中，鍵入 金鑰標籤 (例如,MyCertificate) 及組織。
9. 按一下 國家清單選取國家或地區，鍵入有效期間或接受預設值，然後按一下 確定。
10. 從 IKeyman 金鑰資料庫檔案 功能表，選取關閉，然後 (從同一個的功能表) 按一結束。

您現在應該可以在現行目錄中看到剛才建立的 Keyring。

使用新的憑證啟動 PROXY 伺服器

啟動 PROXY 伺服器之前，請確定 PROXY 伺服器的 CLASSPATH 有包含 jt400.jar、sslightx.zip 以及 PROXY 伺服器的 Keyring 位置。

使用您剛建立的憑證來啟動 PROXY 伺服器。請使用 -keyringName 與 -keyringPassword 參數將此資訊傳送給 PROXY 伺服器。例如：

```
 java com.ibm.as400.access.ProxyServer -keyringName ProxyServerKeyring -
keyringPassword pxypswrd
```

在虛擬從屬站中設置 SSL

下列程序會引導您新增伺服器憑證到從屬站中的憑證資料庫， 該憑證資料庫儲存在一個 Java .class 檔案中。新增伺服器憑證到從屬站是必要的步驟， 因為伺服器使用的是自行簽署的憑證。

請完成下列作業，設置虛擬從屬站交換加密資料：

1. 設置 PROXY 伺服器處理已加密的資料，然後啟動 PROXY 伺服器。
2. 設置您的從屬站使用 SSL。
3. 使用 KeyringDB 以取得 PROXY 伺服器的伺服器憑證。
4. 設置從屬站使用已更新的 KeyRing.class 檔
5. 設定從屬站上的安全 Proxy 設定值。

設置從屬站使用 SSL

用來下載憑證 (KeyringDB) 的工具是一支 Java 程式。若要使用此程式， 您的從屬站必須執行 Java 1.1.8 或 Jav JVM。KeyringDB 是 IBM iSeries Client Encryption 授權程式 (5722-CE2 或 5722-CE3) ssltools.jar 檔案的一部份。用來設置從屬站使用 SSL 的程序，取決於所執行的授權程式版本而定。

設置好 PROXY 伺服器後，請完成下列步驟來設置從屬站使用 SSL：

1. 在工作站中選取要放置必要的 JAR 與 Zip 檔案的目錄。
2. 將必要的檔案複製到所選取的目錄：
   - 若使用 56 位元加密，請在載入授權程式 5722-CE2 至 iSeries 後，複製下列檔案至您的工作站：
     - /QIBM/ProdData/http/public/jt400/ssl56/sslightx.zip
     - /QIBM/ProdData/http/public/jt400/ssl56/ssltools.jar
     - /QIBM/ProdData/http/public/jt400/ssl56/cfwk.zip
   - 若使用 128 位元加密，請在載入授權程式 5722-CE3 至伺服器後，複製下列檔案至您的工作站：
     - /QIBM/ProdData/http/public/jt400/ssl128/sslightu.zip
     - /QIBM/ProdData/http/public/jt400/ssl128/ssltools.jar
     - /QIBM/ProdData/http/public/jt400/ssl128/cfwk.zip
3. 新增 JAR 檔與 Zip 檔到 CLASSPATH 陳述式。
4. 在您的從屬站中建立名為 <SSL>\com\ibm\as400\access 的目錄，其中的 <SSL> 是您複製 JAR 與 Zip 檔所在的目錄。

新增 PROXY 伺服器的伺服器憑證

KeyringDB 會建立一個新的包含伺服器憑證的 KeyRing.class 檔案，並將它放到現行目錄下的 com\ibm\as400\access 子目錄中。

請完成下列步驟，使用 KeyringDB 工具新增伺服器憑證到 KeyRing.class：

1. 從放置 JAR 檔與 Zip 檔的目錄，執行下列指令：

```
java utilities.KeyringDB com.ibm.as400.access.KeyRing -connect
```

proxyServerName:port

其中：

- proxyServerName 是執行 PROXY 伺服器的機器名稱
- port 是安全 PROXY 伺服器監聽的埠（在預設的情況下，埠號為 3471）

例如：

```
 java utilities.KeyringDB com.ibm.as400.access.KeyRing -connect
myProxyServer:3471
```

2. 被問到要使用何種憑證時，請選擇網站憑證 0。
3. 當系統提示您輸入憑證名稱時，您可以鍵入任何的英數字串。

設置從屬站使用已更新的 KeyRing.class 檔案

jt400Proxy.jar 檔中包含 KeyRing.class。若要設置從屬站使用已更新的 KeyRing.class 檔案，請確定您的 CLASSPATH 陳述式中包含下列項目：

- 包含 com\ibm\as400\access\KeyRing.class 的目錄
- jt400Proxy.jar 檔案
- sslightx.zip 或 sslightu.zip（視您下載的檔案而定）
- cfwk.zip 檔案

由於 jt400Proxy.jar 檔中包含 KeyRing.class 的預設副本，因此包含 com\ibm\as400\access\KeyRing.class 的
CLASSPATH 中必須位於 jt400Proxy.jar 前面。

註：如不新增內含 KeyRing.class 檔的目錄到 CLASSPATH 陳述式，您可以新增新的 KeyRing.class 到 jt400Proxy.jar 檔案中。新增新的 KeyRing.class 檔案到 jt400Proxy.jar 中會改寫舊版本。

設定從屬站上的安全 Proxy 設定值

若要告知虛擬從屬站透過安全連線和 PROXY 伺服器進行通信，請設定下列 系統內容

```
com.ibm.as400.access.AS400.proxyServer=          proxyServer
```

其中的proxyServer是執行 PROXY 伺服器的機器名稱

```
com.ibm.as400.access.SecureAS400.proxyEncryptionMode=              mode
```

其中的mode 是下列其中一個整數：

- 1 在虛擬從屬站與 PROXY 伺服器之間加密
- 2 在 PROXY 伺服器與 iSeries 伺服器之間加密
- 3 在虛擬從屬站與 PROXY 伺服器、以及 PROXY 伺服器與 iSeries 伺服器之間加密

例如，下列指令會使用 SSL 啟動應用程式：

```
 java -Dcom.ibm.as400.access.AS400.proxyServer=myProxyServer
-Dcom.ibm.as400.access.SecureAS400.proxyEncryptionMode=1 myApplication
```

鑑別服務

類別是由 IBM Toolbox for Java 提供，與 OS/400 提供的安全服務相互作用。
特別是，可提供鑑別使用者身分（有時候稱為體 (principal)）支 援，以及對照 OS/400
使用者登錄的密碼。接著便可建立代表已驗證使用者的認證。 您可以使用認證來改變現行 OS/400 緒的
ID，利用已驗證的使用者權限及許可權執行工作。 實際上，這種 ID
的交換會導致緒的運作如同已驗證的使用者執行登入一般。

附註：建立及交換憑證的服務只有在伺服器的版次是 V5R1M0 或更新版才受到支援。

提供的支援概觀

AS400 物件提供特定使用者設定檔以及密碼和伺服器對照的鑑別。
您也可以擷取代表系統上已鑑別之使用者設定檔與密碼的»Kerberos 通行證及設定檔記號«。

»註 ：您必須安裝 J2SDK，v1.4 並配置 Java General Security Services (JGSS)「應用程式設計介面」，才能使
Kerberos 通行證。有關 JGSS 的詳細資訊，請參閱J2SDK, v1.4 Security Documentati。«

»若要使用 Kerberos 通行證，只要在 AS400 物件中設定系統名稱（不是密碼）。 使用者識別是透過 JGSS
組織架構擷取。在 AS400 物件中一次只能設定一種鑑別方式。 設定密碼會清除任何的 Kerberos
通行證或設定檔記號。 «

若要使用設定檔記號，請使用getProfileToken()方法擷取ProfileTokenCredential
類別的案例。可以將設定檔記號想像成是一個特定伺服器的已驗證使用者設定檔及密碼的代表。
設定檔記號的到期基於時間，最多一個小時，但在某些情形下可以重新整理，並提供延伸的生命期限。

»下 面範例會建立一個系統物件，並使用該物件來產生設定檔記號。
然後此範例會使用設定檔記號來建立另一個系統物件，再使用第二個系統物件來連接至指令服務程式：

```
AS400 system = new AS400("mySystemName", "MYUSERID", "MYPASSWORD");
ProfileTokenCredential myPT = system.getProfileToken();
AS400 system2 = new AS400("mySystemName", myPT);
system2.connectService(AS400.COMMAND);          «
```

設定緒識別

您可以在遠端或本端上下文建立一個認證。一旦建立之後，
您可以序列化或分送此認證，一如呼叫應用程式所需。
當認證被傳遞給相關伺服器上的執行中程序時，此認證可以用來修改或 交換 OS/400
緒識別，並以先前鑑別過的使用者身分執行工作。

這種支援的實際應用方式可能是在兩階應用程式中， 在第一階的圖形式使用者介面（如 PC）
執行使用者設定檔及密碼鑑別， 並在第二階（伺服器）為該使用者執行工作。藉由使用
ProfileTokenCredentials， 應用程式可以避免直接透過網路傳送使用者 ID 和密碼。
此設定檔記號便可接著分送到第二階的程式， 它可以執行up() 並在指派給使用者的 OS/400
權限及許可權之下操作。

備註 ：雖然在本質上因為有限的生命期限， 而比傳送使用者設定檔及密碼更具有安全性，
設定檔記號仍應被應用程式視為敏感資訊，並依此處理。
既然記號代表已驗證使用者及密碼，它就有可能被不良的應用程式利用，
代表該使用者執行工作。所以，確保該認證是在安全的方式存取是應用程式的責任。

範例

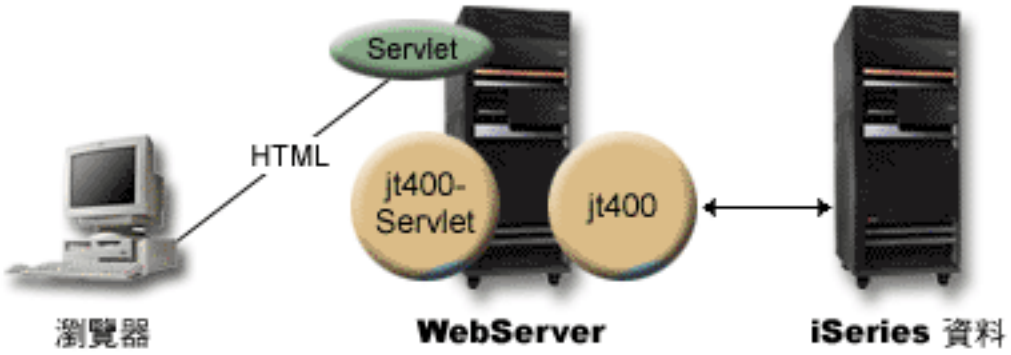
請參照此程式碼，作為如何使用設定檔記號認證來交換 OS/400 緒
ID，並以特定使用者的身份執行工作的範例。

Servlet 類別

IBM Toolbox for Java 隨附的 Servlet 類別與位於 Web 伺服器存取類別一起使用時，可讓您存取 iSeries 伺服器上的資訊。您可以決定如何使用 Servlet 類別來輔助您自己的 Servlet 專案。

下列圖解顯示 Servlet 類別如何在瀏覽器、Web 伺服器和 iSeries 資料之間運作。 瀏覽器會連接到執行 Servlet 的 Web 伺服器。jt400Servlet.jar 以及 jt400.jar 檔案在 Web 伺服器上，因為 Servlet 類別會使用某些存取類別來擷取資料，並以與 HTML 類別呈現資料。 Web 伺服器則是連接到資料所在的 iSeries 系統。

圖 1：Servlets 的運作情形



IBM Toolbox for Java 內含有四種 Servlet 類別：

- Authentication 類別
- RowData 類別
- RowMetaData 類別
- Converter 類別

註：jt400Servlet.jar 檔內含 Servlet 類別。如果您要使用 com.ibm.as400.util.html 以及 com.ibm.as400.util.servlet 套件中的類別，您必須將 CLASSPATH 更新為指向 jt400Servlet.jar 與 jt400.jar。

有關 Servlets 的一般詳細資訊，請參閱參考資訊區段。

鑑別類別

Servlet 套件中的兩個類別會針對 Servlets 與 AuthenticationServlet 與 AS400Servlet 執行鑑別。

AuthenticationServlet 類別

AuthenticationServlet 是一個 HttpServlet 施行方式，可執行 Servlets 基本鑑別。AuthenticationServlet 的次類別應該置換下列一或多個方法：

- 置換 validateAuthority() 方法以執行鑑別（必要的）
- 置換 bypassAuthentication() 方法，使得次類別只能鑑別特定的要求
- 置換 postValidation() 方法，可以在鑑別之後處理其它的要求

AuthenticationServlet 類別提供的方法可讓您：

- 起始設定 Servlet
- 取得已鑑別的使用者 ID
- 略過鑑別之後，設定使用者 ID
- 記載異常情況及訊息

AS400Servlet 類別

AS400Servlet 類別是 AuthenticationServlet（代表一個 HTML Servlet）的摘要次類別。
您可以使用連線儲存區來共用連線並管理 Servlet 使用者可擁有的伺服器連線數目。

AS400Servlet 類別提供的方法可讓您：

- 驗證使用者權限（透過置換 AuthenticationServlet 類別的 validateAuthority() 方法）
- 連接至系統
- 自儲存區取得連線儲存區物件，並將連線儲存區物件傳回儲存區
- 關閉連線儲存區
- 取得並設定 HTML 文件表頭標籤
- 取得並設定 HTML 文件結尾標籤

有關 Servlets 的一般詳細資訊，請參閱參考資訊區段。

RowData 類別

RowData 類別是一個抽象類別，它提供一種說明及存取資料清單的方式。

延伸 RowData 類別的主要有四種類別：

- ListRowData
- RecordListRowData
- ResourceListRowData
- SQLResultSetRowData

RowData 類別可讓您：

- 取得和設定 現行位置
- 使用getObject()方法取得特定直欄的資料列資料
- 取得列的meta 資料
- 取得或設定特定直欄的物件內容
- 使用length()方法取得清單中的列數。

RowData 定位

有幾種方法可讓您取得並設定清單中的現行位置。 下表列出 RowData 類別的 set 和 get 方法。

Set 方法		Get 方法
absolute()	next()	getCurrentPosition()
afterLast()	previous()	isAfterLast()
beforeFirst()	relative()	isBeforeFirst()
first()		isFirst()
last()		isLast()

ListRowData 類別

ListRowData 類別可讓您執行下列作業：

- 新增 及移除結果清單中的列。
- 取得以及設 定資料列
- 使用getMetaData() 方法 取得清單的直欄之相關資訊
- 使用setMetaData() 方法 ， 設定直欄資訊

ListRowData 類別代表資料清單。 使用 IBM Toolbox for Java access 類別， ListRowData 可代表許多類型的資訊，包括以下的：

- 整合檔案系統中的目錄
- 工作 清單
- 訊息佇列中的訊息清單
- 使用者清單
- 印表機清單
- 排存檔 清單

此範例 說明 ListRowData 及 HTMLTableConverter 如何運作。它會顯示 java 程式碼、HTML 程式碼及 HTML 的外觀與感覺。

RecordListRowData 類別

RecordListRowData 類別可讓您執行下列工作：

- 將資料列由記錄清單中來回地新增 和 移除。
- 取得和 設定 資料列
- 使用setRecordFormat方法設定記錄格式
- 取得記錄格式。

RecordListRowData 類別代表一份記錄清單。可以從伺服器以不同格式取得一筆記錄，包括以下幾種：

- 要寫入伺服器檔案或從伺服器檔案讀取的 記錄
- 在資料佇列中的登錄項目
- 來自程式呼叫的參數資料
- 任何需要在伺服器格式與 Java 格式之間作轉換的回傳資料

這個範例 顯示 RecordListRowData 和 HTMLTableConverter 如何運作。它會顯示 java 程式碼、HTML 程式碼及 HTML 的外觀及運作。

ResourceListRowData 類別

ResourceListRowData類別代表資料的資源清單。請使用 ResourceListRowData 物件 來代表任何施行的ResourceList介面。

資源清單會格式化成為一系列的列，各列都含有由直欄數屬性 ID 所決定的有限直欄數。
一列以內的每個直欄都含有一個單獨的資料項目。

ResourceListRowData 類別所提供的方法可讓您執行下列動作：

- 取得和 設定直欄屬性 ID
- 取得和 設定資源清單
- 從清單中 擷取列數
- 為現用列取得直欄資料
- 取得資料物件的內容清單
- 為清單取得 meta 資料

範例： 在 servlet 中顯示出資源清單

SQLResultSetRowData 類別

SQLResultSetRowData 類別將 SQL 結果集呈現成資料清單型態。 這個資料是由 SQL 陳述式經由JDBC 所產生。您可以使用提供的方法, 取得並設定結果集的 meta 資料。

這個範例 可告訴您 ListRowData 和 HTMLTableConverter 的工作方式。它會顯示 Java 程式碼、HTML 程式碼及 HTML 的外觀及運作。

RowMetaData 類別

RowMetaData 類別 定義了一個介面，您可以用它來找出 RowData 物件欄位的相關資訊。

您可以使用 RowMetaData 類別執行下列項目：

- 取得直欄數
- 取得直欄的名稱、類型或大小
- 取得或設定直欄標籤
- 取得直欄資料的精準度或比例
- 判斷直欄資料是否為文字資料

建置 RowMetaData 類別的主要類別有三種。 這些類別除了具有自己的特定功能以外， 還提供所有上述的 RowMetaData 功能。

- ListMetaData
- RecordFormatMetaData
- SQLResultSetMetaData

ListMetaData 類別

ListMetaData可讓您 在ListRowData 類別中取得相關資訊，及變更直欄的設定。 其使用setColumns()方法來設定直欄數、清除所有先前的直欄資訊。另外，當您設定建構元的參數時，您也可以略過直欄數。

此範例 會說 明 ListMetaData、ListRowData 及 HTMLTableConverter 如何運作。它會顯示 java 程式碼、HTML 程式碼及 HTML 的外觀與運作。

RecordFormatMetaData 類別

RecordFormatMetaData 利用 IBM Toolbox for Java RecordFormat 類別。它可讓您在設定建構元的參數時提供記錄格式，或是使用 get 和 set 方法來存取記錄格式。

下列範例會告訴您如何建立 RecordFormatMetaData 物件：

```
// Create a RecordFormatMetaData object from a sequential file's record
format.
RecordFormat recordFormat = sequentialFile.getRecordFormat();
RecordFormatMetaData metadata = new RecordFormatMetaData(recordFormat);

// Display the file's column names.
int numberOfColumns = metadata.getColumnCount();
for (int column=0; column < numberOfColumns; column++)
{
 System.out.println(metadata.getColumnName(column));
}
```

SQLResultSetMetaData 類別

SQLResultSetMetaData 類別會傳回有關SQLResultSetRowData 物件的直欄資訊。您可以在設定建構元的參數時提供結果集，或是使用取 得和 設 定方法來存取結果集的 meta 資料。

下列範例會告訴您如何建立 SQLResultSetMetaData 物件：

```
// Create an SQLResultSetMetaData object from the result set's metadata.
SQLResultSetRowData rowdata = new SQLResultSetRowData(resultSet);
SQLResultSetMetaData sqlMetadata= rowdata.getMetaData();

// Display the column precision for non-text columns.
String name = null;
int numberOfColumns = sqlMetadata.getColumnCount();
for (int column=0; column < numberOfColumns; column++)
{
 name = sqlMetadata.getColumnName(column);
 if (sqlMetadata.isTextData(column))
   {
System.out.println("直欄： " + name + " 包含文字資料。");
   }
else
   {
System.out.println("直欄： " + name + " 的精準度為 " +
sqlMetadata.getPrecision(column));
   }
}
```

轉換器類別

您可使用轉換器類別，將列資料轉換成格式化的字串陣列。結果會 以 HTML 套表呈現，並可供 HTML 網頁使用。下列類別會替您處理轉換：

- StringConverter
- HTMLFormConverter
- HTMLTableConverter

StringConverter 類別

StringConverte類別為一抽象類別，它代表的是一列資料字串的轉換器。 它提供了convert()方法，可用來轉換列資料。這會傳回呈現該列資料的字串陣列。

HTMLFormConverter 類別

HTMLFormConverter 類別會擴充 StringConverter 方法是經由提供額外的轉換方式，此方式稱為 convertToForms() 此方法會將列資料轉換成單一列 HTML 表格的陣列。
您可以使用這些表格標示，在瀏覽器中顯示格式化的資訊。

您可以使用不同的取得及設定方法，檢視或變更套表的屬性以 裁製 HTML
套表的外觀。例如，您可以設定的部份屬性包括：

- 對齊
- 資料格間距
- 標頭超連結
- 寬度

範例

範例： 使用 HTMLFormConverter。(請在執行中的 Web 伺服器上編譯及執行此範例)

HTMLTableConverter 類別

HTMLTableConverter 類別藉由提供 convertToTables() 方法，來延伸 StringConverter。此方法將資料列轉換成 HTML 表格陣列，以便讓 Servlet 用來在瀏覽器中顯示清單。

您可以使用 getTable() 及 setTable() 方法，選擇轉換期間要使用的預設表格。您可以在 HTML 表格物件中設定表格的標頭，或使用標頭資訊的 meta 資料，方法為設定 setUseMetaData() 為 TRUE。

setMaximumTableSize() 方法可讓您限制單一表格的列數。如果資料列超出指定的表格大小，則轉換器會在輸出陣列中，產生另一個 HTML 表格物件。此會一直繼續到所有資料列都被轉換完畢。

範例

下面範例說明如何使用 HTMLTableConverter 類別：

- 範例： 使用 ListRowData
- 範例： 使用 RecordListRowData
- 範例： 使用 SQLResultSetRowData
- 範例： 顯示 Servlet 中的 ResourceList

公用程式類別

公用程式類別可以幫助您執行特定的作業。IBM Toolbox for Java 提供下列公用程式：

- AS400ToolboxInstaller 可讓您安裝並更新從屬站上的 IBM Toolbox for Java 類別。此函數可以當作 Java 程式使用，且含有應用程式設計介面 (API)。
- AS400ToolboxJarMaker：產生載入速度較快的 IBM Toolbox for Java 檔案，其方式是從較大的 JAR 檔案建立一個較小的 JAR 檔案，或選擇解壓縮一個 JAR 檔案來存取個別的內容檔案。
- CommandPrompter ：提示輸入給定指令的參數。CommandPrompter 提供的功能類似於 iSeries CL 指令提示 (按 F4)，且和「管理中心」指令提示相同。
- RunJavaApplication 和 VRunJavaApplication：可讓您在 iSeries 伺服器上，從指令行提示執行 Java 程式。
- JPing: 可讓您查詢伺服器，找出作用中的服務。您也可以指定是否要連通測試 (ping) SSL 埠。

從屬站安裝與更新類別

在伺服器的整合檔案系統中，IBM Toolbox for Java 類別可在其位置上被參考到。 由於暫時性程式修訂 (PTF) 套用於此位置，所以在伺服器中直接存取 這些類別的 Java 程式會自動接收這些更新。存取伺服器的類別並非總是可以成功，尤其是下列情況：

- 如果使用低速通信鏈結連接伺服器與從屬站，則從伺服器載入類別的效能可能很低。
- 如果 Java 應用程式使用 CLASSPATH 環境變數來存取從屬站檔案系統上的類別， 則您需要 iSeries Access for Windows，方可將檔案系統呼叫重新導向至伺服器。iSeries Access for Windows 可能無法常駐在從屬站中。

在這些情況中，在從屬站中安裝這些類別是較佳的解決方案。 當 AS/400 Toolbox for Java 類別常駐在從屬站時， AS400ToolboxInstaller類別 會提供從屬站安裝和更新功能來管理 IBM Toolbox for Java 類別。

使用 AS400ToolboxInstaller

»AS400ToolboxInstaller 物件 不僅是個程式，也是個可程式設計的介面。此物件有一個 main() 方法可以讓您從指令行執行它。 它也有一個公用工作者方法，可以內含在您的應用程式中，或從應用程式呼叫它。

您可以使用 AS400ToolboxInstaller 物件將 Toolbox 檔案安裝在從屬站上， 並使其保持最新。第一次使用 Toolbox 時，檔案會從伺服器被複製到工作站。 將 PTFs 套用到伺服器時，AS400ToolboxInstaller 物件會使用伺服器上的新檔案 更新工作站上的檔案«

AS400ToolboxInstaller 類別會將檔案複製到從屬站的區域檔案系統。此類別可能無法在小型程式中運作；許多瀏覽器不容許 Java 程式寫入到區域檔案系統。

在指令行中執行 AS400ToolboxInstaller 類別

AS400ToolboxInstaller 類別可以當作獨立式程式使用，在指令行中執行。 在指令行中執行 AS400ToolboxInstalle 表示您不需要撰寫程式。 相反的，您可以把它當成 Java 應用程式來執行，以安裝、解除安裝或 更新 IBM Toolbox for Java 類別。

以下列指令呼叫 AS400ToolboxInstaller 類別，指定適當的安裝、解除安裝或 比較項:

```
java utilities.AS400ToolboxInstaller          [options]
```

-source選項表示可以找到 IBM Toolbox for Java 類別的地方，-target表示 要在從屬站中儲存 IBM Toolbox for Java 類別的地方。

您也可以使用選項來安裝完整的工具箱，或調整某些功能»例如， 若要安裝或更新工作站上 Toolbox for Java 存取套件 (jt400.jar) 中的類別， 請使用下列指令：

```
java utilities.AS400ToolboxInstaller -install -package ACCESS -source
myAS400     -target c:\toolbox
```

以上範例假設 c:\toolbox 是包含 Toolbox for Java jar 檔的目錄。

在您的程式中內含 AS400ToolboxInstaller 類別

AS400ToolboxInstaller 類別提供必要的應用程式介面 (API)，以便自從屬 站中的程式安裝、解除安裝及更新 IBM Toolbox for Java 類別。

您可以使用install()方法， 安裝或更新 IBM Toolbox for Java 類別。若要安裝或更新，請提供來源及目標路徑，以及您的 Java 程式中的類別的資料包名稱。 來源 URL 會指向伺服器中控制檔的位置。從伺服器複製目錄結構到從屬站。

install() 方法僅複製檔案；它會更新 CLASSPATH 環境變數。 如果 install() 方法順利完成，則 Java 程式可呼叫 getClasspathAdditions()方法， 來判斷有哪些項目必須新增至 CLASSPATH 環境變數。

下列範例會說明如何使用 AS400ToolboxInstaller 類別，自稱為 "mySystem" 的伺服器安裝檔案到磁 碟機 d: 的目錄 "jt400" 中，然後如何確定要新增哪些項目到 CLASSPATH 環 境變數中：

```
                    // Install the IBM Toolbox for Java
                    // classes on the client.
    URL sourceURL = new
URL("http://mySystem.myCompany.com/QIBM/ProdData/HTTP/Public/jt400/");

    if (AS400ToolboxInstaller.install(
          "ACCESS",
          "d:\\jt400",
          sourceURL))

    {
                    // If the IBM Toolbox for Java classes were installed
                    // or updated, find out what must be added to the
                    // CLASSPATH environment variable.
      Vector additions = AS400ToolboxInstaller.getClasspathAdditions();

                    // If updates must be made to CLASSPATH
      if (additions.size() > 0)
      {
                    // ... Process each classpath addition
      }
    }

                    // ... Else no updates were needed.
```

使用isInstalled()方法來確定 IBM Toolbox for Java 類別是否已安裝在從屬站中。 使用 isInstalled() 方法，可讓您決定要立即完成安裝，或在更適當的時機安裝它。

install() 方法會同時在從屬站中安裝及更新檔案。Java 程式可以呼叫isUpdateNeeded()方法，以確定在呼叫 install() 之前是否有必要更新。

您可以使用unInstall()方法，從從屬站中除去 IBM Toolbox for Java

類別。解除安裝方法僅會除去檔案；不會變更 CLASSPATH 環境變數。 呼叫 getClasspathRemovals() 方法，來判斷可從 CLASSPATH 環境變數中除去哪些項目。

若需有關如何自從屬工作站中的程式安裝及更新 the AS400ToolboxInstaller 類別的其餘範例， 請參照 Install/Upda範例。

AS400ToolboxJarMaker

當 JAR 檔案格式的設計目的在加快 Java 程式檔的下載速度時，AS400ToolboxJarMaker 類別會使用它的能力，從較大的檔案建立一個較小的 JAR 檔，產生一個能更快下載的 IBM Toolbox for Java JAR 檔。

同時，AS400ToolboxJarMaker 類別可以替您解壓縮 JAR 檔，以便能存取個別的內容檔以進行基本使用。

AS400ToolboxJarMaker 的彈性

所有 AS400ToolboxJarMaker 功能均是透過 JarMaker 類別及 AS400ToolboxJarMaker 次類別來執行：

- 同屬 JarMaker 工具可在任何 JAR 或 Zip 檔中運作；它會分割 jar 檔，或除去不使用的類別以減少 jar 檔的大小。
- AS400ToolboxJarMaker 會自訂及擴充 JarMaker 功能，以便更易於使用 IBM Toolbox for Java JAR 檔。

依據您的需求，您可以從自己的 Java 程式或指令行呼叫 AS400ToolboxJarMaker 方法。
請使用下列語法從指令行呼叫 AS400ToolboxJarMaker：

```
java utilities.JarMaker [options]
```

其中

- 選項 = 一或多個可用選項

欲取得可在指令行提示上執行的選項的完整集，請參閱下列：

- JarMaker 基礎類別的選項
- AS00ToolboxJarMaker 子類別的擴充選項

使用 AS400ToolboxJarMaker

解壓縮 JAR 檔

假設您僅想要解壓縮 JAR 檔案內的一個檔案集。AS400ToolboxJarMaker 可讓您將檔案展開到下列其中一項：

- 現行目錄 (extract(jarFile))
- 另一個目錄 (extract(jarFile, outputDirectory))

例如，透過下列程式，您將從 jt400.jar 取出 AS400.class 及其所有相依類別：

```
java utilities.AS400ToolboxJarMaker -source jt400.jar
    -extract outputDir
```

```
-requiredFile com/ibm/as400/access/AS400.class
```

將單一 JAR 檔分割成多個、較小的 JAR 檔

假設您想要依照您對 JAR 檔大小上限的偏好，將一個大型的 JAR 檔分割成較小的 JAR 檔。AS400ToolboxJarMaker 會相對地提供 split(jarFile, splitSize) 功能。

在下列程式碼中，jt400.jar 會分割成一組較小的 JAR 檔，每個檔都不大於 300K：

```
java utilities.AS400ToolboxJarMaker -split 300
```

從 JAR 檔除去不使用的檔案　使用 AS400ToolboxJarMaker，　您可以只選取讓應用程式執行所必須的 IBM Toolbox for Java 元件、　語言及 CCSID，以排除應用程式不需要的 IBM Toolbox for Java 檔案。AS400ToolboxJarMaker 也提供您一個選項，讓您可以併入或排除您所選取元件相關的 JavaBean 檔。

例如，下列指令會建立一個 JAR 檔，　其中只包含 IBM Toolbox for Java 類別要使 IBM Toolbox for Java 之 CommandCall 及 ProgramCall 元件產生作用所需的項目：

```
java utilities.AS400ToolboxJarMaker -component CommandCall,ProgramCall
```

此外，如果不需要在 Unicode 與雙位元組字 集 (DBCS) 轉換表機傳換字串，您可以使用 -ccsid 選項略過不必要的換表，以建立更小的 400K 位元組 JAR 檔：

```
java utilities.AS400ToolboxJarMaker -component CommandCall,ProgramCall -
ccsid 61952
```

註：轉換類別不會與程式呼叫類別一同併入。　在併入程式呼叫類別時，程式所使用的轉換類別也必須經由 -ccsid 選項明確地併入。

IBM Toolbox for Java 支援的元件

下列是當您呼叫 AS400ToolboxJarMaker 工具時可以指定的元件 ID。

- 第一欄列出元件的一般名稱。
- 第二欄列出您在使用 -component 選項標示時，應該指定的關鍵字。
- 第三欄列出您應該在 setComponents() 及 getComponents() 中指定的整數值。

元件	關鍵字	常數
伺服器物件	AS400	AS400ToolboxJarMaker.AS400
指令呼叫	CommandCall	AS400ToolboxJarMaker.COMMAND_CALL
連線儲存區	ConnectionPool	AS400ToolboxJarMaker.CONNECTION_POOL
資料區	DataArea	AS400ToolboxJarMaker.DATA_AREA
資料說明與轉換	DataDescription	AS400ToolboxJarMaker.DATA_DESCRIPTION
資料佇列	DataQueue	AS400ToolboxJarMaker.DATA_QUEUE
數位憑證	DigitalCertificate	AS400ToolboxJarMaker.DIGITAL_CERTIFICATE
FTP	FTP	AS400ToolboxJarMaker.FTP
整合檔案系統	IntegratedFileSystem	AS400ToolboxJarMaker.INTEGRATED_FILE_SYSTEM
JAAS	JAAS	AS400ToolboxJarMaker.JAAS
Java 應用程式呼叫	JavaApplicationCall	AS400ToolboxJarMaker.JAVA_APPLICATION_CALL
JDBC	JDBC	AS400ToolboxJarMaker.JDBC
工具及工具佇列	工作	AS400ToolboxJarMaker.JOB
訊息及訊息佇列	訊息	AS400ToolboxJarMaker.MESSAGE
數值資料類型	NumericDataTypes	AS400ToolboxJarMaker.NUMERIC_DATA_TYPES
NetServer	NetServer	AS400ToolboxJarMaker.NETSERVER
網路列印	列印	AS400ToolboxJarMaker.PRINT
程式呼叫	ProgramCall	AS400ToolboxJarMaker.PROGRAM_CALL
記錄層次存取	RecordLevelAccess	AS400ToolboxJarMaker.RECORD_LEVEL_ACCESS
安全伺服器	SecureAS400	AS400ToolboxJarMaker.SECURE_AS400
服務程式呼叫	ServiceProgramCall	AS400ToolboxJarMaker.SERVICE_PROGRAM_CALL
系統狀態	SystemStatus	AS400ToolboxJarMaker.SYSTEM_STATUS
系統值	SystemValue	AS400ToolboxJarMaker.SYSTEM_VALUE
追蹤及記載	Trace	AS400ToolboxJarMaker.TRACE
使用者和群組	User	AS400ToolboxJarMaker.USER
使用者空間	UserSpace	AS400ToolboxJarMaker.USER_SPACE
目視伺服器物件	AS400Visual	AS400ToolboxJarMaker.AS400_VISUAL
目視指令呼叫	CommandCallVisual	AS400ToolboxJarMaker.COMMAND_CALL_VISUAL
目視資料佇列	DataQueueVisual	AS400ToolboxJarMaker.DATA_QUEUE_VISUAL
目視整合檔案系統	IntegratedFileSystemVisual	AS400ToolboxJarMaker.INTEGRATED_FILE_SYSTEM_VISUAL

目視 Java 應用程式呼叫	JavaApplicationCallVisual	AS400ToolboxJarMaker.JAVA_APPLICATION_CALL_VISUAL
目視 JDBC	JDBCVisual	AS400ToolboxJarMaker.JDBC_VISUAL
目視工作及工作佇列	JobVisual	AS400ToolboxJarMaker.JOB_VISUAL
目視訊息及訊息佇列	MessageVisual	AS400ToolboxJarMaker.MESSAGE_VISUAL
目視網路列印	PrintVisual	AS400ToolboxJarMaker.PRINT_VISUAL
目視程式呼叫	ProgramCallVisual	AS400ToolboxJarMaker.PROGRAM_CALL_VISUAL
目視記錄層次存取	RecordLevelAccessVisual	AS400ToolboxJarMaker.RECORD_LEVEL_ACCESS_VISUAL
目視使用者與群組	UserVisual	AS400ToolboxJarMaker.USER_VISUAL

IBM Toolbox for Java 支援的 CCSID 及編碼值

有一份轉換表隨 IBM Toolbox for Java 一起出貨且根據 CCSID 命名。 當 IBM Toolbox for Java 類別 (例如 CharConverter) 在轉換傳送到/自 iSeries 或 AS/400e 伺服器的資料時, 會在內部使用這些表格。例如, CCSID 10 的轉換表是在檔案com/ibm/as400/access/ ConvTable1027 .class 之中。 下列 CCSID 的轉換表內含在 IBM Toolbox for Java jar 檔中；其它編碼要使用 JDK 才受支援。
在執行期間不再使用伺服器上的中央伺服器來下載表格。任何轉換表或 JDK 編碼找不到的指定 CCSID 將會導致異常發生。其中有些表格對於內含在您 JDK 中的表格而言可能是多餘的。IBM Toolbox for Java 目前支援下列 122 個不同的 iSeries 及 AS/400e CCSID。

關於 CCSID 的額外資訊, 包括 iSeries 及 AS/400e 伺服器可識別的完整清單, 請參閱「資訊中心」中的全球化主題 。

在 IBM Toolbox for Java 中受支援的 CCSID

CCSID	格式	說明
37	單位元組 EBCDIC	美國與其它地區
273	單位元組 EBCDIC	澳洲、德國
277	單位元組 EBCDIC	丹麥、挪威
278	單位元組 EBCDIC	芬蘭、瑞典
280	單位元組 EBCDIC	義大利
284	單位元組 EBCDIC	西班牙、拉丁美洲
285	單位元組 EBCDIC	英國
290	單位元組 EBCDIC	日文片假名 (限單位元組)
297	單位元組 EBCDIC	法國
300	雙位元組 EBCDIC	日文圖形 (16684 的子集)
367	ASCII/ISO/Windows	ASCII (ANSI X3.4 標準)
420	單位元組 EBCDIC (雙向)	阿拉伯文 EBCDIC ST4
423	單位元組 EBCDIC	希臘文 (用於相容性；請參閱 875)
424	單位元組 EBCDIC (雙向)	希伯來文 EBCDIC ST4
437	ASCII/ISO/Windows	ASCII (USA PC 資料)
500	單位元組 EBCDIC	拉丁語-1 (MNCS)
720	ASCII/ISO/Windows	阿拉伯文 (MS-DOS)
737	ASCII/ISO/Windows	希臘文 (MS-DOS)
775	ASCII/ISO/Windows	波羅的海語系 (MS-DOS)
813	ASCII/ISO/Windows	ISO 8859-7 (希臘文/拉丁文)
819	ASCII/ISO/Windows	ISO 8859-1 (拉丁文-1)

833	單位元組 EBCDIC	韓文 (限單位元組)
834	雙位元組 EBCDIC	韓文圖形 (4930 的子集)
835	雙位元組 EBCDIC	繁體中文圖形
836	單位元組 EBCDIC	簡體中文 (限單位元組)
837	雙位元組 EBCDIC	簡體中文圖形
838	單位元組 EBCDIC	泰文
850	ASCII/ISO/Windows	拉丁文-1
851	ASCII/ISO/Windows	希臘文
852	ASCII/ISO/Windows	拉丁文-2
855	ASCII/ISO/Windows	斯拉夫文
857	ASCII/ISO/Windows	土耳其文
860	ASCII/ISO/Windows	葡萄牙文
861	ASCII/ISO/Windows	冰島
862	ASCII/ISO/Windows (雙向)	希伯來文 ASCII ST4
863	ASCII/ISO/Windows	加拿大
864	ASCII/ISO/Windows (雙向)	阿拉伯文 ASCII ST5
865	ASCII/ISO/Windows	丹麥/挪威
866	ASCII/ISO/Windows	斯拉夫文/俄文
869	ASCII/ISO/Windows	希臘文
870	單位元組 EBCDIC	拉丁文-2
871	單位元組 EBCDIC	冰島
874	ASCII/ISO/Windows	泰文 (9066 的子集)
875	單位元組 EBCDIC	希臘文
878	ASCII/ISO/Windows	俄文
880	單位元組 EBCDIC	多種語言斯拉夫文 (用於相容性；請參閱 1025)
912	ASCII/ISO/Windows	ISO 8859-2 (拉丁文-2)
914	ASCII/ISO/Windows	ISO 8859-4 (拉丁文-4)
915	ASCII/ISO/Windows	ISO 8859-5 (八位元的斯拉夫文)
916	ASCII/ISO/Windows (雙向)	ISO 8859-8 (希伯來文) ST5
920	ASCII/ISO/Windows	ISO 8859-9 (拉丁文-5)
921	ASCII/ISO/Windows	ISO 8859-13 (八位元的波羅的海語系)
922	ASCII/ISO/Windows	愛沙尼亞 ISO-8
923	ASCII/ISO/Windows	ISO 8859-15 (拉丁文-9)
930	混合位元組 EBCDIC	日文 (5026 的子集)
933	混合位元組 EBCDIC	韓文 (1364 的子集)

935	混合位元組 EBCDIC	簡體中文 (1388 的子集)
937	混合位元組 EBCDIC	繁體中文
939	混合位元組 EBCDIC	日文 (5035 的子集)
1025	單位元組 EBCDIC	斯拉夫文
1026	單位元組 EBCDIC	土耳其文
1027	單位元組 EBCDIC	日文拉丁文 (限單位元組)
1046	ASCII/ISO/Windows (雙向)	Windows 阿拉伯文 ST5
1089	ASCII/ISO/Windows (雙向)	ISO 8859-6 (阿拉伯文) ST5
1112	單位元組 EBCDIC	波羅的海語系多種語言
1122	單位元組 EBCDIC	愛沙尼亞
1123	單位元組 EBCDIC	烏克蘭
1125	ASCII/ISO/Windows	烏克蘭
1129	ASCII/ISO/Windows	越南文
1130	單位元組 EBCDIC	越南文
1131	ASCII/ISO/Windows	白俄羅斯
1132	單位元組 EBCDIC	寮國
1140	單位元組 EBCDIC	美國與其訰地區 (歐元支援)
1141	單位元組 EBCDIC	澳洲、德國 (歐元支援)
1142	單位元組 EBCDIC	丹麥、挪威 (歐元支援)
1143	單位元組 EBCDIC	芬蘭、瑞典 (歐元支援)
1144	單位元組 EBCDIC	義大利 (歐元支援)
1145	單位元組 EBCDIC	西班牙、拉丁美洲 (歐元支援)
1146	單位元組 EBCDIC	英國 (歐元支援)
1147	單位元組 EBCDIC	法國 (歐元支援)
1148	單位元組 EBCDIC	拉丁-1 (MNCS) (歐元支援)
1149	單位元組 EBCDIC	冰島 (歐元支援)
1200	Unicode	Unicode UCS-2 (小位元組排序法)
1250	ASCII/ISO/Windows	Windows 拉丁文-2
1251	ASCII/ISO/Windows	Windows 斯拉夫文
1252	ASCII/ISO/Windows	Windows 拉丁文-1
1253	ASCII/ISO/Windows	Windows 希臘文
1254	ASCII/ISO/Windows	Windows 土耳其
1255	ASCII/ISO/Windows (雙向)	Windows 希伯來文 ST5
1256	ASCII/ISO/Windows (雙向)	Windows 阿拉伯文 ST5

1257	ASCII/ISO/Windows	Windows 波羅的海語系
1258	ASCII/ISO/Windows	Windows 越南
1364	混合位元組 EBCDIC	日文
1388	混合位元組 EBCDIC	簡體中文
1399	混合位元組 EBCDIC	日文 (在 V4R5 或更新的版本中)
4396	雙位元組 EBCDIC	日文 (300 的子集)
4930	雙位元組 EBCDIC	韓文
4931	雙位元組 EBCDIC	繁體中文 (835 的子集)
4933	雙位元組 EBCDIC	簡體中文 GBK 圖形
4948	ASCII/ISO/Windows	拉丁文-2 (852 的子集)
4951	ASCII/ISO/Windows	斯拉夫文 (855 的子集)
5026	混合位元組 EBCDIC	日文
5035	混合位元組 EBCDIC	日文
5123	單位元組 EBCDIC	日文 (限單位元組、歐元支援)
5351	ASCII/ISO/Windows (雙向)	Windows 希伯來文 (歐元支援) ST5
8492	雙位元組 EBCDIC	日文 (300 的子集)
8612	單位元組 EBCDIC	阿拉伯文 EBCDIC ST5
9026	雙位元組 EBCDIC	韓文 (834 的子集)
9029	雙位元組 EBCDIC	簡體中文 (4933 的子集)
9066	ASCII/ISO/Windows	泰文 (延伸 SBCS)
12588	雙位元組 EBCDIC	日文 (300 的子集)
13122	雙位元組 EBCDIC	韓文 (834 的子集)
16684	雙位元組 EBCDIC	日文 (可於 V4R5 中取得)
17218	雙位元組 EBCDIC	韓文 (834 的子集)
12708	單位元組 EBCDIC	阿拉伯文 EBCDIC ST7
13488	Unicode	Unicode UCS-2 (大位元組排序法)
28709	單位元組 EBCDIC	繁體中文 (限單位元組)
61952	Unicode	iSeries 及 AS/400e Unicode (主要用於 IFS 中)
62211	單位元組 EBCDIC	希伯來文 EBCDIC ST5
62224	單位元組 EBCDIC	阿拉伯文 EBCDIC ST6
62235	單位元組 EBCDIC	希伯來文 EBCDIC ST6
62245	單位元組 EBCDIC	希伯來文 EBCDIC ST10

CommandPrompter 類別

CommandPrompter 類別會提示給定指令的參數。CommandPrompter 提供類似 iSeries CL 指 令提示 (按 F4) 及與管理中心指令提示相同的功能。

若要使用 CommandPrompter，伺服器必須執行 OS/400 V4R4 或更新的版本。詳細資訊，請參閱 iSeries iSeries 領航員資訊 APAR，及檢視「需要的修正程式」取得圖形指令提示器支援。

使用 CommandPrompter 亦需要您在 CLASSPATH 中具有下列 jar 檔：

- jt400.jar
- jui400.jar
- util400.jar
- x4j400.jar
- jhall.jar

除了 jhall.jar 以外，所有 jar 檔都內含在 Toolbox for Java 之中。 關於 Toolbox for Java jar 檔的詳細資訊，請參閱 Jar 檔。關於下載 jhall.jar 的詳細資訊，請參閱 Sun JavaHelp(TM) 網站。

若要建構 CommandPrompter 物件，您必須傳送啟動提示器的上層框架參數、 提示指令所在的 AS400 物件、以及指令字串給它。指令字串可以是指令名稱、完整指令字串、或部份指令名稱，例如* 。

CommandPrompter 顯示器是一個使用者在返回上層框架之前，必須先關閉的模組對話框。CommandPrompter 會處理提示期間出現的任何錯誤。

範例： 將 CommandPrompter 與 CommandCall 及 AS400Message 類別一起使用以提示及執行指令

RunJavaApplication

RunJavaApplication 和 VRunJavaApplication 兩個類別是公用程式，可用來在 iSeries JVM 上執行 Java 程式。不像 JavaApplicationCall 和 VJavaApplicationCall 兩個類別，必須從您的 Java 程式呼叫，RunJavaApplication 及 VRunJavaApplication 是完整的程式。

RunJavaApplication 類別為指令行公用程式。它可以設定 Java 程式的環境（例如：CLASSPATH 及內容）。您可以指定 Java 程式的名稱及它的參數，然後您可以啟動程式。一旦啟動，您可以傳送「輸入」至 Java 程式，它可以透過標準的輸入裝置接收。Java 程式將輸出寫入標準輸出及標準錯誤。

VRunJavaApplication 公用程式具有相同的功能。不同的是 VJavaApplicationCall 使用圖形式使用者介面時，JavaApplicationCall 為指令行介面。

JPing 類別

JPing 類別為一指令行公用程式， 可讓您查詢伺服器以找出正在執行的服務為何，及提供服務的埠有哪些。若要從 Java 應用程式內部進行伺服器的查詢，請使用 JPing 類別。

請參閱 JPing javadoc 以取得在 Java 應用程式內使用 JPing 的其它相關資訊。

使用下列語法，由指令行呼叫 JPing：

```
java utilities.JPing System [options]
```

其中：

- System = 您要查詢的 iSeries 伺服器
- [options] = 一或多個可用選項

選項

您可使用下列選項之一或其中數個項目。如果選項有縮寫，縮寫會列於括弧內。

-help (-h or -?)
: 顯示說明本文。

-service (-s OS/400_Service)
: 指定一項要進行連通測試 (ping) 的特定服務。 預設動作為對所有服務進行連通測試 (ping)。
您可使用此選項來指定下列其中一項服務：as-file、as-netprt、as-rmtcmd、as-dtaq、as-database、as-ddm、as-central 及 as-signon。

-ssl
: 指定是否要對 ssl 埠進行連通測試 (ping)。 預設動作為不要對 ssl 埠進行連通測試 (ping)。

-timeout (-t)
: 指定逾時期間 (以毫秒為單位)。預設設定為 20000，或 20 秒。

範例：在指令行使用 JPing

例如，使用下列指令對 as-dtaq 服務 (包括 ssl 埠) 進行 連通測試 (ping)，逾時時間為 5 秒：

```
java utilities.JPing myServer -s as-dtaq -ssl -t 5000
```

Vaccess 類別

IBM Toolbox for Java 在 vaccess 套件中提供一組圖形使用者介面 (GUI)。這些類別會使用存取類別來擷取資料，並呈現資料供使用者查看。

使用 IBM Toolbox for Java vaccess 類別的 Java 程式需要 Swing 1.1。您可以執行 Java 2，或從 Sun Microsystems, Inc. 下載 Swing 1.1 來取得 Swing 1.1。 過去，IBM Toolbox for Java 需要使用 Swing 1.0.3，V4R5 是第一個支援 Swing 1.1 的版次。 以前若要移到 Swing 1.1，要進行一些程式設計變更；因此，您可能也要進行一些程式設計變更。請參閱 Sun Microsystems, Inc. 頁，取得 Swing 的其餘相關資訊。

關於 IBM Toolbox for Java GUI 類別、「存取」類別與 Java Swing 之間的關係的詳細資訊， 請參閱 Vaccess 類別圖解。

您可以使用 AS400 窗格 類別，來顯示 iSeries 資料

API 可用來存取下列 iSeries 資源及其工具：

- 指令呼叫
- 資料佇列
- 錯誤事件 *
- 整合檔案系統
- JavaApplicationCall
- JDBC
- 工作 *
- 訊息 *
- 許可權
- 列印 * 包括已排存的檔案檢視器
- ProgramCall 及 ProgramParameter
- 記錄層次存取
- 資源清單
- 系統狀態
- 系統值
- 使用者和群組

備註： AS400 窗格會與其它 vaccess 類別一起 使用 (請參閱上面以星號標示的項目)，以呈現 iSeries 資源並容許操作它們。

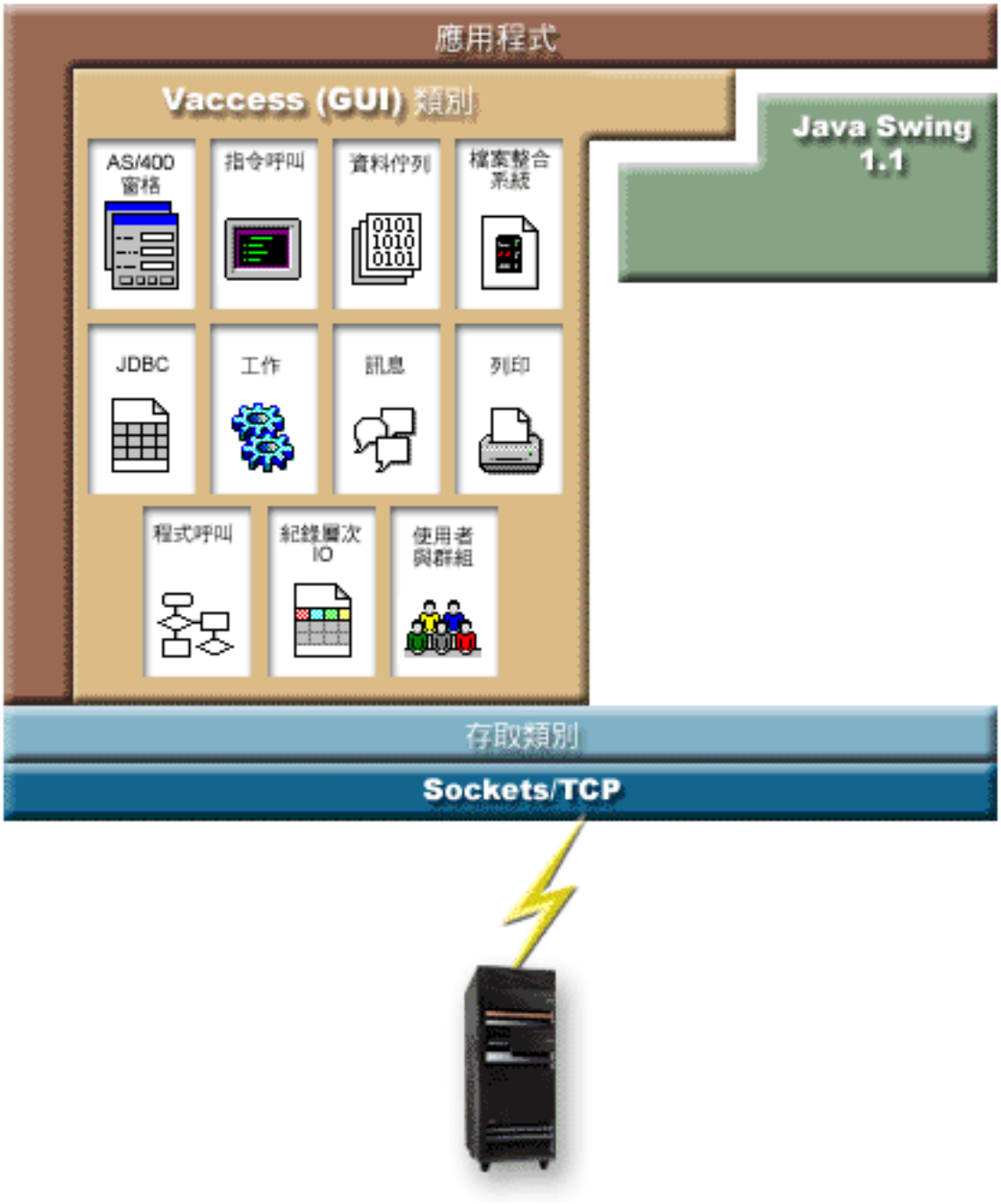
以 IBM Toolbox for Java 圖形使用者介面元件進行程式設計時，請使用錯誤事件類別，向使用者報告及處理錯誤事件。

請參閱 存取類別，以取得存取 iSeries 資料的其餘相關資訊。

Vaccess 類別

IBM Toolbox for Java 在 vaccess 套件中提供圖形式使用者介面 (GUI) 類別，可用來擷取及顯示伺服器資料，並在某些情況下操作該伺服器資料。 這些類別使用 Java Swing 1.1 組織架構。圖 1 顯示這些類別之間的關係：

圖 1：Vaccess 類別



AS400Panes

AS400Panes 是 vaccess 套件的元件，它們會在 GUI 中呈現及容許操作一個或多個伺服器資源。每一個伺服器資源的行為會隨著資源類型而有所

所有窗格會擴充「Java 元件」類別。因此，它們可新增到任何「AWT 頁框」、「視窗」或「容器」中。

下列是可用的 AS400Panes：

- AS400DetailsPane會在表格中呈現伺服器資源清單， 其中的每一列則顯示單一資源的各種詳細資訊。表格容許選擇一個或多個資源。
- AS400ExplorerPane 結合了 AS400TreePane 與 AS400DetailsPane， 以便在詳細畫面中在呈現樹狀結構中所選取的資源。
- AS400JDBCDataSourcePane 會呈現 AS400JDBCDataSource 物件的內容值。
- AS400ListPane 會呈現伺服器資源清單，並容許選擇一或多個資源。
- AS400TreePane 會以樹狀階層結構呈現伺服器資源，並容許選擇一或多個資源。

伺服器資源

在圖形使用者介面中，伺服器資源是以圖示及文字來表示。 伺服器資源是透過階層關係來定義，因此資源可能有一個上層，
以及零或多個子項。這些是預先定義的關係，而且可以用來指定哪些資源要顯示在 AS400Pane 中。例如，VJobList 是零或多個 VJobs 的上層，此B這種階層關係在 AS/400Pane 中會以圖形方式來表示。

IBM Toolbox for Java 可以存取下列伺服器資源：

- VIFSDirectory代表整合檔案系統中的目錄
- VJob 和 VJobList 分別代表工作或工作清單
- VMessageList 和 VMessageQueue 分別代表從 CommandCall 或 ProgramCall 或訊息佇列所傳回的訊息清單
- VPrinter、VPrinters 和 VPrinterOutput分別代表印表機、 印表機清單或排存檔的清單
- VUserList 代表使用者清單

所有資源都是VNode 介面的實作。

設定根物件

若要指定哪些伺服器資源會呈現在 AS400Pane 中，請使用建構元或 setRoot() 方法設定根。根物件會依據窗格來定義不同方式使用的最上層物件

- AS400ListPane 在其清單中呈現根的所有子項
- AS400DetailsPane在其表格中呈現根的所有子項
- AS400TreePane 使用根作為其樹狀結構的根
- AS400ExplorerPane 使用根作為其樹狀結構的根

窗格與根物件可做任意組合。

下列範例會建立一個 AS400DetailsPane，來呈現系統中所定義的使用者清單：

```
// Create the server resource
```

```
 // representing a list of users.
 // Assume that "system" is an AS400
 // object created and initialized
 // elsewhere.
 VUserList
userList = new VUserList (system);

 // Create the AS400DetailsPane object
 // and set its root to be the user
 // list.

AS400DetailsPane detailsPane = new AS400DetailsPane ();

detailsPane.setRoot (userList);

 // Add the details pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (detailsPane);
```

載入內容

當建立 AS400Pane 物件與伺服器資源物件時，會將它們起始設定為預設狀態。在建立期間，不會載入構成窗格內容的相關資訊。

若要載入內容，應用程式必須以明確方式呼叫 load() 方法。在大多數情況下， 這會起始與伺服器的通信，來收集相關資訊。因為有時會需要時間來收集此資訊，
所以應用程式可以精確地控制收集時間。例如，您可以：

- 將窗格新增到頁框之前先載入內容。頁框不會出現，直到載入所有資訊為止。
- 將窗格新增到頁框且顯示該頁框之後再載入內容。頁框會出現，但它不會含有許多資訊。「等待游標」會出現且在載入後該資訊就會填入。

下面範例會在將明細窗格的內容新增到頁框之前，先載入該內容：

```
 // Load the contents of the details
 // pane. Assume that the detailsPane
 // was created and initialized
 // elsewhere.

detailsPane.load ();

 // Add the details pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (detailsPane);
```

動作與特性窗格

在執行時間，使用者可以選取任何伺服器資源中的蹦現功能表。蹦現功能表會呈現可供資源使用的相關動作的清單。當使用者選取動作時，能夠執行該動作。每一資源均有一個不同的已定義動作。

在某些情況下，蹦現功能表也會呈現一個容許使用者檢視特性窗格的項目。特性窗格會顯示關於資源的不同明細，且可能容許使用者變更那些明細

該應用程式可以在窗格中使用 setAllowActions() 方法，來控制是否可以使用動作與內容窗格。

模型

AS400Panes 是使用模型-概略表-控制器參照範例來實作， 其中的資料與使用者介面被分成不同的類別。AS400Panes 整合了 IBM Toolbox for Java 模型與 Java GUI 元件。這些模型會管理伺服器資源，而 vaccess 元件則會以圖形方式來顯示它們及處理使用者互動。

AS400Panes 會提供足夠的功能來符合大部份的需求。不過， 如果應用程式需要更進一步控制 JFC 元件，則應用程式可以直接存取伺服器模型與不同的 vaccess 元件的自訂整合。

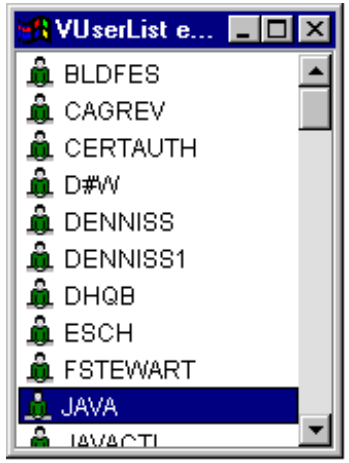
下列是可用的模型：

- AS400ListModel 會將 JFC ListModel 介面實作為伺服器資源的清單。這可與 JFC JList 物件搭配使用。
- AS400DetailsModel 將 JFC TableModel 介面實作為伺服器資源的表格， 其中的每一列皆包含單一資源的各種相關明細。這可與 JFC JTable 物件搭配使用。
- AS400TreeModel 會將 JFC TreeModel 介面實作為伺服器資源的樹狀階層。 這可與 JFC JTree物件搭配使用。

範例

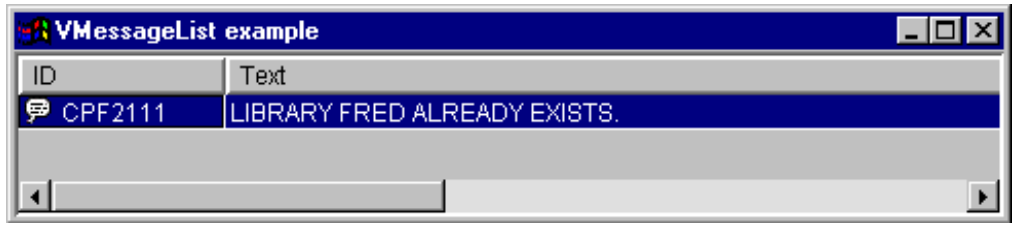
- 使用AS400ListPane 與 VUserList 物件，在系統上呈現使用者清單。圖 1 顯示完成的產品：

  圖 1：搭配使用 AS400ListPane 和 VUserList 物件

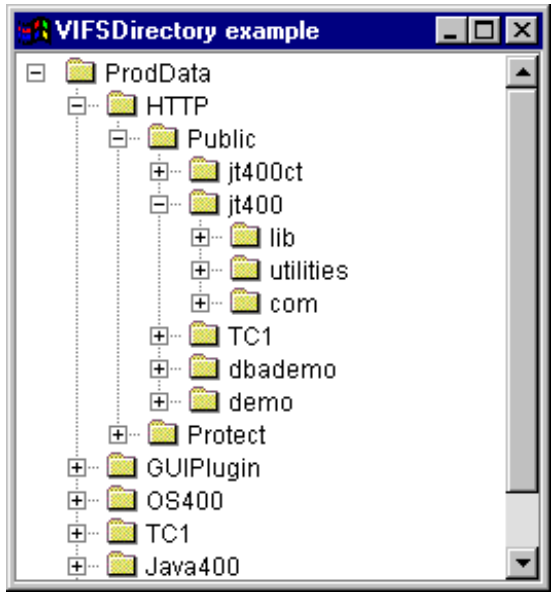
- 使用 VMessageList 物件與 AS400DetailsPane， 來呈現指令呼叫所產生的訊息清單。圖 2 顯示完成的產品：

  圖 2：搭配使用 AS400DetailsPane 和 VMessageList 物件



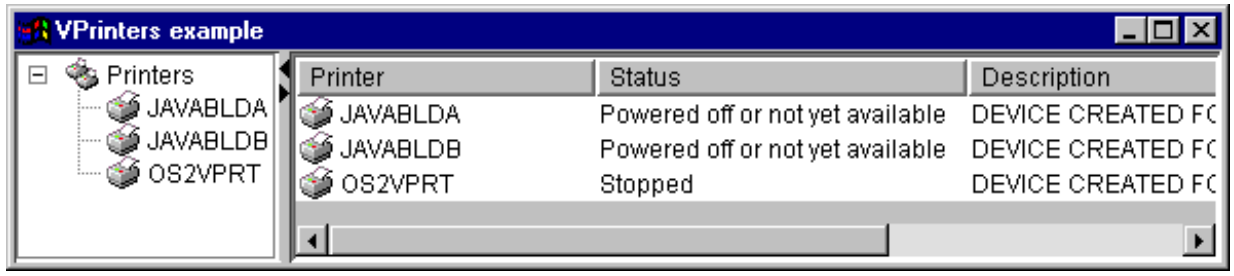
- 使用具有 VIFSDirectory 物件的 AS400TreePane， 呈現整合檔案系統目錄階層。圖 3 顯示完成的產品：

  圖 3：搭配使用 AS400TreePane 和 VIFSDirectory 物件



- 搭配使用 AS400ExplorerPane 和 VPrinters 物件來呈現列印資源。圖 4 顯示完成的產品：

  圖 4：搭配使用 AS400ExplorerPane 和 VPrinters 物件



Vaccess 指令呼叫

指令呼叫 vaccess (GUI) 元件可讓 Java 程式呈現一個按鈕或功能表項目， 用來呼叫非交談式的伺服器指令。

CommandCallButton 物件代表在按下時會呼叫伺服器指令的按鈕。CommandCallButton 類別會擴充 Java Foundation Classes (JFC) JButton 類別，使所有按鈕均有一致的外觀及行為。

同樣地，CommandCallMenuItem 物件代表在選取時，會呼叫伺服器指令的功能表項目。CommandCallMenuItem 類別會擴充 JFC JMenuItem 類別，使所有功能表項目均有一致的外觀與行為。

若要使用指令呼叫圖形使用者介面，請同時設定系統與指令特性。這些特性可使用建構元或透過 setSystem() 及 setCommand() 方法來設定。

下列範例會建立 CommandCallButton。在執行期間， 當按下該按鈕時，它會建立一個叫「FRED」的檔案庫：

```
 // Create the CommandCallButton
 // object. Assume that "system" is
 // an AS400 object created and
 // initialized elsewhere.The button
 // text says "Press Me", and there is
 // no icon.
CommandCallButton button = new CommandCallButton ("Press Me", null, system);

 // Set the command that the button will run.
button.setCommand ("CRTLIB FRED");

 // Add the button to a frame. Assume
 // that "frame" is a JFrame created
 // elsewhere.
frame.getContentPane ().add (button);
```

當執行伺服器指令時，它可能會傳回零或多個伺服器訊息。若要偵測伺服器指令的執行，請使用 addActionCompletedListener() 方法，將 ActionCompletedListener 新增到按鈕或功能表項目中。如此，每當指令執行時， 它會傳送 ActionCompletedEvent 給所有這類接收程式。接收程式可以使用 getMessageList() 方法，來擷取該指令所產生的任何伺服器訊息。

此範例會新增一個 ActionCompletedListener，來處理該指令所產生的所有伺服器訊息：

```
 // Add an ActionCompletedListener that
 // is implemented using an anonymous
 // inner class. This is a convenient
 // way to specify simple event
 // listeners.

button.addActionCompletedListener (new ActionCompletedListener ()
     {
public void actionCompleted(ActionCompletedEvent event)
         {
 // Cast the source of the event to a
```

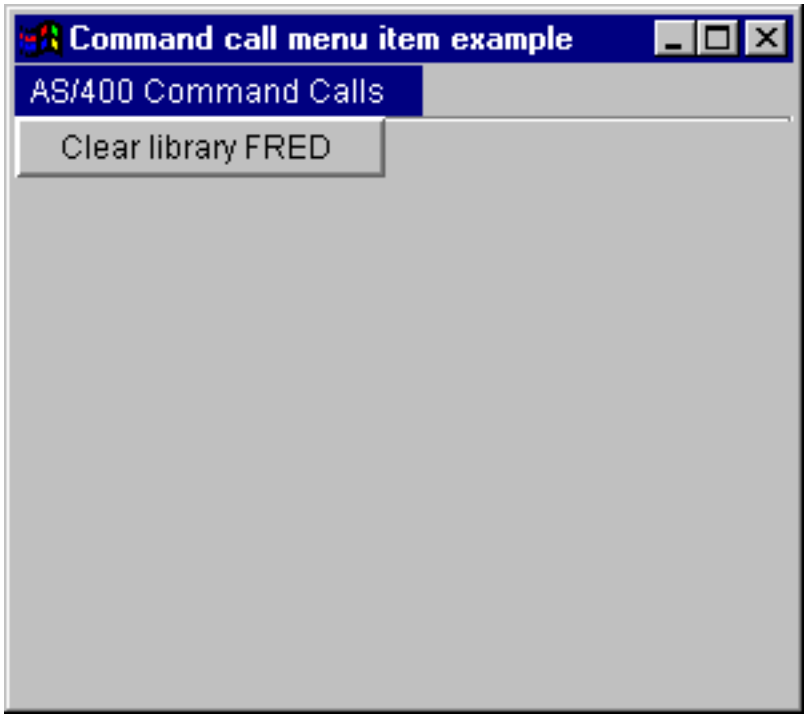
```
 // CommandCallButton。
 CommandCallButton sourceButton =
(CommandCallButton) event.getSource ();

 // Get the list of server messages
 // that the command generated.
 AS400Message[] messageList
= sourceButton.getMessageList ();

 // ... Process the message list.
          }
     });
```

範例

此範例顯示如何在應用程式中使用CommandCallMenuItem 。

圖 1 顯示 CommandCall 圖形式使用者介面元件：

圖 1：CommandCall GUI 元件



Vaccess 資料佇列

資料佇列圖形式元件容許 Java 程式使用任何 Java Foundation Classes (JFC)
圖形式文字元件，來讀取或寫入伺服器資料佇列中。

DataQueueDocument 和 KeyedDataQueueDocument 類別為「JFC 文件」介面的實作。這些類別可直接與任何 JFC
圖形文字元件搭配使用。有數種文字元件（如單一行欄位 (JTextField) 與多行文字區 (JTextArea)）均可在 JFC 中使

資料佇列文件會將文字元件的內容與伺服器資料佇列連結。(文字元件即是用來顯示使用者可選用性編輯的文字的圖形元
Java program 可隨時在文字元件與資料佇列之間讀取及寫入資料。若對資料佇列使用 DataQueueDocument，
對索引資料佇列則使用 KeyedDataQueueDocument。

若要使用 DataQueueDocument，請同時設定系統與路徑特性。這些特性可使用建構元或透過 setSystem() 與 setPath()
方法來設定。接著，DataQueueDocument 物件會被「拉進」文字元件， 通常是使用文字元件建構元或 setDocument()
方法。 KeyedDataQueueDocuments 也是同樣的運作方式。

下面範例會建立一個 DataQueueDocument，其內容會與資料佇列連結：

```
 // Create the DataQueueDocument
 // object. Assume that "system" is
 // an AS400 object created and
 // initialized elsewhere.
 DataQueueDocument dqDocument = new DataQueueDocument (system,
"/QSYS.LIB/MYLIB.LIB/MYQUEUE.DTAQ");

 // Create a text area to present the
 // document.
 JTextArea textArea = new JTextArea (dqDocument);

 // Add the text area to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (textArea);
```

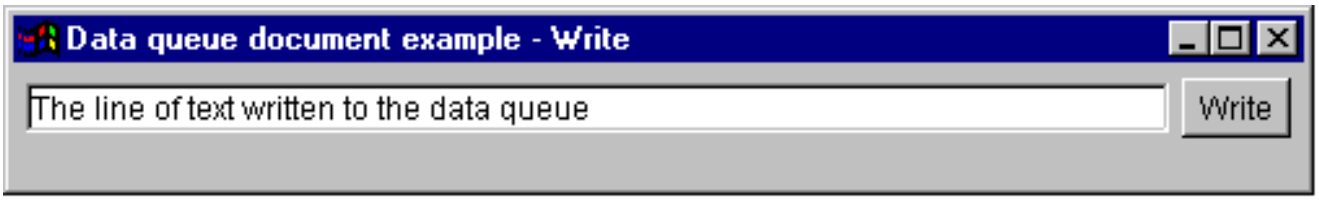
始初，文字元件的內容是空的。使用 read() 或 peek()，以佇列中的下一個登錄填入內容。 使用 write()
將文字元件的內容寫入到資料佇列中。請注意，這些文件只適用於「字串」資料佇列項目。

範例

在應用程式中使用 DataQueueDocument 的範例。

圖 1 顯示在 JTextField 中使用 DataQueueDocument 圖形式使用者介面元件。 其中新增了一個按鈕，提供一個 GUI
介面讓使用者將測試欄位的內容寫入資料佇列中。

圖 1：DataQueueDocument GUI 元件


Data queue document example - Write
The line of text written to the data queue
Write

錯誤事件

在大多數情況中，IBM Toolbox for Java 圖形使用者介面 (GUI) 會發出錯誤事件，而不是擲出 異常。

錯誤事件是內部元件所丟出的異常情況的外層。

您可以提供一錯誤接收器，來處理特殊圖形使用者介面元件所發出的所有錯誤事件。
每當丟出一個異常情況，即會呼叫接收器，它可以提供適當的錯誤報告。依據預設值，當發出錯誤事件時，不會採取任何動作。

IBM Toolbox for Java 提供一個稱為 ErrorDialogAdapter
的圖形使用者介面，每當發生錯誤事件時，它會自動對使用者顯示一個對話框。

範例

下列範例告訴您如何處理錯誤及定義一個簡單錯誤接受器。

範例：經由顯示對話框來處理錯誤事件

下列範例將經由顯示對話框，告訴您如何處理錯誤事件：

```
                    // ... all the setup work to lay out
                    // a graphical user interface
                    // component is done. Now add an
                    // ErrorDialogAdapter as a listener
                    // to the component. This will report
                    // all error events fired by that
                    // component through displaying a
                    // dialog.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (parentFrame);

component.addErrorListener (errorHandler);
```

您可以撰寫一個自訂錯誤接收器，以不同方式處理錯誤。 使用 ErrorListener 介面來完成此工作。

範例：定義錯誤接收器

下列範例將告訴您如何定義一個簡單的錯誤接收器，僅將錯誤列印到 System.out：

```
class MyErrorHandler
implements ErrorListener
{
                    // This method is invoked whenever
                    // an error event is fired.
        public void errorOccurred(ErrorEvent event)
        {
             Exception e = event.getException ();
             System.out.println ("Error: " + e.getMessage ());
        }
}
```

範例：使用錯誤接收器來處理錯誤事件

下列範例將告訴您如何使用此自訂的處理程式，來處理圖形使用者介面元件的錯誤事件：

```
    MyErrorHandler errorHandler = new MyErrorHandler ();
component.addErrorListener (errorHandler);
```

Vaccess 整合檔案系統

整合檔案系統圖形式使用者介面元件容許 Java 程式以 GUI 方式，呈現位於伺服器上整合檔案系統中的目錄及檔案。

可用的元件如下：

- IFSFileDialog會呈現一個對話框，
  以容許使用者經由瀏覽目錄架構來選擇一個目錄並選取一個檔案
- VIFSDirectory是一種資源，它代表AS400Panes 所使用的整合檔案系統的目錄。
- IFSTextFileDocument代表一個要在任何 Java Foundation Classes (JFC) 圖形文字元件中使用的文字檔。

若要使用整合檔案系統圖形使用者介面元件，請同時設定系統與路徑或目錄特性。這些特性可使用建構元或透過 setDirectory() (用於 IFSFileDialog) 或 setSystem() 與 setPath() methods (用於 VIFSDirectory 及 IFSTextFileDocument) 來設定。

您應該設定不同於 "/QSYS.LIB"的路徑，因為這個目錄通常很大，在下載內容時會花很長的時間。

檔案對話框

IFSFileDialog類別是一個對話框，可讓使用者遍訪位於伺服器上整合檔案系統中的目錄並選取檔案。呼叫程式可設定對話框上的按鈕中的文字。此外，呼叫程式可以使用FileFilter物件，讓使用者限制只選擇某些檔案。

如果使用者在對話框中選取檔案，請使用getFileName()方法，來取得所選取檔案的名稱。使用getAbsolutePath()方法可以取得所選取檔案的完整路徑名稱。

下面範例設置一個具有兩個檔案過濾條件的整合檔案系統檔案對話框：

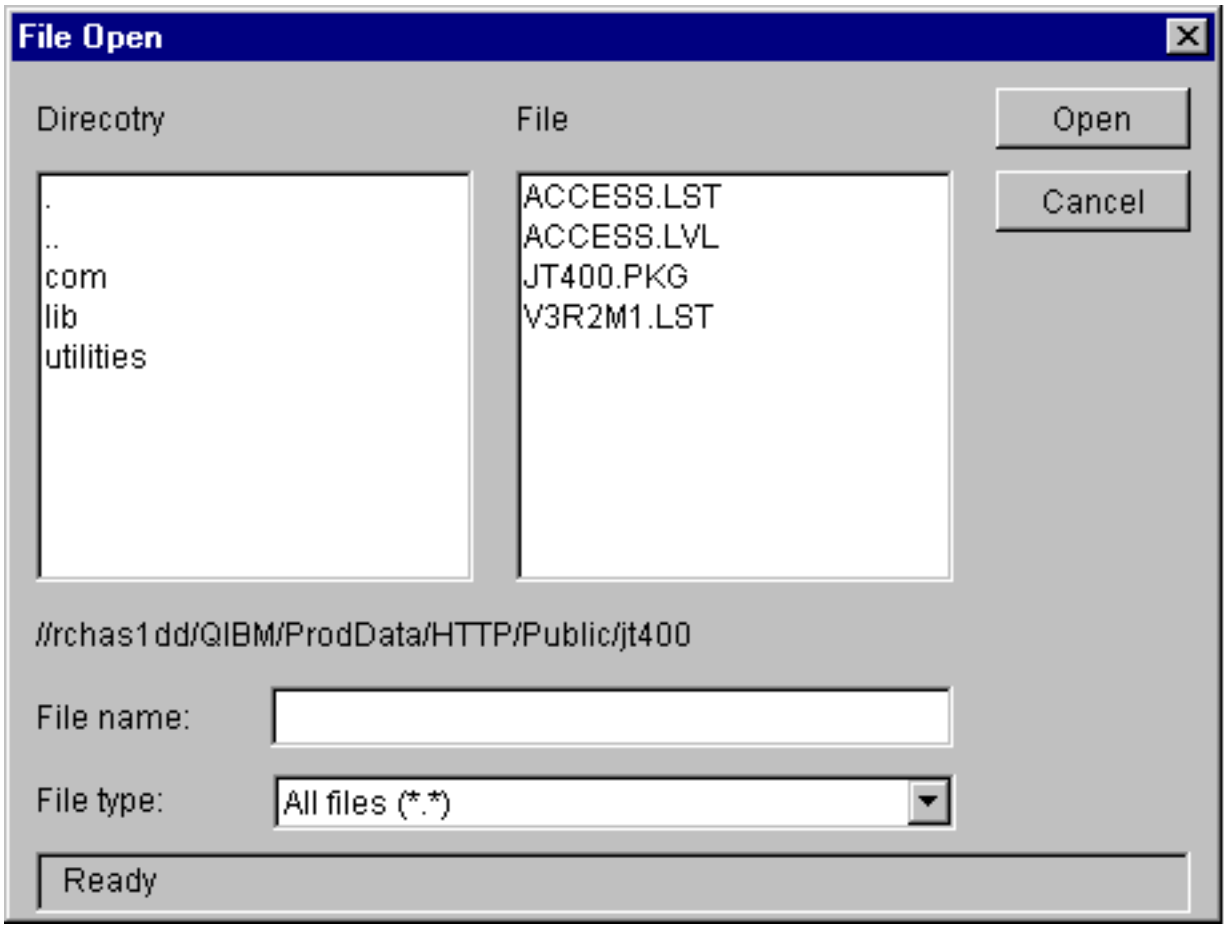
```
// Create a IFSFileDialog object
// setting the text of the title bar.
// Assume that "system" is an AS400
// object and "frame" is a JFrame
// created and initialized elsewhere.
 IFSFileDialog dialog = new IFSFileDialog (frame, "Select a file", system);

// Set a list of filters for the dialog.
// The first filter will be used
// when the dialog is first displayed.
 FileFilter[] filterList = {new FileFilter ("All files (*.*)", "*.*"),
new FileFilter ("HTML files (*.HTML", "*.HTM")};
// Then, set the filters in the dialog.
 dialog.setFileFilter(filterList, 0);

// Set the text on the buttons.
 dialog.setOkButtonText ("Open");
 dialog.setCancelButtonText ("Cancel");

// Show the dialog. If the user
// selected a file by pressing the
// "Open" button, then print the path
// name of the selected file.
 if (dialog.showDialog() == IFSFileDialog.OK)
System.out.println(dialog.getAbsolutePath());
```

範例

呈現IFSFileDialog並列印選擇， 若有的話

圖 1 顯示 IFSFileDialog 圖形式使用者介面元件：

圖 1：IFSFileDialog GUI 元件


File Open
Direcotry
File
Open
Cancel
.
..
com
lib
utilities
ACCESS.LST
ACCESS.LVL
JT400.PKG
V3R2M1.LST
//rchas1dd/QIBM/ProdData/HTTP/Public/jt400
File name:
File type:
All files (*.*)
Ready

AS400Panes 的目錄

AS400Panes 是一些 GUI 元件， 它們會呈現及容許操作一個或多個伺服器資源。IFSDirectory物件是一種資源，它代表 AS400Panes 中使用的整合檔案系統的目錄。 您可以一起使用 AS400Pane 與 VIFSDirectory 物件，來呈現整合檔案系統的多種檢視方式，並容許使用者導覽、操作及選取目錄與檔案。

若要使用 VIFSDirectory，請同時設定系統與路徑特性。 這些特性可使用建構元或透過 setSystem() 與 setPath() 方法來設定。然後，您可以使用建檔元件或 AS400Pane 的 setRoot() 方法， 將 VIFSDirectory 物件插入 AS400Pane，作為根目錄。

VIFSDirectory 含有其它一些有用的內容，有助於定義 AS400Panes 中所呈現的目錄及檔案集。使用 setInclude() 指定要出現目錄、檔案或兩者使用
setPattern()，經由指定檔案名稱必須符合的型樣，對要顯示的項目設定一個過濾條件。您可以使用萬用字元，例如：「*」及「?」在此型樣中。同樣地，使用 setFilter() 可以設定IFSFileFilter物件的過濾字元。

當建立 AS400Pane 物件與 VIFSDirectory 物件時，它們會起始設定為預設狀態。構成根目錄內容的次目錄及檔案尚未載入。若要載入內容，呼叫程式必須在其中一個物件中明確地呼叫 load()方法，起始與伺服器的通信，以便收集目錄的內容。

在執行期間，使用者可以在任何目錄或檔案上按一下滑鼠右鍵，以便顯示環境功能表來針對那些目錄或檔案執行動作。目錄環境功能表可能包括下列項目：

- 建立檔案 - 在目錄中建立檔案。這將給與檔案預設名稱
- 建立目錄 - 以預設名稱建立子目錄
- 更名 - 變更目錄名稱
- 刪除 - 刪除目錄
- 內容 - 顯示如位置、檔案和子目錄數，及修正日期等內容

檔案環境功能表可能包括下列項目：

- 編輯 - 在不同視窗中編輯文字檔
- 檢視 - 在不同視窗中檢視文字檔
- 更名 - 變更檔案名稱
- 刪除 - 刪除檔案
- 內容 - 顯示如位置、大小、修改日期及屬性等內容

使用者僅能讀取或寫入它們有權使用的目錄及檔案。此外，呼叫程式可在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

下面範例會建立一個 VIFSDirectory，並在 AS400ExplorerPane 中呈現它：

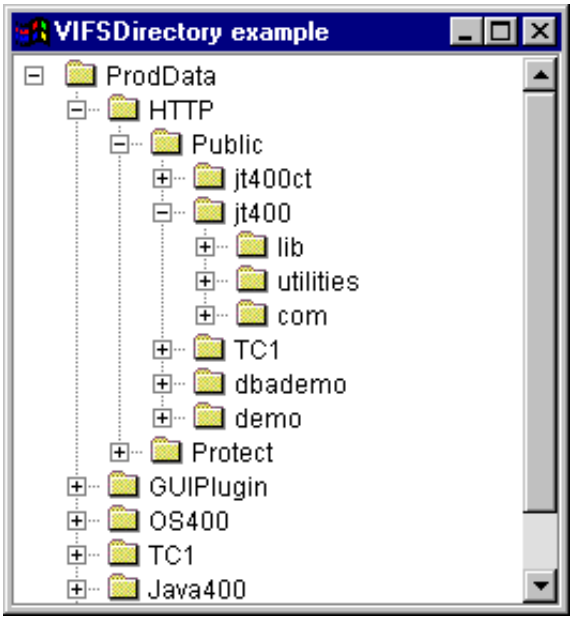
```
// Create the VIFSDirectory object.
// Assume that "system" in an AS400
// object created and initialized
// elsewhere.
VIFSDirectory root = new VIFSDirectory (system, "/DirectoryA/DirectoryB");

// Create and load an AS400ExplorerPane object.

AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

// Add the explorer pane to a frame.
// Assume that "frame" is a JFrame
// created elsewhere.
frame.getContentPane ().add (explorerPane);
```

範例

使用具有IFSDirectory物件的 AS400TreePane 來呈現整合檔案系統目錄架構

圖 1 顯示 VIFSDirectory 圖形式使用者介面元件：

圖 1：VIFSDirectory GUI 元件



IFSTextFileDocument

文字檔文件容許 Java 程式使用任何 Java Foundation Classes (JFC)
圖形式文字元件，來編輯或檢視伺服器上的整合檔案系統中的文字檔。(文字元件即是用來顯示使用者可選用性編輯的文字的圖形元件

IFSTextFileDocument類別是「JFC 文件」介面的實作。它可可直接與任何 JFC 圖形文字元件搭配使用。有數種文字元件
(如單一行欄位 (JTextField) 與多行文字區 (JTextArea)) 均可在 JFC 中使用。

文字檔文件會使文字元件的內容與文字檔產生關聯。Java 程式可隨時在文字元件與文字檔之間載入與儲存。

若要使用 IFSTextFileDocument，請同時設定系統與路徑特性。這些特性可使用建構元或透過 setSystem() 與 setPath()
方法來設定。然後， IFSTextFileDocument 物件會被「拉進」文字元件， 通常是使用文字元件建構元或 setDocument() 方法。

文字元件的起始內容是空的。使用 load() 從文字檔中載入內容。使用 save() 將文字元件的內容儲存到文字檔中。

下面範例會建立及載入 IFSTextFileDocument：

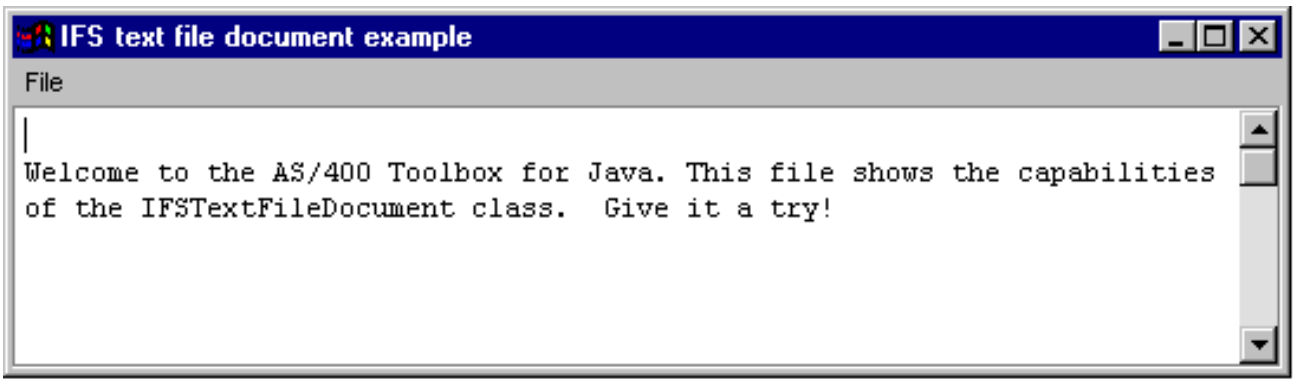
```
 // Create and load the
 // IFSTextFileDocument object. Assume
 // that "system" is an AS400 object
 // created and initialized elsewhere.
 IFSTextFileDocument ifsDocument = new IFSTextFileDocument (system,
"/DirectoryA/MyFile.txt");

ifsDocument.load ();

 // Create a text area to present the
 // document.
 JTextArea textArea = new JTextArea (ifsDocument);

 // Add the text area to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (textArea);
```

範例

在 JTextPane 中呈現IFSTextFileDocument

圖 1 顯示 IFSTextFileDocument 圖形式使用者介面元件：

圖 1：IFS 文字檔文件範例



VJavaApplicationCall 類別

VJavaApplicationCall 類別可讓您使用圖形式使用者介面 (GUI)，自從屬 站執行伺服器上的 Java 應用程式。

GUI 是具有兩個區段的畫面。上面的區段是輸出視窗，顯示 Java 程式寫入 標準輸出及標準錯誤的輸出。下面的區段是輸入欄位，使用者可在其中輸入 Java 環境、要使用參數執行 的 Java 程式，及 Java 程式透過標準輸入所接收的輸入。 請參閱 Java 指令 選項 以取得其餘相關資訊。

例如，這個程 式碼會建立下列 GUI 供您的 Java 程式使用。

VJavaApplicationCall 是一個您可以從 Java 程式呼叫的類別。 然而，IBM Toolbox for Java 也提供一公用程式，該公用程式是完整 的 Java 應用程式，可以用來從工作站呼叫您的 Java 程式。請參閱 RunJavaApplication 類別 以取得其餘相關資訊。

JDBC 類別

JDBC 圖形式使用者介面元件容許 Java 程式呈現不同的檢視及控制方式，以便使用 SQL（結構化查詢語言）陳述式及查詢來存取資料庫。

可用的元件如下：

- SQLStatementButton 和 SQLStatementMenuItem
  分別是按鈕或功能表項目，當您按一下或選取它時，會發出一個 SQL 陳述式。
- SQLStatementDocument 是一份可與 Java Foundation Classes (JFC) 圖形文字元件搭配使用來發出 SQL 陳述式的文件
- SQLResultSetFormPane 會以表單方式呈現 SQL 查詢的結果
- SQLResultSetTablePane會以表格呈現 SQL 查詢的結果
- SQLResultSetTableModel 會管理表格中 SQL 查詢的結果
- SQLQueryBuilderPane 會呈現可動態建置 SQL 查詢的交談式工具

所有 JDBC 圖形使用者介面元件均會使用 JDBC 驅動程式與資料庫進行通信。必須已用 JDBC驅動程式管理程式登記了 JDBC 驅動程式，以便使這些元件的任一個均可運作。下面範例會登記 AS/400 Toolbox for Java JDBC 驅動程式：

```
 // Register the JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());
```

SQL 連接

SQLConnection 物件代表使用 JDBC 的資料庫連線。SQLConnection 物件會與所有 JDBC圖形使用者介面元件搭配使用。

若要使用 SQLConnection，請使用建構元或setURL() 來設定 URL 內容。這會識別對其產生連接的資料庫。可以設定其它選用性特性：

- 使用setProperties()可以指定一組 JDBC 連線內容。
- 使用setUserName()可以指定連線的使用者名稱。
- 使用setPassword()可以指定連線的密碼。

當建立 SQLConnection 物件時，並不會產生實際的資料庫連線。相反地，它是當您呼叫getConnection() 時建立連線。正常情況下，JDBC 圖形使用者介面元件會自動呼叫此方法，但也可以隨時呼叫，以便控制產生連接的時間。

下面範例會建立及起始設定 SQLConnection 物件：

```
 // Create an SQLConnection object.

SQLConnection connection = new SQLConnection ();
```

```
// Set the URL and user name properties of the connection.
connection.setURL ("jdbc:as400://MySystem");
connection.setUserName ("Lisa");
```

SQLConnection 物件可對多個 JDBC
圖形使用者介面元件使用。所有如此的元件將使用同一個連接，因而可增進效能及資源使用。另一種方法，每一 JDB
圖形使用者介面元件均可使用不同的 SQL物件。有時候使用個別連接是必需的，以便在不同異動中發出 SQL 陳述式。

當不再需要連線時，請使用close()來關閉 SQLConnection 物件。這會同時釋放從屬站與伺服器中的 JDBC
資源。

SQL 按鈕與功能表項目

SQLStatementButton代表一個按鈕，當您按下它時，它會發出一個 SQL (Structured Query Language) 陳述式。SQLStatementButton 類別會擴充 Java Foundation Classes (JFC) JButton 類別，以便所有按鈕均有一致的外觀及行為。

同樣地，SQLStatementMenuItem 物件代表在選取時會發出 SQL 陳述式的功能表項目。SQLStatementMenuItem類別會擴充 JFC JMenuItem 類別， 以便所有功能表項目均有一致的外觀與行為。

若要使用這兩個類別之一，請同時設定連接與 SQLStatement 特性。這些特性可使用建構元或透過 setConnection() 與 setSQLStatement() 方法來設定。

下面範例會建立 SQLStatementButton。在執行期間，若按下此按鈕，它會刪除表格中的所有記錄：

```
 // Create an SQLStatementButton object.
 // The button text says "Delete All",
 // and there is no icon.
 SQLStatementButton button = new SQLStatementButton ("Delete All");

 // Set the connection and SQLStatement
 // properties.Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 button.setConnection (connection);
 button.setSQLStatement ("DELETE FROM MYTABLE");

 // Add the button to a frame. Assume
 // that "frame" is a JFrame created
 // elsewhere.
frame.getContentPane ().add (button);
```

在發出 SQL 陳述式後，請使用 getResultSet(), getMoreResults(), getUpdateCount() 或 getWarnings() 來擷取

SQLStatementDocument 類別

SQLStatementDocument 類別是「Java Foundation Classes (JFC) 文件」介面的實作。它可直接與任何 JFC 圖形文字元件搭配使用。 有數種文字元件 (如單一行欄位 (JTextField) 與多行文字區 (JTextArea)) 均可在 JFC 中使用。SQLStatementDocument 物件會使文字元件的內容與與 SQLConnection 物件產生關聯。Java 程式可以隨時執行文件內容中所含有 SQL 陳述式, 然後處理結果 (若有的話)。

若要使用 SQLStatementDocument, 您必須設定 connection 特性。經由使用 建構元或 setConnection() 方法來設定此特性。然後, SQLStatementDocument 物件會被「拉進」文字元件, 通常是使用文字元件建構元或 setDocument() 方法。您可以隨時使用execute()來執行文件所包含的 SQL 陳述式。

下面範例會在 JTextField 中建立 SQLStatementDocument：

```
 // Create an SQLStatementDocument
 // object. Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 // The text of the document is
 // initialized to a generic query.
 SQLStatementDocument document = new SQLStatementDocument (connection,
"SELECT * FROM QIWS.QCUSTCDT");

 // Create a text field to present the
 // document.
 JTextField textField = new JTextField ();

textField.setDocument (document);

 // Add the text field to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (textField);

 // Run the SQL statement that is in
 // the text field.
 document.execute ();
```

發出 SQL 陳述式之後, 請使用getResultSet()、getMoreResults()、getUpdateCount()或getWarnings()來擷取結果。

SQLResultSetFormPane 類別

SQLResultSetFormPane 會將 SQL（結構化查詢語言）查詢的結果呈現在套表中。
表單會一次顯示一筆記錄，並提供一些按鈕，容許使用者向前捲動、向後捲動到第一筆或最後一筆記錄，
或是復新結果的檢視畫面。

若要使用 SQLResultSetFormPane，請設定連接與查詢特性。 這些內容可以使用建構元或setConnection()和
setQuery()方法來設定。使用load()以執行查詢並呈現結果集中的第一筆記錄。當您不再需要結果時，請呼叫
close()來確定關閉結果集。

下面範例會建立 SQLResultSetFormPane 物件，並將它新增到頁框中：

```
 // Create an SQLResultSetFormPane
 // object. Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 SQLResultSetFormPane formPane = new SQLResultSetFormPane (connection,
"SELECT * FROM QIWS.QCUSTCDT");

 // Load the results.
 formPane.load ();

 // Add the form pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (formPane);
```

SQLResultSetTablePane 類別

SQLResultSetTablePane會在表格中呈現 SQL（結構化查詢語言）
查詢的結果。表格中的每一列顯示來自結果集的記錄，而每一欄則顯示欄位。

若要使用 SQLResultSetTablePane，請設定連接與查詢特性。這些內容可以使用建構元或setConnection(和 setQuery()
方法來設定。使用load()以執行查詢並將結果呈現在表格中。當您不再需要結果時，請呼叫close()來確定關閉結果集。

下面範例會建立 SQLResultSetTablePane 物件，並將它新增到頁框中：

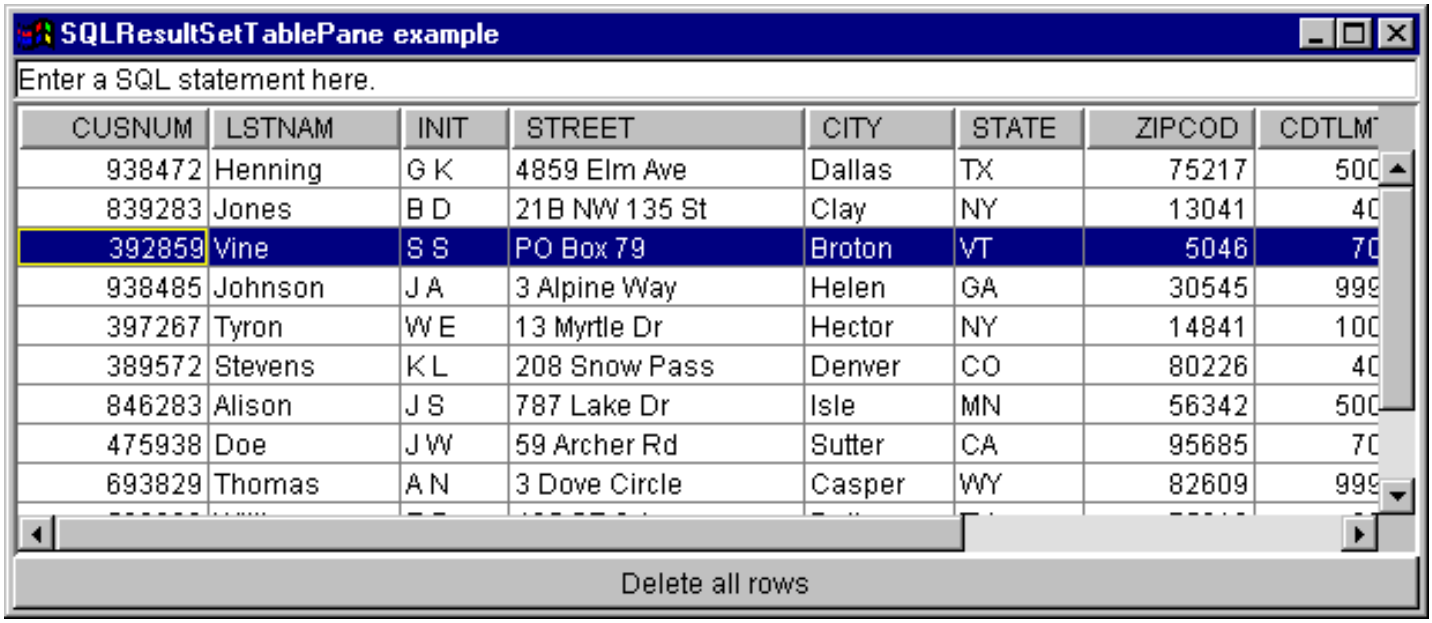
```
 // Create an SQLResultSetTablePane
 // object. Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 SQLResultSetTablePane tablePane = new SQLResultSetTablePane (connection,
"SELECT * FROM QIWS.QCUSTCDT");

 // Load the results.
 tablePane.load ();

 // Add the table pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (tablePane);
```

範例

呈現會顯示表格內容的SQLResultSetTablePane 此範例使用 SQLStatementDocument（用文字表示的下列影像，「在此輸入 SQL 陳述式
，它容許使用者鍵入任何的 SQL 陳述式，及 SQLStatementButton（用文字表示，「刪除全部的列」），它容許使用者從表格中刪除全

圖 1 顯示 SQLResultSetTablePane 圖形式使用者介面元件：

圖 1：SQLResultSetTablePane GUI 元件



SQLResultSetTableModel 類別

SQLResultSetTablePane 是使用模型-概略表-控制器範例來實施，
在此資料與使用者介面被分成不同的類別。此施行會將SQLResultSetTableModel與 Java Foundation Classes' (JFC) JTable 整合。SQLResultSetTableModel 類別會管理查詢的結果， 而 JTable 則以圖形方式顯示結果並處理使用者互動。

SQLResultSetTablePane 會提供足夠的功能來符合大部份的需求。不過，如果呼叫程式需要更進一步控制 JFC 元件，則呼叫程式可以直接使用 SQLResultSetTableModel，並提供與不同圖形使用者介面元件的自訂整合。

若要使用 ResAltSotAoTebSAMmdAS，請設定連接與查詢特性。這些內容可以使用建構元或 setConnection() 和 setQuery()方法來設定。使用load()以執行查詢並載入結果。當您不再需要結果時，請呼叫close() 來確定關閉結果集。

下面範例會建立 SQLResultSetTableModel 物件，並以 JTable 呈現它：

```
 // Create an SQLResultSetTableModel
 // object. Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 SQLResultSetTableModel tableModel = new SQLResultSetTableModel (connection,
"SELECT * FROM QIWS.QCUSTCDT");

 // Load the results.
 tableModel.load ();

 // Create a JTable for the model.
 JTable
table = new JTable (tableModel);

 // Add the table to a frame.Assume
 // that "frame" is a JFrame created
 // elsewhere.
 frame.getContentPane ().add (table);
```

SQL 查詢建置器

SQLQueryBuilderPane 會呈現一個交談式工具，可用來動態建置 SQL 查詢。

若要使用 SQLQueryPane，請設定連接特性。您可以使用建構元或setConnection()方法來設定這個內容。使用load()可以載入查詢建置器圖形式使用者介面所需的資料。使用getQuery()可以取得使用者已建立的 SQL 查詢。

下面範例會建立 SQLQueryBuilderPane 物件，並將它新增到頁框中：

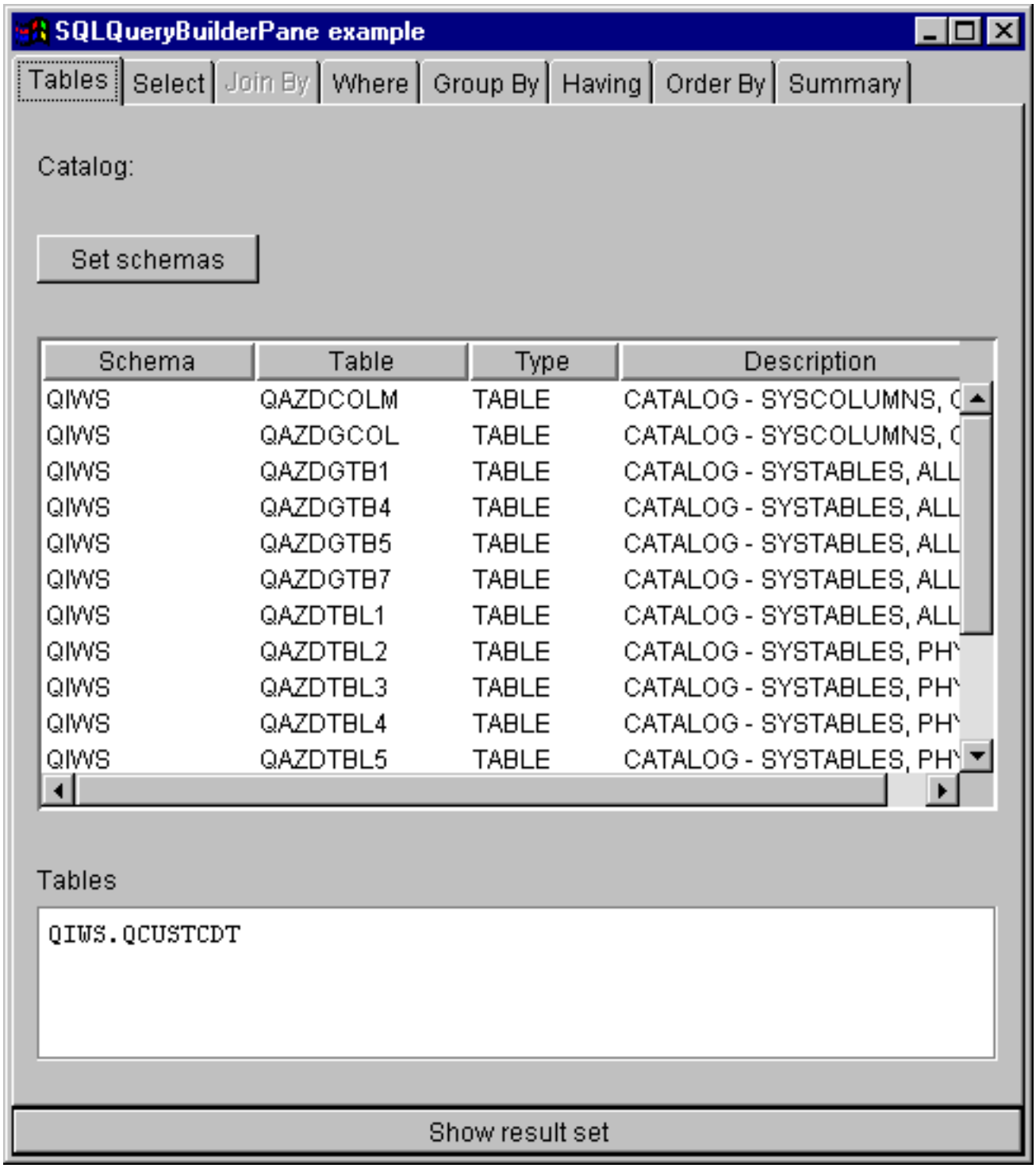
```
 // Create an SQLQueryBuilderPane
 // object. Assume that "connection"
 // is an SQLConnection object that is
 // created and initialized elsewhere.
 SQLQueryBuilderPane queryBuilder = new
SQLQueryBuilderPane (connection);

 // Load the data needed for the query
 // builder.
 queryBuilder.load ();

 // Add the query builder pane to a
 // frame. Assume that "frame" is a
 // JFrame created elsewhere.
 frame.getContentPane ().add
(queryBuilder);
```

範例

呈現SQLQueryBuilderPane 及按鈕當按一下按鈕時，會在另一個頁框中呈現 SQLResultSetFormPane 中查詢的結果。

圖 1 顯示 SQLQueryBuilderPane 圖形式使用者介面元件：

圖 1：SQLQueryBuilderPane GUI 元件


SQLQueryBuilderPane example
Tables
Select
Join By
Where
Group By
Having
Order By
Summary
Catalog:
Set schemas
Schema
Table
Type
Description
QIWS QAZDCOLM TABLE CATALOG - SYSCOLUMNS, C
QIWS QAZDGCOL TABLE CATALOG - SYSCOLUMNS, C
QIWS QAZDGTB1 TABLE CATALOG - SYSTABLES, ALL
QIWS QAZDGTB4 TABLE CATALOG - SYSTABLES, ALL
QIWS QAZDGTB5 TABLE CATALOG - SYSTABLES, ALL
QIWS QAZDGTB7 TABLE CATALOG - SYSTABLES, ALL
QIWS QAZDTBL1 TABLE CATALOG - SYSTABLES, ALL
QIWS QAZDTBL2 TABLE CATALOG - SYSTABLES, PH
QIWS QAZDTBL3 TABLE CATALOG - SYSTABLES, PH
QIWS QAZDTBL4 TABLE CATALOG - SYSTABLES, PH
QIWS QAZDTBL5 TABLE CATALOG - SYSTABLES, PH
Tables
QIWS.QCUSTCDT
Show result set

Vaccess 工作

vaccess (GUI) 元件容許 Java 程式在 GUI 中呈現伺服器工作清單及工作日誌訊息之類的工作。

可用的元件如下：

- VJobList 物件是一種資源，它會呈現 AS400Panes 中所使用的伺服器工作清單。
- VJob 物件是一種資源，它會呈現要在 AS400Panes 中使用的工作日誌的訊息清單。

您可以合併使用 AS400Panes、VJobList 物件與 VJob 物件，以多種檢視方式呈現工作清單或工作日誌。

若要使用 VJobList，請設定系統、名稱、號碼及使用者特性。 這些內容可以使用建構元或透過 setSystem()、setName()、setNumber() 和 setUser() 內容來設定。

若要使用 VJob，請設定系統特性。此內容可以使用建構元或透過 setSystem() 方法來設定。

VJobList 或 VJob 物件接著將「插入」AS400Pane 來作為根， 方式是使用窗格的建構元或 setRoot() 方法。

VJobList 另有一些特性有助於定義 AS400Panes 中所呈現的工作集。使用 setName() 可以指定只顯示具有特定名稱的工作。使用 setNumber() 可以指定只顯示具有特定編號的工作。同樣地，使用 setUser() 可以指定只顯示具有特定使用者的工作。

當建立 AS400Pane、VJobList 與 VJob 物件時，它們會起始設定為預設狀態。
建立時，不會載入工作或工作日誌訊息的清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load()方法。
這會起始與伺服器系統的通信，來收集清單的內容。

在執行時間，以滑鼠右鍵按一下工作、工作清單或工作日誌訊息，以顯示捷徑功能表。
從執行功能表中選取內容，針對選取的物件執行動作：

- 工作 - 使用內容，例如類型和狀態。您也可以變更部份內容的值。
- 工作清單 - 使用內容，例如名稱、號碼及使用者內容。您也可以變更清單的內容。
- 工作日誌訊息 - 顯示內容，例如：全文、嚴重性及傳送時間。

使用者僅能存取他們有權使用的工作。此外，Java 程式可在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

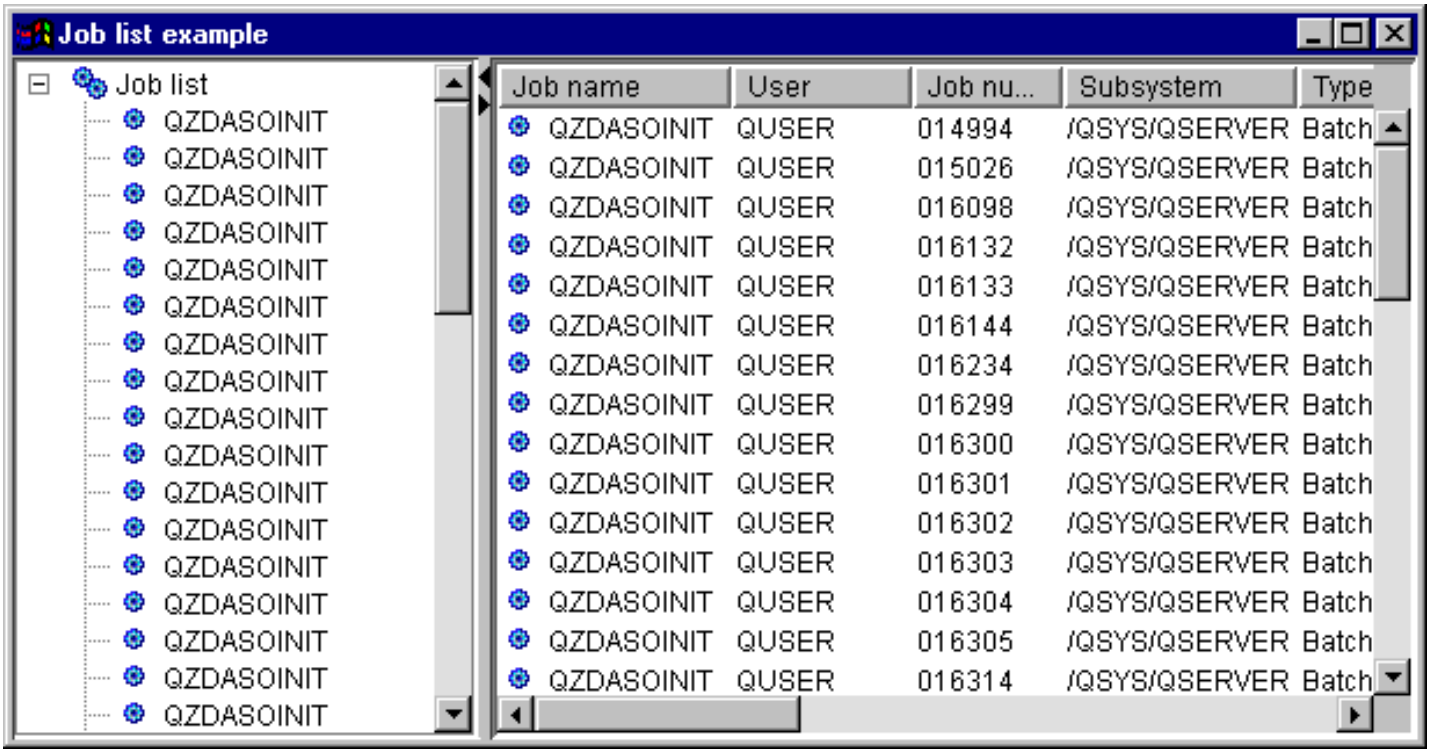
下面範例會建立 VJobList 並在 AS400ExplorerPane 中呈現它：

```
 // Create the VJobList object. Assume
 // that "system" is an AS400 object
 // created and initialized elsewhere.
 VJobList root = new VJobList
(system);

 // Create and load an
 // AS400ExplorerPane object.

AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

 // Add the explorer pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (explorerPane);
```

範例

此VJobList 範例會呈現已填入工作清單的 AS400ExplorerPane。此清單會顯示系統中具有相同工作名稱的工作。

下面影像顯示 VJobList 圖形使用者介面元件：



Vaccess 訊息類別

訊息圖形式使用者介面元件容許 Java 程式在 GUI 中呈現伺服器訊息清單。

可用的元件如下：

- 訊息清單 物件是一種資源， 代表在 AS400Panes 中所使用的訊息清單。這是針對指令或程式呼叫所產生的訊息清單。
- 訊息佇列物件是一種資源，代表 AS400Ppanes 中所使用的伺服器訊息佇列中的訊息。

AS400Panes 是圖形式使用者介面元件， 它們會呈現及容許操作一個或多個伺服器資源。VMessageList 和 VMessageQueue 物件是代表 AS400Panes 中的伺服器訊息清單的資源。

您可以合併使用 AS400Pane、VMessageList 與 VMessageQueue 物件，以不同檢視方式呈現訊息清單，並允許使用者選取訊息及對它們執行動作。

VMessageList 類別

VMessageList 物件是一種資源，代表AS400Panes
中所使用的訊息清單。這是針對指令或程式呼叫所產生的訊息清單。下列方法會傳回訊息清單：

- CommandCall.getMessageList()
- CommandCallButton.getMessageList()
- CommandCallMenuItem.getMessageList()
- ProgramCall.getMessageList()
- ProgramCallButton.getMessageList()
- ProgramCallMenuItem.getMessageList()

若要使用 VMessageList，請設定 messageList 特性。這個內容可以使用建構元或透過setMessageList()方法來設定。接著使用
AS400Pane 的建構元或 setRoot() 方法，將 VMessageList 物件「插入」 AS400Pane 中作為根。

當建立 AS400Pane 與 VMessageList
物件時，它們會起始設定為預設狀態。建立時不會載入訊息清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫
load() 方法。

在執行期間，使用者可以滑鼠右鍵按一下訊息來顯示環境功能表，以對它執行動作。
訊息環境功能表可能包括名為內容的項目，它會顯示諸如嚴重性、類型及日期等內容。

呼叫程式可以經由在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

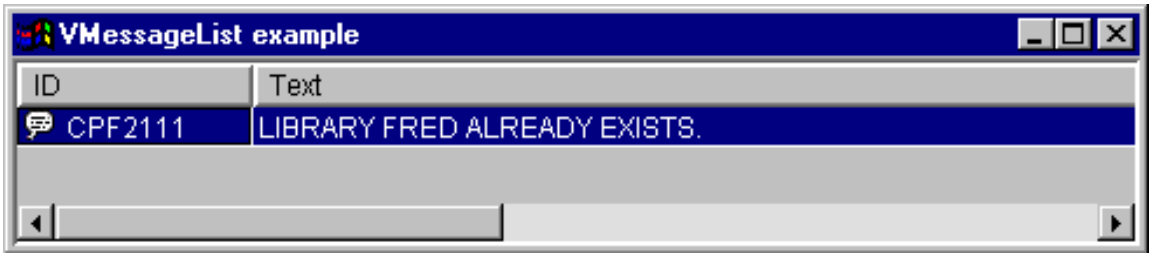
下面範例會建立 VMessageList 供指令呼叫所產生的訊息使用， 並在 AS400DetailsPane 中呈現：

```
 // Create the VMessageList object.
 // Assume that "command" is a
 // CommandCall object created and run
 // elsewhere.

VMessageList root = new VMessageList (command.getMessageList ());

 // Create and load an AS400DetailsPane
 // object.
 AS400DetailsPane detailsPane = new
AS400DetailsPane (root);

detailsPane.load ();

 // Add the details pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (detailsPane);
```

範例

呈現使用 AS400DetailsPane 的指令呼叫，其中包含VMessageList 物件所產生的訊息清單。圖 1 顯示 VMessageList
圖形式使用者介面元件：

圖 1：VMessageList GUI 元件


VMessageList example
ID
Text
CPF2111
LIBRARY FRED ALREADY EXISTS.

VMessageQueue 類別

VMessageQueue 物件是一種資源，代表AS400Panes 中所使用的伺服器訊息佇列中的訊息。

若要使用 VMessageQueue，請設定系統與路徑特性。 您可以使用建構元或透過setSystem()和 setPath()方法來設定這些內容。接著使用 AS400Pane 的建構元或 setRoot() 方法， 將 VMessageQueue 物件「插入」AS400Pane 中作為根。

VMessageQueue 另有一些其它有用的內容，可用來定義 AS400Panes 所呈現的訊息集。setSeverity()使用指定應出現訊息的嚴重性。使用setSelection()來指定應出現的訊息類型。

當建立 AS400Pane 與 VMessageQueue 物件時，它們會起始設定為預設狀態。建立時不會載入訊息清單。
若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。 這會起始與伺服器系統的通信，來收集清單的內容。

在執行期間，使用者可以滑鼠右鍵按一下訊息或訊息佇列來顯示環境功能表，以對它執行動作。
訊息佇列的環境功能表可能包括下列項目：

- 清除 - 清除訊息佇列
- 內容 - 允許使用者設定嚴重性和選項內容。這可用來變更清單的內容

下列是可對訊息佇列中的訊息使用的動作：

- 移除 - 從訊息佇列中移除訊息
- 回答 - 回覆查詢訊息
- 內容 - 顯示如嚴重性、類型及日期等內容

當然，使用者僅能存取他們有權使用的訊息佇列。此外，呼叫程式可在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

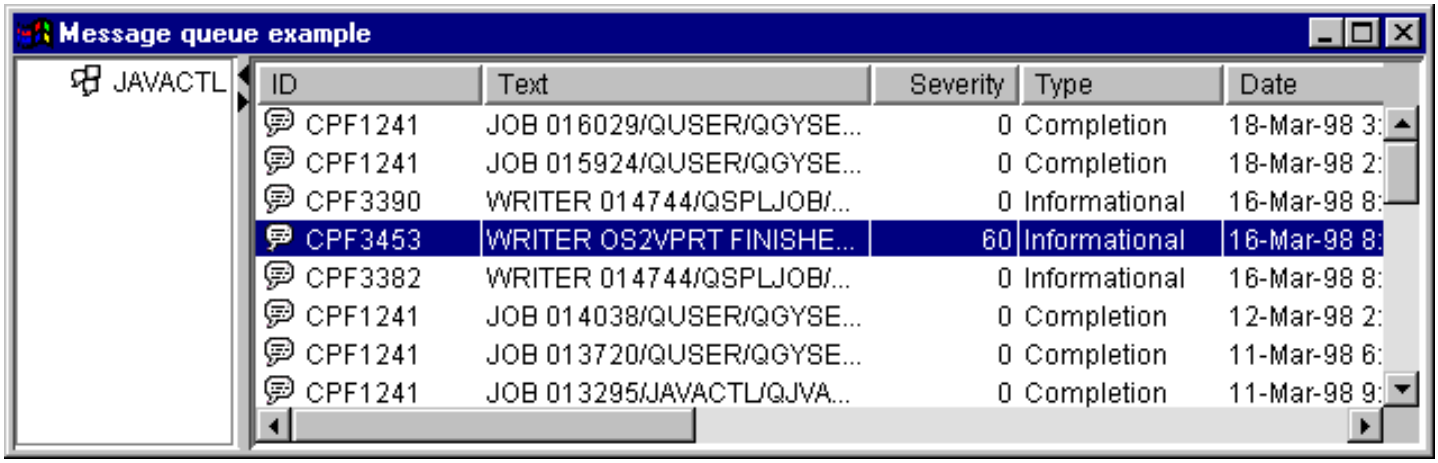
下面範例會建立 VMessageQueue 並在 AS400ExplorerPane 中呈現它：

```
 // Create the VMessageQueue object.
 // Assume that "system" is an AS400
 // object created and initialized
 // elsewhere.
 VMessageQueue root = new VMessageQueue (system,
"/QSYS.LIB/MYLIB.LIB/MYMSGQ.MSGQ");

 // Create and load an
 // AS400ExplorerPane object.
 AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);
 explorerPane.load ();

 // Add the explorer pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (explorerPane);
```

範例

使用 AS400ExplorerPane，其中包括MessageQueue 物件，呈現訊息佇列中的訊息清單。圖 1 顯示 VMessageQueue 圖形式使用者介面元件：

圖 1：VMessageQueue GUI 元件


Message queue example
JAVACTL
ID
Text
Severity
Type
Date
CPF1241 JOB 016029/QUSER/QGYSE... 0 Completion 18-Mar-98 3:
CPF1241 JOB 015924/QUSER/QGYSE... 0 Completion 18-Mar-98 2:
CPF3390 WRITER 014744/QSPLJOB/... 0 Informational 16-Mar-98 8:
CPF3453 WRITER OS2VPRT FINISHE... 60 Informational 16-Mar-98 8:
CPF3382 WRITER 014744/QSPLJOB/... 0 Informational 16-Mar-98 8:
CPF1241 JOB 014038/QUSER/QGYSE... 0 Completion 12-Mar-98 2:
CPF1241 JOB 013720/QUSER/QGYSE... 0 Completion 11-Mar-98 6:
CPF1241 JOB 013295/JAVACTL/QJVA... 0 Completion 11-Mar-98 9:

Permission 類別

您可以透過VIFSFile 和 VIFSDirectory類別，在圖形式使用者介面 (GUI) 中使用Permission 類別。許可權已被新增作為這些類別的每一個的動作。

下面範例告訴您「許可權」如何搭配 VIFSDirectory 類別一起使用：

```
// Create AS400 object
AS400 as400 = new AS400();

// Create an IFSDirectory using the system name
// and the full path of a QSYS object
VIFSDirectory directory = new VIFSDirectory(as400,
"/QSYS.LID/testlib1.lib");

// Create as explorer Pane
AS400ExplorerPane pane = new AS400ExplorerPane((VNode)directory);

// Load the information
pane.load();
```

Vaccess 列印類別

vaccess 套件中的下列元件容許 Java 程式以圖形式使用者介面來呈現伺服器列印資源的清單。

- VPrinters物件是一種資源， 它代表要在 AS400Panes 中使用的印表機清單。
- VPrinter物件是一種資源， 它代表要在 AS400Panes 中使用的印表機及其排存檔。
- VPrinterOutput物件是一種資源， 它代表要在 AS400Panes 中使用的排存檔清單。
- SpooledFileViewer物件為以視覺化的方式呈現排存檔的資源。

AS400Panes 是一些 GUI 元件， 它們會呈現及容許操作一個或多個伺服器資源。VPrinters、VPrinter 及 VPrinterOutput 物件為資源，代表 AS400Panes 中的伺服器列印資源的清單。

您可以合併使用 AS400Pane、VPrinters、VPrinter 與 VPrinterOutput
物件，以呈現列印資源的不同檢視方式，以及容許使用者選取它們並對它們執行作業。

VPrinters 類別

VPrinters物件是一種資源，它代表要在AS400Panes 中使用的印表機清單。

若要使用 VPrinters 物件，請設定系統特性。此內容可以使用建構元或透過setSystem()
方法來設定。然後，即會使用窗格的建構元或 setRoot() 方法， 將 VPrinters 物件「插入」AS400Pane 作為根。

VPrinters 物件有另一種有用的內容，可用來定義 AS400Panes 中所呈現的印表機集。使用setPrinterFilter()
來指定過濾字元，以定義應該出現的印表機。

當建立 AS400Pane 與 VPrinters
物件時，它們會起始設定為預設狀態。尚未載入印表機清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。

在執行期間，使用者可以在任何印表機清單或印表機上按一下滑鼠右鍵，
以便顯示環境功能表來針對它們執行動作。排存檔清單環境功能表可以包括名為內容的項目，
讓使用者設定印表機過濾內容，來變更清單的內容。

印表機環境功能表可能包括下列項目：

- 保留 - 保留印表機
- 釋放 - 釋放印表機
- 啟動 - 啟動印表機
- 停止 - 停止印表機
- 可使用- 使印表機可用
- 不可使用- 使印表機無法使用
- 內容 - 顯示印表機的內容並容許使用者設定過濾條件

使用者僅能存取他們有權使用的印表機。此外，呼叫程式會在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

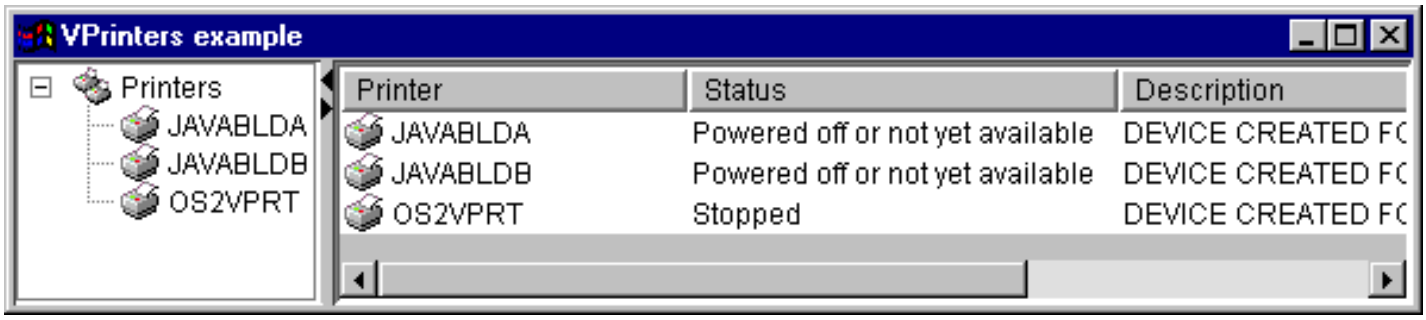
下面範例會建立 VPrinters 物件並在 AS400TreePane 中呈現它

```
 // Create the VPrinters object.
 // Assume that "system" is an AS400
 // object created and initialized
 // elsewhere.

VPrinters root = new VPrinters (system);

 // Create and load an AS400TreePane
 // object.
 AS400TreePane treePane = new AS400TreePane (root);

treePane.load ();

 // Add the tree pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (treePane);
```

範例

搭配使用 AS400ExplorerPane與VPrinters物件，來呈現列印資源。圖 1 顯示 VPrinters 圖形式使用者介面元件：

圖 1：VPrinters GUI 元件


VPrinters example
Printers
JAVABLDA
JAVABLDB
OS2VPRT
Printer
Status
Description
JAVABLDA
Powered off or not yet available
DEVICE CREATED FC
JAVABLDB
Powered off or not yet available
DEVICE CREATED FC
OS2VPRT
Stopped
DEVICE CREATED FC

VPrinter

VPrinter物件是一種資源，它代表要在AS400Panes 中使用的伺服器印表機及其排存檔。

若要使用 VPrinter，請設定印表機特性。此內容可以使用建構元或透過setPrinter()方法來設定。然後，即會使用窗格的建構元或 setRoot() 方法， 將 VPrinter 物件「插入」AS400Pane 中作為根。

當建立 AS400Pane 與 VPrinter 物件時，它們會起始設定為預設狀態。在建立期間，不會載入印表機的屬性與排存檔清單。

若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。 這會起始與伺服器系統的通信，來收集清單的內容。

在執行期間，使用者可以滑鼠右鍵按一下任何印表機或排存檔，
以便顯示環境功能表來針對它們執行動作。訊息佇列的環境功能表可能包括下列項目：

- 保留 - 保留印表機
- 釋放 - 釋放印表機
- 啟動 - 啟動印表機
- 停止 - 停止印表機
- 可使用- 使印表機可用
- 不可使用- 使印表機無法使用
- 內容 - 顯示印表機的內容並容許使用者設定過濾條件

針對印表機所列示的排存檔的環境功能表可能包括下列項目：

- 回答 - 回答排存檔
- 保留 - 保留排存檔
- 釋放 - 釋放排存檔
- 列印下一個 - 列印下一個排存檔
- 傳送 - 傳送排存檔
- 移動 - 移動排存檔
- 刪除 - 刪除排存檔
- 內容 - 顯示排存檔的許多內容，並容許使用者變更其中的某些內容

使用者僅能存取他們有權使用的印表機與排存檔。此外，呼叫程式可在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

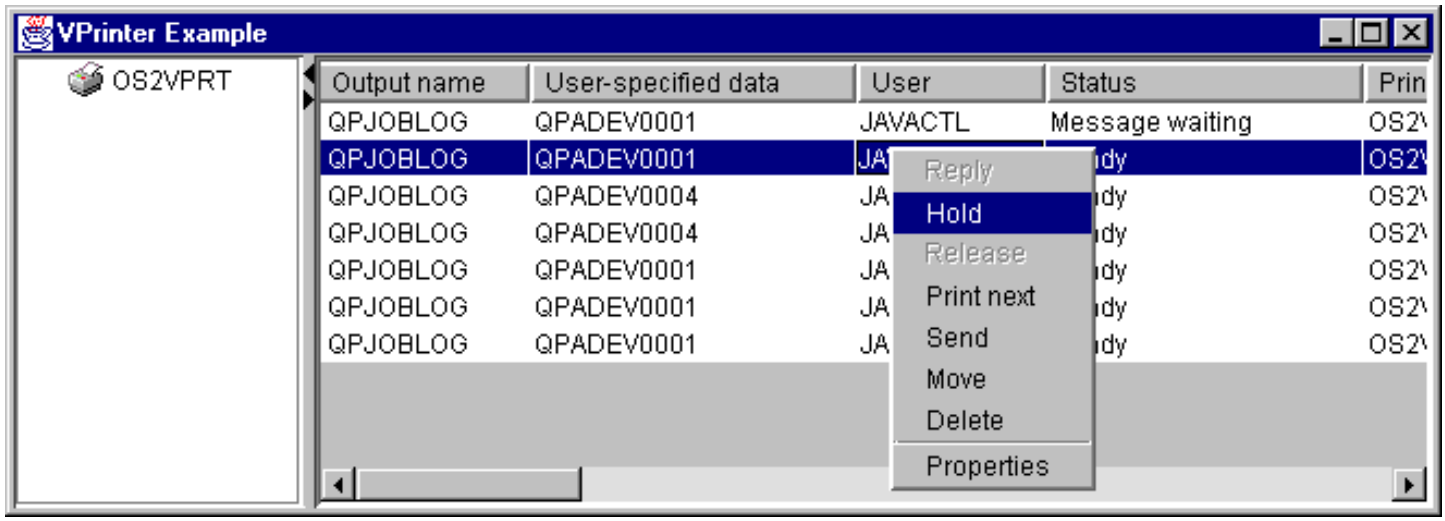
下面範例會建立 VPrinter 並在 AS400ExplorerPane 中呈現它：

```
 // Create the VPrinter object.
 // Assume that "system" is an AS400
 // object created and initialized
 // elsewhere.
 VPrinter root = new VPrinter (new Printer (system, "MYPRINTER"));

 // Create and load an
 // AS400ExplorerPane object.

AS400ExplorerPane explorerPane = new AS400ExplorerPane (root);

explorerPane.load ();

 // Add the explorer pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (explorerPane);
```

範例

使用具有VPrinter物件的 AS400ExplorerPane 來呈現列印資源。圖 1 顯示 VPrinter 圖形式使用者介面元件：

圖 1：VPrinter GUI 元件


VPrinter Example
OS2VPRT
Output name
User-specified data
User
Status
Prin
QPJOBLOG
QPADEV0001
JAVACTL
Message waiting
OS2\
QPJOBLOG
QPADEV0001
JA
dy
OS2\
QPJOBLOG
QPADEV0004
JA
dy
OS2\
QPJOBLOG
QPADEV0004
JA
dy
OS2\
QPJOBLOG
QPADEV0001
JA
dy
OS2\
QPJOBLOG
QPADEV0001
JA
dy
OS2\
QPJOBLOG
QPADEV0001
JA
dy
OS2\
Reply
Hold
Release
Print next
Send
Move
Delete
Properties

VPrinterOutput 類別

VPrinterOutput物件是一種資源，它代表可在AS400Panes 中使用的伺服器上的排存檔清單。

若要使用 VPrinterOutput 物件，請設定系統特性。這些內容可使用建構元或setSystem()方法來設定。接著使用 AS400Pane 的建構元或 setRoot() 方法， 將 VPrinterOutput 物件「插入」AS400Pane 中作為根。

VPrinterOutput 物件含有其它有用的內容，有助於定義 AS400Panes 中所呈現的排存檔集。使用setTypeFilter() 來指定應該出現的套表類型。使用 setUserDataFilter()來指定應該出現的使用者資料。最後，使用setUserFilter() 來指定應該出現的使用者排存檔。

當建立 AS400Pane 與 VPrinterOutput 物件時，它們會起始設定為預設狀態。
在建立期間，不會載入排存檔清單。若要載入內容，呼叫程式必須在這兩個物件之一中以明確方式呼叫 load() 方法。這會起始與伺服器系統的通信，來收集清單的內容。

在執行期間，使用者可以在任何排存檔或排存檔清單上按一下滑鼠右鍵，
以便顯示環境功能表來針對它們執行動作。排存檔清單環境功能表可以包括名為內容的項目，
讓使用者設定過濾內容，來變更清單的內容。

排存檔環境功能表可能包括下列項目：

- 回答 - 回答排存檔
- 保留 - 保留排存檔
- 釋放 - 釋放排存檔
- 列印下一個 - 列印下一個排存檔
- 傳送 - 傳送排存檔
- 移動 - 移動排存檔
- 刪除 - 刪除排存檔
- 內容 - 顯示排存檔的許多內容，並容許使用者變更其中的某些內容

當然，使用者僅能存取他們有權使用的排存檔。此外，呼叫程式會在窗格中使用 setAllowActions() 方法，阻止使用者執行動作。

下面範例會建立 VPrinterOutput 並在 AS400ListPane 中呈現它：

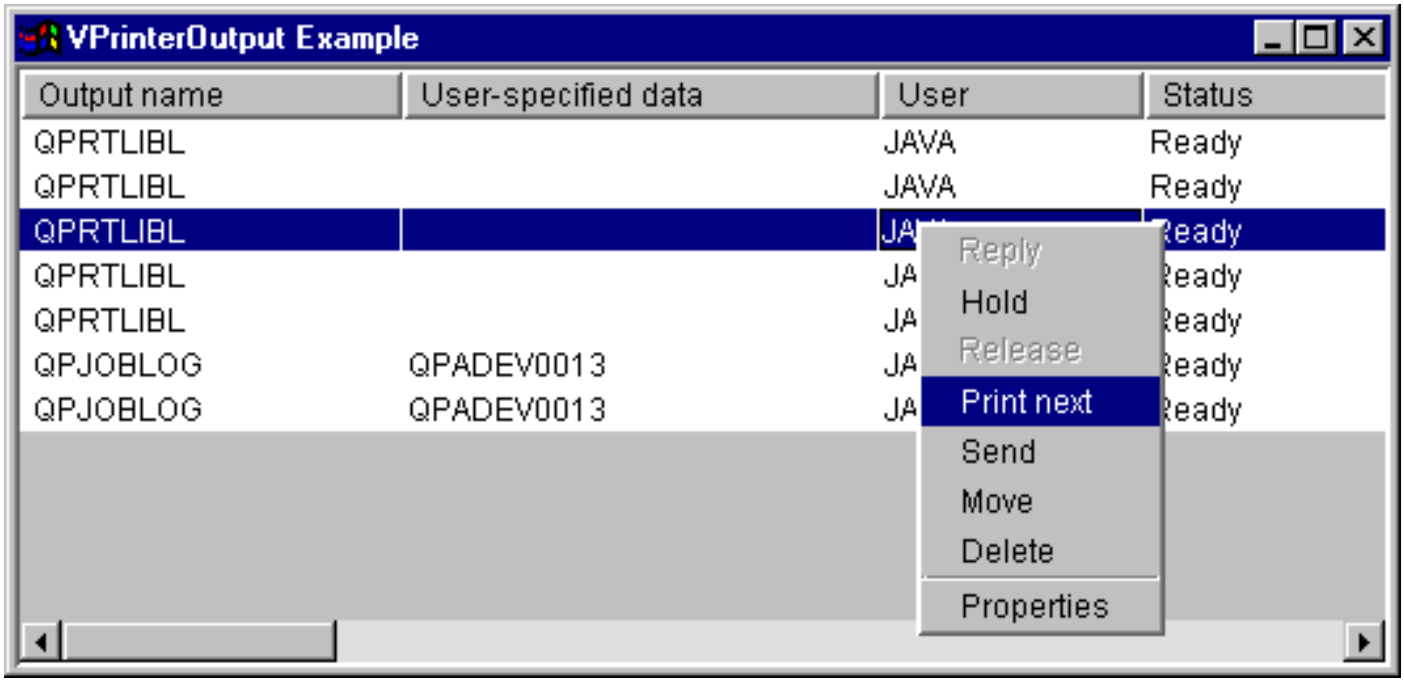
```
 // Create the VPrinterOutput object.
 // Assume that "system" is an AS400
 // object created and initialized
 // elsewhere.

VPrinterOutput root = new VPrinterOutput (system);

 // Create and load an AS400ListPane
 // object.
 AS400ListPane listPane = new AS400ListPane (root);

listPane.load ();

 // Add the list pane to a frame.
 // Assume that "frame" is a JFrame
```

```
// created elsewhere.
frame.getContentPane ().add (listPane);
```

範例

使用列印資源VPrinterOutput物件，來呈現排存檔清單。圖 1 顯示 VPrinterOutput 圖形式使用者介面元件：

圖 1：VPrinterOutput GUI 元件



SpooledFileViewer 類別

SpooledFileViewer 類別會建立一個視窗，用來檢視已經排存要進行列印的「進階功能列印 (AFP)」以及「系統網路架構」字串 (SCS) 檔案。這個類別基本上與大部份的文書處理程式一樣，會新增一個「預覽列印」功能到您的排存檔，圖 1 中所示。

當檢視檔案佈置的精確度比列印檔案更重要時，
或是當檢視資料比列印更加經濟、或是無法使用印表機時， 此排存檔檢視器特別有用。

附註：主電腦伺服器上必須已安裝 SS1 選項 8 (AFP 相容字形)。

使用 SpooledFileViewer 類別

有三種建構元方法可用來建立 SpooledFileViewer 類別的實例。SpooledFileViewer()
建構元可以用來建立檢視器，但不需要相關的排存檔。如果使用這個建構元，稍後將需要使用 setSpooledFile(SpooledFi設定已排存的檔案。 SpooledFileViewer(SpooledFile)
建構元可以用來建立特定排存檔的檢視器，並以第一頁為起始檢視畫面。
最後，SpooledFileViewer(spooledFile, int)
建構元可以用來建立特定排存檔的檢視器，並以特定頁做為起始檢視畫面。不論使用哪一個建構元，建立好檢視器之後，就要執行 load()呼叫，以便真正地擷取排存檔資料。

然後，您的程式可以使用下列方法，遍訪排存檔的各個頁面：

- load FlashPage()
- load Page()
- pageBack()
- pageForward()

不過，如果您需要更詳細檢查文件的某些段落，您可以用下列方式改變每一頁的比例，
來放大或縮小文件的頁影像：

- fitHeight()
- fitPage()
- fitWidth()
- actualSize()

您的程式可在呼叫lose()方法後結束，它會關閉輸入串流並釋出與該串流連結的任何資源。

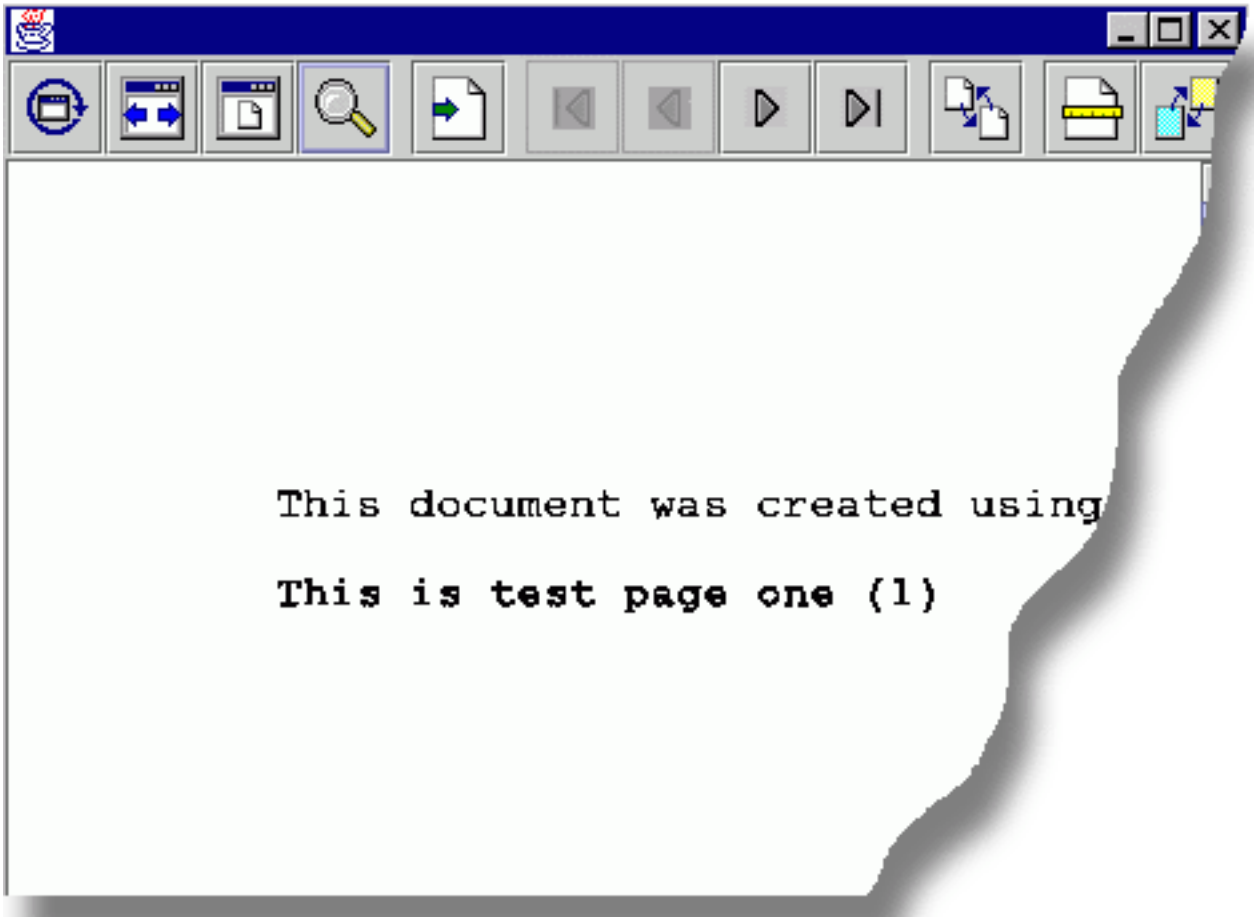
使用 SpooledFileViewer

SpooledFileViewer 類別的案例實際上是檢視器的圖形式表示法， 它可以顯示並導覽 AFP 或 SCS 排存檔。例如，下列程式可建立圖 1 中的排存檔檢視器，來顯示先前在伺服器上建立的排存檔。

附註：您可以選取圖 1影像上的按鈕來取得按鈕功能說明，或者 (如果您的瀏覽器未啟 用 JavaScript)參閱工具列說明。

```
// Assume splf is the spooled file.
// Create the spooled file viewer
SpooledFileViewer splfv = new SpooledFileViewer(splf, 1);
splfv.load();
// Add the spooled file viewer to a frame
JFrame frame = new JFrame("我的視窗");
frame.getContentPane().add(splfv);
frame.pack();
frame.show();
```

圖 1：SpooledFileViewer



SpooledFileViewer 工具列說明


實際大小按鈕會使用 actualSize() 方法，將排存檔還原為它的原始大小。


固定寬度按鈕會使用 fitWidth() 方法，將排存檔頁影像伸展到檢視器資料框的左右兩側。


固定頁面按鈕會使用 fitPage()
方法，將排存檔頁影像上下伸展，直到佔滿整個排存檔檢視器的資料框為止。


「縮放」按鈕可讓您增加或減少排存檔頁影像的大小，
其法為選取預設的百分比，或在選取縮放按鈕後，於對話框中的文字欄位中輸入您自己的百分比。


當選取「跳至」按鈕時， 它可讓您跳至排存檔內的特定頁面。


當選取「第一頁」按鈕時， 它會將您帶到排存檔的第一頁，並在停用時，指出您在第一頁上。


當選取「上一頁」按鈕時， 它會立即將您帶到您正在檢視的頁面的上一頁。


當選取下一頁按鈕時， 它會立即將您帶到您正在檢視的頁面的後一頁。


當選取最後一頁按鈕時， 它會將您帶到排存檔的最後一頁，並在停用時，指出您在最後一頁上。


當您選取載入快閃頁頁按鈕時， 它會使用 loadFlashPage() 方法，載入先前檢視過的頁面。


當您選取設定紙張大小按鈕時， 它可讓您設定紙張大小。


當您選取設定清晰度按鈕時， 它可讓您設定檢視清晰度。

Vaccess ProgramCall 類別

vaccess 套件中的程式呼叫元件，可讓 Java 程式呈現一個按鈕或功能表項目來呼叫伺服器程式。您可以使用 ProgramParameter物件來指定輸入、輸出和輸入/輸出參數。當執行程式時，
輸出及輸入/輸出參數會含有伺服器程式所傳回的資料。

ProgramCallButton物件代表一個按鈕，在按下時，會呼叫伺服器程式。ProgramCallButton 類別會擴充 Java Foundation Classes (JFC) JButton 類別， 以便所有按鈕均有一致的外觀及行為。

同樣地，ProgramCallMenuItem 物件代表一個功能表項目，在選取時，會呼叫伺服器程式。ProgramCallMenuItem 類別會擴充 JFC JMenuItem 類別， 以便所有功能表項目也會具有一致的外觀與行為。

若要使用 vaccess 程式呼叫元件，請同時設定系統與程式內容。經由使用建構元或透過 setSystem()與 setProgram 方法來設定這些特性。

下列範例會建立一個 ProgramCallMenuItem。在執行期間，當選取功能表選項時，它會呼叫程式：

```
 // Create the ProgramCallMenuItem
 // object. Assume that "system" is
 // an AS400 object created and
 // initialized elsewhere. The menu
 // item text says "Select Me", and
 // there is no icon.
 ProgramCallMenuItem menuItem = new ProgramCallMenuItem ("Select Me", null,
system);

 // Create a path name object that
 // represents program MYPROG in
 // library MYLIB
 QSYSObjectPathName programName = new QSYSObjectPathName("MYLIB", "MYPROG",
"PGM");

 // Set the name of the program.
 menuItem.setProgram (programName.getPath());

 // Add the menu item to a menu.
 // Assume that the menu was created
 // elsewhere.
 menu.add (menuItem);
```

在執行伺服器程式時，它可能會傳回零或多則伺服器訊息。若要偵測伺服器程式的執行，請使用 addActionCompletedListener() 方法，將ActionCompletedListener新增到按鈕或功能表項目。每當程式執行時，它會傳送一個ActionCompletedEvent給所有這類接收程式。接收程式可以使用 getMessageList() 方法來擷取該程式所產生的任何伺服器訊息。

此範例將新增 ActionCompletedListener 來處理該程式所產生的所有伺服器訊息：

```
 // Add an ActionCompletedListener
 // that is implemented by using an
```

```
 // anonymous inner class. This is a
 // convenient way to specify simple
 // event listeners.
 menuItem.addActionCompletedListener (new ActionCompletedListener ()
     {
public void actionCompleted (ActionCompletedEvent event)
         {
 // Cast the source of the event to a
 // ProgramCallMenuItem.
 ProgramCallMenuItem sourceMenuItem = (ProgramCallMenuItem) event.getSource
();

 // Get the list of server messages
 // that the program generated.
 AS400Message[] messageList = sourceMenuItem.getMessageList ();

 // ... Process the message list.
         }
     });
```

參數

您可以使用ProgramParameter物件，在 Java 程式與伺服器程式之間傳遞參數資料。 輸入資料是使用 setInputData() 方法來設定的。在程式執行後，可以使用getOutputData()方法來擷取輸出資料。

每一個參數都是一個位元組陣列。Java 程式要負責在 Java 與伺服器格式之間轉換位元組陣列資料轉換類別 類別會提供一些方法來轉換資料。

您可以使用 addParameter() 方法，一次將一個參數新增到程式呼叫圖形使用者介面元件中， 或是使用 setParameterList() 方法，一次將所有參數新增到程式呼叫圖形使用者介面元件中。

關於如何使用 ProgramParameter 物件的詳細資訊，請參閱ProgramCall 存取類別。

下列範例會新增兩個參數：

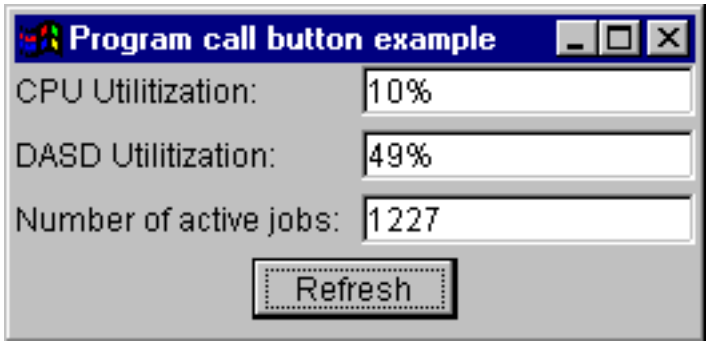
```
 // The first parameter is a String
 // name of up to 100 characters.
 // This is an input parameter.
 // Assume that "name" is a String
 // created and initialized elsewhere.

AS400Text parm1Converter = new AS400Text (100, system.getCcsid (),
system);
 ProgramParameter parm1 = new ProgramParameter
(parm1Converter.toBytes (name));
 menuItem.addParameter (parm1);

 // The second parameter is an Integer
 // output parameter.
 AS400Bin4 parm2Converter = new AS400Bin4 ();
```

```
ProgramParameter parm2 = new ProgramParameter
(parm2Converter.getByteLength ());
 menuItem.addParameter (parm2);

 // ... after the program is called,
 // get the value returned as the
 // second parameter.
 int result = parm2Converter.toInt (parm2.getOutputData
());
```

範例

在應用程式中使用ProgramCallButton的範例。圖 1 顯示 ProgramCallButton 的外觀：

圖 1：在應用程式中使用 ProgramCallButton



Vaccess 記錄層次存取類別

vaccess 套件中的記錄層次存取類別可讓 Java 程式以各種檢視畫面來呈現伺服器檔案。

可用的元件如下：

- RecordListFormPane以套表來呈現伺服器檔案的記錄清單。
- RecordListTablePane以表格來呈現伺服器檔案的記錄清單。
- RecordListTableModel以表格管理伺服器檔案的記錄清單。

索引存取

您可以使用伺服器檔案使用記錄層次存取圖形式使用者介面元件以及索引存取。 索引存取表示 Java 程式可指定索引來存取檔案的記錄。

索引存取與每一個記錄層次存取圖形式使用者介面元件的運作方式是相同的。 請使用 setKeyed() 來指定索引存取，而非循序存取。使用建構元或 setKey() 方法來指定索引鍵。請參閱指定索引鍵 ，取得金鑰的指定方法詳細資訊。

依據預設值，僅有其索引等於指定的索引的記錄才會顯示。若要變更，請使用建構元或 setSearchType() 方法，指定 searchType 特性。可能的選擇如下：

- KEY_EQ - 顯示其索引等於指定的索引的記錄。
- KEY_GE - 顯示其索引大於或等於指定的索引的記錄。
- KEY_GT - 顯示其索引大於指定的索引的記錄。
- KEY_LE - 顯示其索引小於或等於指定的索引的記錄。
- KEY_LT - 顯示其索引小於指定的索引的記錄。

下面範例會建立一個 RecordListTablePane 物件，來顯示小於或等於 某個索引的所有記錄。

```
 // Create a key that contains a
 // single element, the Integer 5.
 Object[] key = new Object[1];
 key[0] = new Integer (5);

 // Create a RecordListTablePane
 // object. Assume that "system" is an
 // AS400 object that is created and
 // initialized elsewhere. Specify
 // the key and search type.
 RecordListTablePane tablePane = new RecordListTablePane (system,
"/QSYS.LIB/QGPL.LIB/PARTS.FILE", key, RecordListTablePane.KEY_LE);

 // Load the file contents.
 tablePane.load ();

 // Add the table pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
```

```
frame.getContentPane ().add (tablePane);
```

RecordListFormPane 類別

RecordListFormPane
以套表呈現伺服器檔案內容。表單會一次顯示一筆記錄，並提供一些按鈕，容許使用者向前捲動、向後捲動到第一筆或最後一筆記錄，或是復新檔案內容的檢視畫面。

若要使用 RecordListFormPane，請設定系統與 fileName 特性。 使用建構子或 setSystem()和 setFileName()方法來設定這些內容。使用 load()來擷取檔案內容並顯示第一筆記錄。當檔案內容不再需要時，請呼叫 close() 以確保結束檔案。

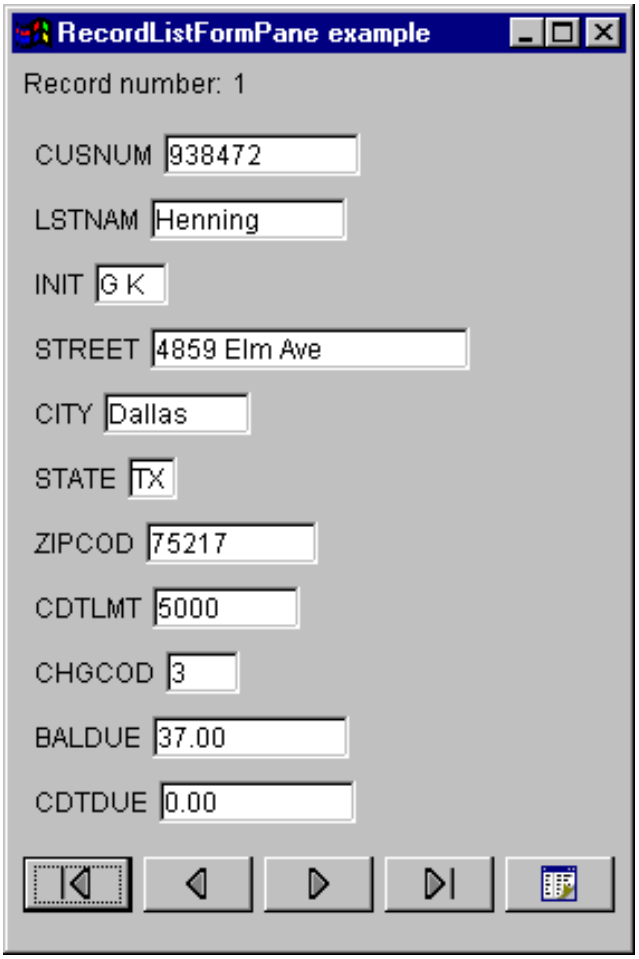
下面範例會建立 RecordListFormPane 物件，並將它新增到頁框中：

```
 // Create a RecordListFormPane
 // object. Assume that "system" is
 // an AS400 object that is created
 // and initialized elsewhere.
 RecordListFormPane formPane = new RecordListFormPane (system,
"/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

 // Load the file contents.
 formPane.load ();

 // Add the form pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (formPane);
```

範例

呈現會顯示檔案內容的RecordListFormPane 圖 1 顯示 RecordListFormPane 圖形式使用者介面元件：

圖 1：RecordListFormPane GUI 元件


RecordListFormPane example
Record number: 1
CUSNUM 938472
LSTNAM Henning
INIT G K
STREET 4859 Elm Ave
CITY Dallas
STATE TX
ZIPCOD 75217
CDTLMT 5000
CHGCOD 3
BALDUE 37.00
CDTDUE 0.00

RecordListTablePane 類別

RecordListTablePane
以表格呈現伺服器檔案內容。表格中的每一列顯示來自檔案的記錄，而每一欄則顯示欄位。

若要使用 RecordListTablePane，請設定系統與 fileName 特性。使用建構子或 setSystem() 和 setFileName() 方法來設定這些內容。使用 load()
來擷取檔案內容，並在表格中呈現記錄。當檔案內容不再需要時，請呼叫 close() 以確保結束檔案。

下面範例會建立 RecordListTablePane 物件，並將它新增到頁框中：

```
 // Create an RecordListTablePane
 // object. Assume that "system" is
 // an AS400 object that is created
 // and initialized elsewhere.
 RecordListTablePane tablePane = new RecordListTablePane (system,
"/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

 // Load the file contents.
 tablePane.load ();

 // Add the table pane to a frame.
 // Assume that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (tablePane);
```

RecordListTablePane 與 RecordListTableModel 類別

RecordListTablePane 是使用模型-概略表-控制器範例來實施，
在此資料與使用者介面被分成不同的類別。此施行將整合 RecordListTableModel 與 Java Foundation Classes (JFC) JTable。RecordListTableModel 類別會擷取並管理檔案內容，而 JTable 則會以圖形方式顯示檔案內容並處理使用者互動。

RecordListTablePane 會提供足夠的功能來符合大部份的需求。不過，如果呼叫程式需要更進一步控制 JFC 元件，則呼叫程式可以直接使用 RecordListTableModel，並提供與不同圖形使用者介面元件的自訂整合。

若要使用 RecordListTableModel，請設定系統與 fileName 特性。 使用建構子或 setSystem() 和 setFileName() 方法來設定這些內容。使用 load() 來擷取檔案內容。當檔案內容不再需要時，請呼叫 close() 以確保結束檔案。

下面範例會建立 RecordListTableModel 物件，並以 JTable 呈現它：

```
 // Create a RecordListTableModel
 // object. Assume that "system" is
 // an AS400 object that is created
 // and initialized elsewhere.
 RecordListTableModel tableModel = new RecordListTableModel (system,
"/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

 // Load the file contents.
 tableModel.load ();

 // Create a JTable for the model.
 JTable
table = new JTable (tableModel);

 // Add the table to a frame. Assume
 // that "frame" is a JFrame
 // created elsewhere.
 frame.getContentPane ().add (table);
```

ResourceListPane 與 ResourceListDetailsPane

使用ResourceListPane和 ResourceListDetailsPane類別，在圖形式使用者介面 (GUI) 中呈現資源清單。

- ResourceListPane 會在圖形式的 javax.swing.JList 中顯示資源清單的內容。清單中所顯示的每個項目代表資源清單中的一個資源物件。
- ResourceListDetailsPane 會在圖形式的 javax.swing.JTable 中顯示資源清單的內容。表格中所顯示的每一列代表資源清單中的一個資源物件。

ResourceListDetailsPane 的表格直欄將指定為直欄屬性 ID 的陣列。
每一個陣列的元素是表格中的一個直欄，而每一個資源物件是表格中的一列。

在預設的情況下，ResourceListPane 和 ResourceListDetailsPane 二者都會啟用蹦現功能表。

大部份的錯誤將報告為com.ibm.as400.vaccess.ErrorEvent，而非丟出異常。因此，請接收 ErrorEvents，來診斷錯誤狀況，並從錯誤狀況中回復。

範例：在 GUI 中顯示資源清單

此範例會針對系統上的所有使用者建立一個 ResourceList，並將其顯示在 GUI 中 (明細窗格)：

```
 // Create the resource list.
 AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
 RUserList userList = new RUserList(system);

 // Create the ResourceListDetailsPane.In this example,
 // there are two columns in the table.The first column
 // contains the icons and names for each user.The
 // second column contains the text description for each
 // user.
 Object[] columnAttributeIDs = new Object[] { null, RUser.TEXT_DESCRIPTION
};
 ResourceListDetailsPane detailsPane = new ResourceListDetailsPane();
 detailsPane.setResourceList(userList);
 detailsPane.setColumnAttributeIDs(columnAttributeIDs);

 // Add the ResourceListDetailsPane to a JFrame and show it.
JFrame frame = new JFrame("My Window");
 frame.getContentPane ().add (detailsPane);
frame.pack();
frame.show();

 // The ResourceListDetailsPane will appear empty until
 // we load it.This gives us control of when the list
 // of users is retrieved from the iSeries.

detailsPane.load ();
```

Vaccess 系統狀態類別

vaccess 套件中的「系統狀態」元件可讓您使用現有的 VSystemPanes，來建立 GUI。您也具有一個選項，使用 Java Foundation Classes (JFC) 建立自己的 GUI。VSystemStatus 物件代表伺服器上的系統狀態。VSystemPool 物件代表伺服器上的系統儲存區。VSystemStatusPane 代表顯示系統狀態資訊的視覺化窗格。

VSystemStatus 類別可讓您在 GUI 環境內，取得伺服器階段作業狀態的相關資訊：

- getSystem() 方法將傳回包含系統狀態資訊的伺服器
- getText() 方法將傳回說明文字
- setSystem() 方法將設定系統狀態資訊所在的伺服器

除了上面提到的方法外，您也可以在 GUI 中存取及變更系統儲存資訊。

您可以將 VSystemStatus 與 VSystemStatusPane 搭配使用。VSystemPane 是顯示系統狀態及系統儲存區資訊的視效窗格。

VSystemStatusPane 類別

VSystemStatusPane類別容許 Java 程式顯示系統狀態及系統儲存區資訊。

VSystemStatusPane 包括下列方法：

- getVSystemStatus()將 VSystemStatus 資訊傳回 VSystemStatusPane 中。
- setAllowModifyAllPools()設定該值，決定是否可以修改系統儲存區資訊。

下面範例將告訴您如何使用 VSystemStatusPane 類別：

```
// Create an as400 object.
AS400 mySystem = new AS400("mySystem.myCompany.com");

// Create a VSystemStatusPane
VSystemStatusPane myPane = new VSystemStatusPane(mySystem);

// Set the value to allow pools to be modified
myPane.setAllowModifyAllPools(true);

//Load the information
myPane.load();
```

系統值 GUI

vaccess 套件中的系統值元件容許 Java 程式藉由使用現存的 AWT Panes，或藉由使用 Java Foundation Classes (JFC) 來建立您自己的窗格，來建立 GUI。VSystemValueList 物件代表伺服器上的系統值清單。

若要使用系統值 GUI 元件，請使用建構元或透過 setSystem() 方法，來設定系統名稱。

範例

下面範例使用 AS400Explorer 窗格建立一個系統值 GUI：

```
//Create an AS400 object
AS400 mySystem = newAS400("mySystem.myCompany.com");
VSystemValueList mySystemValueList = new VSystemValueList(mySystem);
as400Panel=new AS400ExplorerPane((VNode)mySystemValueList);
//Create and load an AS400ExplorerPane object
as400Panel.load();
```

Vaccess 使用者與群組類別

vaccess 套件中的使用者與群組元件可讓您透過 User 類別，來呈現伺服器使用者與群組清單。

可用的元件如下：

- AS400Panes 是一些 GUI 元件， 它們會呈現及容許操作一個或多個伺服器資源。
- VUserList 物件是一種資源，它代表要在 AS400Panes 中使用的伺服器使用者與群組的清單。
- VUserAndGroup 物件是要在 AS400Panes 中使用的資源， 用來代表伺服器使用者的群組。它容許 Java 程式列出所有使用者，列出所有群組或列出不在群組中的使用者。

AS400Pane 與 VUserList 物件可以一起使用，以呈現清單的多種檢視方式。 它們也可以用來容許使用者選取使用者和群組。

若要使用 VUserList，首先您必須設定系統特性。此內容可以使用建構元或透過setSystem()方法來設定。接著使用 AS400Pane 的建構元或 setRoot() 方法， 將 VUserList 物件「插入」AS400Pane 中作為根。

VUserList 含有其它部份有用的內容，可用來定義 AS400Panes 中所呈現的使用者與群組集：

- 使用setUserInfo()方法來指定應該出現的使用者類型。
- 使用setGroupInfo()方法來指定群組名稱。

您可以使用VUserAndGroup 物件，來取得系統上的「使用者與群組」相關資訊。在您能取得特定物件的相關資訊之前，您需要該資訊，才能存取它。 您可以使用 getSystem 方法，來顯示能在其中找到該資訊的伺服器。

當建立 AS400Pane 物件與 VUserList 或 VUserAndGroup 物件時，它們會起始設定為預設狀態。尚未載入使用者和群組的清單。若要載入內容，Java 程式必須在任一物件中明確地呼叫 load() 方法，來起始與伺服器的通信，以便收集清單的內容。

在執行時間，以滑鼠右鍵按一下使用者、使用者清單或群組以顯示捷徑功能表。從捷徑功能表中選取 針對選取的物件執行動作：

- 使用者 -
  顯示使用者資訊的清單，包括說明、使用者類別、狀態、工作說明、輸出資訊、訊息資訊、國際資訊、安全資訊及群組資訊。
- 使用者清單 - 處理使用者資訊和群組資訊內容。您也可以變更清單的內容。
- 使用者與群組 - 顯示內容，例如使用者名稱及說明。

使用者僅能存取他們有權使用的使用者和群組。此外，Java 程式會在窗格中使用 setAllowActions()方法， 阻止使用者執行動作。

下面範例會建立 VUserList 並在 AS400DetailsPane 中呈現它：

```
// Create the VUserList object.
// Assume that "system" is an AS400
// object created and initialized
// elsewhere.
VUserList root = new VUserList (system);

// Create and load an
// AS400DetailsPane object.
AS400DetailsPane detailsPane = new AS400DetailsPane (root);
detailsPane.load ();

// Add the details pane to a frame.
// Assume that "frame" is a JFrame
// created elsewhere.
frame.getContentPane ().add (detailsPane);
```

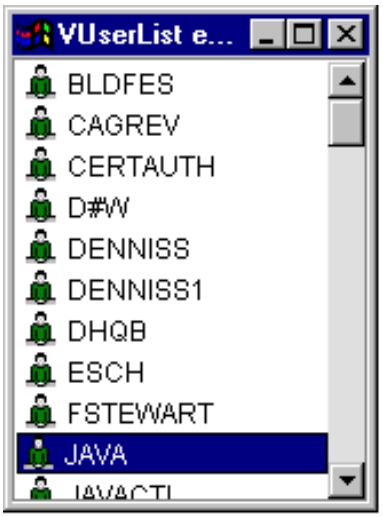
下列的範例顯示如何使用 VUserAndGroup 物件：

```
// Create the VUserAndGroup object.
// Assume that "system" is an AS400 object created and initialized
elsewhere.
VUserAndGroup root = new VUserAndGroup(system);

// Create and Load an AS400ExplorerPane
AS400ExplorerPane explorerPane = new AS400ExplorerPane(root);
explorerPane.load();

// Add the explorer pane to a frame
// Assume that "frame" is a JFrame created elsewhere
frame.getContentPane().add(explorerPane);
```

其它範例

使用 AS400ListPane 搭配 VUserList 物件， 呈現系統中的使用者清單。

下面影像顯示 VUserList 圖形使用者介面元件：



Graphical Toolbox

Graphical Toolbox 是一套 UI 工具集，能輕鬆地使用 Java 建立自訂的使用者介面畫面。 您可以把畫面納入 Java 應用程式、applet、或「iSeries 領航員」外掛程式。畫面可能包含由 iSeries 所取得的資料，或由其它來源取得的資料，如本端檔案系統中的檔案，或網路上的程式。

GUI Builder是 WYSIWYG 視覺編輯器，可用來建立 Java 對話框、 內容表和精靈。利用 GUI Builder，您可以新增、排列或編輯畫面中的 使用者介面控制項，然後預覽畫面以驗證佈置結果是否如您預期。您所建立的畫面定義可以用於對話框中、插入內容表及精靈中， 或放入分割窗格、重疊窗格及標籤窗格。GUI Builder 也可讓您建置功能表列、工具列及上下文功能表 定義。您還可以把 JavaHelp 納入畫面中，包括上下文相關的說明在內。

Resource Script Converte會將 Windows resource script 轉換為 Java 程式可以使用的 XML 表示法。 利用 Resource Script Converter， 您可以在您現存的 Windows 對話框及功能表中處理「Windows 資源 script (RC 檔)」。然後，這些已轉換的檔案可以使用 GUI Builder 加以編輯。使用 Resource Script Converter 及 GUI Builder，可以從 RC 檔製作特性表及精靈。

這兩種工具背後的基礎在於名為nel Definition Markup Language或PDML 的新技術。PDML 是基於 Extensible Markup Language (XML)，它定義出說明使用者介面元素佈置方式用的獨立式平台語言。 只要於 PDML 中定義您的畫面，您就可以使用由 Graphical Toolbox 所提供的 執行時間 API 來顯示畫面。API 會經由解譯 PDML， 並使用 Java Foundation Classes 轉譯您的使用者介面，來顯示您的畫面。

Graphical Toolbox的好處

撰寫的程式較少，並可節省時間

使用 Graphical Toolbox，您就可以迅速地、簡易地建立 Java 使用者介面。GUI Builder 可讓您精確地控制畫面中 UI 元素的佈置情形。因為佈置是 以 PDML 來說明，所以您不需要為了定義使用者介面而開發任何 Java 程式碼，也不需要為了進行變更而重新編譯程式碼。因此，只需要以 相當少的時間，即可建立及維護您的 Java 應用程式。Resource Script Converter 可讓您迅速且輕易地將大量的 Windows 畫面移轉到 Java。

自訂解說

於 PDML 中定義使用者介面可帶來其它好處。因為一張畫面的所有資訊都合併成正式的標記語言，所以工具可以強化為代替程式開發者 來執行更多的服務。例如，GUI Builder 和 Resource Script Converter 都能 夠產生畫面線上說明的 HTML 基本架構。由您決定需要哪些解說主題，解說主題就會自動按照您的要求建立起來。解說主題的錨點標籤會直接建置於說明的基本架構當中，此有助於撰寫者專注於開發適當的內容。Graphical Toolbox 執行時間環境會回應使用者的要求，自動顯示出正確的解說主題。

畫面到程式的自動整合

此外 PDML 還能提供一些標記，使畫面中的每一個控制項與 JavaBean 中的屬性產生關聯。一旦您已識別出會提供資料給畫面的 bean 類別，並且也將每個適當的控制項與屬性相結合，便可要求工具產生 bean 物件 的 Java 原始程式架構。到了執行時間，「圖形化工具箱」會自動在 bean 及 您所識別的畫面控制項之間轉送資料。

平台獨立

Graphical Toolbox 執行時間環境可提供事件處理的支援、使用者資料驗證，以及畫面元素之間的一般互動類型。您的使用者介面之正確平台外觀及風格將 自動依據基本的作業系統來設定，而且 GUI Builder 可讓您在外觀及風格之間切換， 以便讓您在不同的平台中評估畫面的外觀效果。

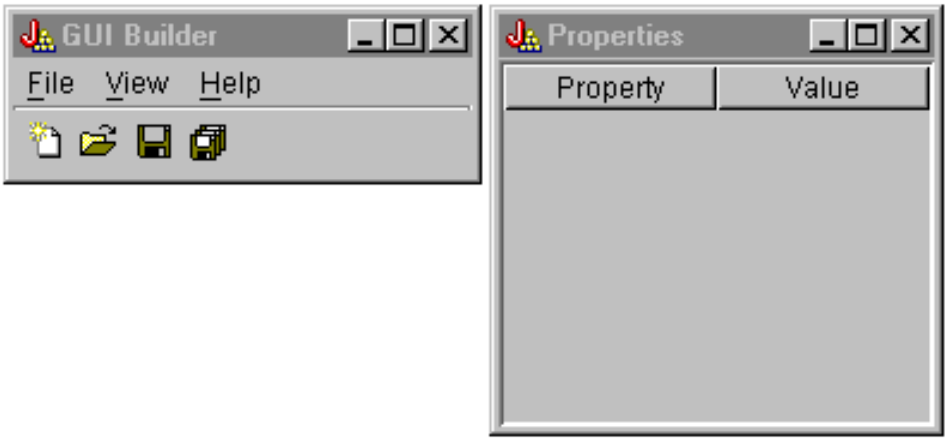
Graphical Toolbox 之內

Graphical Toolbox 提供您兩種工具，因此，有兩種自動建立使用者介面的方法。 您可以使用 GUI Builder 以迅速且輕易地塗銷後建立新畫面，或您可以 使用 Resource Script Converter，將現存的以 Windows 為基礎的畫面 轉換為 Java。然後就可以使用 GUI Builder 編輯已轉換的檔案。 這兩種工具都支援國際化。

GUI Builder

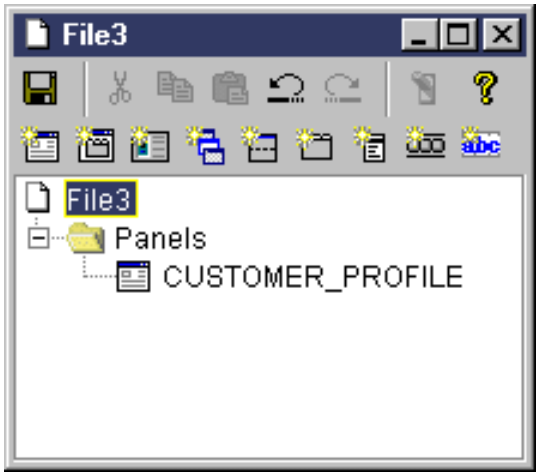
當您第一次呼叫 GUI Builder 時，會顯示兩個視窗，如圖 1 所示：

圖 1： GUI Builder 視窗



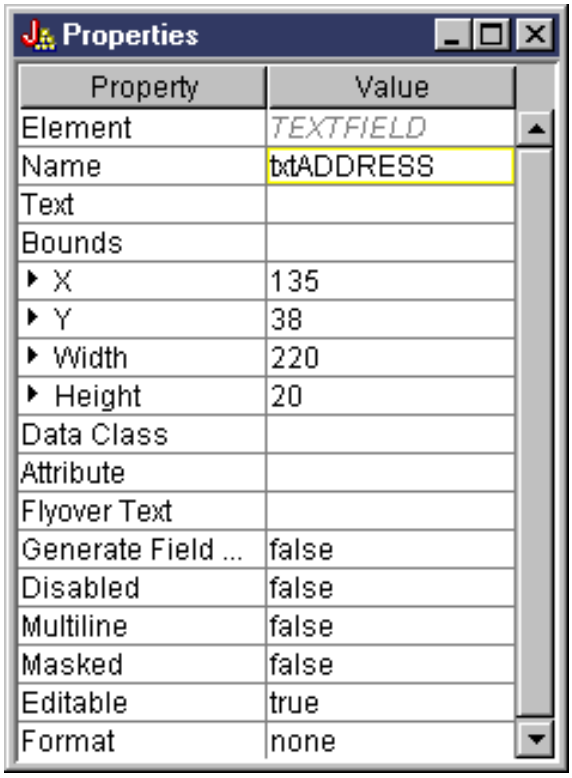
請使用 File Builder 視窗建立及編輯您的 PDML 檔。

圖 2：File Builder 視窗



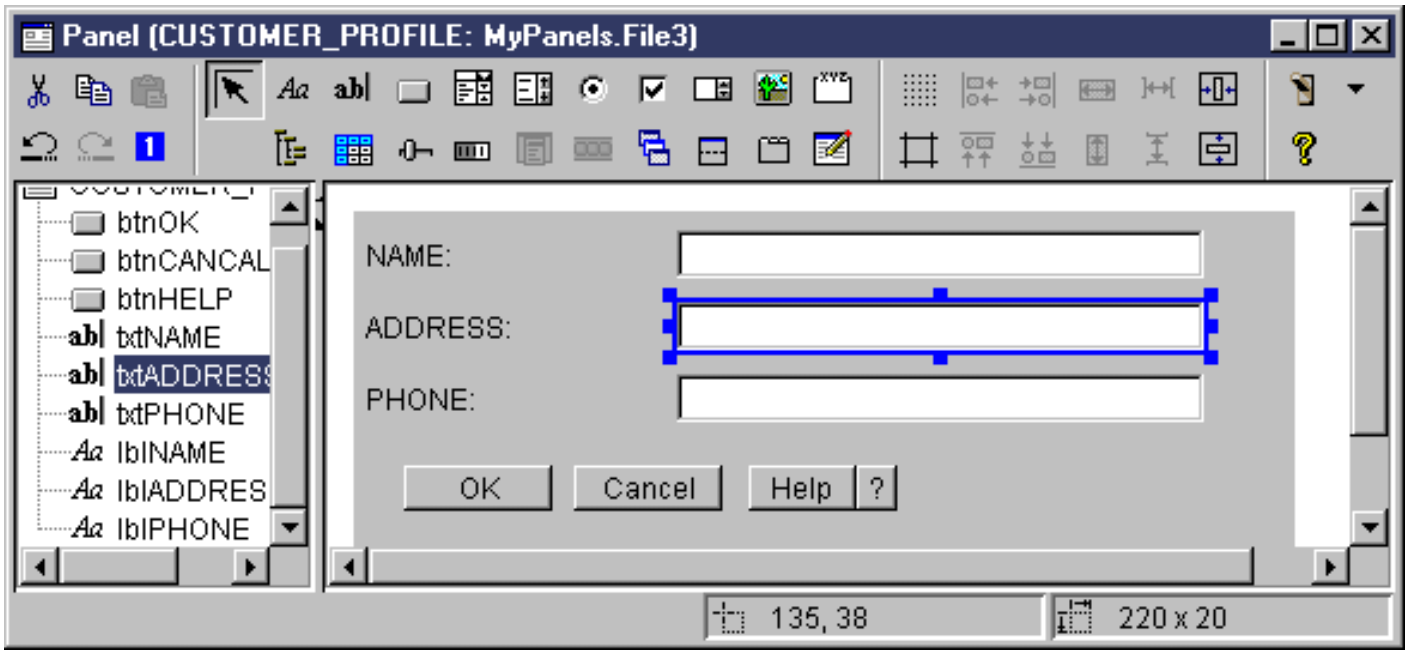
請使用 Properties 視窗檢視或變更目前選取的控制項的內容。

圖 3：Properties 視窗



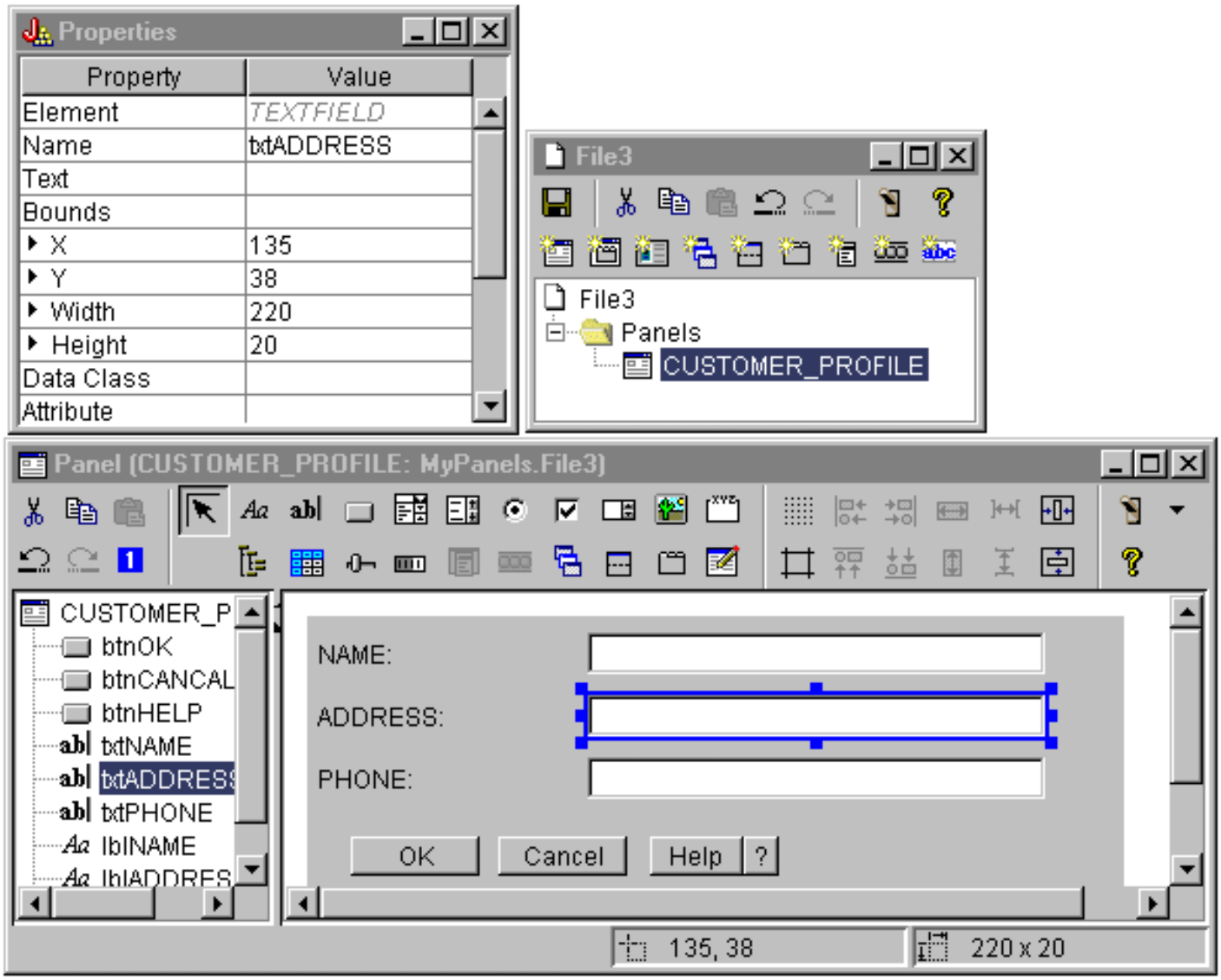
請使用 Panel Builder 視窗建立及編輯您的圖形式使用者介面元件。
從工具列中選取想要的元件，然後按一下畫面，將它放在您要的位置上。
工具列也提供了一些機能，以對齊控制項群組、預覽畫面及要求 GUI Builder 功能的線上說明。 請參閱 Builder Panel Builder 工具的說明以取得每一個圖示功能的說明。

圖 4：Panel Builder 視窗



在 Panel Builder 視窗中會顯示要編輯的畫面。 圖 5 顯示各視窗如何一起作用：

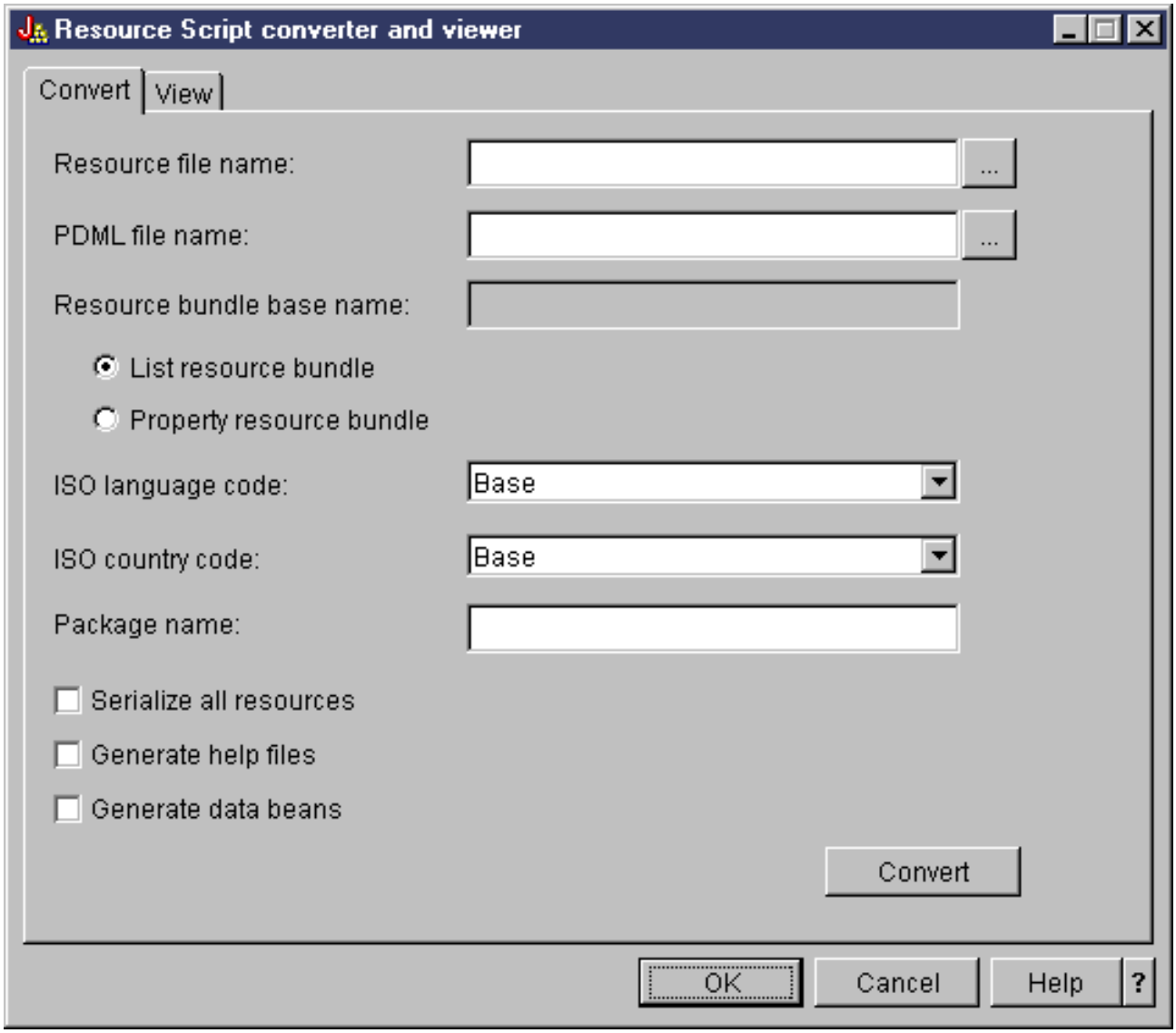
圖 5：GUI Builder 視窗一起作用的範例



Resource Script Converter

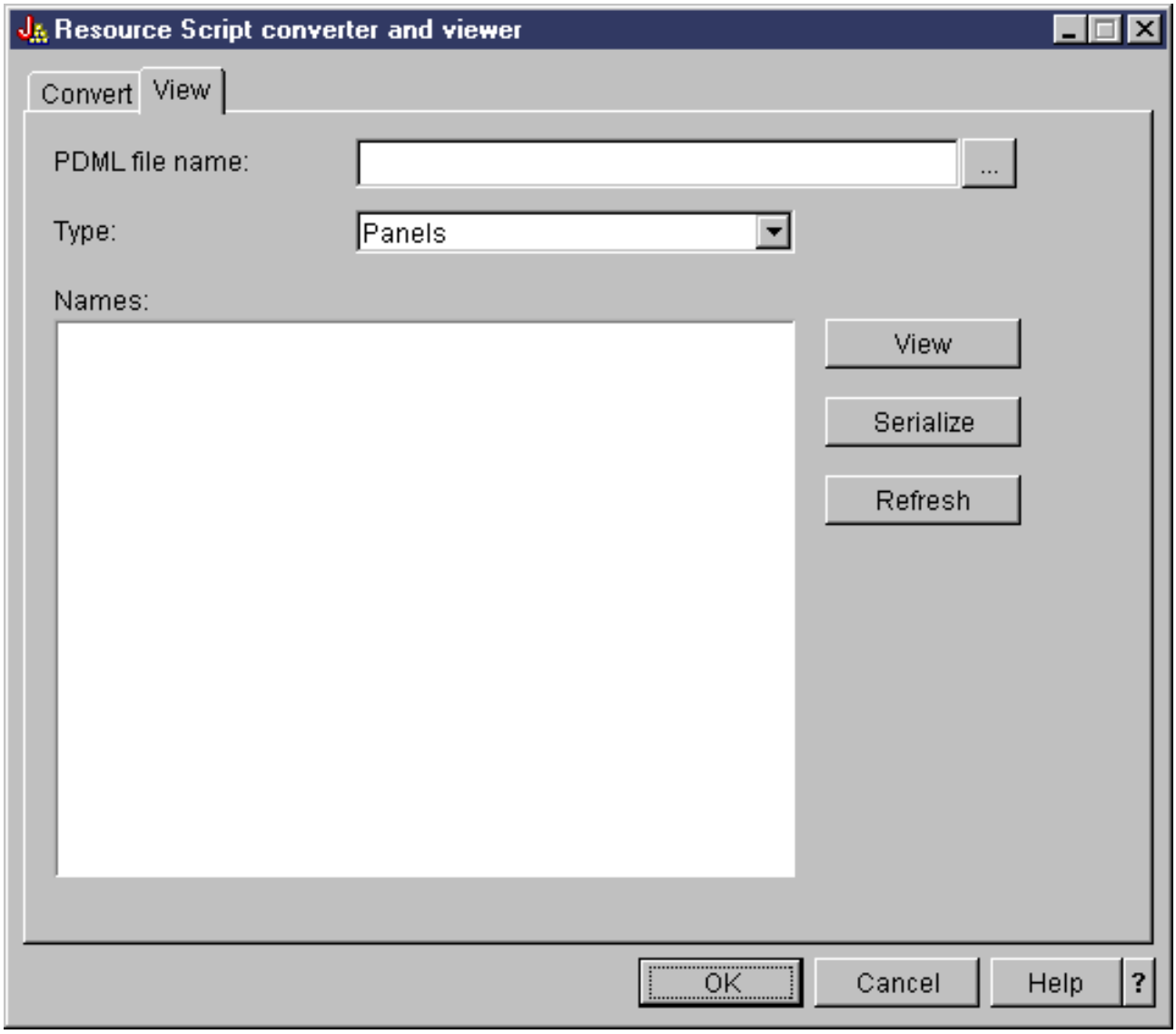
Resource Script Converter 由兩個窗格的含標籤對話框所構成。 於 Convert窗格上，您可以指定要轉換成 PDML 的 Microsoft 或 VisualAge for Windows RC 檔案名稱。 您可以指定目標 PDML 檔的名稱，及將含有畫面的已轉換字串的相關 Java 資源組。 此外，您可以要求為畫面產生線上說明架構，產生提供資料給畫面的物件的 Java 原始程式碼架構，以及將畫面定義序列化，以改善執行時間的效能。Converter 的線上說明提供 Conver 窗格上每一 輸入欄位的詳細說明。

圖 6：Resource Script Converter 的 Convert 窗格



轉換順利執行後，您可以使用View 窗格來 檢視新建立的 PDML 檔的內容，以及預覽新的 Java 畫面。如果需要，您可使用 GUI Builder 來稍微調整 畫面。在執行轉換之前，Converter 一律會檢查現存的 PDML 檔，並嘗試保留任何變更，以便您在稍後執行轉換時可能需要它。

圖 7：Resource Script Converter 檢視窗格



開始使用 Graphical Toolbox

您可以使用下列主題，來更瞭解 Graphical Toolbox：

- 設置 Graphical Toolbox
- 建立使用者介面
- 執行時顯示畫面
- 產生線上說明檔
- Graphical Toolbox範例
- 在瀏覽器中使用 Graphical Toolbox
- 畫面建置器工具列

設置 Graphical Toolbox

Graphical Toolbox 是以一組 JAR 檔來傳遞。 如欲設定 Graphical Toolbox, 您必須在您的工作站上安裝 JAR 檔, 並設定您的 CLASSPATH 環境變數。

您也必須確定您的工作站符合執行 IBM Toolbox for Java 的基本要求。

在您的工作站上安裝 Graphical Toolbox

如欲使用 Graphical Toolbox 開發 Java 程式, 請先在您的工作站上 安裝 Graphical Toolbox JAR 檔。方法有下列幾種：

轉送 JAR 檔

附註：以下的清單代表轉送 JAR 檔的幾種不同方法。您的 iSeries 上必須安裝 IBM Toolbox for Java 授權程式。 此外, 也需要從Sun JavaHelp 網站下載 JavaHelp 的 JAR 檔, 也就是 jhall.jar。

- 使用 FTP (確定您是以二進位模式轉送檔案), 並從目錄 /QIBM/ProdData/HTTP/Public/jt400/lib複製 JAR 檔到您工作站上的本端目錄中
- 使用 iSeries Access for Windows 來對映網路磁碟機。
- 也可以使用 IBM Toolbox for Java 所隨附的 AS400ToolboxInstaller 類別來 安裝 Graphical Toolbox JAR 檔 - 請指定資料包名稱為 OPNAV。
  詳細資訊, 請參閱 從屬站安裝與更新類別。

使用 iSeries Access for Windows 安裝 JAR 檔

您也可以在安裝 iSeries Access for Windows 的時候安裝 Graphical Toolbox。IBM Toolbox for Java 現在隨附於 iSeries Access for Windows 供售銷售。 如果您是要初次安裝 iSeries Access for Windows, 請選擇「自訂安裝」後 再選取安裝功能表上的IBM Toolbox for Java元件。如果您已安裝了 iSeries Access for Windows, 則可以使用「選擇性安裝」 程式來安裝這個元件 (若它未顯示的話)。

設定 classpath

若要使用 Graphical Toolbox, 您必須新增這些 JAR 檔到您 的 CLASSPATH 環境變數中 (或在指令行上的 classpath 選項中指定它們)。

例如, 如果您已經把這些檔案複製到工作站的C:\gtbox\lib目錄下, 則必須新增下列路徑名稱到 classpath：

```
C:\gtbox\lib\uitools.jar;
C:\gtbox\lib\jui400.jar;
C:\gtbox\lib\data400.jar;
C:\gtbox\lib\util400.jar;
C:\gtbox\lib\x4j400.jar;
C:\gtbox\lib\jhall.jar;
```

如果您已使用 iSeries Access for Windows 安裝了 Graphical Toolbox, 則 JAR 檔 (jhall.jar 除外) 將全部常 iSeries Access for Windows 所 安裝之磁碟機的gram Files\Ibm\Client Access\jt400目錄中。iSeries Access for Windows 會將 jhall.jar 安裝在gram Files\Ibm\Client Access\jre目錄中。 您的 classpath 中的路徑名稱應該對此有所反映。

JAR 檔案說明

- uitools.jar含有 GUI Builder 與 Resource Script Converter 工具。
- jui400.jar含有 Graphical Toolbox 的 API執行時間。Java 程式會使用這個 API，顯示透過工具建構的畫面
  這些類別可隨著應用程式一起重新分送。
- data400.jar含有 Program Call Markup Language (PCML) 的執行時間 API。Java 程式使用此 API
  來呼叫其參數及傳回值是使用 PCML 來識別的 iSeries
  程式。這些類別可隨著應用程式一起重新分送。
- util400.jar含有格式化 iSeries 資料及處理 iSeries 訊息的公用程式類別。
  這些類別可隨著應用程式一起分送。
- x4j400.jar含有 API 類別使用的 XML剖析器，來解譯 PDML 與 PCML 文件。
- jhall.jar含有 JavaHelp 類別， 這些類別可為您使用 GUI Builder
  所建置成的畫面顯示出線上說明和上下文相關說明。

註：您可以使用國際化版本的 GUI Builder 與「資源 Script 轉換器」工具。若要執行非美式英文的版本，您必須
Graphical Toolbox 安裝作業中新增適合您語言及國家或地區所使用的正確版本的uitools.jariSeries 伺服器 的
/QIBM/ProdData/HTTP/Public/jt400/Mri29xx 提供了這些 JAR 檔，其中 29xx
就是與您的語言及國家或地區對應的 4 位數 OS/400 NLV 字碼。(各種 Mri29xx 目錄中的 JAR 檔名包含代表 Java
語言碼與國家或地區碼的 2 字元字尾)。 這個額外的 JAR 檔應該新增到您的 classpath 中，並且在搜尋次序中放在
uitools.jar之前。

使用 Graphical Toolbox

一旦安裝了 Graphical Toolbox 後，請遵循這些鏈結來學習如何使用工具：

- 使用 GUI 建置器
- 使用 Resource Script Converter

建立您的使用者介面

執行 GUI Builder

若要啟動 GUI Builder，請按下列呼叫 Java 直譯器：

```
java com.ibm.as400.ui.tools.GUI Builder [-plaf          look and feel   ]
```

如果您未設定 CLASSPATH 環境變數來包含 Graphical Toolbox JAR 檔，則您須要在指令行上，使用classpath 選項來指定它們。 請參閱設置 Graphical Toolbox

選項 -plaf look and feel

> 理想的平台外觀和感覺。此選項可讓您置換預設的外觀與感覺，該
> 預設值是依照您目前開發的平台所設定，您可以預覽畫面，察看它們在不同的
> 作業系統平台上看起來的樣子。系統可接受下列外觀與感覺的值：
>
> - Windows
> - Metal
> - Motif
>
> 目前，Swing 1.1 可以支援的其它外觀與感覺的屬性並不為 GUI Builder 所支援。

使用者介面資源的類型

當您第一次啟動 GUI Builder 時，必須建立新的 PDML 檔。在 GUI Builder 主視窗的功能表列上，i隨選取New File。 建立新的 PDML 檔案之後，您就可以隨意定義下列型類的 UI 資源包含在內。

Panel (畫面)
> 基本資源類型。描述一個矩形區，UI 元素就在這個區域內排列。UI
> 元素可以由簡式控制項所組成，如圓鈕或文字欄位、影像、動畫、 自訂控制項或更複雜的次畫面
> (請參閱底下的分割窗格、疊窗格及標籤窗格)。一個畫面可以定義獨立式視窗或對話框的佈置，或者也可以定義
> 在另一個 UI 資源中包含的次畫面之一。

Menu (功能表)
> 蹦現視窗含有一或多個可選取的動作，每一個都由一個字串代 表 (如 Cut、Copy 及
> Paste)。您可以定義每一個動作的助記符號及快速鍵、
> 插入分隔字元及階式排列子功能表，或定義特殊的勾選或圓鈕功能表項目。
> 功能表資源可以當作獨立式上下文功能表、功能表列中的下拉式功能表使用， 或它可以自己定義與畫面資源相關的功能表列

Toolbar (工具列)
> 視窗是由一系列的按鈕組成，每一個按鈕都代表了可能的使用者動作。 每一個按鈕可以含有文字、圖示或兩者。
> 您可以將工具列定義為可浮動，讓使用者可以將工具列拖曳出畫面， 進入獨立式視窗。

Property Sheet (內容表)
> 獨立式視窗或對話框是由標籤畫面及 OK、Cancel 及 Help 按鈕 所組成。畫面資源會定義每一個標籤視窗的佈置。

Wizard (精靈)
> 由一系列按預定順序顯示給使用者的畫面所組的獨立式視窗或對話框，包括 Back、Next、Cancel、Finish 和 Help
> 按鈕。Wizard (精靈) 視窗也會在畫面左邊顯示作業列示，透過精靈來追蹤使用的進度。

Split Pane (分割窗格)
> 由兩個以分割列區隔的畫面所組成的次畫面。畫面可以水平或垂直排列。

Tabbed Pane (標籤窗格)
> 形成一標籤控制項的子窗格。此標籤控制項可以放在另一個畫面、分割窗格或重疊窗格中。

Deck Pane (重疊窗格)
> 由畫面集合所組成的子窗格。對於這些窗格，一次只能顯示一個畫面。
> 例如，重疊窗格在執行時會依照所提供的使用者動作，變更顯示的畫面。

String Table (字串表格)
> 字串資源及其相關的資源識別字的集合。

產生的檔案

畫面的可翻譯字串並不儲存在 PDML 檔案本身，而是儲存在另外的 Java 資源組。 這些工具可讓您指定資源組的定義方式，可以是 Java PROPERTIES 檔， 或是 ListResourceBundle 次類別。ListResourceBundle 次類別為可轉換資源的 已編譯版，可增強 Java 應用程式的效能。不過，由於 ListResourceBundle 會於 每次儲存作業中編譯，因此會導致 GUI Builder 儲存處理程序變慢。因此，最好 以 PROPERTIES 檔（預設設定）啟動，直到您滿意您的使用者介面設計為止。

您可以使用工具來產生 PDML 檔中每一個畫面的 HTML 架構。 在執行時間，當焦點為在畫面的其中一個控制項上時，在使用者按一下畫面的「說明」 按鈕或按下 F1 時，就會顯示正確的說明主題。 您應在 HTML 中的適當位置上插入說明內容，也就是在 <!-HELPDOC:SEGMENTBEGIN --> 及 <!-HELPDOC:SEGMENTEND --> 標記範圍內。如需其他特定說明 資訊，請參閱 編輯由 GUI Builder 產生的說明文件。

您可以產生將提供資料給畫面的 JavaBeans 的來源程式架構。您可以使用 GUI Builder 的「內容」視窗，替將含有資料的 控制項增 DATACLASS 與 ATTRIBUTE 內容。DATACLASS 內容會識 別 bean 的類別名稱，而且 ATTRIBUTE 內容會指定 bean 類別實施 的 gettor/settor 方法的名稱。一旦您新增了此資訊到 PDML 檔，您便可以使用 GUI Builder 來產生 Java 來源程式架構並編譯它們。執行時， 將呼叫適當的 gettor/settor 方法來填寫畫面的資料。

備註： gettor/settor 方法的數量及類型取決於與 方法相關的 UI 控制項的類型。每一個控制項的方法通訊協定都記錄在 DataBean 類別的類別說明中。

最後，您可以將 PDML 檔案的內容序列化。序列化將產生檔案中所有 UI 資料 的緊密二進位表示法。這將大大地改善使用者介面的效能，因為 PDML 檔不必解譯，即可顯示您的畫面。

彙總：如果已建立一個名為 MyPanels.pdml 的 PDML 檔， 則也會依據您在工具上選取的選項，產生下列檔案：

- MyPanels.properties，若您已定義資源組作為 PROPERTIES 檔的話
- MyPanels.java 與MyPanels.class，若您已定義資源組作為 ListResourceBundle 次類別的話
- PDML 檔中每一畫面的 <panel name>.html，若您已選取要產生線上說明架構的話
- 您在 DATACLASS 內容上指定的每一個唯一的 bean 類別的 <dataclass name>.java及<dataclass name>.class，若您已選取要產生您的 JavaBeans 的來源程式架構
- PDML 檔中定義的每一個 UI 資源 <resource name>.pdml.ser，若已選取要對它的內容進行序列化的話

備註： 如果畫面名稱和條件性行為功能所要連接的畫面名稱 相同，則條件性行為功能 (SELECTED/DESELECTED) 將無法運作。 例如，如果在 FILE1 的 PANEL1 有一個條件性行為參照連接到某一欄位，而 該欄位參照了 FILE2 中 PANEL1 的欄位，則條件性行為事件將不會運作。 如欲修正此情況，只要將 FILE2 中的 PANEL1 更名，然後更新 FILE1 中的條件性行為事件以反映此變更。

執行 Resource Script Converter

若要啟動 Resource Script Converter，請按照下列呼叫 Java 直譯器：

```
java com.ibm.as400.ui.tools.PDMLViewer
```

如果您未設定 CLASSPATH 環境變數來包含 Graphical Toolbox JAR 檔， 則您將需要在指令行上，使用 classpath 選項來指定它們。請參閱 設置 Graphical Toolbox。

您也可以使用下列指令，在批次模式中執行 Resource Script Converter：

java com.ibm.as400.ui.tools.RC2XML file [options]

其中file 是要處理的資料 script (RC 檔) 的名稱選項

-x name
產生的 PDML 的名稱。預設名稱為要處理的 RC 檔的名稱。

-p name
產生的 PROPERTIES 的名稱。預設名稱為 PDML 檔的名稱。

-r name
產生的 ListResourceBundle 的名稱。預設名稱為 PDML 檔的名稱。

-package name
將對其指定產生的資源的資料包的名稱。若未指定, 將不會產生任何資料包陳述式。

-l locale
產生資源的語言環境。如果指定了語言環境, 則適當的 2 字元 ISO 語言及國家或地區碼 就是產生的資源束名稱的字尾。

-h
產生線上說明的 HTML 架構。

-d
產生 JavaBeans 的來源程式架構。

-s
將所有資源序列化。

對映 Windows 資源到 PDML

所有在 RC 檔中找到的對話框、功能表及字串表, 都會在已產生 的 PDML 檔中轉換為相對應的 Graphical Toolbox 資源。您也可以
Windows 控制項的 DATACLASS 及 ATTRIBUTE 內容, 則當您建立 Windows 資源 的 ID
時, 這些內容會遵照簡單的命名慣例, 延伸到新的 PDML 檔。當您 執行慣例時, 這些內容可以用來產生 JavaBeans
的原始程式架構。

Windows 資源識別字的命名慣例為:

IDCB_<class name>_<attribute>

其中<class name> 是您想要指定為控制項的 DATACLASS 內容的 bean 類別 的完整名稱, 而attribute>
則是您想要指定為控制項的 ATTRIBUTE 內容的 bean 內容的名稱。

例如, 具有資源 IDCB_com_MyCompany_MyPackage_MyBean_SampleAttribute 的 Windows 文字 欄位將產生
com.MyCompany.MyPackage.MyBean 的 DATACLASS 內容, 以及產生SampleAttribute的 ATTRIBUTE
內容。當您執行轉換時, 如果您選取要產生 JavaBeans, 則將產生 JAVA 原始程式 MyBean.java, 它含有 資料句陳述式package
com.MyCompany.MyPackage , 以及 SampleAttribute內容 的 gettor 與 settor 內容。

於執行時間顯示畫面

Graphical Toolbox 提供一種可重新分送的 API，您的 Java 程式可使用它， 以顯示使用 PDML 定義的使用者介面畫面。API 會經
PDML， 並使用「Java Foundation 類別」轉譯您的使用者介面，來顯示您的畫面。

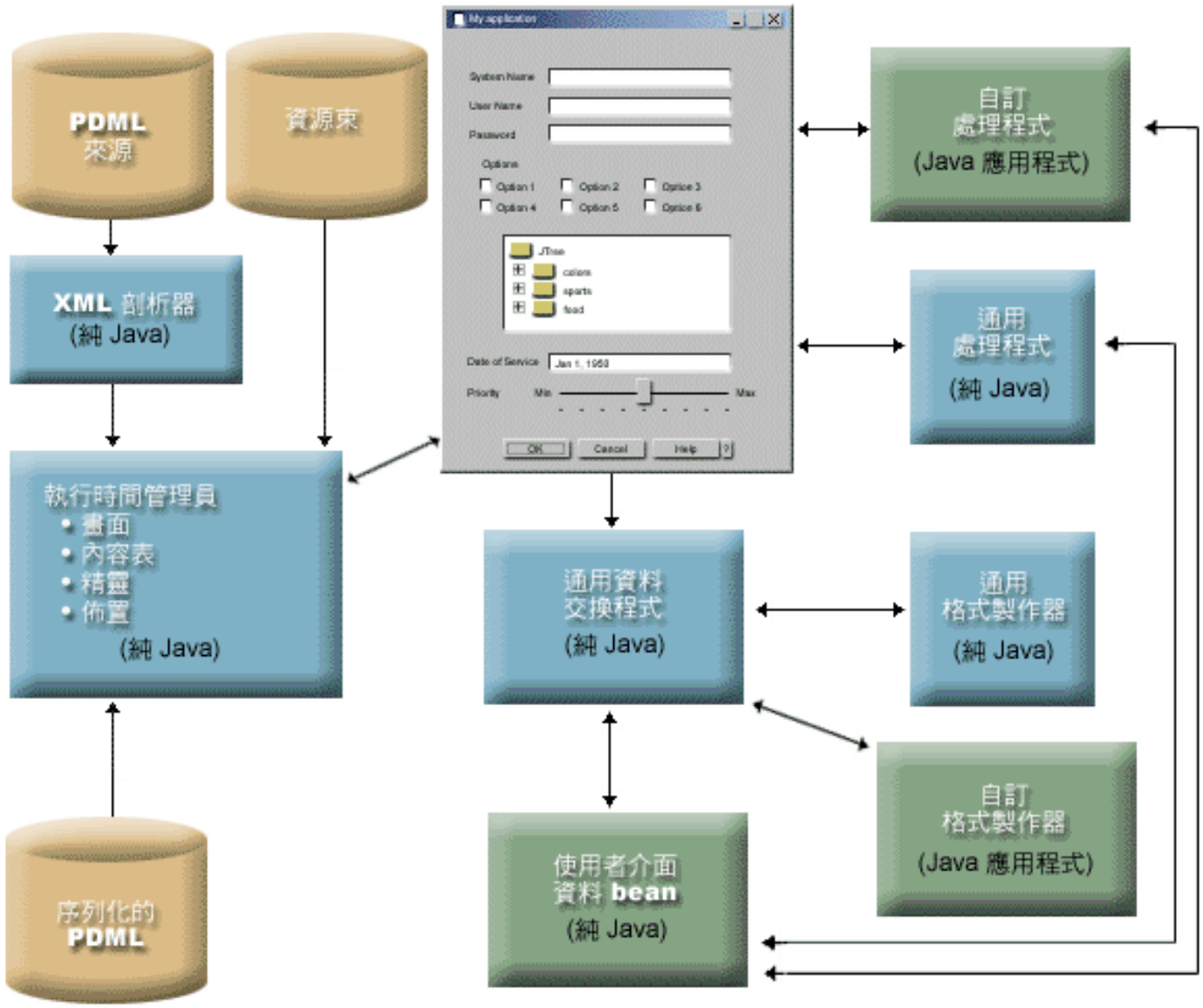
Graphical Toolbox 執行時間環境提供下列服務：

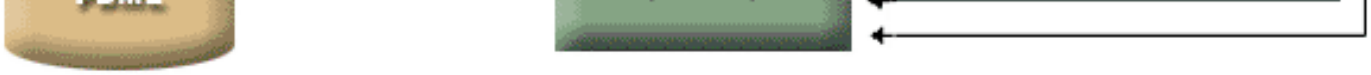
- 處理使用者介面控制與您在 PDML 中所識別的 JavaBeans 之間的所有資料交換。
- 執行一般整數及字元資料類型的使用者資料的驗證，並定義一個介面，
  讓您實施自訂驗證。如果發現資料無效，將顯示錯誤訊息給使用者。
- 定義 Commit、Cancel 與 Help 事件的標準化處理，並提供一個組織架構來處理自訂事件。
- 依據 PDML 中定義的狀態資訊來管理使用者介面控制之間的互動。(例如，每當使用者選取一個特殊圓鈕時，
  您可能想要停用一群控制。)

com.ibm.as400.ui.framework.java 套件中含有 Graphical Toolbox 執行時間 API。

Graphical Toolbox 執行環境的元素如圖所示。 您的 Java 程式與執行時間管理程式方框中一個或多個物件的從屬站。

圖 1：Graphical Toolbox 執行時間環境





範例

假設畫面yPanel 定義於檔案 TestPanels.pdml中，而內容檔 TestPanels.properties與畫面定義相關。
這兩個檔案常駐在目錄 com/ourCompany/ourPackage 中，可從 classpath 中定義的目錄或從 classpath 中定義的 ZIP 或 JAR 檔存取它們。

範例：建立及顯示畫面

下列程式將建立並且顯示出畫面：

```
import com.ibm.as400.ui.framework.java.*;

// Create the panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans omitted

PanelManager pm = null;
      try {
  pm = new PanelManager("com.ourCompany.ourPackage.TestPanels",
                        "MyPanel", null);
}

catch (DisplayManagerException e) {
  e.displayUserMessage(null);
        System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

範例：建立對話框

一旦實施了提供資料給畫面的 DataBeans，以及已在 PDML 中識別了這些屬性後，下列程式可用來建構完整功能的對話框：

```
import com.ibm.as400.ui.framework.java.*;
import java.awt.Frame;

// Instantiate the objects which supply data to the panel
TestDataBean1 db1 = new TestDataBean1();
TestDataBean2 db2 = new TestDataBean2();

// Initialize the objects
db1.load();
db2.load();

// Set up to pass the objects to the UI framework
DataBean[] dataBeans = { db1, db2 };

// Create the panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans
// 4. Owner frame window

Frame owner;
...
```

```
PanelManager pm = null;
      try {
  pm = new PanelManager("com.ourCompany.ourPackage.TestPanels",
                        "MyPanel", dataBeans, owner);
}

catch (DisplayManagerException e) {
  e.displayUserMessage(null);
        System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

範例：使用動態畫面管理員

現存的畫面管理程式已新增一個新的服務。動態的畫面管理程式會在執行時動態調整 畫面。讓我們利用動態畫面管理程式，再看一
MyPanel **範例：**

```
import com.ibm.as400.ui.framework.java.*;

// Create the dynamic panel manager. Parameters:
// 1. Resource name of the panel definition
// 2. Name of panel
// 3. List of DataBeans omitted

DynamicPanelManager dpm = null;
      try {
  pm = new DynamicPanelManager("com.ourCompany.ourPackage.TestPanels",
                               "MyPanel", null);
}

catch (DisplayManagerException e) {
  e.displayUserMessage(null);
        System.exit(-1);
}

// Display the panel
pm.setVisible(true);
```

當您將此畫面應用程式個體化後，您可以看到畫面的動態調整特性。 請將您的游標移到 GUI
顯示畫面的邊緣，當您看見調整大小的箭頭時，您就可以變更畫面的大小。

編輯由 GUI Builder 產生的說明文件

對每一個 PDML 專案檔, GUI Builder 會產生一個說明架構, 並將它放入單一的 HTML 文件中。在使用之前, 針對 PDML 專案的每一個對話, 此 HTML 檔會分成單一主題 的 HTML 檔。這可在使用者使用每一個主題時提供粗略的協助, 並可讓您只管理 少數大型的說明檔。

Help Document 是有效的 HTML 檔, 且可以在任何瀏覽器中檢視及使用大部份 的 HTML 編輯器加以編輯。Help Document 中定義區段的標示是內嵌在註解間, 所以不會在瀏覽器中顯示。註解標示是用來將 Help Document 分成數段：

- 標頭
- 每一對話的主題區段
- 每一個解說啟用控制項的主題區段
- 標底

此外, 您可以在標底之前新增其它的主題區段, 以提供其餘資訊或一般資訊。當標頭及標底建立時, 主題區段在分割之前僅有 html 主體。 當 Help Document 分割後, 處理器會新增標頭及標底到主題區段, 以製作完整的 HTML 檔。在 Help Document 中的標頭及標底會用來作為預設的標頭及標底。 然而, 您可以以您自設的標頭置換預設的標頭。

在 Help Document 內

下列各節會說明 Help Document 的組成部分：

標頭

標頭區段的末端會顯示下列標示：
<!-HELPDOC:HEADEREND -->
如果您想要在主題分割之後, 置換所有個別主題的預設標頭, 請使用 HEADER 關鍵字, 並提供要併入的 html 片段的名稱。 例如：
<!-HELPDOC:HEADEREND HEADER ="defaultheader.html" -->

主題區段

每一個主題都由下列標示包圍：
<!-HELPDOC:SEGMENTBEGIN --> and <!HELPDOC:SEGMENTEND -->
緊接在SEGMENTBEGIN 標示後的是命名區段的錨點標示。它也提供了在 Help Document 分割時所建立的 HTML 文件的檔名。區段名稱合併 了畫面識別字、控制 ID 及未來的副檔名 (html)。例如： 畫面的 "MY_PANEL.MY_CONTROL.html" 區段僅有畫面識別字及未來的副檔名。

解說產生器會將文字置入文件中, 指出您應將解說資訊放在何處：

```
<!-- HELPDOC:SEGMENTBEGIN PDMLSYNCH="YES" --><A
NAME="MY_PANEL.MY_CONTROL.html"></A>
<H2>My favorite control</H2>
在此處插入 "My favorite control" 的解說。
<P><!-- HELPDOC:SEGMENTEND -->
```

必要時, 您可以在錨點標示之後及 SEGMENTEND 標示之前新增其它 的 HTML 2.0 標示。

PDMLSYNCH 標示會控制區段與 PDML 中所定義的控制項的緊密關 係。如果DMLSYCH 是

"YES", 則如果 PDML 中具有相同名稱的控制項移除時, Help Document 區段也會
移除。PDMLSYNCH="NO"　表示主題應留在 Help Document 中, 不 論相對應的控制項是否存於 PDML
中。例如, 當您建立其它的深度主 題或一共同主題時, 就可以使用這個。

所產生的畫面解說已鏈結到畫面中解說所可使用的每一個控制項。這
些鏈結是以本端基準參照產生的, 所以您可以在標準瀏覽器中以內部鏈結的方 式來測試它們。在 Help
Document 分割後, 處理器會移除這些內部鏈結中 的 "#", 製作結果單一主題 HTML
檔中的外部鏈結。因為也許您 會想要在主題中有內部鏈結, 所以當參照中內嵌了 ".html" 時, 處理器僅會
除所有在前面的 "#" 。

如果您想要置換任何特定主題的預設標頭, 請使用 HEADER 關鍵 字, 並提供要併入的 html
片段名稱。例如：
<!-- HELPDOC:SEGMENTBEGIN PDMLSYNCH="YES" HEADER="specialheader.html" -->

標底

Help Document 中的標底會以下列標示起首：
<!-- HELPDOC:FOOTERBEGIN -->　。標準標底是 </BODY></HTML>。此標底會新增到每一個
HTML 檔。

新增鏈結

您可以新增鏈結到任何外部或內部 URL 以及任何其它的區段。 然而, 您必須遵守一些慣例：

- 以標準方式使用外部 URL。這包括內部鏈結到外部 URL
- 相同主題中的內部鏈結是以標準方式撰寫, 但不得有 ".html" 作為部份的
  標示名稱。這是因為在分隔主題時, Help Document 處理器會假設任何具有 .html 的鏈結都
  必須是外部鏈結。 因此, 它會移除前面的 "#"。
- 對其它主題區段的鏈結撰寫, 前面都應附有 "#", 就好比它們是內部的 基準參照。
- 也可以建立對其它主題區段的內部鏈結。 只要在處理程序期間移除前面的 "#"。

備註 ：

- 執行時, PanelManager 類別會在具有與 PDML 檔的名稱相同的子目錄中尋找說明檔。 當處理器分割
  Help Document 時, 就預設值而言, 它會建立此子目錄並在其中放置結果 HTML 檔。
- 處理器不會對相對鏈結的外部 URL 參照進行任何調整。
  當您從個別的主題檔進行鏈結時, 會從新的子目錄搜尋任何相對鏈結。
  因此, 您必須在可以找到資源的地方放置其複本, 如影像, 或在路徑中使 用
  "../", 以便能從畫面目錄進行搜尋。

使用視效編輯程式進行編輯

您幾乎可以使用任何視效 HTML 編輯程式來編輯您的解說內容。 因為 HELPDOC
標示是註解, 在某些編輯程式中可能並不明顯。為了方 便起見, 在緊臨 SEGMENTBEGIN
標示的前方及緊接著 SEGMENTEND 標示的
後方, 會新增一個水平尺規到解說架構中。這些水平尺規在視效編輯
程式中, 提供了整個區段的清楚視覺表示。如果您因為想要移動
、複製或刪除某一區段而加以選取, 請選取周圍的水平尺規, 以確定在您的選 項中已併入
SEGMENTBEGIN 及 SEGMENTEND 標示。這些水平尺規並不會 複製到最終的個別 HTML 檔。

建立其它的主題

您可以在 Help Document 中建立其它的主題區段。只要複製另一個區段，通常此動作是最容易完成的。當您複製區段時，您應複製在 SEGMENTBEGIN 標示之前及 SEGMENTEND 標示之後的水平尺規。這會讓未來在進行視效編輯時容易些，並有助於避免不符的標示。為取得最好的結果，請使用下列秘訣：

- 在分割 Help Document 時，錨點的名稱應與您所想的結果單一檔的 名稱相同。它應以 ".html" 結尾。
- 您應在 SEGMENTBEGIN 標示中使用 PDMLSYNCH="NO" 關鍵字，以避免在解說架構重新產生時，區段會被移除。
- 任何對您新主題的參照應被製成 Help Document 中的內部鏈結，且前面附 有 "#"。當區段被分割成單一檔案時，此 "#" 會在稍後的處理程序中被 移除。

檢查您的鏈結

對於大部份的撰寫，您可以在 Web 瀏覽器中檢視您的文件並選取不同的鏈 結，以檢查鏈結。在單一的 Help Document 中，鏈結仍然是在它們的內部 套表中。

當您即將完成時，或當您想要以您為其開發說明的應用程式測試，您必須 將 Help Document 分割成單一檔案。您可以使用 HTML 處理的 Help Document。

如果您在編輯之後需要重新產生 Help Document，您的撰寫就必須保留。如果您在產生原始解說架構後新增新的控制項，您也許想要重新產生 Help Document。在此情況下，解說產生器會在建立新的架構之前，檢查現存的 Help Document。如果有發現，它會保留任何現存的區段，然後新增新的控制項。

Graphical Toolbox 範例

我們已提供了一些範例，告訴您 如何建置 Graphical Toolbox 中的工具，以完成您自己的 UI 程式。

- 建構並顯示畫面 ：告訴您如何建構
  一個簡單的畫面。然後範例會告訴您如何建置一個顯示畫面的小型 Java 應
  用程式。當使用者在文字欄位中輸入資料，並按一下「關閉」按鈕時， 應用程式會將資料傳回 Java
  主控台。此範例說明 Graphical Toolbox 整體環境 的基本特性及作業。
- 建立及顯示畫面 ： 告訴您當畫面及內容檔是在相同的目錄中時，要如何建立及顯示畫面。
- 建構完整功能的對話框 一旦建置了提供資料給畫面的 DataBeans， 且已在 PDML
  中識別屬性後，本例會告訴您如何建構一個完整功能的對話框。
- 使用動態畫面管理程式調整畫面大 小動態畫面管理程式會在執行時動態調整畫面的大小。
- 可編輯的組合框：告訴您可編輯的組合框的 資料 bean 程式撰寫範例。

下面範例會說明 GUI Builder 如何協助您建立：

- 畫面：告訴您如何建立簡單的畫面，及執行畫 面的資料 bean 程式
- 重疊窗格：告訴您如何建立 Deck Pane 窗格，及最後 Deck Pane 窗格看起來的樣子
- 內容表: 告訴您如何建立 Property Sheet，及最後內容表看起來的樣子
- 分割窗格：告訴您如何建立分割窗格，及最 後分割窗格看起來的樣子
- 標籤窗格 ：告訴您如何建立標籤窗格，及最 後標籤窗格看起來的樣子
- 精靈：告訴您如何建立精靈，及最後產品看起來的樣子
- 工具列: 告訴您如何建立工具列，及最後工 具列看起來的樣子
- 功能表列告訴您如何建立功能表列，及最 後功能表列看起來的樣子
- 說明：說明說明文件是如何產生的，及將 說明文件分割成主題頁的方法。 同時，請參編輯由 GUI Builder 產生的說明文件
- 範例：說明整個 PDML 程式看起來的樣子， 包括畫面、內容表、精靈、選取/取消選取及功能表選項。

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

在瀏覽器中使用 Graphical Toolbox

您可以使用 Graphical Toolbox 以於 Web 瀏覽器中執行的 Java Applet 建立畫面。 本節將描述如何將 Graphical Toolbox 範例的簡單畫畫轉換為可在瀏覽器中執行的畫面。 支援的瀏覽器最低等級是 Netscape 4.05 和 Internet Explorer 4.0。 為了避免必須處理個別瀏覽器的不同特性，建議您的 Applet 最好使用 Sun 的 Java 外掛程式來執行。不然，您建需要建構有符號的 JAR 檔，供 Netscape 使用， 而另外建構有符號的 CAB 檔，供 Internet Explorer 使用。

建構 applet

在 Applet 中顯示畫面的程式碼幾乎就和在 Java 應用程式範例中所使用的程式碼相同， 但程式碼首先必須在 JApplet 次類別的 init 方法重新包裝。 此外，我們必須新增一些程式，確定 applet 畫面的大小符合畫面的 PDML 定義所指定的維度。 底下是我們的範例 applet SampleApplet.java 的原始程式碼。

```
import com.ibm.as400.ui.framework.java.*;

import javax.swing.*;
import java.awt.*;
import java.applet.*;
 import java.util.*;

public class SampleApplet extends JApplet
{
    // The following are needed to maintain the panel's size
    private PanelManager        m_pm;
    private Dimension           m_panelSize;

    // Define an exception to throw in case something goes wrong
    class SampleAppletException extends RuntimeException {}

    public void init()
    {
        System.out.println("In init!");

        // Trace applet parameters
        System.out.println("SampleApplet code base=" + getCodeBase());
        System.out.println("SampleApplet document base=" +
getDocumentBase());

        // Do a check to make sure we're running a Java virtual machine
that's compatible with Swing 1.1
        if (System.getProperty("java.version").compareTo("1.1.5") < 0)
            throw new IllegalStateException("SampleApplet cannot run on Java
VM version " +

System.getProperty("java.version") + " - requires 1.1.5 or higher");

        // Instantiate the bean object that supplies data to the panel
        SampleBean bean = new SampleBean();
```

```
        // Initialize the object
        bean.load();

        // Set up to pass the bean to the panel manager
        DataBean[] beans = { bean };

        // Update the status bar
        showStatus("Loading the panel definition...");

        // Create the panel manager. Parameters:
        // 1. PDML file as a resource name
        // 2. Name of panel to display
        // 3. List of data objects that supply panel data
        // 4. The content pane of the applet

        try { m_pm = new PanelManager("MyGUI", "PANEL_1", beans,
getContentPane()); }
        catch (DisplayManagerException e)
        {
            // Something didn't work, so display a message and exit
            e.displayUserMessage(null);
            throw new SampleAppletException();
        }

        // Identify the directory where the online help resides
        m_pm.setHelpPath("http://MyDomain/MyDirectory/");

        // Display the panel
        m_pm.setVisible(true);
    }

    public void start()
    {
        System.out.println("In start!");

        // Size the panel to its predefined size
        m_panelSize = m_pm.getPreferredSize();
        if (m_panelSize != null)
        {
            System.out.println("Resizing to " + m_panelSize);
            resize(m_panelSize);
        }
        else
            System.err.println("Error: getPreferredSize returned null");
    }

    public void stop()
    {
        System.out.println("In stop!");
    }
```

```
    public void destroy()
    {
        System.out.println("In destroy!");
    }

    public void paint(Graphics g)
    {
        // Call the parent first
        super.paint(g);

        // Preserve the panel's predefined size on a repaint
        if (m_panelSize != null)
            resize(m_panelSize);
    }
}
```

applet 的內容畫面將傳給 Graphical Toolbox，作為要佈置的儲存區。在 start 方法中，我們調整 applet 窗格的大小，使其具有正確的大小，而且我們會實作 paint 方法，以便當重新調整瀏覽器視窗的大小時，保留畫面的大小。

於瀏覽器中執行 Graphical Toolbox 時，無法自 JAR 檔中存取畫面的線上說明 HTML 檔。它們必須以個別的檔案常駐在目錄中，而該目錄則有 Applet 常駐。對 PanelManager.setHelpPath 所進行的呼叫會把這個目錄識別成「圖形化工具箱」，於是就能找到您的說明檔。

HTML 標籤

因為我們建議您使用 Sun 的 Java 外掛程式來提供 Java 執行環境的正確層次，所以識別 Graphical Toolbox applet 的 HTML 不如我們想要的那麼通用。我們建議您使用對另一個 Applet 同一個 HTML 模版，僅需做小小的更動。標註是設計為可於 Netscape Navigator 與 Internet Explorer 中解譯，而且如果瀏覽器沒有安裝 Java 外掛程式，它會產生提示，告知使用者由 Sun 的網站中下載 Java 外掛程式。若需有關 Java 外掛程式作用的其餘詳細資訊，請參閱 Java 外掛程式 HTML 規格。

以下是範例 applet 的 HTML，位在檔案 MyGUI.html 中：

```
<html>

<head>
<title>Graphical Toolbox Demo</title>
</head>

<body>
<h1>Graphical Toolbox Demo Using Java(TM) Plug-in</h1>
<p>

<!-- BEGIN JAVA(TM) PLUG-IN APPLET TAGS -->

<!-- The following tags use a special syntax which allows both Netscape and
Internet Explorer to load  -->
<!-- the Java Plug-in and run the applet in the Plug-in's JRE.  Do not
```

```
modify this syntax.              -->
<!-- For more information see http://java.sun.com/products/jfc/tsc/swingdoc-
current/java_plug_in.html.  -->

<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        width="400"
        height="200"
        align="left"
        codebase="http://java.sun.com/products/plugin/1.1.3/jinstall-113-
win32.cab#Version=1,1,3,0">
    <PARAM name="code"     value="SampleApplet">
    <PARAM name="codebase" value="http://www.mycompany.com/~auser/applets/">
    <PARAM name="archive"
value="MyGUI.jar,jui400.jar,util400.jar,x4j400.jar">
    <PARAM name="type"     value="application/x-java-applet;version=1.1">

    <COMMENT>
    <EMBED type="application/x-java-applet;version=1.1"
           width="400"
           height=200"
           align="left"
           code="SampleApplet"
           codebase="http://www.mycompany.com/~auser/applets/"
           archive="MyGUI.jar,jui400.jar,util400.jar,x4j400.jar"
           pluginspage="http://java.sun.com/products/plugin/1.1.3/plugin-
install.html">
        <NOEMBED>
    </COMMENT>
        No support for JDK 1.1 applets found!
        </NOEMBED>
    </EMBED>
</OBJECT>

<!-- END JAVA(TM) PLUG-IN APPLET TAGS -->

<p>
</body>
</html>
```

設定 1.1.3 的版本資訊是很重要的。

附註：此範例中，我們把 XML 剖析器 JAR 檔案 x4j400.jar儲存在 Web 伺服器中。唯有 PDML 檔案 併入 applet 安裝的一部份時才有必要這樣儲存。因為效能上的理由，通常應該序列化 畫面定義，以便 Graphical Toolbox 不必在執行時間解譯 PDML。 如此一來，可建立畫面的二進位表示法，大幅增進使用者介面的效能。
如需更多資訊，請參閱工具所產生的檔案的說明。

安裝及執行 applet

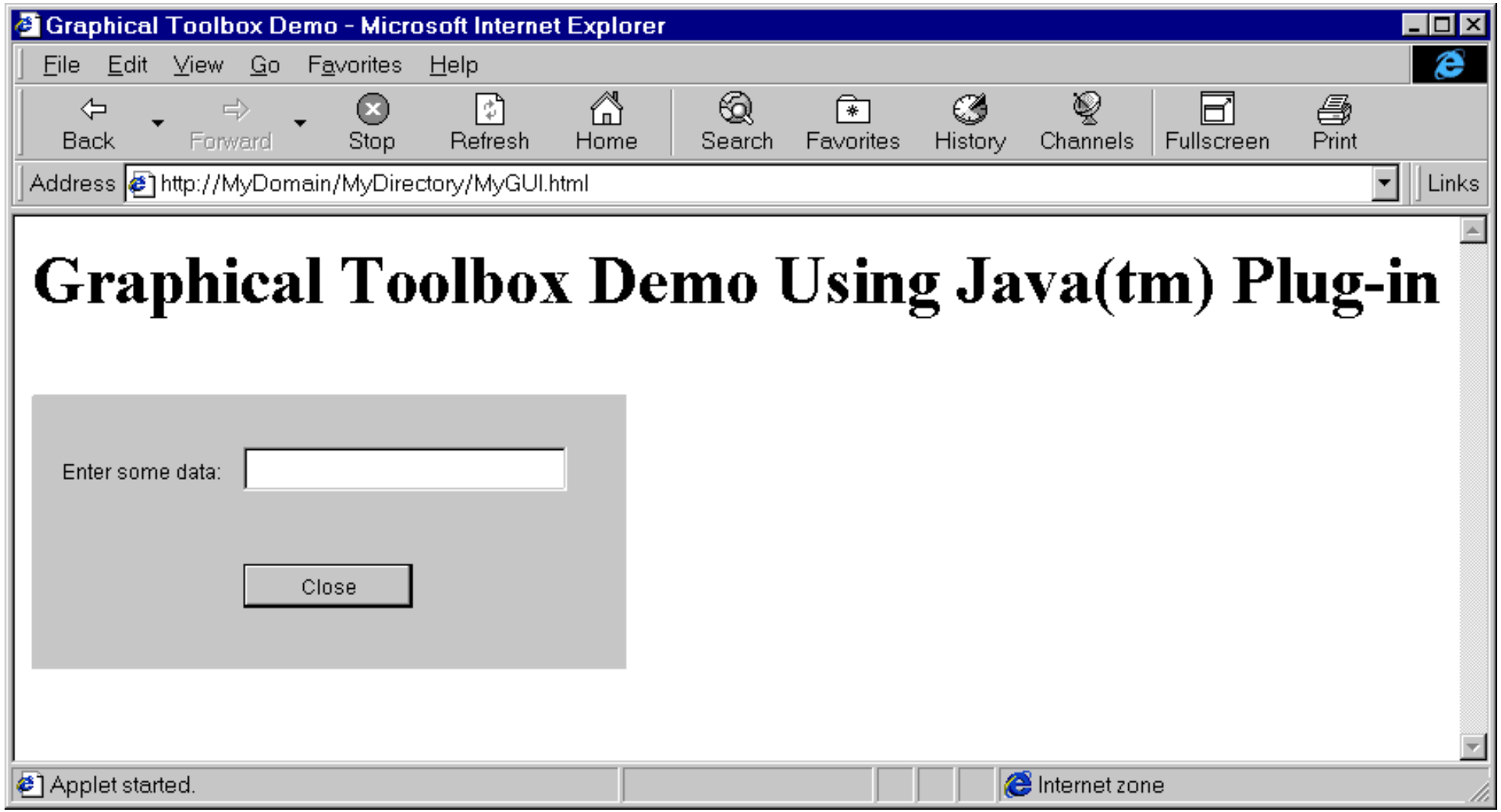
您可以執行下列步驟，在您喜愛的 Web 伺服器上安裝 applet：

- 編譯 SampleApplet.java。
- 建立一個名為 MyGUI.jar 的 JAR 檔，來包含 applet 二進位。包括當您編譯 SampleApplet.java 和 SampleBean.java、PDML 檔 MyGUI.pdml 和 資源組 MyGUI.properties 時所產生的類別檔。
- 將新的 JAR 檔複製到您在 Web 伺服器上選擇的目錄。將含有線上說明的 HTML 檔複製到伺服器目錄中。
- 將 Graphical Toolbox JAR 檔複製到伺服器目錄中。
- 最後，將內含 applet 的 HTML 檔 MyGUI.html 複製到伺服器目錄中。

要訣：在測試 Applet 時，請確定您已在您的工作站中，從 CLASSPATH 環境變數中移除 Graphical Toolbox jar 檔。否則您會看到錯誤訊息，表示伺服器中找不到 Applet 的資源。

現在您可以執行 applet。請將 Web 瀏覽器指向伺服器上的 MyGUI.html 。如果您尚未安裝 Java Plug-in，將詢問您是否要安裝它。一旦安裝了外掛程式且啟動了 applet 後，您的瀏覽器畫面將類似圖 1：

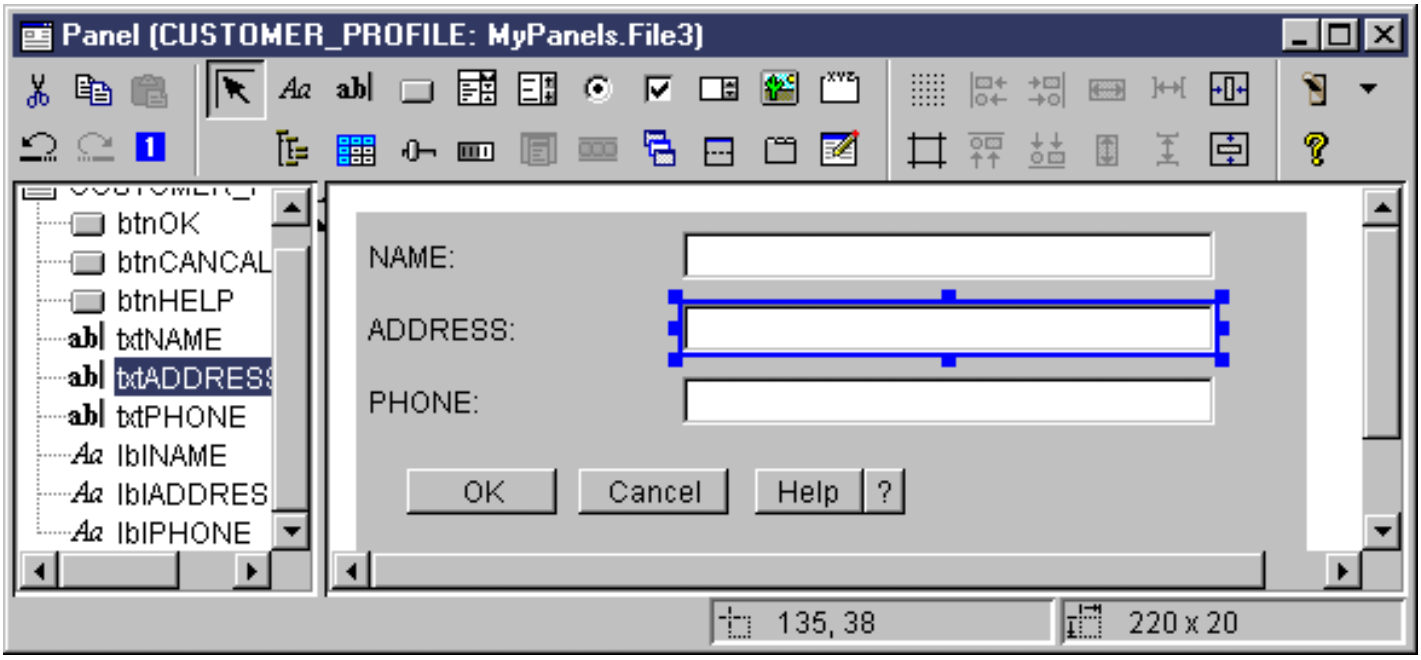
圖 1：在瀏覽器中執行範例 applet


Graphical Toolbox Demo - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Print
Address http://MyDomain/MyDirectory/MyGUI.html
Links
Graphical Toolbox Demo Using Java(tm) Plug-in
Enter some data:
Close
Applet started.
Internet zone

GUI Builder 畫面建置器工具列

圖 1 顯示「GUI 建置器畫面建置器」視窗。在圖 1 之後則是一份清單，
它列出每一個「畫面建置器」工具圖示及其功能的說明。

圖 1：「GUI 建置器畫面」視窗



按一下「指標」可以移動及調整畫面元件的大小。

按一下「標籤」可以在畫面上插入靜態標籤。

按一下「本文」可以在畫面上插入文字框。

按一下「按鈕」可以在畫面上插入按鈕。

按一下「組合框」可以在畫面上插入下拉清單框。

按一下「清單框」可以在畫面上插入清單框。

按一下「圓鈕」可以在畫面上插入圓鈕。

按一下「勾選框」可以在畫面上插入勾選框。

按一下「調整器」可以在畫面上插入調整器。

按一下「影像」可以在畫面上插入影像。

 按一下「功能表列」可以在畫面上插入功能表列。

 按一下「群組框」可以在畫面上插入含標示的群組框。

 按一下「樹」可以在畫面上插入階層樹。

 按一下「表格」可以在畫面上插入表格。

 按一下「滑動區」可以在畫面上插入可調整的滑動區。

 按一下「進度列」可以在畫面上插入進度列。

 按一下「疊窗格」可以在畫面上插入疊窗格。
一個疊窗格包含一疊畫面。使用者可以選取其中的任何畫面，但只能完全看到選取的畫面。

 按一下「分割窗格」可以在畫面上插入分割窗格。一個分割窗格可分成兩個水平窗格或垂直窗格。

 按一下「標籤窗格」可以在畫面上插入含標籤的窗格。一個含標籤窗格含有一組畫面，
每個畫面的頂端有標籤。使用者可以按一下標籤來顯示畫面的內容。會以畫面的標題作為標籤的文字。

 按一下「自訂」可以在畫面上插入自行定義的使用者介面元件。

 按一下「工具列」可以在畫面上插入工具列。

 按一下「輪換格線」可以在畫面上啟用格線。

 按一下「向上對齊」，可以將畫面上的多個元件對齊特定或主要元件的頂端。

 按一下「向下對齊」，可以將畫面上的多個元件對齊特定或主要元件的底端。

 按一下「等高」，可以將多個元件的高度對齊特定或主要元件的高度。

 按一下「垂直置中」，可以將選取的元件垂直地放在畫面的中央。

 按一下「輪換邊距」可以檢視畫面的邊距。

 按一下「靠左對齊」，可以將畫面上的多個元件對齊特定或主要元件的左邊。

 按一下「靠右對齊」，可以將畫面上的多個元件對齊特定或主要元件的右邊。

 按一下「等寬」，可以將多個元件的寬度對齊特定或主要元件的寬度。

 按一下「水平置中」，可以將選取的元件水平地放在畫面的中央。

 按一下「剪下」可以剪下畫面元件。

 按一下「複製」按鈕可以複製畫面元件。

 按一下「貼上」可以在不同的畫面或檔案之間貼上畫面元件。

 按一下「還原」可以還原上一個動作。

 按一下「重做」可以重做上一個動作。

 按一下「標籤次序」，即可在使用者按下 TAB 鍵來導覽畫面時， 控制每一個元件的選擇次序。

 按一下「預覽」可以顯示畫面的外觀預覽。

 按一下「說明」可以取得 Graphical Toolbox 的更多特定資訊。

IBM Toolbox for Java bean

JavaBeans(TM) 是以 Java 所撰寫，並可重複使用的軟體元件。元件是程式的一部份，
它會提供定義良好的功能單元，這種單元可小至視窗中按鈕的標籤，或大至整個應用程式。

JavaBeans 可以是視覺化或非視覺化元件。 非視覺化 JavaBeans 仍具有視覺化的呈現方式
(如圖示或名稱)， 以容許視覺化操作。

所有的 IBM Toolbox for Java 公用類別也是 JavaBeans。 這些類別是依據 Javasoft JavaBean
標準來建立的；因此它們可作為重複使用的元件。IBM Toolbox for Java Bean
的內容及方法與類別的內容及方法相同。

JavaBeans 可以在應用程式內使用，或可在建立器工具程式 如 IBM VisualAge for Java
產品中以視覺化的方式操作它們。

範例

- 範例： IBM Toolbox for Java bean 範例，顯示在程式中使用 JavaBeans 的方式。
- 範例： 視覺化 bean 建置器程式碼範例 ， 顯示以使用視覺化 bean 建置器 (如 IBM Visual Age for Java)， 從 JavaBeans 中建立程式的方式。

程式呼叫語言

概觀

「程式呼叫語言 (PCML)」為一種標示語言，可協助您使用較少的 Java 程式碼 來呼叫伺服器程式。PCML 是以「可擴充標示語言 (XML)」為基礎，
這是一種您可用來說明伺服器程式的輸入及輸出參數的標示語法。PCML 使您定義可完整說明 Java 應用程式所呼叫的伺服器程式之標示。若需 XML 的其餘相關資訊，請參閱 XML 參照 一節。

PCML 的最大好處就是它可讓您少寫些程式。 通常，在伺服器及 IBM Toolbox for Java 物件間需要額外的程式碼來連接、 擷取及轉換資料。然而，藉由使用 PCML，可自動處理以 IBM Toolbox for Java 類別對伺服器進行的呼叫。PCML 類別物件是從 PCML 標示產生的，
可協助將為了從應用程式呼叫伺服器程式所須撰寫的程式碼數量減至最小。

平台需求

雖然 PCML 是設計用來從 Java 平台支援對伺服器程式物件的分散式程式呼叫， 但您也可使用 PCML 從伺服器環境內呼叫伺服器程式。

詳細資訊的主題

請參閱下列如何使用 PCML 的主題：

- 透過 PCML 的協助呼叫程式
- 透過 PCML 標籤 建置程式呼叫
- PCML 範例

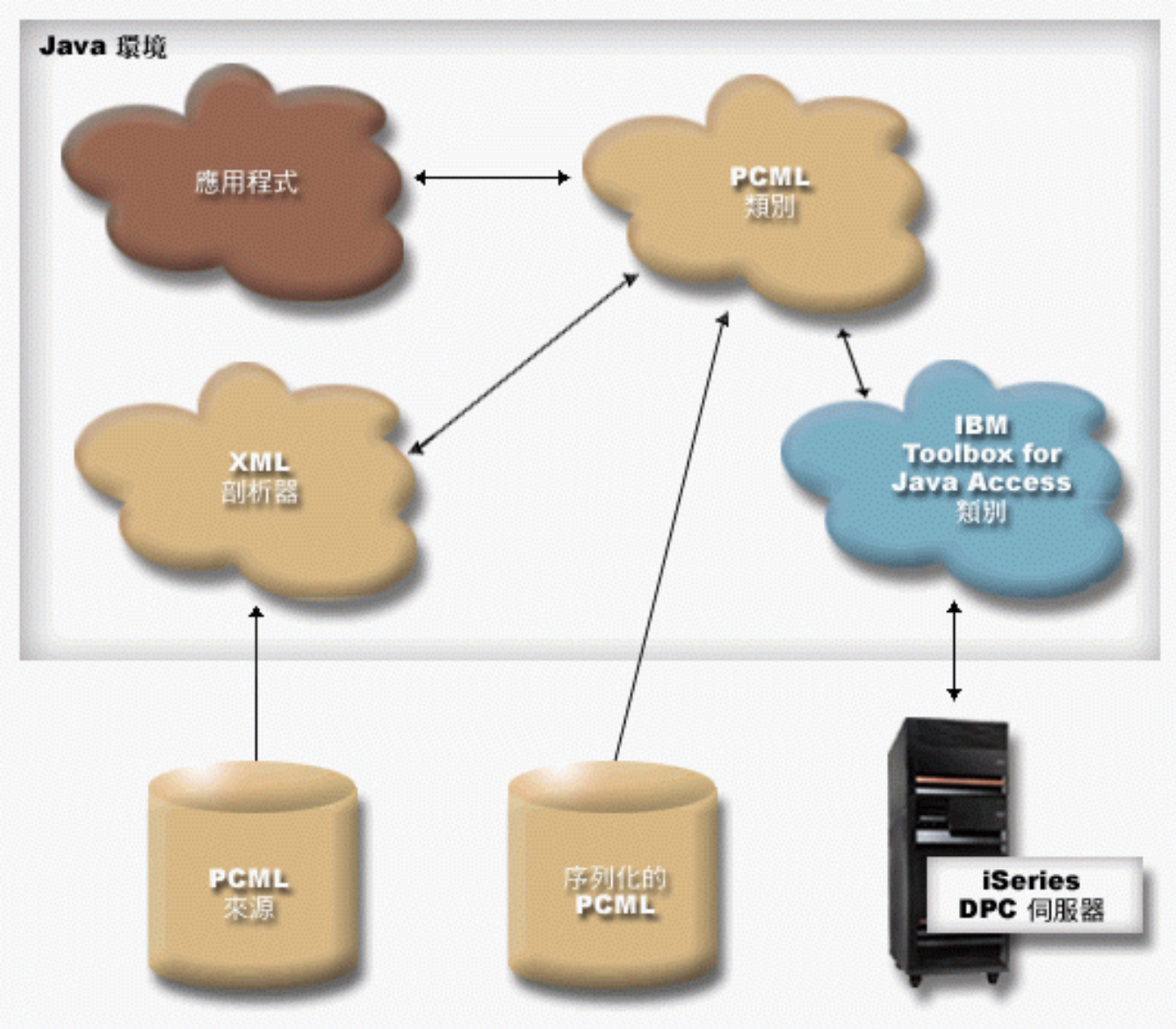
使用 PCML 建置 iSeries 程式呼叫

若要透過 PCML 來建置 iSeries 程式呼叫，首先必須建立 Java 應用程式和 PCML 原始檔。

取決於設計程序不同，您必須撰寫一個或多個 PCML 原始檔， 描述您的 Java 應用程式呼叫的與 iSeries 程式的介面。請參閱 PCML 語法，取得語言的詳細說明。

然後您的 Java 應用程式便與 PCML 類別（本案例中是 ProgramCallDocument 類別）互動。ProgramCallDocument 類別會利用您的 PCML 原始檔在您的 Java 應用 程式及 iSeries 程式之間傳送資訊。圖 1 說明 Java 應用程式如何與 PCML 類別

圖 1，使用 PCML 產生伺服器的程式呼叫



當您的應用程式建構 ProgramCallDocument 物件時，IBM XML 部析器將讀取及解析 PCML 原始程式。

於建立 ProgramCallDocument 類別後，應用程式將使用 ProgramCallDocument 類別的方法， 透過 iSeries 分散式程式呼叫 (DPC) 伺服器，從伺服器擷取必需的資訊。

若要增加執行時間效能，則可在建置產品時，將 ProgramCallDocument 類別序列化。然後，使用序列化的檔案，建構 ProgramCallDocument。 在這個情況中，不會在執行時，使用 IBM XML 剖析器。請參閱 使用序列化 PCML 檔 。

使用 PCML 原始檔

您的 Java 應用程式會以對 PCML 原始程式的參照 建構 ProgramCallDocument 物件，以使用 PCML。ProgramCallDocument 物件會考 慮讓 PCML 原始程式成為 Java 資源。

Java 應用程式 若不是使用 Java CLASSPATH 就是使用 ProgramCallDocument setPath() 方法來尋找 PCML 原始檔。當 Java 應用程式需要在執行時間設定路徑到 PCML 檔案時， 請使用 setPath() 方法。

下列 Java 程式會建構 ProgramCallDocument 物件：

```
    AS400 as400 = new AS400();
    ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400,
"myPcmlDoc");
```

ProgramCallDocument 物件會在稱為 myPcmlDoc.pcml 的檔案中尋找您的 PCML 原始程式。 請注意，.pcml 副檔名未指定於建構元中。

如果您於 Java 套件中開發 Java 應用程式，可以用套件定義 PCML 資源的名稱：

```
    AS400 as400 = new AS400();
    ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400,
"com.company.package.myPcmlDoc");
```

使用序列化的 PCML 檔

若要增加執行時間效能，您可以使用序列化的 PCML 檔。序列化的 PCML 檔含有代表 PCML 的序列化 Java 物件。序列化的物件同於當您從上述的來源檔 建構 ProgramCallDocument 時所建立的物件。

使用序列化的 PCML 檔將給與您更好的效能，因為在執行時，不需要 IBM XML 部析器來處理 PCML 標籤。

您可以使用下列其中一種方法，將 PCML 序列化：

- 從指令行：

  ```
  java com.ibm.as400.data.ProgramCallDocument -serialize mypcml
  ```

  這個方法有助於以批次處理方式來建置您的應用程式。

- 從 Java 程式內：

  ```
  ProgramCallDocument pcmlDoc; // Initialized elsewhere
  pcmlDoc.serialize();
  ```

如果您 PCML 位於名為 myDoc.pcml 的原始檔中，序列化的結果即是一個名為 myDoc.pcml.ser 的檔案。

PCML 原始程式 vs. 序列化 PCML 檔

考慮使用下列程式來建構 ProgramCallDocument：

```
    AS400 as400 = new AS400();
    ProgramCallDocument pcmlDoc = new ProgramCallDocument(as400,
"com.mycompany.mypackage.myPcmlDoc");
```

ProgramCallDocument 建構元首先會嘗試在 Java CLASSPATH 中的 com.mycompany.mypackage 內 尋找名為 myPcmlDoc.pcml.ser 的序列化 PCML 檔。如果序列化的 PCML 檔不存在， 則建構元將嘗試在 Java CLASSPATH 中的 com.mycompany.mypackage 內尋找 名為 myPcmlDoc.pcml 的 PCML 原始檔。如果 PCML 原始程式不存在，將丟出一個異常情況。

完整名稱

您的 Java 應用程式會使用 ProgramCallDocument.setValue() 來設定 所要呼叫的 iSeries 程式的輸入值。同樣地，您的應用 ProgramCallDocument.getValue() 來擷取 iSeries 程式的輸出值。

在存取 ProgramCallDocument 類別中的值時，您必須指定 文件元素或<data> 標示的完整名稱。完整名稱即是所有含有的標籤的名稱的組，每一個名稱之間均是以句點來區隔。

例如，以下列 PCML 來源為例， "nbrPolygons" 項目 的完整名稱為"polytest.parm1.nbrPolygons"存取多邊形某一邊的某一點的 "x" 值的完整名稱 為polytest.parm1.polygon.point.x"

如果產生完整名稱所需的任一元素尚未命名，則該元素的所有衍生元素將沒有完整名稱。 將無法從 Java 程式中存取沒有完整名稱的元素。

```
<pcml version="1.0">
  <program name="polytest" path="/QSYS.lib/MYLIB.lib/POLYTEST.pgm">
    <!-- Parameter 1 contains a count of polygons along with an array of
polygons -->
    <struct name="parm1" usage="inputoutput">
      <data name="nbrPolygons" type="int" length="4" init="5" />
      <!-- Each polygon contains a count of the number of points along with
an array of points -->
      <struct name="polygon" count="nbrPolygons">
        <data name="nbrPoints" type="int" length="4" init="3" />
        <struct name="point" count="nbrPoints" >
          <data name="x" type="int" length="4" init="100" />
          <data name="y" type="int" length="4" init="200" />
    </struct>
    </struct>
    </struct>
  </program>
</pcml>
```

存取陣列中的資料

任何 <data> 或 <struct> 元素可使用count 屬性定義為陣列。或 者，<data> 或 <struct> 元素可包含在定義為陣列的另一個<struct> 元素內。

尤其，<data> 或 <struct> 元素在多維度的陣列中，若含有元素的多個陣列具有指定屬性的話。

為了讓您的應用程式能夠設定或取得定義為陣列或在陣列內定義的值，您必須指定陣列的每一維度的陣列索引。陣列索引將以 int 值的陣列形式傳遞。以上面顯示的多邊形陣列的來源為例，下列 Java 程式可用來擷取多邊形的資訊：

```
    ProgramCallDocument polytest; // Initialized elsewhere
    Integer nbrPolygons, nbrPoints, pointX, pointY;
    nbrPolygons = (Integer) polytest.getValue("polytest.parm1.nbrPolygons");
    System.out.println("多邊形數：" + nbrPolygons);
    indices = new int[2];
    for (int polygon = 0; polygon < nbrPolygons.intValue(); polygon++)
    {
        indices[0] = polygon;
        nbrPoints = (Integer)
polytest.getValue("polytest.parm1.polygon.nbrPoints", indices );
        System.out.println("  端點數：" + nbrPoints);

        for (int point = 0; point < nbrPoints.intValue(); point++)
        {
            indices[1] = point;
            pointX = (Integer)
polytest.getValue("polytest.parm1.polygon.point.x", indices );
            pointY = (Integer)
polytest.getValue("polytest.parm1.polygon.point.y", indices );
            System.out.println("    X:" + pointX + " Y:" + pointY);
        }
    }
```

除錯

當您使用 PCML 呼叫具有複雜的資料結構的程式時，容易在您的 PCML 產生錯誤，導致 ProgramCallDocument 類別發生異常情況。如果錯誤是因為不正確地描述資料的位移與長度而引起的，將很難對異常情況進行除錯。

»請使用 Trace 類別的下列方法來開啟 PCML 追蹤：

```
    Trace.setTraceOn(true);       // Turn on tracing function.
    Trace.setTracePcmlOn(true);   // Turn on PCML tracing.
```

附註：PcmlMessageLog 類別中所有的公用方法，包括 tracing 在 V5R2 中都被取代。

Tracesetfilename() 方讓您能夠傳送下列型類的資訊到特定日誌檔或按照預設傳送到 System.out«：

- Java 應用程式與 iSeries 程式之間轉送的十六進位資料傾出。 在字元資料轉換為 EBCDIC 及整數轉換為 big-endia
  後，這將顯示程式輸入參數。 在輸出參數轉換為 Java 環境之前，它也會顯示這些輸出參數。

  資料會以一般十六進位傾出格式（十六進位數字在左邊，而字元解譯在右邊）來顯示。
  下列即是這種傾出格式的範例：

```
  qgyolobj[6]
  Offset : 0....... 4....... 8....... C....... 0....... 4....... 8.......
  C.......   0...4...8...C...0...4...8...C...
       0 : 5CE4E2D9 D7D9C640 4040
  **USRPRF                        *
```

在上述範例中，傾出顯示第 7 個參數具有設定為 "*USRPRF" 的資料，共有 10 個位元組。

- 對於輸出參數，在十六進位傾出後是文件的資料如何解譯的說明。

```
/QSYS.lib/QGY.lib/QGYOLOBJ.pgm[2]
Offset : 0....... 4....... 8....... C....... 0....... 4....... 8.......
C.......   0...4...8...C...0...4...8...C...
     0 : 0000000A 0000000A 00000001 00000068 D7F0F9F9 F0F1F1F5 F1F4F2F6
F2F5F400  *................P09901151426254.*
    20 : 00000410 00000001 00000000 00000000 00000000 00000000 00000000
00000000  *................................*
    40 : 00000000 00000000 00000000 00000000
*.................                       *
Reading data -- Offset: 0   Length: 4   Name: "qgyolobj.listInfo.totalRcds"
Byte data: 0000000A
Reading data -- Offset: 4   Length: 4   Name:
"qgyolobj.listInfo.rcdsReturned"    Byte data: 0000000A
Reading data -- Offset: 8   Length: 4   Name: "qgyolobj.listInfo.rqsHandle"
Byte data: 00000001
Reading data -- Offset: c   Length: 4   Name: "qgyolobj.listInfo.rcdLength"
Byte data: 00000068
Reading data -- Offset: 10  Length: 1   Name:
"qgyolobj.listInfo.infoComplete"    Byte data: D7
Reading data -- Offset: 11  Length: 7   Name:
"qgyolobj.listInfo.dateCreated" Byte data: F0F9F9F0F1F1F5
Reading data -- Offset: 18  Length: 6   Name:
"qgyolobj.listInfo.timeCreated" Byte data: F1F4F2F6F2F5
Reading data -- Offset: 1e  Length: 1   Name: "qgyolobj.listInfo.listStatus"
Byte data: F4
Reading data -- Offset: 1f  Length: 1   Name: "qgyolobj.listInfo.[8]" Byte
data: 00
Reading data -- Offset: 20  Length: 4   Name:
"qgyolobj.listInfo.lengthOfInfo"    Byte data: 00000410
Reading data -- Offset: 24  Length: 4   Name:
"qgyolobj.listInfo.firstRecord" Byte data: 00000001
Reading data -- Offset: 28  Length: 40  Name: "qgyolobj.listInfo.[11]"
Byte data:
0000000000000000000000000000000000000000000000000000000000000000000000000000
```

當來自 iSeries 程式的輸出資料與 PCML 來源不符時，上述訊息可能非常有助於診斷。當您使用動態長度與位移時，很容易發生這種情況。

PCML 語法

PCML 是由下列標記所組成，每一個均有自己的屬性標記：

- program 標記 開始並結束描述程式的程式碼。
- struct 標記 定義一個已命名的結構，它可指定為程式的引數，或另一個已命名結構內的欄位。結構標籤含有結構中每一欄位的資料或結構標籤。
- data 標記定義程式或結構內的欄位。

以下範例中，PCML 語法描述一個具有某類別的資料及一些隔離的資料的程式。

```
<program>

    <struct>
    <data> </data>
    </struct>

    <data> </data>

  </program>
```

PCML 程式標籤

PCML 程式標籤可以下列元素展開：

```
<program name=   "name"
     [ entrypoint=   "entry-point-name"     ]
»    [ epccsid=  "ccsid"   ]«
     [ path= "path-name"   ]
     [ parseorder=   "name-list"   ]
     [ returnvalue=   "{ void | integer }"     ]
     [ threadsafe=   "{ true | false }"     ]>
</program>
```

以下表格列出程式標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
entrypoint=	entry-point-name	在此程式呼叫的目標服務程式物件中，指定進入點的名稱。
»epccsid=	ccsid	指定服務程式中，進入點的 CCSID。 如需更多資訊，請參閱 ServiceProgramCall javadoc 中有關服務程式進入的附註。«
name=	name	指定程式的名稱。
path=	path-name	指定程式物件的路徑。 預設值假設程式位在 QSYS 檔案庫中。 路徑必須是到 *PGM 或 *SRVPGM 物件的有效 IFS 路徑名稱。如果 *SRVPGM 物件被呼叫， 則必須指定 entrypoint 屬性，以表示被呼叫的 entrypoint 的名稱。 如果沒有指定 entrypoint 屬性，則此屬性的預設值會假設是 QSYS 檔案庫中的 *PGM 物件。 如果已指定 entrypoint 屬性，則此屬性的預設值會假設是 QSYS 檔案庫中的 *SRVPGM 物件。 路徑名稱應全部用大寫字元加以指定。 »如果應用程式需要在執行時間設定路徑，例如有使用者指定安裝所用的程式庫時， 不要使用 path 屬性。在這種情形下，請 使用 ProgramCallDocument.setPath() 方法。«
parseorder=	name-list	指定輸出參數的處理次序。指定的值是以空白區隔的參數名稱清單，將按排列次序處理這些參數。 清單中的名稱必須與<program> 所屬標籤的 name 屬性中指定的名稱相同。 預設值將按標籤出現在文件的次序來處理輸出參數。 某些程式會在參數傳回資訊，描述先前參數中的資訊。例如，假設程式在第一個參數傳回結構陣列，而在第二個參數傳回陣列中的登錄數。在這種情況中，第二個參數必須按 ProgramCallDocument 中的次序處理，來決定第一個參數中要處理的結構數目。
returnvalue=	void 程式沒有傳回值。 integer 程式傳回 4 位元組帶正負號的整數。	指定服務程式呼叫所傳回的值的類型，如果有的話。 這個屬性不可用於 *PGM 物件呼叫。
threadsafe=	true 程式視為安全緒。 false 程式不是安全緒。	當您呼叫同一部伺服器中的 Java 程式以及 iSeries 程式時，請用這個屬性來指定要不要在 Java 程式的同一工作中和同一執行緒上呼叫出 iSeries 程式。如果確知 程式是安全緒，把內容設定為 true 可使效能更佳。 為維持環境的安全，預設為在另外的伺服器工作中呼叫程式。預設值是 false。

PCML struct 標籤

PCML struct 標籤可以下列元素展開：

```
<struct name=  "name"
     [ count= "{number | data-name }"     ]
     [ maxvrm= "version-string"     ]
     [ minvrm= "version-string"     ]
     [ offset= "{number | data-name }"      ]
     [ offsetfrom=  "{number | data-name | struct-name }"         ]
     [ outputsize=  "{number | data-name }"      ]      [ usage= "{ inherit | input |
output | inputoutput }"      ]>
</struct>
```

以下表格列出 struct 標籤的屬性。每個登錄項都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	name	指定 <struct> 元素的名稱
count=	number 其中number 定義固定、從不變更的大小陣列。 data-name 其中data-name 定義將在執行時間含有陣列中元素數目的 PCML 文件內的 <data> 元素的名稱。 指定的data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data> 元素，該元素是以 type="int定義的。 請參閱 解析相對名稱取得如何解析相對名稱的詳細資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過這個屬性，則元素雖然不會定義為陣列，但它可能包含在定義為陣列的另一個元素內。
maxvrm=	version-string	指定元素所存在的最高 AS/400 版本。如果 AS/400 版本高於屬性上指定的版本，則在呼叫程式時， 將不會處理元素及其子項 (若存在的話)maxvrm 元素有助於定義在 OS/400 版次之間不同的程式介面。 版本字串的語法必須是 "VvRrMm"，其中大寫字母 "V"、"R" 及 "M" 是文 字字元，而 "v"、"r" 及 "m" 是一或多個分別代表版本、版次及修正層次的數字。"v" 的值必須是從 1 到 255。"r" 及 "m" 的值必須是從 0 到 255。
minvrm=	version-string	指定元素所存在的最低 AS/400 版本。如果 OS/400 版本低於屬性上指定的版本，則在呼叫程式時， 將不會處理元素及其子項 (若存在的話)。這個屬性有助於定義在 OS/400 版次之間不同的程式介面。 版本字串的語法必須是 "VvRrMm"，其中大寫字母 "V"、"R" 及 "M" 是文 字字元，而 "v"、"r" 及 "m" 是一或多個分別代表版本、版次及修正層次的數字。"v" 的值必須是從 1 到 255。"r" 及 "m" 的值必須是從 0 到 255。

offset=	number 其中number 定義固定、從不變更的位移。 data-name 其中data-name 定義在執行時間含有這個元素的位移的 PCML 文件內的 元素的名稱。 指定的data-name 可為完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data>元素，該 元素是以 type="int" 定義的。 請參閱 解析相對名稱取得如何解析相對名稱的詳細資訊。	指定輸出參數內的 <struct>元素的位移。 有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在任一情況中，可變長度元素的位置通常指定為 參數內的位移或位置。offset屬性係用來描述這個 <struct>元素的位移。 Offset將結合offsetfrom屬性一起使用。 如果未指定 offsetfrom屬性，則 offset 屬性上指定的位移的基本位置將是元素的母項。 請參閱 指定位移，取得如何 使用offset與offsetfrom屬性的詳細資訊。 offset與offsetfrom屬性僅會用來 處理來自程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。 如果略過屬性，則元素的資料位置之後緊跟著參數中的前一個元素，若有的話。
offsetfrom=	number 其中number 定義固定、從不變更的基本位置。 number 屬性 最常用來指定number= "0"，表示位移 是從參數開頭開始的絕對位移。 data-name 其中data-name 定義 <data> 元素的名稱， 它將作為位移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。 來自 offset屬性的值將是這個屬性上指定的元素的相對位置。 指定的 data-name 可為完整的名稱或是與現行元素相對的名稱。 不管哪一種情況，名稱必須參照這個元素的祖先項。 請參閱 解析相對名稱，取得如何解析相對名稱的詳細資訊。 struct-name 其中struct-name定義 <struct>元素的名稱， 它將作為位移的基本位置。指定的元素名稱必須是這個元素的母項或祖先項。 來自 offset屬性的值將是這個屬性上指定的元素的相對位置。 指定的 struct-name可為完整的名稱或是與現行元素相對的名稱。 不管哪一種情況，名稱必須參照這個元素的祖先項。 請參閱 解析相對名稱，取得如何解析相對名稱的詳細資訊。	指定與 位移屬性相對的基本位置。 如果未指定 offsetfrom屬性， 則 offset屬性上指定的位移之基本位置將為 這個元素的母項。請參閱 指定位移， 取得如何使用offset與offsetfrom屬性的詳細資訊。 offset與offsetfrom屬性僅會用來 處理來自程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。
outputsize=	number 其中number 定義固定、從不變更的保留位元組數目。 data-name 其中data-name 定義將在執行時間保留給輸出資料的位元組數目的 PCML 文件 內的data> 元素的名稱。 指定的data-name 可為完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data>元素，該 元素是以 type="int" 定義的。 請參閱 解析相對名稱取得如何解析相對名稱的詳細資訊。	指定要保留給元素的輸出資料的位元組數目。 對於長度可變的輸出參數，需要 outputsize屬性， 方可指定應該保留多少個位元組供將從伺服器程式傳回的資料使用outputsize 可以在所有可變長度欄位及可變大小陣列上指定，或它可以針對含有一個或多個可變長度欄位的整個參數而指定。 不需要 Outputsize，且不應該針對固定大小輸出參數指定它。 屬性上指定的值將作為元素的總大小，包括元素的所有子項。因此，在元素的子項或後代項上，不會處理putsize 屬性。 如果略過這個屬性，將在執行時間決定要保留給輸出資料的位元組數目， 其方法是新增要保留給 <struct> 元素的所有子項使用的位元組數目。
usage=	inherit	用法將繼承自母項元素。如果結構沒有母項，則用法將假設為putoutput。
	input	結構是主程式的輸入值。對於字元與數字類型，將執行適當的轉換。
	output	結構是主程式的輸出值。對於字元與數字類型，將執行適當的轉換。

	inputoutput	結構同時為輸入及輸出值。

指定位移

有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在任一情況中，可變長度元素的位置通常指定為參數內的位移或

位移即是從參數開頭到欄位或結構開頭的距離，以位元組表示。 位置即是從某個結構的開頭到另一結構開頭的距離，以位元組表示。

對於位移，因為距離是從參數開頭開始算起，所以您應該指定 offsetfrom="0"。 下列是從參數開頭開始算起的位移的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains a path -->
    <struct name="receiver" usage="output" outputsize="2048">
      <data name="pathType"         type="int"  length="4" />
      <data name="offsetToPathName" type="int"  length="4" />
      <data name="lengthOfPathName" type="int"  length="4" />
      <data name="pathName"         type="char" length="lengthOfPathName"
                offset="offsetToPathName"  offsetfrom="0"        />
    </struct>
  </program>
</pcml>
```

對於位置，因為距離是從另一個結構開始算起，所以您可以指定與位移相對的結構的名稱。下列是從已命名的結構開頭開始算起的位置的範例：

```
<pcml ="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains an object -->
    <struct name="receiver" usage="output" >
      <data name="objectName"       type="char"  length="10" />
      <data name="libraryName"      type="char"  length="10" />
      <data name="objectType"       type="char"  length="10" />
      <struct name="pathInfo" usage="output" outputsize="2048" >
        <data name="pathType"         type="int"  length="4" />
        <data name="offsetToPathName" type="int"  length="4" />
        <data name="lengthOfPathName" type="int"  length="4" />
        <data name="pathName"         type="char" length="lengthOfPathName"
                  offset="offsetToPathName"  offsetfrom="pathInfo"         />
    </struct>
    </struct>
  </program>
</pcml>
```

PCML 資料標記

PCML 資料標記具有下列屬性。屬性是以方括弧 ([]) 括住，指出屬性是選用性的。如果您指定一個選用性屬性，請不要在您的來源檔中包括方括弧 ([])。有些屬性值會以選擇清單 (以大括弧括住) 顯示，並以垂直線區隔可能的選擇。當您指定這些屬性之一時，請不要在您的來源檔中包括大括弧，並指定顯示的選擇之一。

```
<data type=  "{ char | int | packed | zoned | float | byte | struct }"
     [ bidistringtype   ="{ ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 |
DEFAULT }  "]
     [ ccsid= "{ number | data-name }"      ]
»    [ chartype=  "{ onebyte | twobyte }    "]«
     [ count= "{ number | data-name }"      ]
     [ init= "string"   ]
     [ length=  "{ number | data-name }"      ]
     [ maxvrm=  "version-string"    ]
     [ minvrm=  "version-string"    ]
     [ name= "name"   ]
     [ offset=  "{ number | data-name }"      ]
     [ offsetfrom=   "{ number | data-name | struct-name }"        ]
     [ outputsize=   "{ number | data-name | struct-name }"        ]
     [ passby=   "{ reference | value }"      ]
     [ precision=   "number"   ]
     [ struct=  "struct-name"    ]
»    [ trim= "{ right | left | both | none }"       ]«
     [ usage= "{ inherit | input | output | inputoutput }"          ]>
</data>
```

下列表格列出資料標記屬性。每一個登錄包含屬性名稱、可能的有效值及屬性說明。

屬性	值	說明
type=	char 其中char 指出字元值。char 資料值將以 java.lang.Stri傳回。 其它相關資訊，請參閱長度的 char 值 int 其中int為一整數值。 int資料值將以 java.lang.Lon傳回。 其它相關資訊，請參閱 長度及精準度的nt值 packed 其中packed 是壓縮十進位值。packed 資料值將 以 java.math.BigDecima傳回。其它相關資訊，請參閱 長度及精準度的cked 值 zoned 其中zoned 是區化十進位值。zoned 資料值將 以 java.math.BigDecima傳回。其它相關資訊，請參閱 長度及精準度的ned 值 float 其中float是浮點值。length 屬性會指定位元組數 , "4" 或 "8"。4 位元組的整數將以 java.lang.Flo傳回。8 位元組的整數將 以ava.lang.Doubl傳回。其它相關資訊， 請參閱長度的 float值 byte	指出要使用的資料類型 (character, integer, packed, zoned, floating point, byte 或 struct)。 不同資料類型的長度與精準度屬性之值不同。其它相關資訊，請參閱 長度及精準度的值

	其中byte為一個位元組值。不對資料執行任何轉換byte資料值將以 byte值 byte[] 的陣列傳回。其它相關資訊，請參閱 長度的byte值 struct 其中struct指定 <struct> 元素的名稱。struct可讓您定義結構一次，並在文件內重複使用它多次。 當type="struct"時，它會像指定的結構一般出現在文件中的這個位置。 長度值不可有且精準度沒有struct 值。	
bidistringtype=	DEFAULT 其中 DEFAULT 為「非雙向資料 (LTR)」的 預設字串類型 ST4 其中ST4 為字串類型 4。 ST5 其中ST5 為字串類型 5。 ST6 其中ST6 為字串類型 6。 ST7 其中ST7 為字串類型 7。 ST8 其中ST8 為字串類型 8。 ST9 其中ST9 為字串類型 9。 ST10 其中ST10 為字串類型 10。 ST11 其中ST11 為字串類型 11。	指定雙向字串類型的<data> 元素，且其type="char"。 如果省略此屬性，此元素的字串類型將由 CCSID 所指定 (由其直接指定或為主電腦環境的預設 CCSID)。 字串類型是於 BidiStringType 類別的 javadoc 中所定義。
ccsid=	number 其中number 定義固定、從不變更的 CCSID。 data-name 其中data-name 定義在執行時將含有字元資料的 CCSID 之名稱。 指定的data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱都必須參照data> 元素， 該元素是以 type="int"定義的。請參閱解析相對名稱，以取得相對名稱如何解析的其它相關資訊。	指定 <data> 元素的字元資料之主電腦「編碼字集 ID (CCSID)」。ccsid屬性僅能指定供 <data> 元素使用，該元素具有type="char"。 如果略過這個屬性，這個元素的字元資料將假設具有主電腦環境的預設 CCSID。

»chartype=	onebyte 其中onebyte指定每一個字元的大小。 twobyte 其中twobyte指定每一個字元的大小。 使用chartype時， length="數目"　屬性指定字元的數目，而非位元組的數目。	指定每一個字元的大小。«
count=	number 其中number 定義固定、從不變更的大小陣列中元素數目。 data-name 其中data-name 定義將在執行時含有陣列中元素數目的 PCML 文件內 之data> 元素的名稱。指定的data-name 可以為 完整的名稱，或是與現行元素相對的名稱。不管是哪一種情況，　名稱必須參照data> 元素，該元素是以 type="int"定義的。 請參閱 解析相對名稱，以取得相對名稱如何解析的其它相關資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過ount 屬性，則元素雖然不會定義為陣列，但它可能包含在定義為陣列的另一個元素內。
init=	string	指定 <data> 元素的起始值。 當使用a> 元素（其sage="input"或usage="inputoutput"）時， 如果應用程式沒有明確地設定起始值，便會使用值。 指定的起始值係用來起始設定純量值。 如果元素定義為陣列或包含在定義為陣列的結構內，則指定的起始值將作為陣列中所有登錄的起始值。
length=	»number 其中number 定義資料所需的位元組數。 不過，使用 chartype屬性時，　number 是指定字元數，而非位元組數。 « data-name 其中data-name 定義將在執行時含有長度的 PCML 文件內之data> 元素的名稱。data-name 僅能指定給其 type="char" 或 type="byte"的data> 元素。 指定的data-name 可以為完整的名稱，或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data>元素，該元素是以 type="int"定義的。 請參閱 解析相對名稱，以取得相對名稱如何解析的其它相關資訊。	指定資料元素的長度。這個屬性的用法會隨著資料類型而有所不同。 其它相關資訊，請參閱 長度及精準度的值
maxvrm=	version-string	指定此元素所存在的處之最高 iSeries 版本。 如果 iSeries 版本高於此屬性上指定的版本，則在呼叫程式時，　將不會處理此元素及其子項 (如果有的話)。此屬性有助於定義 iSeries 版次間不同的程式介面。 版本字串的語法必須是 "VvRrMm"，其中大寫字母 "V"、"R" 及 "M" 是文字字元，而 "v"、"r" 及 "m" 是一或多個分別代表版本、版次及修正層次的數字。"v" 的值必須是從 1 到 255。"r" 及 "m" 的值必須是從 0 到 255。

minvrm=	version-string	指定此元素所存在處之最低 iSeries 版本。 如果 iSeries 版本低於此屬性上指定的版本，則在呼叫程式時，將不會處理此元素及其子項 (如果有的話)。此屬性有助於定義 iSeries 版次間不同的程式介面。 版本字串的語法必須是 "VvRrMm"，其中大寫 字母 "V"、"R" 及 "M" 是文字字元，而 "v"、"r" 及 "m" 是一或多個分別代表版本、版次及修正層次的數字。"v" 的值必須是從 1 到 255。"r" 及 "m" 的值必須是從 0 到 255。
name=	name	指定 <data> 元素的名稱。
offset=	number 其中number 定義固定、從不變更的位移。 data-name 其中data-name 定義將在執行時含有此元素位移之 PCML 文件內data的a> 元素之名稱。 指定的data-name 可以為完整的名稱，或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data> 元素，該 元素是以 type="int"定義的。 請參閱 解析相對名稱，以取得相對名稱如何解析的其它相關資訊。	指定輸出參數內的 <data> 元素的位移。 有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在任一情況中，可變長度元素的位置通常指定為參數內的位移或位置。 offset屬性會與 offsetfrom屬性一起使用。 如果未指定 offsetfrom屬性，則在 offset 屬性上指定的 位移之基本位置將為這個元素的母項。 請參閱 指定位移，以取得如何使用ffset及offsetfrom屬性的其它相關資訊。 offset與offsetfrom屬性僅會用來處理來自程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。 如果略過這個屬性，則這個元素的資料位置之後將緊跟著參數中的前一個元素，若有的話。
offsetfrom=	number 其中number 定義固定、從不變更的基本位置。 Number 通常是最常用來指定number= "0"，表示位移是從參數開頭開始的絕對位移。 data-name 其中data-name 定義用來作為位移的基本位置之data> 元素的名稱。 指定的元素名稱必須是這個元素的母項或祖先項。來自ffset 屬性的值將與此屬性上指定的元素之位置相對。 指定的 data-name 可以是完整的名稱或是與現行元素相對的名稱。不管哪一種情況，名稱必須參照這個元素的祖先項。 請參閱 解析相對名稱，取得如何解析相對名稱的詳細資訊。 struct-name 其中struct-name定義用來作為位移的基本位置之struct>元素的名稱。 指定的元素名稱必須是這個元素的母項或祖先項。來自ffset 屬性的值將與此屬性上所指定的元素之位置相對。 指定的 struct-name 可為完整的名稱或是與現行元素相對的名稱。 不管哪一種情況，名稱必須參照這個元素的祖先項。 請參閱 解析相對名稱，取得如何解析相對名稱的詳細資訊。	指定與 位移屬性相對的基本位置。 如果未指定 offsetfrom屬性，則 offset屬性上指定的 位移之基本位置將為這個元素的母項。 請參閱 指定位移，以取得如何使用ffset及offsetfrom屬性的其它相關資訊。 offset與offsetfrom屬性僅會用來處理來自程式的輸出資料。這些屬性不會控制輸入資料的位移或位置。

outputsize=	number 其中number 定義要保留的固定、從不變更的位元組數目。 data-name 其中data-name 定義將在執行時包含保留給輸出資料的位元組數之 PCML 文件內 的data> 元素之名稱。指定的 data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱必須參照 <data>元素，該 元素是以 type="int"定義的。 請參閱 解析相對名稱， 以取得相對名稱如何解析的其它相關資訊。	指定要保留給元素的輸出資料的位元組數目。 對長度可變的輸出參數而言，需要outputsize 屬性， 方可指定應保留多少個位元組供將從 iSeries 程式傳回的資料使用。utputsize 屬性可在所有可變長度欄位及可變大小陣列中指定， 或可針對含有一或多個可變長度欄位的整個參數指定。 不需要 Outputsize 不需要，且不應該針對固定大小輸出參數指定它。 這個屬性上指定的值將作為元素的總大小，包括元素的所有子項。 因此，在元素的子項或後代項上，不會處理utsize屬性。 如果省略outputsize 則將在執行時，藉新增保留給 <struct>元素的所有子項使用之位元組數， 來決定要保留給輸出資料的位元組數。
passby=	reference 其中reference表示參照將傳送參數。當程式被呼叫時，指標會傳送參數值到程式。 value 其中value 表示一整數值。此值只有在 type="int"及length="4" 已指定時，才會被接受。	指定參數是否由參照傳送或由值傳送。 此屬性只有在此元素為定義服務程式呼叫的 <program> 元素之子項時，才會被接受。
precision=	number	指定某些數值資料類型的精確度的位元組數目。 其它相關資訊，請參閱 長度及精準度的值
struct=	name	指定 <data> 元素的struct>的名稱。struct屬性僅能對具有ype="struc的<data> 元素指定。
»trim=	right 其中right為表示會刪除尾端空格的預設行為。 left 其中lef表示將刪除前導空格。 both 其中both 表示會刪除前導及尾端空格兩者。 none 其中none 表示不會刪除空格。	指定如何刪除字元資料的空格。«
usage=	inherit	用法將繼承自母項元素。如果結構沒有母項，則用法將假設為putoutput
	input	定義主程式的輸入值。對於字元與數字類型，將執行適當的轉換。
	output	定義主程式的輸出值。對於字元與數字類型，將執行適當的轉換。
	inputoutput	定義輸入及輸出值。

指定位移

有些程式會傳回具有固定結構的資訊，其後跟著一個或多個可變長度欄位或結構。在任一情況中， 可變長度元素的位置通常指定為參數內的位移

位移即是從參數開頭到欄位或結構開頭的距離，以位元組表示。位置即是從某個結構的開頭到另一結構開頭的距離，以位元組表示。

對於位移，因為距離是從參數開頭開始算起，所以您應該指定 offsetfrom="0"。 下列是從參數開頭開始算起的位移的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains a path -->
    <struct name="接收器" usage="輸出" outputsize="2048">
      <data name="pathType"          type="int"  length="4" />
      <data name="offsetToPathName" type="int"  length="4" />
      <data name="lengthOfPathName" type="int"  length="4" />
      <data name="pathName"          type="char" length="lengthOfPathName"
                  offset="offsetToPathName"  offsetfrom="0"         />
      </struct>
  </program>
</pcml>
```

對於位置，因為距離是從另一個結構開始算起，所以您可以指定與位移相對的結構的名稱。下列是從已命名的結構開頭開始算起的位置的範例：

```
<pcml version="1.0">
  <program name="myprog" path="/QSYS.lib/MYLIB.lib/MYPROG.pgm">
    <!-- receiver variable contains an object -->
    <struct name="接收器" usage="輸出" >
      <data name="objectName"        type="char"  length="10" />
      <data name="libraryName"       type="char"  length="10" />
      <data name="objectType"        type="char"  length="10" />
      <struct name="pathInfo" usage="輸出" outputsize="2048" >
        <data name="pathType"          type="int"  length="4" />
        <data name="offsetToPathName" type="int"  length="4" />
        <data name="lengthOfPathName" type="int"  length="4" />
        <data name="pathName"          type="char" length="lengthOfPathName"
                    offset="offsetToPathName"  offsetfrom="pathInfo"          />
      </struct>
      </struct>
  </program>
</pcml>
```

長度與精準度的值

不同資料類型的長度與精準度屬性之值不同。下表列出每一個資料類型的長度與精準度可能值說明。

資料類型	長度	精準度
type="char "	此元素資料的位元組數，不一定是字元數。您必須指定文字長度或資料名稱 。	未提供
type="int "	此元素資料的位元組數 2、4 或8。您必須指定文字長度。	指出精準度的位元數，及整數是否帶正負號或無正負號： • 針對length="2" ○ 使用precision="15 " 代表帶正負號的 2 位元組整數。此為預設值 ○ 使用precision="16 " 代表無正負號的 2 位元組整數 • 針對length="4" ○ 使用precision="31 " 代表帶正負號的 4 位元組整數 ○ 使用precision="32 " 代表無正負號的 4 位元組整數 • 針對length="8"， 使用precision="63 " 代表帶正負號的 8 位元組整數
type="packed " 或 "zoned "	此元素資料之位數。您必須指定文字長度。	元素的十進位數。此數字必須大於或等於 0，且小於或等於 length 屬性中指定的總位數。
type="float "	此元素資料之位元組數，4 或8。您必須指定文字長度。	未提供
type="byte "	此元素資料之位元組數。您必須指定文字長度 或資料名稱 。	未提供
type="struct "	不允許。	未提供

解析相對名稱

有幾個屬性可讓您指定文件內另一個元素或標籤的名稱，作為屬性值。 指定的名稱可以是與現行標籤相對的名稱。

可以經由查看名稱可否解析為含有現行標籤的標籤的子項或後代項，
來解析名稱。如果無法在這個層次解析名稱，則將在下一含有標籤的最高層次中繼續搜尋。 此解析最後必須產生出 <pcml> 標籤 或 <rfml> 標籤 所組成的一組標籤，在此情況下，會將名稱視為絕對名稱而非相對名稱。

以下為使用 PCML 的範例：

```
<pcml version="1.0">
  <program name="polytest" path="/QSYS.lib/MYLIB.lib/POLYTEST.pgm">
    <!-- Parameter 1 contains a count of polygons along with an array of
polygons -->
    <struct name="parm1" usage="inputoutput">
      <data name="nbrPolygons" type="int" length="4" init="5" />
      <!-- Each polygon contains a count of the number of points along with
an array of points -->
      <struct name="polygon" count="nbrPolygons">
        <data name="nbrPoints" type="int" length="4" init="3" />
        <struct name="point" count="nbrPoints" >
          <data name="x" type="int" length="4" init="100" />
          <data name="y" type="int" length="4" init="200" />
      </struct>
      </struct>
    </struct>
  </program>
</pcml>
```

以下為使用 RFML 的範例：

```
<rfml version="4.0">
```

```
  <struct name="polygon">
    <!-- Each polygon contains a count of the number of points along with an
array of points. -->
    <data name="nbrPoints" type="int" length="4" init="3" />
    <data name="point" type="struct" struct="point" count="nbrPoints" />
      </struct>
  <struct name="point" >
    <data name="x" type="int" length="4" init="100" />
    <data name="y" type="int" length="4" init="200" />
      </struct>
  <recordformat name="polytest">
    <!-- This format contains a count of polygons along with an array of
polygons -->
    <data name="nbrPolygons" type="int" length="4" init="5" />
    <data name="polygon" type="struct" struct="polygon" count="nbrPolygons"
/>
  </recordformat>
</rfml> «
```

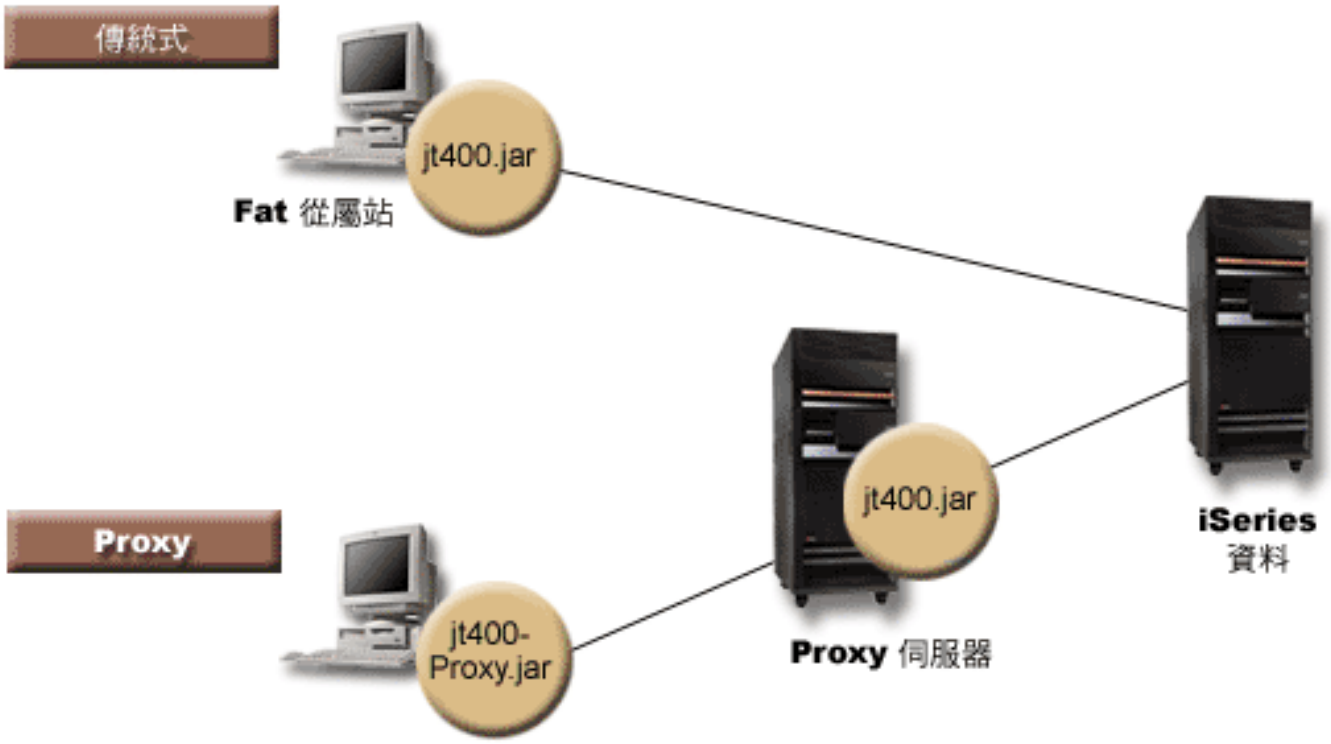
Proxy 支援

IBM Toolbox for Java 包含部份類別的 proxy 支援。Proxy 支援是一種處理程序，IBM Toolbox for Java 需要它在非應用程式實際所在的 Java 虛擬機器 (JVM) 上執行作業。Proxy 支援使用 Secure Sockets Layer (SSL) 通信協定以加密資料。

proxy 類別常駐於 jt400Proxy.jar，隨 IBM Toolbox for Java 其餘部份一併供應。proxy 類別和 IBM Toolbox for Java 中的其它類別相同，構成一套獨立式平台 Java 類別，可在含有 虛擬機器 的任何一部電腦上執行。proxy 類別分派所有的方法呼叫到伺服器應用程式，或是 PROXY 伺服器。完整的 IBM Toolbox for Java 類別 是在 proxy 伺服器上。當從屬站使用 proxy 類別時，此要求會轉送到建立並管理實際 IBM Toolbox for Java 物件的 proxy 伺服器。

圖 1 所示為標準和 proxy 從屬站連線到伺服器的方式。proxy 伺服器可以包含資料的 iSeries。

圖 1：標準從屬站和 proxy 從屬站如何連線到伺服器



使用 proxy 支援的應用程式執行速度要比使用標準的 IBM Toolbox for Java 類別來得慢，這是因為它需要額外的通信來支援較小的 proxy 類別。執行方法呼叫較少的應用程式，其效能較不會降低。

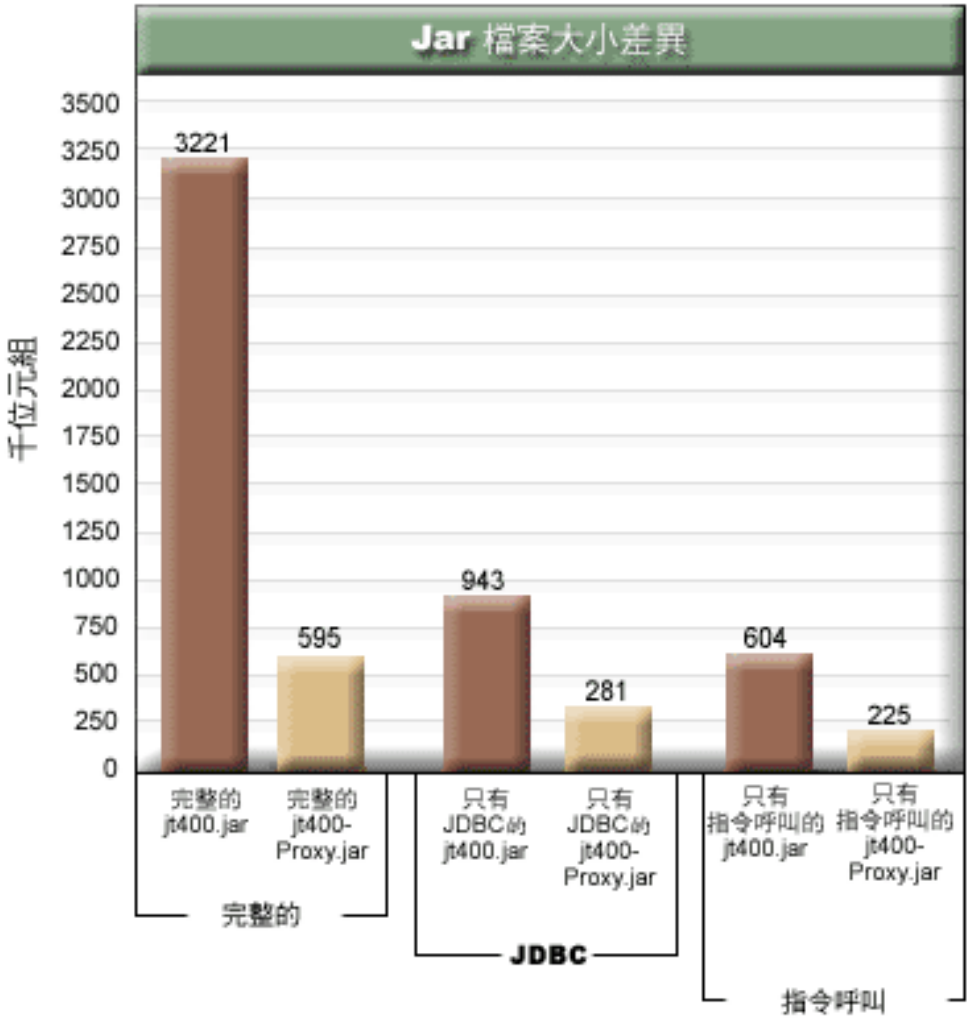
在提供 proxy 支援之前，包含公用介面的類別、所有需要處理 要求的類別和應用程式本身都在同一部 Java 虛擬機器上執行。使用 proxy 支援時，公用介面必須和應用程式在一起，但處理要求用的類別可在另一部 JVM 執行。Proxy 支援並不會變更公用介面。此相同程式可以 和 IBM Toolbox for Java 的 proxy 版本、或是標準版本一起執行。

使用 jt400Proxy.jar 檔案

多重階層的 proxy 實務目標是讓公用介面的 jar 檔案大小儘可能得小， 進而使用最少的時間從 Applet 下載此檔。當您使用 proxy 類別時， 並不需要在從屬站上安裝完整的 IBM Toolbox for Java。 應使用 jt400Proxy.jar 檔案的JT400JarMaker， 只併入必要的元件，以使 jar 檔案儘可能小。

圖 2 與標準的 jar 檔案比較 proxy jar 檔案之大小：

圖 2：proxy jar 檔案與標準 jar 檔案之間大小的比較



另外附帶一個好處就是 proxy 支援要求穿過防火牆開啟埠較少。 使用標準的 IBM Toolbox for Java，您必須開啟多重的埠。 這是因為 IBM Toolbox for Java 每項服務都使用不同的埠與伺服器通信。例如「指令」呼叫使用非 JDBC 的埠，它使用與列印不同的埠...等等。 每一個埠都必須允許它穿過防火牆。 然而，proxy 支援後，所有的資料流程皆經由相同的埠。

標準 proxy 和 HTTP 通道

經由 proxy 執行有兩種選項可用：標準 proxy 和 HTTP 通道：

- 標準 proxy 為 proxy 從屬站和 proxy 伺服器在埠上使用插座通信的地方。 預設埠是 3470。 您可以使用 ProxyServer 類別上的setPort()方法變更預設的埠，或在啟動 Proxy 伺服器時使用port 選項。例如：

```
java com.ibm.as400.access.ProxyServer -port 1234
```

- HTTP 通道為 proxy 從屬站和 proxy 伺服器在埠上經由 HTTP 伺服器通信的地方。IBM Toolbox for Java 提供可處理 proxy 要求的 servlet。Proxy 從屬站經由 HTTP 伺服器呼叫 servlet。通道的優點在於，因為經由 HTTP 埠通信，所以不必開啟額外的埠穿過防火牆。 通道的缺點就是比標準 proxy 速度慢。

IBM Toolbox for Java 用 proxy 伺服器名稱來判斷 使用的是標準 proxy 還是通道 proxy：

- 如果是標準 proxy，請使用伺服器名稱。例如：

```
com.ibm.as400.access.AS400.proxyServer=myServer
```

- 如果是通道 proxy，請用 URL 來強制 proxy 從屬站使用通道。例如：

```
com.ibm.as400.access.AS400.proxyServer=http://myServer
```

執行標準 proxy 時，socket 連線存在於從屬站與伺服器之間。
如果該連線失敗，伺服器會清除該從屬站的相關資源。

使用 HTTP 通道時，如果使用 HTTP 通信協定，proxy 無法連線。 亦即為，每一個資料流都會有新的連線。
因為通信協定無法連線，所以從屬站應用程式不再作用時，伺服器並不知道。
結果伺服器也不知道何時應清除資源。通道伺服器則是依照 預定的間隔（基於逾時值）
使用執行緒來清除資源，因此解決了這個問題。

於預定間隔終了時，執行緒就會執行並清除最近未曾使用的資源。 有兩個支配執行緒：

- com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval 為 執行清除執行緒的頻率，以秒計。預設為每兩小時。
- com.ibm.as400.access.TunnelProxyServer.clientLifetime 為 資源在清除之前可以閒置多久，以秒計。預設是 30 分鐘。

使用 proxy 伺服器

為了使用 IBM Toolbox for Java 類別的 PROXY 伺服器實作， 請完成下列步驟：

1. 於 jt400Proxy.jar 上執行 AS400ToolboxJarMaker， 捨棄不需要的類別。這是選用步驟，但建議使用。
2. 決定如何傳送 jt400Proxy.jar 到從屬站。
   - 若為 Java 程式，請使用 AS400ToolboxInstaller 類別或另一個方法，把它傳送到從屬站。
   - 若為 Java Applet， 您可能可以從 HTML 伺服器下載 jar 檔案。
3. 決定您要用作 PROXY 伺服器的伺服器。
   - 對 Java 應用程式而言，PROXY 伺服器可以是任一部電腦。
   - 對 Java Applet 而言，PROXY 伺服器必須在與 HTTP 伺服器相同的電腦上執行。
4. 確定已將 jt400.jar 置於伺服器上之 CLASSPATH 中。
5. 啟動 proxy 伺服器或使用 proxy servlet：
   - 若為標準 proxy，使用下列指令來啟動 proxy 伺服器：

```
java com.ibm.as400.access.ProxyServer
```

- 若為通道 proxy，請將 HTTP 伺服器配置為使用 proxy servlet。servlet 類別名稱是 com.ibm.as400.access.TunnelProxyServer，包含於 jt400.jar 中。

6. 在從屬站上，設定一 系統 內容以識別 proxy 伺服器。IBM Toolbox for Java 使用此系統內容來判斷所使用的是標準 proxy 還是通道 proxy。
   - 若為標準 proxy，內容值是執行 proxy 伺服器的機器名稱。例如：

     ```
     com.ibm.as400.access.AS400.proxyServer=myServer
     ```

   - 若為通道 proxy，請用 URL 來強制 proxy 從屬站使用通道。例如：

     ```
     com.ibm.as400.access.AS400.proxyServer=http://myServer
     ```

7. 請執行從屬站程式。

如果您都要使用 proxy 類別與不在 jt400Proxy.jar 中的類別兩者，您可以參照 jt400.jar 以代替 jt400Proxy.jar。jt400Proxy.jar 為 jt400.jar 的一個子集，因此所有的 proxy 類別都會內含於 jt400.jar 檔案。

使用 SSL

使用 proxy 時，當資料從 proxy 從屬站流向目標 iSeries 伺服器時，有三個選項可以加密資料。SSL 演算法用來加密資料。

1. proxy 從屬站和 proxy 伺服器之間的資料流可以加密。
2. proxy 伺服器和目標 iSeries 伺服器之間的資料流可以加密。
3. 第 1 和第 2 點。proxy 從屬站和 proxy 伺服器、以及 proxy 伺服器和目標 iSeries 伺服器之間的資料流可以加密。

請參閱 Secure Sockets Layer 以取得其它相關資訊。

範例：使用 proxy 伺服器

我們提供了三種具體的範例，用上列步驟來使用 PROXY 伺服器。

- 使用 Proxy 支援執行 Java 應用程式
- 使用 proxy 支援執行 Java Applet
- 使用通道 Proxy 支援執行 Java 應用程式

已啟用類別來使用 proxy 伺服器

已啟用部份 IBM Toolbox for Java 類別來 使用 PROXY 伺服器應用程式。這包括下列各項：

- JDBC
- 記錄層次存取
- 整合檔案系統

- 列印
- 資料佇列
- 指令呼叫
- 程式呼叫
- 服務程式呼叫
- 使用者空間
- 資料區
- AS400 類別
- SecureAS400 類別

到目前為止，其它類別尚未支援 jt400Proxy。 同時，於僅使用 proxy jar 檔案之下無法使用整合檔案系統許可權。 然而，您可以使用JarMaker 類別， 來含括 jt400.jar 檔案的這些類別。

記錄格式標記語言

記錄格式標記語言 (RFML) 是指定記錄格式用的 XML 延伸語言。IBM Toolbox for Java RFML 元件使您的 Java 應用程式能夠使用 RFML 文件來指定和操作某些記錄種類當中的欄位。

RFML 文件稱為 RFML 原始檔，代表針對 iSeries 系統中實體與邏輯檔案所定義的 資料說明規格 (DDS) 資料類型的實用子集。您可以使用 RFML 文件來管理下列當中的資訊：

- 檔案記錄
- 資料佇列登錄
- 使用者空間
- 自訂資料緩衝區

附註：關於使用 DDS 來說明資料屬性如需更多資訊，請參閱 DDS 參照 。

RFML 非常近似程式呼叫語言 (PCML)，這 是 Toolbox for Java 所支援之 XML 的另一種延伸。RFML 不是 PCML 的子集 也不是 PCML
的超集，而是新增幾個新的元素和屬性，並省略不執行一些元素和屬性的一種兄弟語言。

PCML 可作為使用 ProgramCall 和 ProgramParameter 類別的 XML 導向的選擇方案。 同樣地，RFML 也是 Record、RecordFormat 和 FieldDescription 等類別另一個具有使用親和力、 又容易維護的選擇方案。

若需 RFML 的其它相關資訊，請參閱下列主題：

基本要求
閱讀有關使用 RFML 的基本要求。

RFML 範例
說明在您的應用程式中使用 RFML 如何能夠讓您必須撰寫的程式碼量減少，
有時還能夠簡化。本範例中有 RFML 原始檔的範例。

RecordFormatDocument 類別
閱讀如何與其它 Toolbox for Java 類別一起使用 RecordFormatDocument
類別，來讀取和寫入資料。

RFML 文件與 RFML 語法
瞭解 RFML 資料類型定義中所定義的 RFML 文件 (稱為 RFML 原始檔) 以及 RFML
語法。

RFML 只是把 XML 使用在您的伺服器上的一種方式。如需關於在 iSeries 伺服器使用 XML 的更多資訊， 請參閱 Toolbox for Java XML 延伸和Extensible Markup Language (XML)。

使用 RFML 的基本要求

RFML 元件的工作站 Java 虛擬機器基本要求與其它的 IBM Toolbox for Java 相同。

此外，為了要在執行時間剖析 RFML，應用程式的 CLASSPATH 必須包含 XML 剖析器。XML 剖析器必須延伸 org.apache.xerces.parsers.SAXParser 類別。Toolbox for Java 包含有相容的剖析器 - XML Parser for Java，在 Toolbox for Java x4j400.jar 檔案當中。 詳細資訊，請參閱 檔案。

附註：RFML 的剖析器基本要求與 PCML 相同。如 同 PCML，如果您預先序列化 RFML 檔案，不必把 XML 剖析器併入應用程式 的 CLASSPATH 即可執行應用程式。

»範例： 使用 RFML 與使用 Toolbox for Java Record 類別的比較

本範例說明使用 RFML 和使用 Toolbox for Java Record 類別之間的差異。

使用傳統式的 Record 類別，資料格式規格會與您的應用程式的商務邏輯交織在一起。 新增、變更或刪除欄位就表示您必須編輯和重新編譯 Java 程式碼。不過， 使用 RFML 可使資料格式規格隔離在 RFML 原始檔中，與商務邏輯完全區隔開來。 因應欄位變更時，只須修改 RFML 檔案即可，往往不必變更或重新編譯 Java 應用程式。

本範例假設您的應用程式負責處理客戶記錄，您已將記錄定義 於RFML 原始檔 ，命名為 qcustcdt.rfml。 原始檔代表構成各筆客戶記錄的欄位。

以下報表說明 Java 應用程式可以如何使用 Toolbox for Java 的 Record、RecordFormat 和 FieldDescription 類別來解譯客戶記錄：

```
// Buffer containing the binary representation of one record of information.
    byte[] bytes;

    // ... Read the record data into the buffer ...

    // Set up a RecordFormat object to represent one customer record.
    RecordFormat recFmt1 = new RecordFormat("cusrec");
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(6, 0), "cusnum"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new
AS400Text(8, 37), "lstnam"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new
AS400Text(3, 37), "init"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new
AS400Text(13, 37), "street"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new
AS400Text(6, 37), "city"));
    recFmt1.addFieldDescription(new CharacterFieldDescription(new
AS400Text(2, 37), "state"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(5, 0), "zipcod"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(4, 0), "cdtlmt"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(1, 0), "chgcod"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(6, 2), "baldue"));
    recFmt1.addFieldDescription(new ZonedDecimalFieldDescription(new
AS400ZonedDecimal(6, 2), "cdtdue"));

    // Read the byte buffer into the RecordFormatDocument object.
    Record rec1 = new Record(recFmt1, bytes);

    // Get the field values.
    System.out.println("cusnum: " + rec1.getField("cusnum"));
```

```
    System.out.println("lstnam: " + rec1.getField("lstnam"));
    System.out.println("init:   " + rec1.getField("init"));
    System.out.println("street: " + rec1.getField("street"));
    System.out.println("city:   " + rec1.getField("city"));
    System.out.println("state:  " + rec1.getField("state"));
    System.out.println("zipcod: " + rec1.getField("zipcod"));
    System.out.println("cdtlmt: " + rec1.getField("cdtlmt"));
    System.out.println("chgcod: " + rec1.getField("chgcod"));
    System.out.println("baldue: " + rec1.getField("baldue"));
    System.out.println("cdtdue: " + rec1.getField("cdtdue"));
```

對照比較之下，以下是相同的記錄改為使用 RFML 時的解譯情形。

Java 程式碼使用 RFML 所解譯的客戶資料記錄內容可能會像這樣：

```
    // Buffer containing the binary representation of one record of
information.
    byte[] bytes;

    // ... Read the record data into the buffer ...

    // Parse the RFML file into a RecordFormatDocument object.
    // The RFML source file is called        qcustcdt.rfml    .
    RecordFormatDocument rfml1 = new RecordFormatDocument("qcustcdt");

    // Read the byte buffer into the RecordFormatDocument object.
    rfml1.setValues("cusrec", bytes);

    // Get the field values.
    System.out.println("cusnum: " + rfml1.getValue("cusrec.cusnum"));
    System.out.println("lstnam: " + rfml1.getValue("cusrec.lstnam"));
    System.out.println("init:   " + rfml1.getValue("cusrec.init"));
    System.out.println("street: " + rfml1.getValue("cusrec.street"));
    System.out.println("city:   " + rfml1.getValue("cusrec.city"));
    System.out.println("state:  " + rfml1.getValue("cusrec.state"));
    System.out.println("zipcod: " + rfml1.getValue("cusrec.zipcod"));
    System.out.println("cdtlmt: " + rfml1.getValue("cusrec.cdtlmt"));
    System.out.println("chgcod: " + rfml1.getValue("cusrec.chgcod"));
    System.out.println("baldue: " + rfml1.getValue("cusrec.baldue"));
    System.out.println("cdtdue: " + rfml1.getValue("cusrec.cdtdue"));              «
```

範例：RFML 原始檔

以下的 RFML 原始檔範例定義出 RFML 範例 使用 RFML 與 使用 Toolbox for Java Record 類別的比較中所使用的客戶記錄的格式。 這個 RFML 原始檔是命名為 qcustcdt.rfml 的文字檔。

附註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE rfml SYSTEM "rfml.dtd">

<rfml version="4.0" ccsid="819">

  <recordformat name="cusrec">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8"  ccsid="37" init="A"/>
    <data name="init"   type="char"  length="3"  ccsid="37" init="B"/>
    <data name="street" type="char"  length="13" ccsid="37" init="C"/>
    <data name="city"   type="char"  length="6"  ccsid="37" init="D"/>
    <data name="state"  type="char"  length="2"  ccsid="37" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="zoned" length="6" precision="2" init="4"/>
    <data name="cdtdue" type="zoned" length="6" precision="2" init="5"/>

  </recordformat>

  <recordformat name="cusrec1">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8"  ccsid="37" init="A"/>
    <data name="init"   type="char"  length="3"  ccsid="37" init="B"/>
    <data name="street" type="char"  length="13" ccsid="37" init="C"/>
    <data name="city"   type="char"  length="6"  ccsid="37" init="D"/>
    <data name="state"  type="char"  length="2"  ccsid="37" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="struct" struct="balance"/>
    <data name="cdtdue" type="struct" struct="balance"/>

  </recordformat>

  <recordformat name="cusrecAscii">

    <data name="cusnum" type="zoned" length="6"  precision="0" init="0"/>
    <data name="lstnam" type="char"  length="8" init="A"/>
    <data name="init"   type="char"  length="3" init="B"/>
    <data name="street" type="char"  length="13" init="C"/>
    <data name="city"   type="char"  length="6" init="D"/>
    <data name="state"  type="char"  length="2" init="E"/>
    <data name="zipcod" type="zoned" length="5" init="1"/>
```

```
    <data name="cdtlmt" type="zoned" length="4" init="2"/>
    <data name="chgcod" type="zoned" length="1" init="3"/>
    <data name="baldue" type="zoned" length="6" precision="2" init="4"/>
    <data name="cdtdue" type="zoned" length="6" precision="2" init="5"/>

  </recordformat>

  <struct name="balance">
    <data name="amount" type="zoned" length="6" precision="2" init="7"/>
      </struct>

</rfml>
```

»RecordFormatDocument 類別

RecordFormatDocument 類別讓 您的 Java 程式能夠在 RFML 的資料表示與 Record、RecordFormat 物件之間轉換， 以便使用於其它 Toolbox for Java 元件。

RecordFormatDocument 類別代表 RFML 原始檔，提供的方法允許您的 Java 程式執行下列動作：

- 從 Record 物件、RecordFormat 物件和位元組陣列撰寫 RFML 原始檔
- 建立 Record 物件、RecordFormat 物件和位元組陣列撰寫 RFML 原始檔， 表示出 RecordFormatDocument 物件所包含的資訊
- 取得及設定不同物件和資料類型的值
- 建立 XML (RFML)，代表 RecordFormatDocument 物件所包含的資料
- 序列化 RecordFormatDocument 物件代表的 RFML 原始檔

有關可用的方法之相關資訊，請參閱 javadoc 方法摘要中的 RecordFormatDocument 類別章節。

與其它 Toolbox for Java 類別一起使用 RecordFormatDocument 類別

請與下列 Toolbox for Java 類別一起使用 RecordFormatDocument 類別：

- 記錄導向的類別，包括讀取、操作和撰寫 Record 物件的記錄層級存取檔案類 別 (AS400File、SequentialFile 和 KeyedFile)。此種類還包括 LineDataRecordWriter 類別。
- 位元組導向的類別，包括一次讀取和寫入資料的位元組陣列的某些 DataQueue、UserSpace 和 IFSFile 類別。

切勿與下列 Toolbox for Java 類別一起使用 RecordFormatDocument 類別，這些類別讀取和寫入資料所用的格式 RecordFormatDocument 無法定址：

- DataArea 類別，因為它的讀取和寫入方法只處理「字串」、布林和 BigDecimal 等資料類型。
- IFSTextFileInputStream 和 IFSTextFileOutputStream，因為這些讀取和寫入方法只處理「字串」。
- JDBC 類別，因為 RFML 只把焦點放在 iSeries 資料說明規格 (DDS) 所 說明的資料上頭。 «

記錄格式文件與 RFML 語法

RFML 文件稱為 RFML 原始檔，包含為特定資料格式定義規格用的標籤。

因為 RFML 是基於 PCML，所以它的語法為 PCML 使用者所熟悉。因 為 RFML 是 XML 的延伸，所以 RFML 原始檔既容易讀取，也很容易建立。 例如，使用簡式的文字編輯器就可以建立 RFML 原始檔。 同時 RFML 原始檔說明資料結構的方式也比 Java 這些程式設計語言更容易瞭解。

RFML 的範例 使用 RFML 與使用 Toolbox for Java Record 類別的比較 包含有 RFML 原始檔 這個範例。

RFML DTD

RFML 文件類型定義 (DTD) 定義出有效的 RFML 元素和語法。 為確定 XML 剖析器可以在執行時間驗證您的 RFML 原始檔，請在原始檔中宣告 RFML DTD：

```
<!DOCTYPE rfml SYSTEM "rfml.dtd">
```

RFML DTD 常駐於 jt400.jar 檔案 (com/ibm/as400/data/rfml.dtd)。

RFML 語法

RFML DTD 定義出標籤，每一個都有自己的屬性標籤。RFML 標籤用來宣告和定義 RFML 原始檔中的下列元素：

- rfml 標籤 開始和結束說明資料格式的 RFML 原始檔。
- struct 標籤定義出具名的結構， 您可以在 RFML 原始檔中重覆使用。結構中含有結構中每一個欄位的資料標籤。
- recordformat 標籤定義出記錄格式， 當中不是含有資料元素，就是含有結構元素的參照。
- data 標籤 定義出記錄格式或結構內的欄位。

以下範例中，RFML 語法說明一個記錄格式和一個結構：

```
<rfml>

    <recordformat>
    <data> </data>
    </recordformat>

    <struct>
    <data> </data>
    </struct>

</rfml>
```

»

RFML 文件類型定義

這是 RFML DTD。注意版本是 4.0。RFML DTD 常駐於 jt400.jar 檔案 (com/ibm/as400/data/rfml.dtd)。

```
<!--
記錄格式標記語言 (RFML) 文件類型定義。
RFML 是 XML 語言。典型用法：   <?xml version="1.0"?>
   <!DOCTYPE rfml SYSTEM "rfml.dtd">
   <rfml version="4.0">
   ...
   </rfml>


(C) Copyright IBM Corporation, 2001,2002
All rights reserved. Licensed Materials Property of IBM
US Government Users Restricted Rights
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.
-->

<!-- Convenience entities -->
<!ENTITY % string             "CDATA">    <!-- a string of length 0 or
greater -->
<!ENTITY % nonNegativeInteger "CDATA">    <!-- a non-negative integer -->
<!ENTITY % binary2            "CDATA">    <!-- an integer in range 0-65535 --
>
<!ENTITY % boolean            "(true|false)">
<!ENTITY % datatype  "(char | int | packed | zoned | float | byte |
struct)">
<!ENTITY % biditype  "(ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 |
DEFAULT)">


<!-- The document root element -->
<!ELEMENT rfml (struct | recordformat)+>
<!ATTLIST rfml
          version       %string;    #FIXED "4.0"
          ccsid         %binary2;   #IMPLIED
>
<!-- Note: The ccsid is the default value that will be used for any
contained <data type="char"> elements that do not specify a ccsid. -->


<!-- Note: RFML does not support nested struct declarations.  All struct
elements are direct children of the root node. -->
<!ELEMENT struct (data)+>
<!ATTLIST struct
          name          ID          #REQUIRED
>


<!-- <!ELEMENT recordformat (data | struct)*> -->
```

```
<!ELEMENT recordformat (data)*>
<!ATTLIST recordformat
          name          ID                    #REQUIRED
          description   %string;              #IMPLIED
>
<!-- Note: On the server, the Record "text description" field is limited to
50 bytes. -->

<!ELEMENT data EMPTY>
<!ATTLIST data
          name          %string;              #REQUIRED

          count         %nonNegativeInteger;  #IMPLIED

          type          %datatype;            #REQUIRED
          length        %nonNegativeInteger;  #IMPLIED
          precision     %nonNegativeInteger;  #IMPLIED
          ccsid         %binary2;             #IMPLIED
          init          CDATA                 #IMPLIED
          struct        IDREF                 #IMPLIED

          bidistringtype %biditype;           #IMPLIED
>
<!-- Note: The 'name' attribute must be unique within a given recordformat. -
->
<!-- Note: On the server, the length of Record field names is limited to 10
bytes. -->
<!-- Note: The 'length' attribute is required, except when type="struct". --
>
<!-- Note: If type="struct", then the 'struct' attribute is required. -->
<!-- Note: The 'ccsid' and 'bidistringtype' attributes are valid only when
type="char". -->
<!-- Note: The 'precision' attribute is valid only for types "int",
"packed", and "zoned". -->


<!-- The standard predefined character entities -->
<!ENTITY quot  "&#34;">     <!-- quotation mark -->
<!ENTITY amp   "&#38;#38;"> <!-- ampersand -->
<!ENTITY apos  "&#39;">     <!-- apostrophe -->
<!ENTITY lt    "&#38;#60;"> <!-- less than -->
<!ENTITY gt    "&#62;">     <!-- greater than -->
<!ENTITY nbsp  " ">    <!-- non-breaking space -->
<!ENTITY shy   "&#173;">    <!-- soft hyphen (discretionary hyphen) -->
<!ENTITY mdash "&#38;#x2014;">
<!ENTITY ldquo "&#38;#x201C;">
<!ENTITY rdquo "&#38;#x201D;">
```

«

»RFML 資料標籤

資料標籤具有下列屬性。屬性是以方括弧（[]）括住，指出屬性是選用性的。 如果您指定一個選用性屬性，請不要在您的來源檔中包括方括弧（[）。
有些屬性值會以選擇清單（以大括弧括住）顯示，並以垂直線區隔可能的選擇。 當您指定這些屬性之一時，請不要在您的來源檔中包括大括弧， 禱 w顯示的選擇之一。

```
<data type=  "{ char | int | packed | zoned | float | byte | struct }"          ]
     [ bidistringtype   ="{ ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 |
DEFAULT }"  ]
     [ ccsid= "{  number | data-name     }" ]
     [ count= "{  number | data-name     }" ]
     [ init= "string " ]
     [ length= "{  number | data-name     }" ]
     [ name= "name " ]
     [ precision=  "number " ]
     [ struct=  "struct-name  " ]>
</data>
```

以下表格列出資料標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
type=	char 字元值。char 資料值將以 java.lang.Stri傳回。 詳細資訊，請參閱 char 長度值 int 整數值。 int資料值將以 java.lang.Lon傳回。 詳細資訊，請參閱 int長度和精準度的值 packed 壓縮十進位數值。 packed 資料值將以 java.math.BigDecima傳回。 詳細資訊，請參閱packed 長度和精準度的值。 zoned 區化十進位值。zoned 資料值將以 java.math.BigDecimal 傳回。 詳細資訊，請參閱 zoned 長度和精準度的值 float 浮點值。length 屬性指定位元組數目： 4 或8。4 位元組整數以 java.lang.Floa傳回。8 位元組整數以 java.lang.Doubl傳回。詳細資訊，請參閱長度的float值。	指出要使用的資料類型 (character, integer, packed, zoned, floating point, byte 或 struct)。 長度和精準屬性的值依不同的資料類型各有不同。詳細資訊，請參閱長度和精準度的值。

	byte 位元組值。不對資料執行任何轉換。byte 資料值將以 byte 值的陣列傳回byte[]。 詳細資訊，請參閱長度的byte 值。 struct <struct> 元素的名稱。struct 可讓您定義結構一次，並在文件內重複使用它多次。 當您使用 type="struct " 時，它就會像指定的結構一般出現在文件中的這個位置。struct 不允許長度值，也沒有精準度值。	
bidistringtype=	DEFAULT 其中 DEFAULT 是非雙向資料 (LTR) 的 預設字串類型。 ST4 其中ST4 是 字串類型 4。 ST5 其中ST5 是 字串類型 5。 ST6 其中ST6 是 字串類型 6。 ST7 其中ST7 是 字串類型 7。 ST8 其中ST8 是 字串類型 8。 ST9 其中ST9 是 字串類型 9。 ST10 其中ST10 是 字串類型 10。 ST11 其中ST11 是 字串類型 11。	為具有type="char " 的data> 元素指定雙向字串類型。 如果略過這個屬性，此元素的字串類型由 CCSID 所隱含 (明確指定或主電腦環境的預設 CCSID)。 字串類別定義於 javadoc for the BidiStringType class 中。

ccsid=	number 其中number 定義固定、從不變更的 CCSID。 data-name 其中data-name 定義在執行時間將含有字元資料的 CCSID 的名稱。 指定的data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱都必須參照<data> 元素，該元素是以 type="int" 定義的。 請參閱解析相對名稱，取得如何解析相對名稱的詳細資訊。	指定 <data> 元素的字元資料之主電腦編碼字集 ID (CCSID)。ccsid屬性僅能指定供 <data> 元素使用， 該元素具有type="char" 。 如果略過這個屬性，這個元素的字元資料將假設具有主電腦環境的預設 CCSID。
count=	number 其中number 定義於某種大小陣列中之固定、從不變更的元素數目。 data-name 其中data-name 定義將在執行時間含有陣列中元素數目的 PCML 文件內的<data> 元素的名稱。 指定的data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱都必須參照<data> 元素，該元素是以 type="int" 定義的。 請參閱解析相對名稱，取得如何解析相對名稱的詳細資訊。	指定元素是一個陣列，且識別陣列中的登錄數目。 如果略過count 屬性，則元素雖然不會定義為陣列， 但它可能包含在定義為陣列的另一個元素內。
init=	string	指定 <data> 元素的起始值。 指定的起始值係用來起始設定純量值。 如果元素定義為陣列或包含在定義為陣列的結構內，則指定的起始值將作為陣列中所有登錄的起始值。
length=	number 其中number 定義固定、從不變更的長度。 data-name 其中data-name 定義將在執行時間含有長度之 PCML 文件內的<data> 元素名稱。data-name 僅能指定具有type="char" 或 type="byte" 的data> 元素。指定的data-name 可以是完整的名稱或是與現行元素相對的名稱。 不管是哪一種情況，名稱都必須參照<data> 元素，該元素是以 type="int" 定義的。 請參閱解析相對名稱，取得如何解析相對名稱的詳細資訊。	指定資料元素的長度。這個屬性的用法會隨著資料類型而有所不同。 詳細資訊請參閱 長度和精準度的值。
name=	name	指定 <data> 元素的名稱。
precision=	number	指定某些數值資料類型的精確度的位元組數目。 詳細資訊，請參閱 長度和精準度的值。
struct=	name	指定 <data> 元素的struct>的名稱。struct屬性僅能指定具有 type="struct" 的data> 元素。

«

»RFML rfml 標籤

rfml 標籤可具有下列屬性；屬性是以方括弧 ([])
括住，指出屬性是選用性的。如果您指定一個選用性屬性，請不要在您的來源檔中包括方括弧 ([])。

```
<rfml version   ="version-string   "
     [ ccsid=  "number "  ]>
</rfml>
```

以下表格列出 rfml 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
version=	version-string 固定版本的 RFML DTD 。對於 V5R2 來說，4.0 是唯一的有效值。	指定 RFML DTD 的版本，您可以用來驗證值是否正確。
ccsid=	number 固定、永不會變更的編碼字集 ID (CCSID)。	指定主電腦 CCSID，其套用到所有含括、沒有指定 CCSID 的 <data type"char"> 元素。 詳細資訊，請參閱 RFML <data> 標籤 。當您省略不執行這個屬性時，則使用主電腦環境的預設 CCSID。

«

»RFML recordformat 標籤

recordformat 標籤可具有下列屬性。屬性是以方括弧 ([])
括住，指出屬性是選用性的。如果您指定一個選用性屬性，請不要在您的來源檔中包括方括弧 ([])。

```
<recordformat name    =" name "
     [ description=   "說明 " ]>
</recordformat>
```

以下表格列出 recordformat 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	name	指定 recordformat 的名稱。
description=	說明	指定 recordformat 的說明。

«

»RFML struct 標籤

struct 標籤可具有下列屬性；屬性是以方括弧（[]）
括住，指出屬性是選用性的。如果您指定一個選用性屬性，請不要在您的來源檔中包括方括弧（[]）。

<struct name =" 名稱 ">
</struct>

以下表格列出 struct 標籤的屬性。每個登錄都包括屬性名稱、可能的有效值和屬性的說明。

屬性	值	說明
name=	name	指定 <struct> 元素的名稱。

«

»

Secure Sockets Layer 及 Java Secure Socket Extension

IBM Toolbox for Java 使用 Java Secure Socket Extension (JSSE) 來支援 Java Secure Sockets Layer (SSL) 連線。JSSE 是 Java2 Platform, 標準版 (J2SE), 版本 1.2 以及 1.3 的選用套件。JSSE 已整合在 J2SE, 版本 1.4 中。

有關 JSSE 的詳細資訊, 請參閱Sun JSSE 網站。

JSSE 提供的功能包括：執行伺服器鑑別、啟用安全通信以及加密資料。 使用 JSSE 時, 您可以在透過 TCP/IP 執行任何應用程式通信協定 (例如, HTTP 及 FTP) 的從屬站與伺服器之間, 提供安全的資料交換。

安裝及配置 JSSE 之後, Toolbox for Java 便可以在預設的情況下使用 JSSE。 您應考慮移轉到 JSSE, 因為 Sslig 已經不會再進行加強。關於使用 Sslight 來啟用 SSL 的資訊僅內含做為向後相容性。

在 IBM Toolbox for Java 中開始使用 SSL 前, 您必須先瞭解憑證的信任 «

»iSeries 系統除錯器

「IBM iSeries 系統除錯器」在 iSeries 伺服器上提供一個新的圖形式使用者除錯環境。 使用「iSeries 系統除錯
iSeries 伺服器上執行的程式進行除錯及測試，包括那些在 OS/400 PASE 環境中執行的程式。

如需「iSeries 系統除錯器」的相關資訊，請參閱下列主題：

元件
讀取「iSeries
系統除錯器」中不同組成元件的相關資訊，以及它們如何一起運作以提供有力的除錯工具。

安裝
找出有關「iSeries 系統除錯器」安裝的基本要求，以及如何將它安裝到工作站中。

執行「iSeries 系統除錯器」
檢視如何執行不同的除錯元件。

«

iSeries 系統除錯器元件

「iSeries 系統除錯器」包含下列元件：

- 從屬站端的除錯管理程式
- 從屬站端的系統除錯器
- 從屬站端的 OS/400 PASE 除錯器
- 主電腦端的除錯 Hub
- 主電腦端的除錯伺服器

下面的說明僅提供有關「iSeries 系統除錯器」元件的一般資訊。若要得知有關各元件的詳細資訊，請參閱「iSeries 系統除錯器」並諮詢線上說明。 若要檢視「iSeries 系統除錯器」線上說明，請執行下列其中一項動作：

- 從任一除錯器介面中的說明功能表，按一下說明
- 按下 F1

除錯管理程式

「除錯管理程式」利用除錯 Hub 來註冊從屬站，針對所選取的系統使用圖形式除錯模式來啟用除錯管理程式。註冊之後，從屬站從模擬階段作業發出「啟動除錯 (STRDBG)」CL 指令時，就會啟動「系統除錯器」。

請使用「除錯管理程式」來管理您的除錯作業及連線：

- 新增及移除系統
- 新增及移除使用者
- 啟動除錯作業
- 啟動系統除錯器

系統除錯器

使用「系統除錯器」可以對 iSeries 伺服器上執行的程式除錯。您可以為系統內現有工作中執行的程式除錯，或者使用「系統除錯器」來啟動並接著為系統批次作業中的程式除錯。

您可以設定「系統除錯器」自動啟動、從工作站指令提示中以手動方式啟動，或使用除錯管理程式介面。

使用「系統除錯器」可以執行除錯活動，包括：

- 設定岔斷點
- 通過程式
- 檢驗變數
- 檢查呼叫堆疊
- 檢查與程式變數相關的記憶體
- 檢查緒活動

OS/400 PASE 除錯器

使用「OS/400 PASE 除錯器」可以對 OS/400 PASE 環境中執行的程式除錯。您可以為系統上現有程序中執行的程式除錯，或使用「OS/400 PASE 除錯器」啟動，然後在程式中除錯。

您可以直接從指令行或使用除錯管理程式面來啟動「OS/400 PASE 除錯器」。

除了之前列出的「系統除錯器」的除錯活動之外，您還可以使用「OS/400 PASE 除錯器」來執行 OS/400 PASE 特定的除錯活動，包括：

- 使用程式 loadmap 來除錯
- 檢視原始檔與方法清單
- 追蹤上層及子程序
- 檢查登記

除錯 Hub

「除錯 Hub」會提供下列功能：

- 作為想要使用「系統除錯器」的從屬站的一種登錄 OS/400 PASE 除錯器
- 處理啟動除錯伺服器的輸入要求

使用除錯管理程式面，以利用「除錯 Hub」註冊從屬站。 註冊從屬站可以將使用者資訊以及從屬站的 TCP/IP 位址儲存在登錄中。自模擬階段作業使用 「啟動除錯 (STRDBG)」CL 指令時，即會連接「除錯 Hub」，查看執行指令的使用者是否利用「除錯管理程式」進行註冊。
它也會檢查所執行的指令是否與「除錯管理程式」來自相同的 TCP/IP 位址。當這些條件都符合時，就會啟動圖形式的「iSeries 系統除錯」應用程式，取代傳統的除錯環境。

「除錯 Hub」也扮演所有「iSeries 系統除錯」應用程式的單一聯絡點。當「iSeries 系統除錯器」執行啟動除錯作業時，「除錯 Hub」就會代表使用者提交「除錯伺服器」工作，並將該工作傳送給相關的 TCP/IP 連線。

除錯伺服器

「除錯伺服器」是一個 TCP/IP 伺服器，當其中一個除錯器發出啟動除錯的要求時， 除錯 Hub 即會啟動它。然後伺服器工作就會服務正在除錯的作業，並發出適當的除錯器 API 與指令。

安裝 iSeries 系統除錯器

若要執行「iSeries 系統除錯器」，您的從屬工作站必須符合下列的硬體及軟體基本要求。

硬體基本要求

您必須在從屬站中安裝下列硬體：

- CPU：400 - 500 MHz
- 記憶體：最小為 128 MB (建議使用 256 MB)

軟體基本要求

您必須在從屬站中安裝下列軟體：

- 安裝下列其中一項：
  - Java 2 Platform 標準版 (J2SE) 或 企業版 (J2EE)，版本 1.3 或更新版本
  - Java 2 Runtime Environment (JRE) 標準版，版本 1.3.1 或更新版本
- jhall.jar (JavaHelp™ 中的其中一個 JAR 檔)

註：請確定新增 jhall.jar 到從屬站 CLASSPATH 環境變數。

有關安裝先前提到的軟體資訊，請參閱 Sun Java 網站。

安裝「iSeries 系統除錯器」JAR 檔

「iSeries 系統除錯器」是 IBM Toolbox for Java 套件的一部份。如果您的從屬站中沒有安裝 Toolbox for Java jt400.jar 檔案，當您安裝「iSeries 系統除錯器」時，就必須安裝該檔案。

安裝「iSeries 系統除錯器」之前，請確定從屬站系統符合前所列的基本要求。 若要安裝「iSeries 系統除錯器」，請完成下列步驟：

1. 安裝 Toolbox for Java，確定複製 jt400.jar 及 tes.jar 到從屬站。

   註：如果您已經在伺服器中安裝了 Toolbox for Java，jt400.jar 與 tes.jar 檔案就會位於伺服器中的同一個目錄下：

   /QIBM/ProdData/HTTP/Public/jt400/lib/

2. 複製 JAR 檔到從屬站後，請將它們新增到從屬站 CLASSPATH 環境變數。

現在您可以使用從屬站來執行「iSeries 系統除錯器」。

執行 iSeries 系統除錯器

您可以使用從屬站指令提示來啟動「除錯管理程式」啟動「系統除錯器」或啟動「OS/400 PASE 除錯器」。

若要瞭解「iSeries 系統除錯器」的相關資訊，請啟動「iSeries 系統除錯器」並查閱線上說明。若要檢視「iSeries 系統除錯器」線上說明，請下列其中一項動作：

- 從任一除錯器介面中的說明功能表，按一下說明
- 按下 F1

啟動「除錯管理程式」

若要由從屬站中的指令提示啟動「除錯管理程式」，請執行下列指令：

```
java utilities.DebugMgr
```

啟動「系統除錯器」

若要在從屬站的指令提示中啟動「系統除錯器」，請執行下列指令：

```
java utilities.Debug <args>
```

其中的args> 代表下列任何一個指令引數：

- -u = 使用者
- -s = 系統名稱
- -j = 工作說明，其格式為：工作編號/工作使用者/工作名稱
- -p = 要執行的程式，其格式為：程式庫/程式名稱

註： 一旦使用「除錯管理程式」來登記從屬站， 您就可以從模擬階段作業發出「啟動除錯 (STRDBG)」CL 指令來啟動「系統除錯器」。 您也可以直接從「系統除錯管理程式」啟動「系統除錯器」。

啟動「OS/400 PASE 除錯器」

若要在從屬站的指令提示中啟動「OS/400 PASE 除錯器」，請執行下列指令：

```
java utilities.DebugPASE <args>
```

其中的args> 代表下列任何一個指令引數：

- -u = 使用者
- -s = 系統名稱
- -p = 要執行的程式之完整路徑
- -pid = 程序 ID

註： 您可以從「系統除錯管理程式」中直接啟動「OS/400 PASE 除錯器」。
與「系統除錯器」不同，您不能從模擬器階段作業中啟動「OS/400 PASE 除錯器」。

若要得知有關元件的詳細資訊，請執行「iSeries 系統除錯器」並諮詢線上說明。 若要檢視「iSeries 系統除錯器」線上說明，請執行下列其中一項動作：

- **從任一「iSeries 系統除錯器」介面說明功能表，按一下說明**
- **從任一「iSeries 系統除錯器」介面，按下F1**

範例：內容檔

```
#=========================================================#
# IBM Toolbox for Java#
#---------------------------------------------------------#
# 內容檔範例#
#                                                         #
# 此檔案命名為 jt400.properties 且儲存於#
# com/ibm/as400/access 目錄, 此目錄是即為 #
# classpath 指向的目錄。#
#=========================================================#

#---------------------------------------------------------#
# PROXY 伺服器系統內容#
#---------------------------------------------------------#

# 此系統內容指定 PROXY 伺服器主電腦名稱
# 以及埠號, 指定的格式如下：hostName:portNumber
# 埠號是可選用的。
com.ibm.as400.access.AS400.proxyServer=hqoffice

# 此系統內容指定透過 SSL 加密的虛擬
# 資料流程。有效值為：
# 1 - Proxy 從屬站到 PROXY 伺服器
# 2 - Proxy 伺服器到 AS/400
# 3 - Proxy 從屬站到 Proxy, 以及 PROXY 伺服器到 AS/400
com.ibm.as400.access.SecureAS400.proxyEncryptionMode=1

# 此系統內容指定 PROXY 伺服器尋找閒置中連線的頻率
# (以秒為單位)。PROXY 伺服器啟動一個尋找從屬站的緒,
# 所尋找的是已不再保持通信的從屬站。使用此內容來設
# 定緒尋找閒置中連線的頻率。
#
com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval=7200

# 此系統內容指定在清除從屬站之前其可閒置的時間
# (以秒為單位)。PROXY 伺服器啟動一個尋找不再通信
# 之從屬站的緒。使用此內容設定清除從屬站之前, 其
# 可閒置的時間。
#
com.ibm.as400.access.TunnelProxyServer.clientLifetime=2700


#---------------------------------------------------------#
# 追蹤系統內容#
#---------------------------------------------------------#

# 此系統內容指定要啟用的追蹤種類。
# 這是一個包含任何追蹤種類組合的逗點分界式清單。
# 完整的追蹤種類清單定義於 Trace 類別中。
#
com.ibm.as400.access.Trace.category=error,warning,information

# 此系統內容指定追蹤輸出要寫入的檔案。
# 預設為將追蹤輸出寫入 System.out。
```

```
com.ibm.as400.access.Trace.file=c:\\temp\\trace.out


#-------------------------------------------------------#
# 指令呼叫系統內容#
#-------------------------------------------------------#

# 此系統指定是否應將 CommandCalls 假設為安全緒。
# 如果為 true, 則假設所有的 CommandCalls 為
# 安全緒。如果為 false, 則假設所有的 CommandCalls 為
# 非安全緒。如果已在物件上執行 CommandCall.setThreadSafe(true/false)
# 或 AS400.setMustUseSockets(true), 則系統將不處理
# 所指定的 CommandCall 物件。
#
com.ibm.as400.access.CommandCall.threadSafe=true


#-------------------------------------------------------#
# 程式呼叫系統內容#
#-------------------------------------------------------#

# 此系統指定是否應將 ProgramCalls 假設為安全緒。
# 如果為 true, 則假設所有的 ProgramCalls 為
# 安全緒。如果為 false, 則假設所有的 ProgramCalls 為
# 非安全緒。如果已在物件上執行 ProgramCall.setThreadSafe(true/false)
# 或 AS400.setMustUseSockets(true), 則系統將不處理
# 所指定的 ProgramCall 物件。
#
com.ibm.as400.access.ProgramCall.threadSafe=true


# 結束
```

範例：系統內容類別原始檔

```
//=============================================================
// IBM Toolbox for Java
//-------------------------------------------------------------
// Sample properties class source file
//
// Compile this source file and store the class file in
// the classpath.
//=============================================================
package com.ibm.as400.access;

public class Properties
extends java.util.Properties
{
 public Properties ()
   {
      /*-----------------------------------------------------------*/
/* Proxy server system properties*/
      /*-----------------------------------------------------------*/

// This system property specifies the proxy server host name
// and port number, specified in the format: hostName:portNumber
// The port number is optional.
put ("com.ibm.as400.access.AS400.proxyServer", "hqoffice");

// This system property specifies which portion of the proxy
// data flow is encrypted via SSL.Valid values are:
// 1 - Proxy client to proxy server
// 2 - Proxy server to iSeries or AS/400e server
// 3 - Proxy client to proxy, and proxy server to iSeries or AS/400e server
put("com.ibm.as400.access.SecureAS400.proxyEncryptionMode", "1");

// This system property specifies how often, in seconds,
// the proxy server will look for idle connections. The
// proxy server starts a thread to look for clients that are
// no longer communicating. Use this property to set how
// often the thread looks for idle connections.
put("com.ibm.as400.access.TunnelProxyServer.clientCleanupInterval", "7200");

// This system property specifies how long, in seconds, a
// client can be idle before it is cleaned up. The proxy server
// starts a thread to look for clients that are no longer
// communicating. Use this property to set long a client can
// be idle before it is cleaned up.
put("com.ibm.as400.access.TunnelProxyServer.clientLifetime", "2700");

      /*-----------------------------------------------------------*/
/* Trace system properties */
      /*-----------------------------------------------------------*/

// This system property specifies which trace categories to enable.
// This is a comma-delimited list containing any combination of trace
// categories.The complete list of trace categories is defined in
```

```
// the Trace class.
put ("com.ibm.as400.access.Trace.category", "error,warning,information");

// This system property specifies the file to which trace output
// is written.The default is to write trace output to System.out.
put ("com.ibm.as400.access.Trace.file", "c:\temp\trace.out");


      /*-------------------------------------------------------*/
/* Command Call system properties*/
      /*-------------------------------------------------------*/

// This system property specifies whether CommandCalls should
// be assumed to be thread-safe. If true, all CommandCalls are
// assumed to be thread-safe. If false, all CommandCalls are
// assumed to be non-thread-safe. This property is ignored
// for a given CommandCall object if either
// CommandCall.setThreadSafe(true/false) or
// AS400.setMustUseSockets(true) has been performed on the object.
put ("com.ibm.as400.access.CommandCall.threadSafe", "true");


      /*-------------------------------------------------------*/
/* Program Call system properties*/
      /*-------------------------------------------------------*/

// This system property specifies whether ProgramCalls should
// be assumed to be thread-safe. If true, all ProgramCalls are
// assumed to be thread-safe. If false, all ProgramCalls are
// assumed to be non-thread-safe. This property is ignored
// for a given ProgramCall object if either
// ProgramCall.setThreadSafe(true/false) or
// AS400.setMustUseSockets(true) has been performed on the object.
put ("com.ibm.as400.access.ProgramCall.threadSafe", "true");
   }
}
```

»IBM Toolbox for Java 2 Micro Edition

IBM Toolbox for Java 2 Micro Edition 套件 (com.ibm.as400.micro) 可讓您 撰寫具下列功能的 Java 程式：允許
Tier0 無線裝置 (如個人數位助理 (PDA) 及行動電話) 直接存取 iSeries 資料與資源。

ToolboxME for iSeries 的其它相關資訊, 請參閱下列主題：

基本要求
說明使用 ToolboxME for iSeries 開發應用程式, 並在 Tier0 裝置上執行那些應用程式的基本要求。

下載及設定 ToolboxME for iSeries
告訴您如何將 ToolboxME for iSeries 下載至伺服器、工作站及 Tier0 裝置, 及在這些地方設定 ToolboxME for iSeries。

概念
簡介開發於 Tier0 裝置上執行的應用程式之重要概念定義。

ToolboxME for iSeries 類別
介紹 ToolboxME for iSeries 元件 (com.ibm.as400.micro 套件) 中的類別。 這些類別提供 Toolbox for Java access 類別、JDBC 支援及其它項目中的可用函數精簡集。

建立 ToolboxME for iSeries 程式
告訴您建立 Tier0 裝置上執行的 ToolboxME for iSeries 程式之步驟。
您可遵循指令來建立第一個自己的 ToolboxME for iSeries 程式。

範例
檢查、下載並執行 ToolboxMe for iSeries
工作範例, 以協助您瞭解如何建立及使用無線應用程式。 «

下載及設定 ToolboxME for iSeries

您必須單獨下載 ToolboxME for iSeries (jt400Micro.jar)，其包含於 JTOpen 中。 您可由也同時提供 ToolboxME for iSeries 設定之其餘資訊的 IBM Toolbox for Java/JTOpen 網站，下載 ToolboxME for iSeries。

在 Tier0 裝置、開發工作站及伺服器上設定 ToolboxME for iSeries 的方法不同：

- 為您的無線裝置建置一個應用程式 (使用 jt400Micro.jar)，並依建置製造商提供的文件中之說明安裝應用程式。
- 請確定 iSeries Host Servers 於包含目標資料的伺服器上啟動。
- 請確定要執行 MEServer 的系統能存取 jt400.jar。其它相關資訊，請參閱在開發工作站安裝 Toolbox for Java 及在 iSeries 伺服器安裝 Toolbox for Java。

使用 ToolboxME for iSeries 的重要概念

在進行 ToolboxME for iSeries Java 應用程式的開發之前， 您必須瞭解下列支配此項開發作業的概念及標準。

Java 2 Platform, Micro Edition (J2ME)

J2ME (TM) 施行 Java 2 標準，提供 Tier0 無線 裝置 (如個人數位助理 (PDA) 及行動電話) Java 執行時間環境。IBM Toolbox for Java 2 Micro Edition 遵守此標準。

Tier0 裝置

使用無線技術與電腦及網路連線的無線裝置，如 PDA 及行動電話，稱為 Tier0 裝置。此名稱的命名方式源自常用的 3 層應用程式模型。3 層模型描述組織成三個主要部份的分散式程式，每一個部份常駐在不同的電腦或網路：

- 第三層為資料庫及相關程式，通常常駐於與第二層不同的伺服器中。本層提供其它層用來執行作業使用的資訊，及對這些資訊的存取權。
- 第二層為商業邏輯，通常常駐於網路上分享出去的不同電腦 (通常為伺服器) 中。
- 第一層通常是應用程式部份，常駐在工作站中，包括使用者介面。

Tier0 裝置通常體積小、可攜帶、資源受限，像是 PDA 及行動電話。Tier0 裝置可代替第一層的裝置，或補其功能之不足。

連接限制裝置配置 (CLDC)

定義提供與大型裝置相同功能所需的最小 API 集，及必要 Java 虛擬機器 (Java VM) 功能之配置。CLDC 適用於廣泛的資源受限裝置，包括 Tier0 裝置。

其它相關資訊，請參閱 CLDC 及「K 虛擬 機器 (KVM」) 。

行動資訊裝置設定檔 (MIDP)

代表建置在現有配置上、適用於特定類型的裝置或作業系統之 API 集的設定檔。MIDP 建置於 CLDC 之上，提供一標準執行時間環境，可讓您對 Tier0 裝置動態地部署應用程式及服務。

其它相關資訊，請參閱 行動資訊 裝置設定檔 (MIDP)。

用於無線裝置的 Java 虛擬機器 (Java VM)

為了執行 Java 應用程式，您的 Tier0 裝置需要特別為無線裝置的有限資源設計 之 Java 虛擬機器 (Java VM)。您可使用的 JVM 包括：

- IBM J9 虛擬機器，IBM WebSphere Micro Environment 的一部份

- Sun K Virtual Machine (KVM)
- MIDP

相關資訊

您可使用任何一種建立用來協助您建置無線 Java 應用程式的開發工具。 有關此種工具的簡略清單，請參閱 Toolbox for Java 的相關資訊

若要得知無線裝置模擬器的其它相關資訊，及要下載無線裝置模擬器，請諮詢您要於其中執行應用程式的裝置或作業系統之網站。

ToolboxME for iSeries 類別

com.ibm.as400.micro 套件提供了撰寫應用程式的必要類別，可使用 Tier 0 裝置存取 iSeries 伺服器資料與資源。

註：若要使用 ToolboxMe for iSeries 類別，您必須先下載與設定 ToolboxME for iSeries 元件。

ToolboxME for iSeries 提供下列類別：

- MEServer 轉達從 Tier0 裝置到主電腦伺服器的要求
- 數個提供來自 Toolbox for Java access 套件的函數之子集的類別
  - AS400 - 登入 iSeries 系統
  - CommandCall - 呼叫 iSeries 指令
  - DataQueue - 讀取及寫入 iSeries 伺服器資料佇列
  - ProgramCall - 呼叫 iSeries 伺服器程式，並存取程式執行後傳回的資料
- 其它提供 JDBC 支援的類別，也包括來自 java.sql 套件的最小可用方法與資料集

»MEServer 類別

您可使用MEServer 類別， 履行來自使用 ToolboxME for iSeries jar 檔案的從屬站
應用程式之要求。MEServer 會建立 Toolbox for Java 物件，並代表從屬站應用程式呼叫這些物件中的方法。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。
其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

您可使用下列指令來啟動 MEServer：

```
java com.ibm.as400.micro.MEServer [options]
```

其中 [options] 為下列其中一項或數個項目：

-pcml pcml-doc1 [;pcml_doc2;...]

指定 PCML 文件以預載及剖析。您可以使用-pc 來縮寫此選項。

使用此選項的重要相關資訊，請參閱MEServer javadoc。

-port port
指定用來接受來自從屬站連線的埠。此選項可縮寫為 -po。預設埠是 3470。 您可以使用p
來縮寫此選項。

-verbose [true|false]
指定是否要將狀態及連線資訊列印至 System.out。您可以使用 來縮寫此選項。

-help
將用法資訊列印至 System.out。您可以使用或-? 來縮寫此選項。 預設值為不要列印用法資訊。

如果指定的埠已有另一個伺服器正於作用中，則 MEServer 將不會啟動。«

AS400 類別

micro 套件內的 AS400 類別 (com.ibm.as400.micro.AS400) 提供了 access 套件內的 AS400 類別 (com.ibm.as400.access.AS400) 中之可用函數的修正子集。 您可使用 ToolboxMe for iSeries AS400 類別，從裝置登入 iSeries 系統。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

AS400 類別提供下列功能：

- 連接 MEServer
- 從 MEServer 切斷連接

與 MEServer 的連線是間接地達成。例如，建立 AS400 物件後，便可在 CommandCall 中使用 run() 方法，來自動執行 connect()。 換句話說，您毋需直接呼叫 connect() 方法，除非要控制何時建立連線。

下列範例說明如何使用 AS400 類別來登入 iSeries 系統：

```
AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
  try
{
    system.connect();
}
    catch (Exception e)
{
    // Handle the exception
}
// Done with the system object.
system.disconnect();
```

»CommandCall 類別

micro 套件中的 CommandCall 類別 (com.ibm.as400.micro.CommandCall) 提 供了access 套件內的 CommandCall 類別 (com.ibm.as400.access.CommandCall) 中之 可用函數的修正子集。您可使用 CommandCall 類別, 從 Tier0 裝置呼叫 iSeries 指令。

註：若要使用 ToolboxMe for iSeries 類別, 您必須分下載與設定 ToolboxME for iSeries 元件

CommandCall run() 方 需要一個 String (您想要執行的指令), 並會在執行 String 之後, 傳回產生的所有訊息。如果指令執行完畢後未產生任何訊息, 則 run() 方法會傳回空的 String 陣列。

下列範例示範如何使用 CommandCall：

```
// Work with commands.
AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
 try
{
    // Run the command "CRTLIB FRED."
    String[] messages = CommandCall.run(system, "CRTLIB FRED");
    if (messages != null)
    {
        // Note that there was an error.
        System.out.println("指令失敗：");
        for (int i = 0; i < messages.length; ++i)
        {
            System.out.println(messages[i]);
        }
    }
     else
    {
        System.out.println("指令成功！");
    }
}
   catch (Exception e)
{
    // Handle the exception
}
// Done with the system object.
system.disconnect();
```

«

»

DataQueue 類別

micro 套件內的 DataQueue 類別 (com.ibm.as400.micro.DataQueue) 提供了 access 套件內的 DataQueue 類別 (com.ibm.as400.access.DataQueue) 中之可用函數的修正子集。 您可使用 DataQueue 類別讓 裝置可讀取或寫入 iSeries 伺服器中的資料佇列。

註：若要使用 ToolboxMe for iSeries 類別，您必須分散與 設定 ToolboxME for iSeries 元件。

DataQueue 類別包含下列方法：

- 讀取或撰寫 為 String 的登錄
- 讀取或撰寫 為位元組陣列的登錄

若要讀取或撰寫項目，您必須提供常駐資料佇列的 iSeries 系統之名稱，
及資料佇列之完整的整合檔案系統路徑名稱。沒有可用的項目時，讀取項目會傳回 NULL 值。

下列範例示範如何使用 DataQueue 類別從 iSeries 系統中的資料佇列讀取項目， 及將項目寫入資料佇列：

```
   AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");
    try
   {
       // Write to the Data Queue.
       DataQueue.write(system, "/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ",
"部份文字");

       // Read from the Data Queue.
       String txt = DataQueue.read(system,
"/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ");
   }
      catch (Exception e)
   {
       // Handle the exception
   }
   // Done with the system object.
   system.disconnect();
```

«

ProgramCall 類別

micro 套件內的 ProgramCall 類別 (com.ibm.as400.micro.ProgramCall) 提供了 access 套件內的 ProgramCall 類別 (com.ibm.as400.access.ProgramCall) 中之可用函數的修正子集。 您可使用 ProgramCall 類別，讓裝置呼叫 iSeries 程式， 並存取執行程式後傳回的資料。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。其它相關資訊，請參閱 ToolboxME for iSeries 基本要求及安裝。

若要使用 ProgramCall.run() 方法， 您必須提供下列參數：

- 要執行程式的系統
- 程式呼叫語言 (PCML) 文件的名稱
- 想要執行程式的名稱
- 包含您要設定的一或多個程式參數的名稱及關聯值之雜湊表
- 包含程式執行後將傳回的所有參數之名稱的字串陣列

ProgramCall 使用 PCML 來描述程式的輸入及輸出參數。PCML 檔案所在的機器必須與 MEServer 相同，且您必須在該機器的 CLASSPATH 下 包含 PCML 檔案的目錄。

您必須對 MEServer 註冊每一個 PCML 文件。註冊 PCML 文件的用意 即在告訴 MEServer 您要執行哪一個 PCML 定義的程式。 您可在執行期間或啟動 MEServer 時進行 PCML 文件的註冊。

包含程式參數的雜湊表或如何註冊 PCML 文件的其它相關資訊， 請參閱 ToolboxME for iSeries ProgramCall javadoc。 PCML 的其它相關資訊，請參閱 程式呼叫語言。

下列範例說明如何使用 ProgramCall 類別來使用 Integer 0 裝置執行伺服器中的程式。

```
   // Call programs.
   AS400 system = new AS400("mySystem", "myUserid", "myPwd", "myMEServer");

   String pcmlName = "qsyrusri.pcml"; // The PCML document describing the
program we want to use.
   String apiName = "qsyrusri";

   Hashtable parametersToSet = new Hashtable();
   parametersToSet.put("qsyrusri.receiverLength", "2048");
   parametersToSet.put("qsyrusri.profileName", "JOHNDOE" };

   String[] parametersToGet = { "qsyrusri.receiver.userProfile",
               "qsyrusri.receiver.previousSignonDate",
               "qsyrusri.receiver.previousSignonTime",
               "qsyrusri.receiver.displaySignonInfo" };

   String[] valuesToGet = null;

    try
   {
       valuesToGet = ProgramCall.run(system, pcmlName, apiName,
```

```
parametersToSet, parametersToGet);

        // Get and display the user profile.
        System.out.println("User profile: " + valuesToGet[0]);

        // Get and display the date in a readable format.
        char[] c = valuesToGet[1].toCharArray();
        System.out.println("Last Signon Date: " +
c[3]+c[4]+"/"+c[5]+c[6]+"/"+c[1]+c[2] );

        // Get and display the time in a readable format.
        char[] d = valuesToGet[2].toCharArray();
        System.out.println("Last Signon Time: " + d[0]+d[1]+":"+d[2]+d[3]);

        // Get and display the signon info.
        System.out.println("Signon Info: " + valuesToGet[3] );
    }
    catch (MEException te)
    {
        // Handle the exception.
    }
    catch (IOException ioe)
    {
        // Handle the exception
    }

    // Done with the system object.
    system.disconnect();
```

«

JdbcMe 類別

ToolboxME for iSeries 類別提供 JDBC 支援，包括 支援 java.sql 套件。這些類別主要是設計用來於 Tier 0 裝置中 執行的程式內使用。

下節討論 資料的存取與使用，及說 明JdbcMe 中有什麼，包括 單獨 JdbcMe 類別 的相關資訊。

資料的存取與使用

使用 Tier0 裝置來存取與更新資料時，您會想要裝置的功能運作如同您辦公室的系統一般。 但是，大部份 Tier0 裝置的開發著重於資料同步化。藉著使用資料同步化， 每一個 Tier0 裝置可具有主資料庫的特定資料之副本。使用者可定期地將裝置上 的資料與主資料庫的資料同步化。

動態資料的資料同步化不易實行。將動態資料同步化需要能對最新資料的即時存取。對許多企業而言，不希望在存取同步化資料時要花時間等候。再者，進行資料同步化的伺服器及裝置也需要特殊的硬體與軟體需求。

為了協助解決資料同步模型先天上的問題，ToolboxME for iSeries 中的 JdbcMe 類別可讓您執行即時的更新及存取主資料庫，但仍能在離線情況下進行資料儲存。因此，您的應用程式可在對主資料庫進行即時更新的同時，存取重要的離線資料。此扮演中間角色的方式兼具同步資料模型及即時資料模型的優點。

JdbcMe 中有什麼

根據定義， Tier0 裝置的任何一種驅動程式必須非常小。 但是 JDBC API 卻非常大。為此，JdbcMe 類別必須極小但仍能支援 JDBC 介面， 如此 Tier0 裝置才能用它來執行有意義的工作。

JdbcMe 類別提供下列 JDBC 功能：

- 插入或更新資料
- 異動控制及修改異動隔離層次的能力
- 同時可捲動及可更新的結果集
- 提供對儲存程序及磁碟機觸發程式呼叫之 SQL 支援

此外，JdbcMe 類別還包含一些唯一的特性：

- 可讓大部份的配置明細於伺服器端的單一點合併之廣用驅動程式
- 使資料能留存於離線儲存體的標準機制

JdbcMe 包含下列類別：

- JdbcMeConnection
- JdbcMeDriver
- JdbcMeException
- JdbcMeLiveResultSet

- JdbcMeOfflineData
- JdbcMeOfflineResultSet
- JdbcMeResultSetMetaData
- JdbcMeStatement

採用 SQL

ToolboxME for iSeries 提供 java.sql 套件，其遵循 JDBC 規格，但僅包含最小可用類別與方法集。提供的最小 SQL 函數集可讓 JdbcMe 類別的大小夠小，但仍足以用來執行常用的 JDBC 作業。

JdbcMeConnection 類別

JdbcMeConnection 類別提供 了 Toolbox for Java AS400 JDBCConnection 類別 中的 可用函數子集。您可使用 JdbcMeConnection 來啟用Tier0 裝置， 以存取主電腦伺服器的「DB2 廣用資料庫 (UDB)」資料庫。

註： 若要使用 ToolboxMe for iSeries 類別, 您必須分別下載及 設定 ToolboxME for iSeries 元件。其它相關資訊, 請參閱 ToolboxME for iSeries 基本要求及安裝

使用JdbcMeDriver.getConnection與 伺服器資料庫連接。getConnection() 方法會將全球資源定位器 (URL) 字串視為引數、 使用者 ID 及密碼。主電腦伺服器上的 JDBC 驅動程式管理員會試圖尋找一驅動程式, 該驅動程式可連接至由此 URL 代表的資料庫。JdbcMeDriver 使用下列的 URL 語法：

```
      jdbc:as400://server-name/default-
schema;meserver=<server>[:port];[other properties];
```

您必須指定一個伺服器名稱, 否則 JdbcMeDriver 會丟出異常。預設的綱目是可選用的。
如果不指定一個埠, JdbcMeDriver 會使用埠 3470。同時, 您可在 URL 內設定不同的 JDBC 內容。
若要設定內容, 請使用下列語法：

```
      name1=value1;name2=value2;...
```

請參閱 JDBC 內容, 以取得 JdbcMeDriver 支援的內容清單。

範例：使用 JdbcMeDriver 與伺服器連接

範例 1： 在不指定預設綱目、埠或 JDBC 內容的情況下 與伺服器資料庫連接。使用者 ID 及密碼會被指定為方法中的參數。

```
        // Connect to system 'mysystem'. No default schema, port or
        // properties are specified.
    Connection c  = JdbcMeDriver.getConnection
        ("jdbc:as400://mysystem.helloworld.com;
          meserver=myMeServer;"
                    "auser",
                    "apassword");
```

範例 2： 在指定綱目及 JDBC 內容的情況下與伺服器資料庫連接。使用者 ID 及密碼會被指定為方法中的參數。

```
        // Connect to system 'mysystem'. Specify a schema and
        // two JDBC properties. Do not specify a port.
    Connection c2 = JdbcMeDriver.getConnection
        ("jdbc:as400://mysystem.helloworld.com/mySchema;
         meserver=myMeServer;
         naming=system;
         errors=full;"
                    "auser",
                    "apassword");
```

範例 3：與伺服器資料庫連接；使用全球資源定位器 (URL) 指定內容 (包括使用者 ID 和密碼)。

```
        // Connect using properties. The properties are set on the URL
                    // instead of through a properties
    Connection c = DriverManager.getConnection
         ("jdbc:as400://mySystem;
           meserver=myMeServer;
           naming=sql;
           errors=full;
           user=auser;
           password=apassword");
```

範例 4：中斷資料庫的連線。對連線物件使用 close() 方法，中斷與伺服器的連接：

```
    c.close();
```

«

JdbcMeDriver 類別

JdbcMeDriver 類別提供 了 Toolbox for Java AS400JDBCDriver 類別 中 可用函數子集。您可在 ier0 從屬站應用程式中使用 JdbcMeDriver， 來執行沒有參數的簡單 SQL 陳述式，並取得陳述式產生的 esultSets。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

您並非直接地註冊 JdbcMeDriver；而是於 JdbcMeConnection.getConnection() 方法中 在 URL 指定的 er 來決定驅動程式。例如，若要載 入 IBM Developer Kit for Java JDBC 驅動程式 (稱為「原有的」驅動程式)，請使用下列程式碼：

```
Connection c = JdbcMeDriver.getConnection
     ("jdbc:as400://mysystem.helloworld.com;
       meserver=myMeServer;
       driver=native;
       user=auser;
       password=apassword");
```

IBM Toolbox for Java JDBC 驅動程式和其它從伺服器取得資料的 IBM Toolbox for Java 類別不一樣，它不需要 AS400 物件作為輸入參數。不過，其會在內部使用 AS400 物件，且您必須直接提供使用者 ID 和密碼。請在 URL 中提供使用者 ID 和密碼，或以 getConnection() 方法的參數之形式提供使用者 ID 和密碼。

使用 getConnection() 的範例，請參閱 JDBCMeConnection 。«

JdbcMeResultSet 類別

ToolboxME for iSeries 結果集類別為：

- JdbcMeLiveResultSet
- JdbcMeOfflineResultSet
- JdbcMeResultSetMetaData

JdbcMeLiveResultSet 與 JdbcMeOfflineResultSet 包含相同的功能，除了：

- JdbcMeLiveResultSet 擷取資料的方法是對伺服器的資料庫進行呼叫
- JdbcMeOfflineResultSet 由本機上之資料庫擷取資料

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。
其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

JdbcMeLiveResultSet

JdbcMeLiveResultSet 類別提供了 Toolbox for Java AS400JDBCResultSet 類別中的可用函數子集。您可在 Tier0 從屬站應用程式中使用 JdbcMeLiveResultSet，來存取執行查詢所產生的資料表。

JdbcMeLiveResultSet 會依順序擷取表格列。在一列中，您可以任何次序存取欄位值。JdbcMeLiveResultSet 包含的方法可讓您執行下列作業：

- 擷取儲存在結果集中不同類型的資料
- 移動游標至指定的列（上一列、現行列、下一列等等）
- 插入、更新及刪除列
- 更新直欄（使用 String 及 int 值）
- 擷取描述於結果集中直欄的 ResultSetMetaData 物件

游標為一內部指標，結果集用它來指向結果集中正被 Java 程式存取的列。JDBC 2.0 另外提供一些方法，來存取資料庫內的特定位置：

可捲動的游標位置	
absolute	moveToInsertRow
first	previous
last	relative
moveToCurrentRow	

捲動功能

如果透過執行一個陳述式來建立結果集，您可以向後（最後一個到第一個）或向前（第一個到最後一個）移動

(捲動) 表格中的橫列。

支援這個移動的結果集稱為可捲動的結果集。可捲動的結果集也支援相對與絕對定位。
相對位置可讓您經由指定相對於現行列的位置，來移至結果集中的某一列。絕對位置可讓您經由在
結果集中指定一個位置，直接移到該橫列。

透過 JDBC 2.0，您有兩個額外捲動功能可在使用 ResultSet 類別時使用：不可捲動及可捲動結果集。

不可捲動結果集通常不會感應到當開啟它時所做的變更，而可捲動結果則可感應到變更。IBM Toolbox for Java
JDBC 驅動程式不支援不可捲動的結果集。

可更新的結果集

在您的應用程式中，您可以使用使用唯讀並行處理 (不對資料做任何變更) 的結果集，或 可更新並行處理
(容許更新資料並使用資料寫入鎖定來控制不同異動對同一資料的存取權) 的結果集。
在可更新結果集中，橫列可被更新、插入及刪除。

請參閱 方法總結， 以取得 JdbcMeResultSet 中可用更新方法的完整報表。

範例：可更新的結果集

下面範例將告訴您如何使用當開啟時容許變更資料 (更新並行處理)，以及容許變更結果集 (可捲動) 的結果集。

```
        // Connect to the server.
    Connection c = JdbcMeDriver.getConnection
        ("jdbc:as400://mySystem;
          meserver=myMeServer;
          user=auser;
          password=apassword");

                    // Create a Statement object.  Set the result set
            // concurrency to updatable.
    Statement s = c.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_UPDATABLE);

                    // Run a query. The result is placed
                    // in a ResultSet object.
    ResultSet rs = s.executeQuery ("SELECT NAME,ID FROM MYLIBRARY.MYTABLE
FOR UPDATE");

                    // Iterate through the rows of the ResultSet.
        // the row, we will update it with a new ID.
    int newId = 0;
    while (rs.next ())
    {

            // Get the values from the ResultSet. The first value
            // is a string, and the second value is an integer.
       String name = rs.getString("NAME");
       int id = rs.getInt("ID");
```

```
        System.out.println("Name = " + name);
        System.out.println("Old id = " + id);

              // Update the id with a new integer.
        rs.updateInt("ID", ++newId);

                     // Send the updates to the server.
        rs.updateRow ();

        System.out.println("New id = " + newId);
    }

         // Close the Statement and the Connection.
    s.close();
    c.close();
```

JdbcMeOfflineResultSet 類別

JdbcMeOfflineResultSet 類別提供了 Toolbox for Java AS400JDBCResultSet 類別中的可用函數子集。您可在 Tier0 從屬站應用程式中 使用 JdbcMeOfflineResultSet，來存取執行查詢所產生的資料表。

您可使用 JdbcMeOfflineResultSet 類別與常駐在 Tier0 裝置的資料搭配使用。
常駐在裝置中的資料可能已存在於該處，或您可藉由呼叫 JdbcMeStatement.executeToOfflineData() 方法將資料存放該處。executeToOfflineData() 方法會下載所有查詢所得的資料，並將它們儲存在裝置中。
然後您可使用 JdbcMeOfflineResultSet 類別來存取儲存的資料。

JdbcMeOfflineResultSet 包含的方法可讓您執行下列動作：

- 擷取儲存於結果集中不同類型的資料
- 移動游標至指定的列 (上一列、現行列、下一列等等)
- 插入、更新及刪除列
- 更新直欄 (使用 String 及 int 值)
- 擷取描述於結果集中直欄的 ResultSetMetaData 物件

您可使用 JdbcMe 類別中所示的函數，提供將本機資料庫與 iSeries 伺服器的資料庫同步化之功能。

JdbcMeResultSetMetaData 類別

JdbcMeResultSetMetaData 類別提供了 Toolbox for Java AS400JDBCResultSetMetaData 類別 中的可用函數子集。您可在 Tier0 從屬站應用程式中使用 JdbcMeResultSetMetaData，來決定 JDBCResultSet 中的直欄類型與特性。

下列範例說明如何使用 JdbcMeResultSetMetaData 類別：

```
        // Connect to the server.
    Connection c = JdbcMeDriver.getConnection
        ("jdbc:as400://mySystem;
          meserver=myMeServer;
```

```
            user=auser;
            password=apassword");

                        // Create a Statement object.
     Statement s = c.createStatement();

          // Run a query. The result is placed in a ResultSet object.
     JdbcMeLiveResultSet rs = s.executeQuery ("SELECT NAME,ID FROM
MYLIBRARY.MYTABLE");

                        // Iterate through the rows of the ResultSet.
     while (rs.next ())
     {

               // Get the values from the ResultSet. The first value is
               // a string, and the second value is an integer.
          String name = rs.getString("NAME");
          int id = rs.getInt("ID");

          System.out.println("Name = " + name);
          System.out.println("ID = " + id);
     }

          // Close the Statement and the Connection.
     s.close();
     c.close();
```

«

JdbcMeOfflineData 類別

JdbcMeOfflineData 類別為一離線資料儲存庫，設計使用於 Tier0 裝置上。此儲存庫是通用的，不受您使用的設定檔 及 Java 虛擬機器 (Java VM) 的影響。其它相關資訊，請參閱 ToolboxME for iSeries 概念。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

JdbcMeOfflineData 類別提供的方法可讓您執行下列功能：

- 建立離線資料儲存庫
- 開啟現有的儲存庫
- 取得儲存庫中的記錄數
- 取得及刪除個別記錄
- 更新記錄 (Robb: the set() method, right?)
- 新增記錄至儲存庫的尾端
- 關閉儲存庫

使用 JdbcMeOfflineData 類別的範例，請參閱 ToolboxMe for iSeries 範例 使用 ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java。

JdbcMeStatement 類別

JdbcMeStatement 類別提供 了 Toolbox for JAva AS400JDBCStatement 類別 中的 可用函數子集。您可在 Tier0 從屬站應用程式中使用 JdbcMeStatement，來執行沒有參數的簡單 SQL 陳述式， 及取得陳述式所產生的 ResultSets。

註：若要使用 ToolboxMe for iSeries 類別，您必須分別下載及設定 ToolboxME for iSeries 元件。其它相關資訊，請參閱 下載及設定 ToolboxME for iSeries。

Statement 類別

您可使用JdbcMeConnection.createStatement()來建立新的 Statement 物件。

下列範例說明如何使用 JdbcMeStatement 物件：

```
        // Connect to the server.
    JdbcMeConnection c = JdbcMeDriver.getConnection(
        "jdbc:as400://mysystem.helloworld.com/mylibrary;
         naming=system;
         errors=full;
         meserver=myMeServer;
         user=auser;
         password=apassword");

                    // Create a Statement object.
    JdbcMeStatement s = c.createStatement();

        // Run an SQL statement that creates a table in the database.
    s.executeUpdate("CREATE TABLE MYLIBRARY.MYTABLE (NAME VARCHAR(20), ID
INTEGER)");

        // Run an SQL statement that inserts a record into the table.
    s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES
('DAVE', 123)");

        // Run an SQL statement that inserts a record into the table.
    s.executeUpdate("INSERT INTO MYLIBRARY.MYTABLE (NAME, ID) VALUES
('CINDY', 456)");

                    // Run an SQL query on the table.
    JdbcMeLiveResultSet rs = s.executeQuery("SELECT * FROM
MYLIBRARY.MYTABLE");

        // Close the Statement and the Connection.
    s.close();
    c.close();
```

»建立及執行 ToolboxME for iSeries 程式

此資訊可讓您編輯、編譯並執行範例 ToolboxME for iSeries 程式。 您也可將此資訊用來作為建立、測試及執行 ToolboxME for iSeries 工作範例 及您自己的 ToolboxME for iSeries 應用程式之一般手冊使用。

範例程式使用「K 虛擬機器 (KVM)」，並允許使用者執行所有 JDBC 查詢。 接著，使用者可對查詢結果執行 JDBC 動作 (next、previous、close、commit 及 rollback)。

在開始建立任一 ToolboxME for iSeries 範例前，請先確定您的環境 符合ToolboxME for iSeries 基本要求

建立 ToolboxME for iSeries 範例

若要建立 Tier0 裝置的 ToolboxME for iSeries 範例程式，請完成下列步驟：

1. 複製 ToolboxME for iSeries 範例的 Java 程式，稱為 JdbcDemo.java。
2. 在所選的文字或 Java 編輯器中，如程式說明所述般變更程式碼，然後以名稱 JdbcDemo.java 儲存檔案。

   註： 請考慮使用無線應用程式開發工具，如此可更容易地完成剩餘的步驟。
   某些無線應用程式開發工具可在單一步驟中編譯、預先驗證及建立程式，然後自動在模擬器中加以執行。

3. 編譯 JdbcDemo.java，請確定是指向包含 KVM 類別的 .jar 檔案。
4. 預先驗證可執行檔，方法是使用無線應用程式開發工具，或使用 Java preverify 指令。
5. 為 Tier0 裝置的作業系統建立適當類型的可執行檔。例如，對 Palm OS，可建立稱為 JdbcDemo.prc 的檔案。
6. 測試程式。如果已安裝了模擬器，便可在模擬器中執行程式，以測試程式並看其執行後的情況如何。

   註： 如果在您的無線裝置上測試程式但不使用無線應用程式開發工具， 請確定已先在裝置中載入所選的 Java 虛擬機器 (Java VM) 或 MIDP。

請參閱 ToolboxME for iSeries 概念 以取得概念、無線應用程式開發工具及模擬器的相關資訊。

執行 ToolboxME for iSeries 範例

若要在 Tier0 裝置中執行 ToolboxME for iSeries 範例程式，請完成下列作業：

- 使用 Tier0 裝置製造商所提供的指令，將可執行檔載入裝置。
- 啟動 MEServer
- 按一下 JdbcDemo 圖示，在 Tier0 裝置中執行 JdbcDemo 程式。

ToolboxME for iSeries 工作範例

以下 ToolboxMe for iSeries 工作範例說明 ToolboxME for iSeries 與行動資訊裝置設定檔 (MIDP) 搭配使用的方法。
您可使用下列鏈結來檢視所選的範例原始檔，或下載所有建置工作範例無線應用程式所需之 exource 檔案：

範例：ToolboxME for iSeries、MIDP 及 JDBC

範例：ToolboxME for iSeries、MIDP 及 IBM Toolbox for Java

下載 ToolboxME for iSeries 工作範例

如何建置 ToolboxME for iSeries 應用程式的其它相關資訊，請參閱建置及執行 ToolboxME for iSeries 程式。

常見問題 (FAQ)

IBM Toolbox for Java 常見問題 (FAQ) 提供與最佳化您的 IBM Toolbox for Java 效能、執行疑難排解、使用 JDBC 及其它動作有關的問題答案：

- IBM Toolbox for Java FAQ：尋找許多問題類型的答案，包括增進效能、使用 OS/400、執行疑難排解等。
- IBM Toolbox for Java JDBC FAQ：尋找關於使用 JDBC 與 Toolbox for Java 的問題答案

程式碼範例

下列清單為 IBM Toolbox for Java 主題中都會用到的許多範例提供進入點的鏈結。

存取類別	Bean	Graphical Toolbox
HTML 類別	PCML	ReportWriter 類別
資源類別	RFML	Security 類別
Servlet 類別	簡單範例	程式設計秘訣
ToolboxMe for iSeries	公用程式類別	Vaccess 類別

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

範例：存取類別

本節會列出存取類別文件中提供的程式碼範例。

AS400JPing

- 範例：在 Java 程式內使用 AS400JPing

BidiTransform

- 範例：使用 AS400BidiTransform 類別來轉換雙向文字

CommandCall

- 範例：使用 CommandCall 來執行伺服器上的指令
- 範例：使用 CommandCall 提示伺服器的名稱、要執行的指令及列印結果

ConnectionPool

- 範例：使用 AS400ConnectionPool 建立伺服器的連線

DataArea

- 範例：建立及寫入到小數資料區

資料轉換與說明

- 範例：使用 FieldDescription、RecordFormat 及 Record 類別
- 範例：將資料放在佇列上
- 範例：從佇列讀取資料

DataQueue

- 範例：如何建立 DataQueue 物件、讀取資料及切斷連線
- 範例：將資料放在佇列上
- 範例：從佇列讀取資料

數位認證

- 範例：列出屬於使用者的數位認證

EnvironmentVariable

- 範例：建立、設定及取得環境變數

異常情況

- 範例：抓取一個丟出的異常情況、擷取回覆碼及顯示異常情況文字

FTP

- 範例：使用 FTP 類別自伺服器目錄中，複製一組檔案
- 範例：使用 AS400FTP 類別自目錄中複製一組檔案

整合檔案系統

- 範例：使用 IFSFile
- 範例：使用 IFSFile.listFiles() 方法列出目錄內容
- 範例：使用 IFSFile 類別複製檔案
- 範例：使用 IFSFile 類別列出目錄內容
- 範例：如何使用 IFSJavaFile 代替 java.io.File
- 範例：使用 IFSFile 類別列出伺服器上的目錄內容

JavaApplicationCall

- 範例：自從屬站執行伺服器上的程式，輸出 "Hello World!"

JDBC

- 範例：使用 JDBC 驅動程式來建立及擴大表格。
- 範例：使用 JDBC 驅動程式來查詢表格並輸出它的內容。

工作

- 範例：使用快取記憶體擷取及變更工作資訊
- 範例：列出所有作用中的工作
- 範例：列印工作日誌中特定使用者的所有訊息
- 範例：列出特定使用者的工作識別資訊
- 範例：取得伺服器上的工作清單，並列出工作狀態及工作識別字
- 範例：顯示工作日誌中屬於現行使用者的工作的訊息

訊息佇列

- 範例：如何使用訊息佇列物件
- 範例：列印訊息佇列的內容
- 範例：擷取及列印訊息
- 範例：列出訊息佇列的內容
- 範例：使用 Using AS400Message 與 CommandCall
- 範例：使用 Using AS400Message 與 ProgramCall

NetServer

- 範例：使用 NetServer 物件來變更 NetServer 的名稱

列印

- 範例：從輸入串流建立排存檔
- 範例：使用 SCS3812Writer 類別產生一個 SCS 資料串流
- 範例：讀取現存的排存檔
- 範例：以非同步方式列出使用 PrintObjectListListener 介面的所有排存檔
- 範例：以非同步方式列出系統中所有的排存檔,不使用 PrintObjectListListener 介面
- 範例：以同步方式列出所有排存檔

許可權

- 範例：設定 AS400 物件的權限

程式呼叫

- 範例：使用 ProgramCall
- 範例：使用 ProgramCall 來擷取系統狀態
- 範例：傳送具有程式參數物件的參數資料

QSYSObjectPathName

- 範例：建置一個整合檔案系統名稱
- 範例：使用 QSYSObjectPathName.toPath() 建置一個 AS400 物件名稱
- 範例：使用 QSYSObjectPathName 來剖析整合檔案系統路徑名稱

記錄層次存取

- 範例：循序存取檔案
- 範例：使用記錄層次存取類別來讀取檔案
- 範例：使用記錄層次存取類別, 依索引來讀取記錄
- 範例：使用 LineDataRecordWriter 類別

服務程式呼叫

- 範例：使用 ServiceProgramCall 呼叫程序

系統狀態

- 範例：以 SystemStatus 類別使用快取記憶體

系統儲存區

- 範例：設定 SystemPool 的最大錯誤大小

SystemValue

- 範例：使用 SystemValue 及 SystemValueList

追蹤

- 範例：使用 Trace.setTraceOn() 方法
- 範例：使用追蹤的偏好方式
- 範例：使用元件追蹤

使用者群組

- 範例：擷取使用者的清單
- 範例：列出群組的所有使用者

使用者空間

- 範例：如何建立使用者空間

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

範例：使用 CommandCall

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Command call example.This program prompts the user
// for the name of the server and the command to run, then
// prints the result of the command.
//
// This source is an example of IBM Toolbox for Java "CommandCall"
//
///////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class CommandCallExample extends Object
{
 public static void main(String[] parmeters)
   {

// Created a reader to get input from the user
BufferedReader inputStream = new BufferedReader(new
InputStreamReader(System.in),1);



// Declare variables to hold the system name and the command to run
String systemString= null;
String commandString = null;

System.out.println( " " );



// Get the system name and the command to run from the user
 try
      {
 System.out.print("系統名稱：");
 systemString = inputStream.readLine();

 System.out.print("指令：");
 commandString = inputStream.readLine();
      }
catch (Exception e)

System.out.println( " " );



// Create an AS400 object.This is the system we send the command to
AS400 as400 = new AS400(systemString);
```

```
// Create a command call object specifying the server that will
// recieve the command.
CommandCall command = new CommandCall( as400 );



 try
      {
 // Run the command.
 if (command.run(commandString))
 System.out.print( "指令順利完成" );
else
 System.out.print( "指令失敗" );



 // If messages were produced from the command, print them
 AS400Message[] messagelist = command.getMessageList();

 if (messagelist.length > 0)
         {
 System.out.println( ", 指令訊息：" );
 System.out.println( " " );
         }

 for (int i=0; i < messagelist.length; i++)
         {
System.out.print( messagelist[i].getID() );
System.out.print( ": " );
System.out.println( messagelist[i].getText() );
         }
      }
catch (Exception e)
      {
 System.out.println( "指令 " + command.getCommand() + " 未執行" );
      }

 System.exit(0);
   }
}
```

範例：使用 AS400ConnectionPool

註： 請詳讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// AS400ConnectionPooling example.  This program uses an AS400ConnectionPool
to
// create connections to an iSeries system.
// Command syntax:
//    AS400ConnectionPooling system myUserId myPassword
//
// For example,
//    AS400ConnectionPooling MySystem MyUserId MyPassword
//
////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

public class AS400ConnectionPooling
{
    public static void main(String[] parameters)
   {
      // Check the input parameters.
      if (parameters.length != 3)
      {
         System.out.println("");
         System.out.println("Usage:");
         System.out.println("");
         System.out.println("   AS400ConnectionPooling system userId
password");
         System.out.println("");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("");
         System.out.println("   AS400ConnectionPooling MySystem MyUserId
MyPassword");
         System.out.println("");
    return;
      }

      String system   = parameters[0];
      String userId   = parameters[1];
      String password = parameters[2];

     try
      {
         // Create an AS400ConnectionPool.
         AS400ConnectionPool testPool = new AS400ConnectionPool();

         // Set a maximum of 128 connections to this pool.
         testPool.setMaxConnections(128);

         // Set a maximum lifetime for 30 minutes for connections.
```

```
        testPool.setMaxLifetime(1000*60*30);     // 30 min  Max lifetime
since created.

        // Preconnect 5 connections to the AS400.COMMAND service.
        testPool.fill(system, userId, password, AS400.COMMAND, 1);
        System.out.println();
        System.out.println("Preconnected 1 connection to the AS400.COMMAND
service");

        // Call getActiveConnectionCount and getAvailableConnectionCount to
see how many
        // connections are in use and available for a particular system.
        System.out.println("Number of active connections: " +
testPool.getActiveConnectionCount(system, userId));
        System.out.println("Number of available connections for use: " +
testPool.getAvailableConnectionCount(system, userId));

        // Create a connection to the AS400.COMMAND service. (Use the
service number constants
        // defined in the AS400 class (FILE, PRINT, COMMAND, DATAQUEUE,
etc.))
        // Since connections have already been filled, the usual time spent
connecting to the command
        // service is avoided.
        AS400 newConn1 = testPool.getConnection(system, userId, password,
AS400.COMMAND);

        System.out.println();
        System.out.println("getConnection gives out an existing connection
to user");
        System.out.println("Number of active connections: " +
testPool.getActiveConnectionCount(system, userId));
        System.out.println("Number of available connections for use:  " +
testPool.getAvailableConnectionCount(system, userId));

        // Create a new command call object and run a command.
        CommandCall cmd1 = new CommandCall(newConn1);
        cmd1.run("CRTLIB FRED");

        // Return the connection to the pool.
        testPool.returnConnectionToPool(newConn1);

        System.out.println();
        System.out.println("Returned a connection to pool");
        System.out.println("Number of active connections: " +
testPool.getActiveConnectionCount(system, userId));
        System.out.println("Number of available connections for reuse: " +
testPool.getAvailableConnectionCount(system, userId));

        // Create a connection to the AS400.COMMAND service.  This will
return the same
        // object as above for reuse.
        AS400 newConn2 = testPool.getConnection(system, userId, password,
AS400.COMMAND);

        System.out.println();
        System.out.println("getConnection gives out an existing connection
to user");
```

```
            System.out.println("Number of active connections: " +
testPool.getActiveConnectionCount(system, userId));
            System.out.println("Number of available connections for reuse: " +
testPool.getAvailableConnectionCount(system, userId));

            // Create a connection to the AS400.COMMAND service.  This will
create a new
            // connection as there are not any connections in the pool to
reuse.
            AS400 newConn3 = testPool.getConnection(system, userId, password,
AS400.COMMAND);

            System.out.println();
            System.out.println("getConnection creates a new connection since
there are no connections available");
            System.out.println("Number of active connections: " +
testPool.getActiveConnectionCount(system, userId));
            System.out.println("Number of available connections for reuse: " +
testPool.getAvailableConnectionCount(system, userId));

            // Close the test pool.
            testPool.close();
         }
         catch (Exception e)
         {
            // If any of the above operations failed say the pool operations
failed
            // and output the exception.

            System.out.println("Pool operations failed");
           System.out.println(e);
               e.printStackTrace();
         }
      }
}
```

範例：使用 DataQueue 類別以在佇列上放置資料

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Data Queue example.This program uses the DataQueue class to put
// records on a data queue.
//
// This example uses the Record and Record format classes to put data
// on the queue.String data is converted from Unicode to ebcdic
// and numbers are converted from Java to the server format.Since data
// is converted the data queue entries can be read by a server program
// or a iSeries Access for Windows program as well as another Java program.
//
// This is the producer side of the producer/consumer example.It puts work
// items on the queue for the consumer to process.
//
// Command syntax:
//DQProducerExample system
//
///////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQProducerExample extends Object
{
 // Create a reader to get input from the user.
 static BufferedReader inputStream = new BufferedReader(new
InputStreamReader(System.in),1);

public static void main(String[] parameters)
   {
System.out.println( " " );

// if the system name was not specified, display help text and exit.
if (parameters.length >= 1)
      {
 try
        {
 // The first parameter is the system that contains the data queue.
 String system = parameters[0];

 // Create an AS400 object for the server that has the data queue.
 AS400 as400 = new AS400(system);

 // Build a record format for the format of the data queue entry.
 // This format matches the format in the DQConsumer class.A
 // record consists of:
 //- a four byte number-- the customer number
 //- a four byte number-- the part number
 //- a 20 character string -- the part description
```

```
 //- a four byte number-- the number of parts in this order
 // First create the base data types.
 BinaryFieldDescription customerNumber =
new BinaryFieldDescription(new AS400Bin4(), "CUSTOMER_NUMBER");

 BinaryFieldDescription partNumber =
new BinaryFieldDescription(new AS400Bin4(), "PART_NUMBER");

 CharacterFieldDescription partName =
new CharacterFieldDescription(new AS400Text(20, as400), "PART_NAME");

 BinaryFieldDescription quantity =
new BinaryFieldDescription(new AS400Bin4(), "QUANTITY");

 // Build a record format and fill it with the base data types.
 RecordFormat dataFormat = new RecordFormat();
 dataFormat.addFieldDescription(customerNumber);
 dataFormat.addFieldDescription(partNumber);
 dataFormat.addFieldDescription(partName);
 dataFormat.addFieldDescription(quantity);

 // Create the library that contains the data queue using CommandCall.
 CommandCall crtlib = new CommandCall(as400);
 crtlib.run("CRTLIB JAVADEMO");

 // Create the data queue object.
 DataQueue dq = new DataQueue(as400,
"/QSYS.LIB/JAVADEMO.LIB/PRODCONS.DTAQ");

 // Create the data queue just in case this is the first time this
 // program has run.The queue already exists exception is caught
 // and ignored.
 try
             {
dq.create(96);
             }
 catch (Exception e) {};

 // Get the first field of data from the user.
 System.out.print("輸入客戶號碼 (或 0 以退出)：");
 int customer = getInt();

 // While there is data to put on the queue.
 while (customer > 0)
             {
// Get the rest of the data for this order from the user.
System.out.print("輸入產品編號"
int part = getInt();

System.out.print("輸入數量：");
int quantityToOrder = getInt();

String description = "組件 " + part;

// Create a record based on the record format.The record
// is empty now but will eventually contain the data.
Record data = new Record(dataFormat);
```

```
// Set the values we received from the user into the record.
data.setField("CUSTOMER_NUMBER",new Integer(customer));
data.setField("PART_NUMBER",new Integer(part));
data.setField("QUANTITY", new Integer(quantityToOrder));
data.setField("PART_NAME", description);

// Convert the record into a byte array.The byte array is what
// is actually put to the data queue.
byte [] byteData = data.getContents();

System.out.println("");
System.out.println("將記錄寫入伺服器 ...");
System.out.println("");

// Write the record to the data queue.
dq.write(byteData);

// Get the next value from the user.
System.out.print("輸入客戶號碼 (或 0 以退出)：");
customer = getInt();
            }
         }
catch (Exception e)
         {
// If any of the above operations failed say the data queue
// operation failed and output the exception.

System.out.println("資料佇列作業失敗");
System.out.println(e);
         }
      }

// Display help text when parameters are incorrect.
else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法為："
 System.out.println("");
 System.out.println(" DQProducter 系統"
 System.out.println("");
 System.out.println("其中"
 System.out.println("");
 System.out.println(" system = 具有資料佇列的伺服器");
 System.out.println("");
 System.out.println("例如："
 System.out.println("");
 System.out.println(" DQProducerExample mySystem");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);
   }

 // This is the subroutine that gets a character string from the user
```

```
 // and converts it into an int.
 static int getInt()
   {
int i = 0;
boolean Continue = true;

while (Continue)
      {
 try
        {
String s = inputStream.readLine();

i = (new Integer(s)).intValue();
Continue = false;
        }
catch (Exception e)
        {
System.out.println(e);
System.out.print("請輸入一個數字 ==>");
        }
      }

return i;
   }
}
```

範例：使用 DataQueue 類別以讀取資料佇列登錄

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Data Queue example.This program uses the Data Queue classes to read
// entries off a data queue on the server.The entries were put on the
// queue with the DQProducer example program.
//
// This is the consumer side of the producer/consumer example.It reads
// entries off the queue and process them.
//
// Command syntax:
//DQConsumerExample system
//
////////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.net.*;
import com.ibm.as400.access.*;

public class DQConsumerExample extends Object
{
public static void main(String[] parameters)
   {
System.out.println( " " );

// if a system name was not specified, display help text and exit.
if (parameters.length >= 1)
      {
 try
        {

 // The first parameter is the system that contains the data queue.
 String system = parameters[0];

 // Create an AS400 object for the server that has the data queue.
 AS400 as400 = new AS400(system);


 // Build a record format for the format of the data queue entry.
 // This format matches the format in the DQProducer class.A
 // record consists of:
 //- a four byte number -- the customer number
 //- a four byte number -- the part number
 //- a 20 character string -- the part description
 //- a four byte number -- the number of parts in this order

 // First create the base data types.
 BinaryFieldDescription customerNumber =
new BinaryFieldDescription(new AS400Bin4(), "CUSTOMER_NUMBER");

 BinaryFieldDescription partNumber =
```

```
new BinaryFieldDescription(new AS400Bin4(), "PART_NUMBER");

 CharacterFieldDescription partName =
new CharacterFieldDescription(new AS400Text(20, as400), "PART_NAME"

 BinaryFieldDescription quantity =
new BinaryFieldDescription(new AS400Bin4(), "QUANTITY"

 // Build a record format and fill it with the base data types.
 RecordFormat dataFormat = new RecordFormat();

 dataFormat.addFieldDescription(customerNumber);
 dataFormat.addFieldDescription(partNumber);
 dataFormat.addFieldDescription(partName);
 dataFormat.addFieldDescription(quantity);

 // Create the data queue object that represents the data queue on
 // the server.
 DataQueue dq = new DataQueue(as400,
"/QSYS.LIB/JAVADEMO.LIB/PRODCONS.DTAQ");

 boolean Continue = true;

 // Read the first entry off the queue.The timeout value is
 // set to -1 so this program will wait forever for an entry.
 System.out.println("*** 等待登錄處理 ***");

 DataQueueEntry DQData = dq.read(-1);

 while (Continue)
             {

// We just read an entry off the queue.Put the data into
// a record object so the program can access the fields of
// the data by name.The Record object will also convert
// the data from server format to Java format.
Record data = dataFormat.getNewRecord(DQData.getData());

// Get two values out of the record and display them.
Integer amountOrdered = (Integer) data.getField("QUANTITY");
StringpartOrdered = (String)data.getField("PART_NAME");

System.out.println("需要" + amountOrdered + "的" + partOrdered);
System.out.println(" ");
System.out.println("*** 等待登錄處理 ***");

// Wait for the next entry.
DQData = dq.read(-1);
             }
          }
catch (Exception e)
          {
 // If any of the above operations failed say the data queue operation
 // failed and output the exception.
 System.out.println("資料佇列作業失敗");
System.out.println(e);
          }
      }
```

```
// Display help text when parameters are incorrect.
else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法是：");
 System.out.println("");
 System.out.println(" DQConsumerExample 系統");
 System.out.println("");
 System.out.println("其中");
 System.out.println("");
 System.out.println(" system = 具有資料佇列的伺服器");
 System.out.println("");
 System.out.println("例如:");
 System.out.println("");
 System.out.println(" DQConsumerExample mySystem");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);
   }
}
```

範例：使用 IFSFile

下面範例顯示如何使用 IFSFile 類別：

- 範例： 建立目錄
- 範例： 使用 IFSFile 異常以追蹤錯誤
- 範例： 列出具有 .txt 副檔名的檔案
- 範例： 使用 IFSFile listFiles() 方法以列出目錄內容

範例：建立目錄

```
                    // Create an AS400 object.  This new
                    // directory will be created on this
                    // iSeries.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create a file object that
                    // represents the directory.
IFSFile aDirectory = new IFSFile(sys, "/mydir1/mydir2/newdir");

                    // Create the directory.
if (aDirectory.mkdir())
   System.out.println("Create directory was successful");

                    // Else the create directory failed.
   else
{
                    // If the object already exists,
                    // find out if it is a directory or
                    // file, then display a message.
   if (aDirectory.exists())
   {
      if (aDirectory.isDirectory())
         System.out.println("Directory already exists");
   else
         System.out.println("File with this name already exists");
   }
   else
      System.out.println("Create directory failed");
}

                    // Disconnect since I am done
                    // accessing files.
sys.disconnectService(AS400.FILE);
```

範例：使用 IFSFile 異常以追蹤錯誤

發生錯誤時，IFSFile 類別會擲出 ExtendedIOException
異常。此異常情況包含一個指出失敗原因的回覆碼。即使 IFSFile 複製的 java.io 類別不丟出異常情況，IFSFile
類別也 會擲出此異常情況。例如，來自 java.io.File 的刪除方法傳回一布耳值來指出順 利完成或失敗。IFSFile

中的刪除方法傳回一布耳值，但如果刪除失敗的話則會擲出 ExtendedIOException。ExtendedIOException 提供 Java 程式關於刪除失敗原因的詳細資訊。

```
    // Create an AS400 object.
     AS400 sys = new AS400("mySystem.myCompany.com");

                        // Create a file object that
                        // represents the file.
     IFSFile aFile = new IFSFile(sys, "/mydir1/mydir2/myfile");

  // Delete the file.
     try
     {
        aFile.delete();

                        // The delete was successful.
        System.out.println("Delete successful ");
     }

                        // The delete failed. Get the return
                        // code out of the exception and
                        // display why the delete failed.
     catch (ExtendedIOException e)
     {
        int rc = e.getReturnCode();

        switch (rc)
        {
           case ExtendedIOException.FILE_IN_USE:
              System.out.println("Delete failed, file is in use ");
              break;

           case ExtendedIOException.PATH_NOT_FOUND:
              System.out.println("Delete failed, path not found ");
              break;

                           // ... for every specific error
                           // you want to track.

           default:
              System.out.println("Delete failed, rc = ");
              System.out.println(rc);
        }
     }
```

範例：列出具有 .txt 副檔名的檔案

範例 3：在列出目錄中的檔案時，Java 程式可以選用性地指定符合的基準。 符合的基準可以減少伺服器傳回至 IFSFile 物件的檔案數，以增進效能。下面範例說明如何列示具有副檔名 .txt 的檔案：

```
                        // Create the AS400 object.
     AS400 system = new AS400("mySystem.myCompany.com");

                        // Create the file object.
```

```
IFSFile directory = new IFSFile(system, "/");

                    // Generate a list of all files with
                    // extension .txt
String[] names = directory.list("*.txt");

                    // Display the names.
if (names != null)
  for (int i = 0; i < names.length; i++)
    System.out.println(names[i]);
   else
  System.out.println("No .txt files");
```

範例：使用 IFSFile listFiles() 方法以列出目錄內容

註： 請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// IFSListFiles example.  This program uses the integrated file system
// classes to list the contents of a directory on the server.
//
// Command syntax:
//    IFSListFiles system directory
//
// For example,
//    IFSListFiles MySystem /path1
//
///////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSListFiles extends Object
{
    public static void main(String[] parameters)
    {
        System.out.println( " " );

        String directoryName = "";
        String system        = "";


        // if both parameters were not specified, display help text and
exit.

        if (parameters.length >= 2)
        {

            // Assume the first parameter is the system name and
            // the second parameter is the directory name

            system = parameters[0];
            directoryName = parameters[1];

     try
            {
                // Create an AS400 object for the server that holds the
files.

                AS400 as400 = new AS400(system);


                // Create the IFSFile object for the directory.

                IFSFile directory = new IFSFile(as400, directoryName);
```

```
                // Generate a list of IFSFiles.  Pass the listFiles method
                // the directory filter object and the search match
                // criteria.  This method caches attribute information.  For
                // instance, when isDirectory() is called on an IFSFile
                // object in the file array returned in the following code,
                // no call to the server is required.
                //
                // However, with the user of the listFiles method, attribute
                // information will not be refreshed automatically from the
                // server. This means attribute information can become
                // inconsistent with information on the server.

                IFSFile[] directoryFiles = directory.listFiles(new
MyDirectoryFilter(),"*");

                // Tell the user if the directory doesn't exist or is empty

                if (directoryFiles == null)
                {
                    System.out.println("目錄不存在");
    return;
                }

                else if (directoryFiles.length == 0)
                {
                    System.out.println("目錄是空的");
    return;
                }

                for (int i=0; i< directoryFiles.length; i++)
                    {
                        // Print out information on list.
                        // Print the name of the current file

                        System.out.print(directoryFiles[i].getName());


                        // Pad the output so the columns line up

                        for (int j = directoryFiles[i].getName().length(); j
<18; j++)
                            System.out.print(" ");


                        // Print the date the file was last changed.

                        long changeDate = directoryFiles[i].lastModified();
                        Date d = new Date(changeDate);
                        System.out.print(d);
                        System.out.print("  ");


                        // Print if the entry is a file or directory

                        System.out.print("   ");

                        if (directoryFiles[i].isDirectory())
                            System.out.println("");
        else
```

```
                        System.out.println(directoryFiles[i].length());

                    }
                }

  catch (Exception e)
                {
                    // If any of the above operations failed say the list failed
                    // and output the exception.

                    System.out.println("列示失敗");
        System.out.println(e);
                }
        }

        // Display help text when parameters are incorrect.

        else
        {
            System.out.println("");
            System.out.println("");
            System.out.println("");
            System.out.println("參數不正確。  指令語法是：");
            System.out.println("");
            System.out.println("   IFSListFiles as400 目錄");
            System.out.println("");
            System.out.println("其中");
            System.out.println("");
            System.out.println("   as400 = 含有檔案的系統");
            System.out.println("   directory = 要列出的目錄");
            System.out.println("");
            System.out.println("例如：");
            System.out.println("");
            System.out.println("   IFSListFiles mySystem /dir1/dir2");
            System.out.println("");
            System.out.println("");
        }

       System.exit(0);
    }
}


////////////////////////////////////////////////////////////////////////
//
// The directory filter class prints information from the file object.
//
// Another way to use the filter is to simply return true or false
// based on information in the file object.  This lets the mainline
// function decide what to do with the list of files that meet the
// search criteria.
//
////////////////////////////////////////////////////////////////////////


class MyDirectoryFilter implements IFSFileFilter
{
```

```
        public boolean accept(IFSFile file)
    {
     try
        {
            // Keep this entry.  Returning true tells the IFSList object
            // to return this file in the list of entries returned to the
            // .list() method.

            return true;
        }

  catch (Exception e)
        {
            return false;
        }
    }
}
```

範例：使用 IFS 類別以將檔案從某一目錄複製到另一目錄

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// IFSCopyFile example.This program uses the installable file system classes
// to copy a file from one directory to another on the server.
//
// Command syntax:
//IFSCopyFile system sourceName TargetName
//
// For example,
//IFSCopyFile MySystem/path1/path2/file.ext/path3/path4/path5/file.ext
//
////////////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSCopyFile extends Object
{
public static void main(String[] parameters)
   {
System.out.println( " " );

String sourceName = "";
String targetName = "";
String system = "";
byte[] buffer = new byte[1024 * 64];

IFSFileInputStreamsource = null;
IFSFileOutputStream target = null;

// if all three parameters were not specified, display help text and exit.

if (parameters.length > 2)
      {

 // Assume the first parameter is the system name,
 // the second parameter is the source name and
 // the third parameter is the target name.

 system = parameters[0];
 sourceName = parameters[1];
 targetName = parameters[2];

 try
        {
// Create an AS400 object for the server that holds the files.

AS400 as400 = new AS400(system);
```

```
// Open the source file for exclusive access.

source = new IFSFileInputStream(as400,
sourceName,
IFSFileInputStream.SHARE_NONE);

System.out.println("原始檔順利開啟");



// Open the target file for exclusive access.

target = new IFSFileOutputStream(as400,
 targetName,
IFSFileOutputStream.SHARE_NONE,
false);

System.out.println("目標檔順利開啟");



// Read the first 64K bytes from the source file.

int bytesRead = source.read(buffer);



// While there is data in the source file copy the data from
// the source file to the target file.

while (bytesRead > 0)
            {
 target.write(buffer, 0, bytesRead);
 bytesRead = source.read(buffer);
            }

System.out.println("資料已順利複製");



// Clean up by closing the source and target files.

source.close();
target.close();



// Get the last changed date/time from the source file and
// set it on the target file.

IFSFile src = new IFSFile(as400, sourceName);
long dateTime = src.lastModified();

IFSFile tgt = new IFSFile(as400, targetName);
tgt.setLastModified(dateTime);

System.out.println("已順利設定目標檔上的日期/時間");
System.out.println("複製順利完成");

         }
```

```
catch (Exception e)
        {
// If any of the above operations failed say the copy failed
// and output the exception.

System.out.println("複製失敗");
System.out.println(e);
        }
      }

// Display help text when parameters are incorrect.

else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法為："
 System.out.println("");
 System.out.println(" IFSCopyFile as400 來源目標");
 System.out.println("");
 System.out.println("其中"
 System.out.println("");
 System.out.println(" as400= 含有檔案的系統");
 System.out.println(" source = /path/path/name 格式的原始檔");
 System.out.println(" target = /path/path/name 格式的目標檔");
 System.out.println("");
 System.out.println("例如："
 System.out.println("");
 System.out.println(" IFSCopyFile myAS400/dir1/dir2/a.txt/dir3/b.txt");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);
   }
}
```

範例：使用 IFS 範例列出目錄內容

註： 請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// IFSListFile example.  This program uses the integrated file system
classes
// to list the contents of a directory on the server.
//
// Command syntax:
//    IFSList system directory
//
// For example,
//    IFSList MySystem /path1
//
////////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class IFSList extends Object
{
    public static void main(String[] parameters)
   {
      System.out.println( " " );

      String directoryName = "";
      String system        = "";


      // if both parameters were not specified, display help text and exit.

      if (parameters.length >= 2)
      {

         // Assume the first parameter is the system name and
         // the second parameter is the directory name

         system = parameters[0];
         directoryName = parameters[1];

     try
         {
            // Create an AS400 object for the server that holds the files.

            AS400 as400 = new AS400(system);


            // Create the IFSFile object for the directory.

            IFSFile directory = new IFSFile(as400, directoryName);


            // Generate the list of name.  Pass the list method the
```

```
            // directory filter object and the search match criteria.
            //
            // Note - this example does the processing in the filter
            // object.  An alternative is to process the list after
            // it is returned from the list method call.

            String[] directoryNames = directory.list(new
MyDirectoryFilter(),"*");



            // Tell the user if the directory doesn't exist or is empty

            if (directoryNames == null)
               System.out.println("The directory does not exist");

            else if (directoryNames.length == 0)
               System.out.println("The directory is empty");
         }

  catch (Exception e)
         {
            // If any of the above operations failed say the list failed
            // and output the exception.

            System.out.println("List failed");
        System.out.println(e);
         }
      }

      // Display help text when parameters are incorrect.

        else
      {
         System.out.println("");
         System.out.println("");
         System.out.println("");
         System.out.println("Parameters are not correct.  Command syntax
is:");
         System.out.println("");
         System.out.println("   IFSList as400 directory");
         System.out.println("");
         System.out.println("Where");
         System.out.println("");
         System.out.println("   as400 = system that contains the files");
         System.out.println("   directory = directory to be listed");
         System.out.println("");
         System.out.println("For example:");
         System.out.println("");
         System.out.println("   IFSCopyFile mySystem /dir1/dir2");
         System.out.println("");
         System.out.println("");
      }

       System.exit(0);
   }
}
```

```
////////////////////////////////////////////////////////////////////////
//
// The directory filter class prints information from the file object.
//
// Another way to use the filter is to simply return true or false
// based on information in the file object.  This lets the mainline
// function decide what to do with the list of files that meet the
// search criteria.
//
////////////////////////////////////////////////////////////////////////


class MyDirectoryFilter implements IFSFileFilter
{
        public boolean accept(IFSFile file)
   {
     try
      {
         // Print the name of the current file

         System.out.print(file.getName());



         // Pad the output so the columns line up

         for (int i = file.getName().length(); i < 18; i++)
            System.out.print(" ");



         // Print the date the file was last changed.

         long changeDate = file.lastModified();
         Date d = new Date(changeDate);
         System.out.print(d);
         System.out.print("  ");



         // Print if the entry is a file or directory

         System.out.print("   ");

         if (file.isDirectory())
            System.out.println("<DIR>");
        else
            System.out.println(file.length());



         // Keep this entry.  Returning true tells the IFSList object
         // to return this file in the list of entries returned to the
         // .list() method.

         return true;
      }
```

```
  catch (Exception e)
      {
         return false;
      }
   }
}
```

範例：使用 JDBCPopulate 以建立及輸入表格資料

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// JDBCPopulate example.This program uses the IBM Toolbox for Java JDBC
driver to
// create and populate a table.
//
// Command syntax:
//JDBCPopulate system collectionName tableName
//
// For example,
//JDBCPopulate MySystem MyLibrary MyTable
//
////////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCPopulate
{



// Strings to be added in the WORD column of the table.
private static final String words[]
= { "一","二","三","四", "五",
"六","七","八","九", "十",
"十一", "十二", "十三", "十四", "十五",
"十六","十七","十八", "十九", "二十" };



public static void main(String[] parameters)
    {
// Check the input parameters.
if (parameters.length != 3) {
System.out.println("");
System.out.println("用法：");
System.out.println("");
System.out.println(" JDBCPopulate system collectionName tableName");
System.out.println("");
System.out.println("");
System.out.println("例如：");
System.out.println("");
System.out.println("");
System.out.println(" JDBCPopulate MySystem MyLibrary MyTable");
System.out.println("");
return;
        }

String system = parameters[0];
String collectionName = parameters[1];
String tableName= parameters[2];
```

```
Connection connection = null;

try {

// Load the IBM Toolbox for Java JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());

// Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
            //
// Note that we provide a default schema here so
// that we do not need to qualify the table name in
// SQL statements.
            //
connection = DriverManager.getConnection ("jdbc:as400://"
+ system + "/" + collectionName);

// Drop the table if it already exists.
try {
Statement dropTable = connection.createStatement ();
dropTable.executeUpdate ("DROP TABLE " + tableName);
            }
catch (SQLException e) {
// Ignore.
            }

// Create the table.
Statement createTable = connection.createStatement ();
createTable.executeUpdate ("CREATE TABLE " + tableName
+ " (I INTEGER, WORD VARCHAR(20), SQUARE INTEGER, "
+ " SQUAREROOT DOUBLE)");

// Prepare a statement for inserting rows.Since we
// execute this multiple times, it is best to use a
// PreparedStatement and parameter markers.
PreparedStatement insert = connection.prepareStatement ("INSERT INTO "
+ tableName + " (I, WORD, SQUARE, SQUAREROOT) "
+ " VALUES (?, ?, ?, ?)");

// Populate the table.
for (int i = 1; i <= words.length; ++i) {
insert.setInt (1, i);
insert.setString (2, words[i-1]);
insert.setInt (3, i*i);
insert.setDouble (4, Math.sqrt(i));
insert.executeUpdate ();
            }

// Output a completion message.
System.out.println ("表格 " + collectionName + "." + tableName
+ " 已輸入資料。");
        }

catch (Exception e) {
System.out.println ();
System.out.println ("錯誤：" + e.getMessage());
```

```
        }

finally {

// Clean up.
try {
if (connection != null)
connection.close ();
            }
catch (SQLException e) {
// Ignore.
            }
        }

 System.exit(0);
    }


}
```

範例：使用 JDBCQuery 以查詢表格

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// JDBCQuery example.This program uses the IBM Toolbox for Java JDBC driver
to
// query a table and output its contents.
//
// Command syntax:
//JDBCQuery system collectionName tableName
//
// For example,
//JDBCQuery MySystem qiws qcustcdt
//
///////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCQuery
{



// Format a string so that it has the specified width.
private static String format (String s, int width)
    {
String formattedString;

// The string is shorter than specified width,
// so we need to pad with blanks.
if (s.length() < width) {
StringBuffer buffer = new StringBuffer (s);
for (int i = s.length(); i < width; ++i)
buffer.append (" ");
formattedString = buffer.toString();
        }

// Otherwise, we need to truncate the string.
else
formattedString = s.substring (0, width);

return formattedString;
    }



public static void main(String[] parameters)
    {
// Check the input parameters.
if (parameters.length != 3) {
System.out.println("");
System.out.println("用法：");
System.out.println("");
System.out.println(" JDBCQuery system collectionName tableName");
System.out.println("");
```

```
System.out.println("");
System.out.println("例如：");
System.out.println("");
System.out.println("");
System.out.println(" JDBCQuery mySystem qiws qcustcdt");
System.out.println("");
return;
        }

String system = parameters[0];
String collectionName = parameters[1];
String tableName= parameters[2];

Connection connection = null;

try {

// Load the IBM Toolbox for Java JDBC driver.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());

// Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
connection = DriverManager.getConnection ("jdbc:as400://" + system);
DatabaseMetaData dmd = connection.getMetaData ();

// Execute the query.
Statement select = connection.createStatement ();
ResultSet rs = select.executeQuery ("SELECT * FROM "
+ collectionName + dmd.getCatalogSeparator() + tableName);

// Get information about the result set.Set the column
// width to whichever is longer: the length of the label
// or the length of the data.
ResultSetMetaData rsmd = rs.getMetaData ();
int columnCount = rsmd.getColumnCount ();
String[] columnLabels = new String[columnCount];
int[] columnWidths = new int[columnCount];
for (int i = 1; i <= columnCount; ++i) {
columnLabels[i-1] = rsmd.getColumnLabel (i);
columnWidths[i-1] = Math.max (columnLabels[i-1].length(),
rsmd.getColumnDisplaySize (i));
            }

// Output the column headings.
for (int i = 1; i <= columnCount; ++i) {
System.out.print (format (rsmd.getColumnLabel(i), columnWidths[i-1]));
System.out.print (" ");
            }
System.out.println ();

// Output a dashed line.
StringBuffer dashedLine;
for (int i = 1; i <= columnCount; ++i) {
for (int j = 1; j <= columnWidths[i-1]; ++j)
System.out.print ("-");
System.out.print (" ");
```

```
            }
System.out.println ();

// Iterate throught the rows in the result set and output
// the columns for each row.
while (rs.next ()) {
for (int i = 1; i <= columnCount; ++i) {
String value = rs.getString (i);
if (rs.wasNull ())
value = "<null>";
System.out.print (format (value, columnWidths[i-1]));
System.out.print (" ");
                }
System.out.println ();
            }

        }

catch (Exception e) {
System.out.println ();
System.out.println ("錯誤：" + e.getMessage());
        }

finally {

// Clean up.
try {
if (connection != null)
connection.close ();
            }
catch (SQLException e) {
// Ignore.
            }
        }

 System.exit(0);
    }



}
```

範例：使用 JobList 以列出工作識別資訊

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// This program is an example of the Job support in the IBM Toolbox
// for Java.It lists job identification information for a specific
// user on the system.
//
// Command syntax:
//listJobs2 system userID password
//
////////////////////////////////////////////////////////////////////////

import java.io.*;
import java.lang.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class listJobs2 extends Object
{

// Create an object in case we want to call
// any non-staic methods.
public static void main(String[] parameters)
   {
listJobs2 me = new listJobs2();
me.Main(parameters);

 System.exit(0);
   }



 void Main(String[] parameters)
   {

// If a system was not specified, display help text and exit.
if (parameters.length == 0)
      {
 showHelp();
return;
      }

// Assign the parameters to variables.The
// first parameter is assumed to be the system
// name the second is a userID and the third
// is a password.
String systemName = parameters[0];
String userID = null;
String password = null;

if (parameters.length > 1)
 userID = parameters[1].toUpperCase();
```

```
if (parameters.length >= 2)
 password = parameters[2].toUpperCase();

System.out.println(" ");



 try
      {

// Create an AS400 object using the system name
// specified by the user. Set the userid and
// password if specified by the user.
 AS400 as400 = new AS400(parameters[0]);

 if (userID != null)
as400.setUserId(userID);

 if (password != null)
as400.setPassword(password);



 System.out.println("正在擷取清單 ... ");


// Create a jobList object.This object is used
// to retrieve the list of active jobs on the server.
 JobList jobList = new JobList(as400);


 // Get the list of active jobs.
 Enumeration list = jobList.getJobs();


// For each job in the list ...
 while (list.hasMoreElements())
         {

// Get a job off the list.If a userID was
// specified then print identification information
// only if the job's user matches the userID.If
// no userID was specified then print information
// for every job on the system.
Job j = (Job) list.nextElement();

if (userID != null)
           {
if (j.getUser().trim().equalsIgnoreCase(userID))
              {
System.out.println(j.getName().trim() + "." +
 j.getUser().trim() + "." +
j.getNumber());
              }
           }
else
```

```
 System.out.println(j.getName().trim() + "." +
j.getUser().trim() + "." +
j.getNumber());
         }

      }
catch (Exception e)
      {
 System.out.println("異常錯誤");
e.printStackTrace();
      }
   }



// Display help text when parameters are incorrect.
 void showHelp()
   {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法是：");
 System.out.println("");
 System.out.println(" listJobs2 System UserID Password");
 System.out.println("");
 System.out.println("其中");
 System.out.println("");
 System.out.println(" System = 連線的伺服器");
 System.out.println(" UserID = 該系統上有效的使用者 ID");
 System.out.println(" Password = 使用者 ID 的密碼 (選用)");
 System.out.println("");
 System.out.println("例如：");
 System.out.println("");
 System.out.println(" listJobs2 MYAS400 JavaUser pwd1");
 System.out.println("");
 System.out.println("");
   }
}
```

範例：使用 JobList 以取得工作清單

註：請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// This program is an example of the "job" classes in the IBM
// Toolbox for Java.It gets a list of jobs on the server and
// outputs the job's status followed by job identifier.
//
//
// Command syntax:
//listJobs system userID password
//
// (UserID and password are optional)
//
////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class listJobs extends Object
{
public static void main(String[] parameters)
   {
listJobs me = new listJobs();
me.Main(parameters);

 System.exit(0);
   }



 void Main(String[] parameters)
   {

// If a system was not specified, display help text and exit.
if (parameters.length == 0)
      {
 showHelp();
return;
      }

// Set up AS400 object parms.The first is the system name and must
// be specified by the user.The second and third are optional.They
// are the userid and password.Convert the userid and password
// to uppercase before setting them on the AS400 object.
String userID = null;
String password = null;

if (parameters.length > 1)
 userID = parameters[1].toUpperCase();

if (parameters.length >= 2) password = parameters[2].toUpperCase();
```

```
 System.out.println(" ");

 try
      {

 // Create an AS400 object using the system name specified by the user.
 AS400 as400 = new AS400(parameters[0]);

 // If a userid and/or password was specified, set them on the
 // AS400 object.
 if (userID != null)
as400.setUserId(userID);

 if (password != null)
as400.setPassword(password);

 // Create a job list object.Input parm is the AS400 we want job
 // information from.
 JobList jobList = new JobList(as400);

 // Get a list of jobs running on the server.
 Enumeration listOfJobs = jobList.getJobs();

 // For each job in the list print information about the job.
 while (listOfJobs.hasMoreElements())
         {
printJobInfo((Job) listOfJobs.nextElement(), as400);
         }

      }
catch (Exception e)
      {
 System.out.println("異常錯誤");
System.out.println(e);
      }
   }



 void printJobInfo(Job job, AS400 as400)
   {

// Create the various converters we need
 AS400Bin4 bin4Converter = new AS400Bin4();
AS400Text text26Converter = new AS400Text(26, as400);
AS400Text text16Converter = new AS400Text(16, as400);
AS400Text text10Converter = new AS400Text(10, as400);
AS400Text text8Converter= new AS400Text(8,as400);
AS400Text text6Converter= new AS400Text(6,as400);
AS400Text text4Converter= new AS400Text(4,as400);


// We have the job name/number/etc. from the list request.Now
// make a server API call to get the status of the job.
 try
      {
```

```
 // Create a program call object
 ProgramCall pgm = new ProgramCall(as400);

 // The server program we call has five parameters
 ProgramParameter[] parmlist = new ProgramParameter[5];

 // The first parm is a byte array that holds the output
 // data.We will allocate a 1k buffer for output data.
 parmlist[0] = new ProgramParameter( 1024 );

 // The second parm is the size of our output data buffer (1K).
 Integer iStatusLength = new Integer( 1024 );
 byte[]statusLength = bin4Converter.toBytes( iStatusLength );
 parmlist[1] = new ProgramParameter( statusLength );

 // The third parm is the name of the format of the data.
 // We will use format JOBI0200 because it has job status.
 byte[] statusFormat = text8Converter.toBytes("JOBI0200");
 parmlist[2] = new ProgramParameter( statusFormat );

 // The fourth parm is the job name is format "name user number".
 // Name must be 10 characters, user must be 10 characters and
 // number must be 6 characters.We will use a text converter
 // to do the conversion and padding.
 byte[] jobName = text26Converter.toBytes(job.getName());

 inti = text10Converter.toBytes(job.getUser(),
jobName,
10);

i = text6Converter.toBytes(job.getNumber(),
 jobName,
 20);

 parmlist[3] = new ProgramParameter( jobName );

 // The last paramter is job identifier.We will leave this blank.
 byte[] jobID = text16Converter.toBytes("");
 parmlist[4] = new ProgramParameter( jobID );


 // Run the program.
 if (pgm.run( "/QSYS.LIB/QUSRJOBI.PGM", parmlist )==false)
         {
// if the program failed display the error message.
AS400Message[] msgList = pgm.getMessageList();
System.out.println(msgList[0].getText());
         }
else
         {
// else the program worked.Output the status followed by
// the jobName.user.jobID
byte[] as400Data = parmlist[0].getOutputData();
System.out.print("" + text4Converter.toObject(as400Data, 107) + "");

System.out.println(job.getName().trim() + "." +
```

```
 job.getUser().trim() + "." +
 job.getNumber() + " ");
         }

      }
catch (Exception e)
      {
System.out.println(e);
      }

   }



// Display help text when parameters are incorrect.
 void showHelp()
   {
System.out.println("");
System.out.println("");
System.out.println("");
 System.out.println("參數不正確。指令語法是:");
System.out.println("");
 System.out.println(" listJobs System UserID Password");
System.out.println("");
 System.out.println("其中");
System.out.println("");
 System.out.println(" System = 連線的伺服器");
 System.out.println(" UserID = 該系統上的有效使用者 ID (選用)");
 System.out.println(" Password = 使用者 ID 的密碼 (選用)");
System.out.println("");
 System.out.println("例如：");
System.out.println("");
 System.out.println(" listJobs MYAS400 JavaUser pwd1");
System.out.println("");
System.out.println("");
   }
}
```

範例：使用 JobLog 以顯示工作日誌中的訊息

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// This program is an example of the job log fuction of the IBM
// Toolbox for Java.It will display the messages in the job log
// for a job that belongs to the current user.
//
// Command syntax:
//jobLogExample system userID password
//
// (Password is optional)
//
////////////////////////////////////////////////////////////////////////

import java.lang.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class jobLogExample
{


 public static void main(String[] args)
    {
// If a system and user were not specified, display help text and exit.
if (args.length < 2)
        {
System.out.println("用法： jobLogExample system userid <password>");
return;
        }

String userID = null;

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.If a userid
// and password were passed on the command line,
// set those as well.
AS400 system = new AS400 (args[0]);

if (args.length > 1)            {
 userID = args[1];
 system.setUserId(userID);
            }

if (args.length > 2)
 system.setPassword(args[2]);


// Create a job list object.This object will be used to get
// the list of active jobs on the system.Once the list of
```

```
// jobs is retrieved, the program will find a job for the
// current user.
JobList jobList = new JobList(system);

 // Get the list of active jobs on
 Enumeration list = jobList.getJobs();

boolean Continue = true;

// Look through the list to find a job for the current user.
while (list.hasMoreElements() && Continue)
            {
Job j = (Job) list.nextElement();

if (j.getUser().trim().equalsIgnoreCase(userID))
                {
// A job matching the current user was found.Create
// a job log object for this job.
 JobLog jlog = new JobLog(system,
j.getName(),
j.getUser(),
j.getNumber());

// Enumerate the messages in the job log then print them.
 Enumeration messageList = jlog.getMessages();

 while (messageList.hasMoreElements())
                    {
 AS400Message message = (AS400Message) messageList.nextElement();
 System.out.println(message.getText());
                    }

// We found one job matching the current user so exit.
Continue = false;
                }
            }
        }
catch (Exception e)
        {
 System.out.println ("錯誤：" + e.getMessage ());
        }

 System.exit(0);
    }

}
```

範例：建立排存檔

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Example that shows creating a spooled file on a server from an input
stream.
//
///////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;

import com.ibm.as400.access.*;

class NPExampleCreateSplf
{

// method to create the spooled file on the specified system, in the
specified
// output queue from the given input stream.
public SpooledFile createSpooledFile(AS400 system,
 OutputQueue outputQueue,
 InputStream in)
{
SpooledFile spooledFile = null;
 try
    {
byte[] buf = new byte[2048];
int bytesRead;
SpooledFileOutputStream out;
PrintParameterList parms = new PrintParameterList();

// create a PrintParameterList with the values that we want
// to override from the default printer file...we will override
// the output queue and the copies value.
parms.setParameter(PrintObject.ATTR_COPIES, 4);
if (outputQueue != null)
        {
 parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, outputQueue.getPath());
        }
out = new SpooledFileOutputStream(system,
parms,
null,
null);

// read from the inputstream in until end of stream, passing all data
// to the spooled file output stream.
do
        {
bytesRead = in.read(buf);
if (bytesRead != -1)
            {
```

```
out.write(buf);
            }
} while (bytesRead != -1);

out.close();// close the spooled file

spooledFile = out.getSpooledFile(); // get a reference to the new spooled
file

    }
catch (Exception e)
    {
//...handle exception...
    }
return spooledFile;
}

}
```

範例：建立 SCS 排存檔

此範例使用 SCS3812Writer 類別來產生 SCS 資料串流，並將它寫入伺服器上的排存檔。

此應用程式可以採用下列引數，或可以使用已定義的預設值：

- 接收排存檔的伺服器名稱。
- 伺服器上接收排存檔的輸出佇列名稱。

註：請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

註：請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "SCS3812Writer".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

class NPExampleCreateSCSSplf
{
private static final String DEFAULT_SYSTEM = new String("RCHAS1");
private static final String DEFAULT_OUTQ = new
String("/QSYS.LIB/QUSRSYS.LIB/PRT01.OUTQ");

 public static void main(String[] args)
    {
 try
        {
AS400 system;
SpooledFileOutputStream out;
PrintParameterList parms = new PrintParameterList();
SCS3812Writer scsWtr;

// Process the arguments.
if (args.length >= 1)
            {
system = new AS400(args[0]);// Create an AS400 object
} else {
system = new AS400(DEFAULT_SYSTEM);
            }

if (args.length >= 2) // Set the outq
            {
parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, args[1]);
} else {
parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, DEFAULT_OUTQ);
            }

out = new SpooledFileOutputStream(system, parms, null, null);
```

```
scsWtr = new SCS3812Writer(out, 37);

// Write the contents of the spool file.
scsWtr.setLeftMargin(1.0);
scsWtr.absoluteVerticalPosition(6);
scsWtr.setFont(scsWtr.FONT_COURIER_BOLD_5);
scsWtr.write("Java 列印中");
scsWtr.newLine();
scsWtr.newLine();
scsWtr.setCPI(10);
scsWtr.write("此文件是使用 IBM Toolbox for Java 建立的。");
scsWtr.newLine();
scsWtr.write("此文件的其餘部份顯示可以使用");
scsWtr.newLine();
scsWtr.write(" SCS3812Writer 類別執行的部份作業。");
scsWtr.newLine();
scsWtr.newLine();
scsWtr.setUnderline(true); scsWtr.write("設定字型：");
scsWtr.setUnderline(false);
scsWtr.newLine();
scsWtr.setFont(scsWtr.FONT_COURIER_10); scsWtr.write("Courier 字型");
scsWtr.setFont(scsWtr.FONT_COURIER_BOLD_10); scsWtr.write(" Courier
粗體字型");
scsWtr.setFont(scsWtr.FONT_COURIER_ITALIC_10); scsWtr.write(" Courier
斜體字型");
scsWtr.newLine();
scsWtr.setBold(true); scsWtr.write("Courier 粗斜體字型");
scsWtr.setBold(false);
scsWtr.setCPI(10);
scsWtr.newLine();
scsWtr.newLine();
scsWtr.setUnderline(true); scsWtr.write("每吋行數：");
scsWtr.setUnderline(false);
scsWtr.newLine();
scsWtr.write("下列各行應每英吋列印 8 行。");
scsWtr.newLine();
scsWtr.newLine();
scsWtr.setLPI(8);
scsWtr.write("行 1"); scsWtr.newLine();
scsWtr.write("行 2"); scsWtr.newLine();
scsWtr.write("行 3"); scsWtr.newLine();
scsWtr.write("行 4"); scsWtr.newLine();
scsWtr.write("行 5"); scsWtr.newLine();
scsWtr.write("行 6"); scsWtr.newLine();
scsWtr.write("行 7"); scsWtr.newLine();
scsWtr.write("行 8"); scsWtr.newLine();
scsWtr.endPage();
scsWtr.setLPI(6);
scsWtr.setSourceDrawer(1);
scsWtr.setTextOrientation(0);
scsWtr.absoluteVerticalPosition(6);
scsWtr.write("本頁應從紙匣 1 以直向列印。");
scsWtr.endPage();
scsWtr.setSourceDrawer(2);
```

```
scsWtr.setTextOrientation(90);
scsWtr.absoluteVerticalPosition(6);
scsWtr.write("本頁應從紙匣 2 橫向列印。");
scsWtr.endPage();
scsWtr.close();
System.out.println("已建立範例排存檔。");
 System.exit(0);
        }
catch (Exception e)
        {
// Handle error.
System.out.println("建立排存檔時發生異常。" + e);
 System.exit(0);
        }
    }
}
```

範例：讀取排存檔

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Example that reads an existing server spooled file.
//
// This source is an example of IBM Toolbox for Java
"PrintObjectInputStream".
//
////////////////////////////////////////////////////////////////////////
try{
byte[] buf = new byte[2048];
int bytesRead;
AS400 sys = new AS400();
SpooledFile splf = new SpooledFile( sys,// AS400
"MICR", // splf name
17, // splf number
"QPRTJOB",// job name
"QUSER",// job user
"020791" ); // job number

// open the spooled file for reading and get the input stream to
// read from it.
InputStream in = splf.getInputStream(null);

do
        {
// read up to buf.length bytes of raw spool data into
// our buffer.The actual bytes read will be returned.
// The data will be a binary printer data stream that is the
// contents of the spooled file.
bytesRead = in.read( buf );
if( bytesRead != -1 )
            {
// process the spooled file data.
 System.out.println( "讀取 " + bytesRead + " 位元組" );
            }
} while( bytesRead != -1 );

in.close();
    }
catch (Exception e)
    {
// exception
    }
```

範例：非同步列出排存檔（使用接收程式）

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a system Asynchronously
using
// the PrintObjectListListener interface to get feedback as the list is
being built.
// Listing Asynchronously allows the caller to start processing the list
objects
// before the entire list is built for a faster perceived response time
// for the user.
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;
import com.ibm.as400.access.ExtendedIllegalStateException;
import com.ibm.as400.access.PrintObjectListListener;
import com.ibm.as400.access.PrintObjectListEvent;

public class NPExampleListSplfAsynch extends Object
 implements PrintObjectListListener
{
 private AS400system_;
private boolean fListError;
private boolean fListClosed;
private boolean fListCompleted;
private Exception listException;
private int listObjectCount;

public NPExampleListSplfAsynch(AS400 system)
    {
system_ = system;
    }

// list all spooled files on the system asynchronously using a listener
public void listSpooledFiles()
    {
fListError = false;
fListClosed = false;
fListCompleted = false;
listException = null;
listObjectCount = 0;

 try
        {
String strSpooledFileName;
boolean fCompleted = false;
int listed = 0, size;
```

```
if (system_ == null)
            {
system_ = new AS400();
            }

System.out.println(" 現在正使用接收程式接收所有排存檔");

SpooledFileList splfList = new SpooledFileList(system_);

// set filters, all users, on all queues
splfList.setUserFilter("*ALL");
splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

// add the listener.
splfList.addPrintObjectListListener(this);

// open the list, openAsynchronously returns immediately
splfList.openAsynchronously();

do
            {
// wait for the list to have at least 25 objects or to be done
waitForWakeUp();

fCompleted = splfList.isCompleted();
size = splfList.size();

// output the names of all objects added to the list
// since we last woke up
while (listed < size)
                {
if (fListError)
                    {
System.out.println(" 清單發生異常 - " + listException);
break;
                    }

if (fListClosed)
                    {
System.out.println(" 清單在完成前結束！");
break;
                    }

SpooledFile splf = (SpooledFile)splfList.getObject(listed++);
if (splf != null)
                    {
// output this spooled file name
strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
System.out.println(" 排存檔 = " + strSpooledFileName);
                    }
                }

} while (!fCompleted);

// clean up after we are done with the list
splfList.close();
```

```
splfList.removePrintObjectListListener(this);
        }

catch( ExtendedIllegalStateException e )
        {
System.out.println(" 清單在完成前結束！");
        }

catch (Exception e)
        {
// ...handle any other exceptions...
e.printStackTrace();
        }

    }

// This is where the foreground thread waits to be awaken by the
// the background thread when the list is updated or it ends.
private synchronized void waitForWakeUp()
 throws InterruptedException
    {
// don''t go back to sleep if the listener says the list is done
if (!fListCompleted)
        {
wait();
        }
    }

// The following methods implement the PrintObjectListListener interface

// This method is invoked when the list is closed.
public void listClosed(PrintObjectListEvent event)
    {
System.out.println("*****清單已結束*****");
fListClosed = true;
synchronized(this)
        {
// Set flag to indicate that the list has
// completed and wake up foreground thread.
fListCompleted = true;
notifyAll();
        }
    }

// This method is invoked when the list is completed.
public void listCompleted(PrintObjectListEvent event)
    {
System.out.println("*****清單已完成*****");
synchronized (this)
        {
// Set flag to indicate that the list has
// completed and wake up foreground thread.
fListCompleted = true;
notifyAll();
        }
```

```
    }

// This method is invoked if an error occurs while retrieving
// the list.
public void listErrorOccurred(PrintObjectListEvent event)
    {
System.out.println("*****清單發生錯誤*****");
fListError = true;
listException = event.getException();
synchronized(this)
        {
// Set flag to indicate that the list has
// completed and wake up foreground thread.
fListCompleted = true;
notifyAll();
        }
    }

// This method is invoked when the list is opened.
public void listOpened(PrintObjectListEvent event)
    {
System.out.println("*****清單已開啟*****");
listObjectCount = 0;
    }

// This method is invoked when an object is added to the list.
public void listObjectAdded(PrintObjectListEvent event)
    {
// every 25 objects we'll wake up the foreground
// thread to get the latest objects...
if( (++listObjectCount % 25) == 0 )
        {
System.out.println("*****已新增 25 個以上的物件到清單*****");
synchronized (this)
            {
// wake up foreground thread
notifyAll();
            }
        }
    }

public static void main( String args[] )
    {
NPExampleListSplfAsynch list = new NPExampleListSplfAsynch(new AS400());
try{
list.listSpooledFiles();
        }
catch (Exception e)
        {
e.printStackTrace();
        }
 System.exit(0);
    }

}
```

範例：非同步列出排存檔（不使用接收程式）

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a system Asynchronously
without
// using the PrintObjectListListener interface. After opening the list the
caller
// can do some additional work before waiting for the list to complete.
//
///////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "PrintObjectList".
//
///////////////////////////////////////////////////////////////////////

import java.util.Enumeration;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;

public class NPExampleListSplfAsynch2 extends Object
{
 private AS400system_;

public NPExampleListSplfAsynch2(AS400 system)
    {
system_ = system;
    }

// list all spooled files on the system asynchronously
public void listSpooledFiles()
    {
 try
        {
String strSpooledFileName;
int listed, size;

if (system_ == null)
            {
system_ = new AS400();
            }

System.out.println(" 現在正在非同步接收所有排存檔, 而不使用接收程式");

SpooledFileList splfList = new SpooledFileList(system_);

// set filters, all users, on all queues
splfList.setUserFilter("*ALL");
splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

// open list, openAsynchronously() returns immediately
```

```
// we have not added any listeners...
splfList.openAsynchronously();

System.out.println(" 等待前, 請先執行一些處理...");

// ... do some processing here while the list is being built....

System.out.println(" 現在, 請等待清單完成。");

// wait for the list to complete
splfList.waitForListToComplete();

Enumeration enum = splfList.getObjects();

// output the name of all objects on the list
 while
(enum.hasMoreElements ())
            {
SpooledFile splf = (SpooledFile)enum.nextElement();
if (splf != null)
                {
// output this spooled file's name
strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
System.out.println(" 排存檔 = " + strSpooledFileName);
                }
            }
// clean up after we are done with the list
splfList.close();
        }

catch (Exception e)
        {
// ...handle any exceptions...
e.printStackTrace();
        }
    }

public static void main( String args[] )
    {
NPExampleListSplfAsynch2 list = new NPExampleListSplfAsynch2(new AS400());
try{
list.listSpooledFiles();
        }
catch (Exception e)
        {
e.printStackTrace();
        }
 System.exit(0);
    }
}
```

範例：同步列出排存檔

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Example that shows listing all spooled files on a system Synchronously.
// Listing Synchronously does not return to the caller until the complete
list
// is built. The user perceives a slower response time then listing
Asynchronously.
//
///////////////////////////////////////////////////////////////////////
//
// This source is an example of IBM Toolbox for Java "PrintObjectList".
//
///////////////////////////////////////////////////////////////////////

import java.util.Enumeration;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.SpooledFileList;
import com.ibm.as400.access.SpooledFile;

public class NPExampleListSplfSynch
{
private AS400 system_ = new AS400();

public NPExampleListSplfSynch(AS400 system)
    {
system_ = system;
    }

public void listSpooledFiles()
    {
try{
String strSpooledFileName;

if (system_ == null)
            {
system_ = new AS400();
            }

System.out.println(" 現在正同步接收所有排存檔");

SpooledFileList splfList = new SpooledFileList( system_ );

// set filters, all users, on all queues
splfList.setUserFilter("*ALL");
splfList.setQueueFilter("/QSYS.LIB/%ALL%.LIB/%ALL%.OUTQ");

// open list, openSynchronously() returns when the list is completed.
splfList.openSynchronously();
Enumeration enum = splfList.getObjects();
```

```
 while
(enum.hasMoreElements ())
            {
SpooledFile splf = (SpooledFile)enum.nextElement();
if ( splf != null )
                {
// output this spooled file's name
strSpooledFileName = splf.getStringAttribute(SpooledFile.ATTR_SPOOLFILE);
System.out.println(" 排存檔 = " + strSpooledFileName);
                }
            }
// clean up after we are done with the list
splfList.close();
        }
catch (Exception e)
        {
// ...handle any exceptions...
e.printStackTrace();
        }
    }

public static void main( String args[] )
    {
NPExampleListSplfSynch list = new NPExampleListSplfSynch(new AS400());
try{
list.listSpooledFiles();
        }
catch (Exception e)
        {
e.printStackTrace();
        }
 System.exit(0);
    }
}
```

範例：使用 ProgramCall

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Program call example.This program calls the QWCRSSTS server program
// to retrieve the status of the system.
//
// Command syntax:
//PCSystemStatusExample system
//
// This source is an example of IBM Toolbox for Java "ProgramCall".
//
///////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import java.math.*;
import java.lang.Thread.*;
import com.ibm.as400.access.*;

public class PCSystemStatusExample extends Object
{
public static void main(String[] parameters)
   {
System.out.println( " " );


// if a system was not specified, display help text and exit.

if (parameters.length >= 1)
      {

 try
        {
// Create an AS400 object for the server that contains the
// program.Assume the first parameter is the system name.

AS400 as400 = new AS400(parameters[0]);



// Create the path to the program.

QSYSObjectPathName programName = new QSYSObjectPathName("QSYS",
"QWCRSSTS",
"PGM");



// Create the program call object.Assocate the object with the
```

```
// AS400 object that represents the server we get status from.

ProgramCall getSystemStatus = new ProgramCall(as400);



// Create the program parameter list.This program has five
// parameters that will be added to this list.

ProgramParameter[] parmlist = new ProgramParameter[5];



// The server program returns data in parameter 1.It is an output
// parameter.Allocate 64 bytes for this parameter.

parmlist[0] = new ProgramParameter( 64 );



// Parameter 2 is the buffer size of parm 1.It is a numeric input
// parameter.Sets its value to 64, convert it to the server format,
// then add the parm to the parm list.

AS400Bin4 bin4 = new AS400Bin4( );
Integer iStatusLength = new Integer( 64 );
byte[]statusLength = bin4.toBytes( iStatusLength );
parmlist[1] = new ProgramParameter( statusLength );



// Parameter 3 is the status-format parameter.It is a string input
// parameter.Set the string value, convert it to the server format,
// then add the parameter to the parm list.

AS400Text text1 = new AS400Text(8, as400);
byte[] statusFormat = text1.toBytes("SSTS0200");
parmlist[2] = new ProgramParameter( statusFormat );



// Parameter 4 is the reset-statistics parameter.It is a string input
// parameter.Set the string value, convert it to the server format,
// then add the parameter to the parm list.

AS400Text text3 = new AS400Text(10, as400);
byte[] resetStats = text3.toBytes("*NO ");
parmlist[3] = new ProgramParameter( resetStats );



// Parameter 5 is the error info parameter.It is an input/output
// parameter.Add it to the parm list.

byte[] errorInfo = new byte[32];
parmlist[4] = new ProgramParameter( errorInfo, 0 );
```

```
// Set the program to call and the parameter list to the program
// call object.

getSystemStatus.setProgram(programName.getPath(), parmlist );



// Run the program then sleep.We run the program twice because
// the first set of results are inflated.If we discard the first
// set of results and run the command again five seconds later the
// number will be more accurate.

getSystemStatus.run();
Thread.sleep(5000);



// Run the program

if (getSystemStatus.run()!=true)
            {

 // If the program did not run get the list of error messages
 // from the program object and display the messages.The error
 // would be something like program-not-found or not-authorized
 // to the program.

 AS400Message[] msgList = getSystemStatus.getMessageList();

 System.out.println("程式未執行。伺服器訊息：");

 for (int i=0; i<msgList.length; i++)
                {
 System.out.println(msgList[i].getText());
                }
            }


// Else the program did run.

else
            {

 // Create a server to Java numeric converter.This converter
 // will be used in the following section to convert the numeric
 // output from the server format to Java format.

 AS400Bin4 as400Int = new AS400Bin4( );



 // Get the results of the program.Output data is in
 // a byte array in the first parameter.

 byte[] as400Data = parmlist[0].getOutputData();
```

```
 // CPU utilization is a numeric field starting at byte
 // 32 of the output buffer.Convert this number from the
 // server format to Java format and output the number.

 Integer cpuUtil = (Integer)as400Int.toObject( as400Data, 32 );
 cpuUtil = new Integer(cpuUtil.intValue()/10);
 System.out.print("CPU 使用率： ");
 System.out.print(cpuUtil);
 System.out.println("%");



 // DASD utilization is a numeric field starting at byte
 // 52 of the output buffer.Convert this number from the
 // server format to Java format and output the number.

 Integer dasdUtil = (Integer)as400Int.toObject( as400Data, 52 );
 dasdUtil = new Integer(dasdUtil.intValue()/10000);
 System.out.print("Dasd 使用率： ");
 System.out.print(dasdUtil);
 System.out.println("%");



 // Number of jobs is a numeric field starting at byte
 // 36 of the output buffer.Convert this number from the
 // server format to Java format and output the number.

 Integer nj = (Integer)as400Int.toObject( as400Data, 36 );
 System.out.print("作用中工作：");
 System.out.println(nj);

           }

 // This program is done running program so disconnect from
 // the command server on the server.Program call and command
 // call use the same server on the server.

as400.disconnectService(AS400.COMMAND);
         }
catch (Exception e)
         {
// If any of the above operations failed say the program failed
// and output the exception.

System.out.println("程式呼叫失敗");
System.out.println(e);
         }
      }

// Display help text when parameters are incorrect.

else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
```

```
 System.out.println("參數不正確。指令語法為：");
 System.out.println("");
 System.out.println(" PCSystemStatusExample myServer");
 System.out.println("");
 System.out.println("其中");
 System.out.println("");
 System.out.println(" myServer = 取得此伺服器的狀態 ");
 System.out.println("");
 System.out.println("例如：");
 System.out.println("");
 System.out.println(" PCSystemStatusExample mySystem");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);
   }
}
```

範例：使用記錄層次存取類別

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// Record level access example.This program will prompt the user
// for the name of the server and the file to display.The file must exist
// and should contain records.Each record in the file will be displayed
// to System.out.
//
// Calling syntax: java RLSequentialAccessExample
//
// This source is an example of IBM Toolbox for Java "RecordLevelAccess"
//
///////////////////////////////////////////////////////////////////////////////



import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLSequentialAccessExample
{
 public static void main(String[] parmeters)
   {
// Created a reader to get input from the user
BufferedReader inputStream = new BufferedReader(new
InputStreamReader(System.in),1);

// Declare variables to hold the system name, library, file and member names
String systemName = "";
String library = "";
String file = "";
String member = "";

// Get the system name and and file to display from the user
System.out.println();
 try
     {
System.out.print("系統名稱：");
systemName = inputStream.readLine();

System.out.print("檔案所在的檔案庫： ");
library = inputStream.readLine();

System.out.print("檔名： ");
file = inputStream.readLine();

System.out.print("成員名稱 (按 Enter 鍵取得第一個成員): ");
member = inputStream.readLine();
if (member.equals(""))
       {
member = "*FIRST";
        }
```

```
System.out.println();
      }
catch (Exception e)
      {
System.out.println("取得使用者輸入時發生錯誤。");
e.printStackTrace();
 System.exit(0);
      }

// Create AS400 object and connect for the record level access service.
AS400 system = new AS400(systemName);
 try
      {
system.connectService(AS400.RECORDACCESS);
      }
catch (Exception e)
      {
System.out.println("無法連線供記錄層次存取。");
System.out.println("查看 ReadMe 檔中關於記錄層次存取的特殊指示");
e.printStackTrace();
 System.exit(0);
      }

// Create a QSYSObjectPathName object to obtain the integrated file system
path name form
// of the file to be displayed.
QSYSObjectPathName filePathName = new QSYSObjectPathName(library, file,
member, "MBR");

// Create a SequentialFile object representing the file to be displayed
SequentialFile theFile = new SequentialFile(system, filePathName.getPath());

 // Retrieve the record format for the file
AS400FileRecordDescription recordDescription = new
AS400FileRecordDescription(system, filePathName.getPath());
 try
      {
RecordFormat[] format = recordDescription.retrieveRecordFormat();

 // Set the record format for the file
theFile.setRecordFormat(format[0]);

// Open the file for reading.Read 100 records at a time if possible.
theFile.open(AS400File.READ_ONLY, 100, AS400File.COMMIT_LOCK_LEVEL_NONE);

// Display each record in the file
System.out.println("Displaying file " + library.toUpperCase() + "/" +
file.toUpperCase() + "(" + theFile.getMemberName().trim() + "):");

Record record = theFile.readNext();
 while(record != null)
       {
System.out.println(record);
record = theFile.readNext();
       }
System.out.println();
```

```
 // 關閉檔案
theFile.close();

// Disconnect from the record level access service
system.disconnectService(AS400.RECORDACCESS);
      }
catch (Exception e)
      {
System.out.println("嘗試顯示檔案時發生錯誤。");
e.printStackTrace();

 try
        {
 // 關閉檔案
theFile.close();
        }
catch(Exception x)
        {
        }

// Disconnect from the record level access service
system.disconnectService(AS400.RECORDACCESS);
 System.exit(0);
      }

// Make sure that the application ends; see readme for details
 System.exit(0);
    }
  }
```

範例：使用記錄層次存取類別來類別檔案中的記錄

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Record-Level Access example.This program uses the record-level
// access classes to read records from a file on the server.
//
// Command syntax:
//java RLReadFile system
//
// This program reads the records from CA/400's sample database file
// (QCUSTCDT in library QIWS).If you change this example to update
// records you should make a copy of QCUSTCDT and update the copy.
//
// This source is an example of IBM Toolbox for Java "Record-level access".
//
///////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.math.*;
import com.ibm.as400.access.*;

public class RLReadFile extends Object
{
public static void main(String[] parameters)
   {

String system = "";

// Continue only if a system name was specified.

if (parameters.length >= 1)
      {

 try
        {

// Assume the first parameter is the system name.

system = parameters[0];

// Create an AS400 object for the server that has the file.

AS400 as400 = new AS400(system);


// Create a record description for the file.The file is QCUSTCDT
// in library QIWS.

ZonedDecimalFieldDescription customerNumber =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,0),
```

```
 "CUSNUM");
CharacterFieldDescription lastName =
new CharacterFieldDescription(new AS400Text(8, as400), "LSTNAM");

CharacterFieldDescription initials =
new CharacterFieldDescription(new AS400Text(3, as400), "INIT");

CharacterFieldDescription street =
new CharacterFieldDescription(new AS400Text(13, as400), "STREET");

CharacterFieldDescription city =
new CharacterFieldDescription(new AS400Text(6, as400), "CITY");

CharacterFieldDescription state =
new CharacterFieldDescription(new AS400Text(2, as400), "STATE");

ZonedDecimalFieldDescription zipCode =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(5,0),
 "ZIPCOD");
ZonedDecimalFieldDescription creditLimit =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(4,0),
 "CDTLMT");
ZonedDecimalFieldDescription chargeCode =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(1,0),
 "CHGCOD");
ZonedDecimalFieldDescription balanceDue =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,2),
 "BALDUE");
ZonedDecimalFieldDescription creditDue =
new ZonedDecimalFieldDescription(new AS400ZonedDecimal(6,2),
 "CDTDUE");


// The record format name should be specified for a DDM file.
// In the case of the QCUSTCDT file, its record format is called CUSREC.

RecordFormat qcustcdt = new RecordFormat("CUSREC");

qcustcdt.addFieldDescription(customerNumber);
qcustcdt.addFieldDescription(lastName);
qcustcdt.addFieldDescription(initials);
qcustcdt.addFieldDescription(street);
qcustcdt.addFieldDescription(city);
qcustcdt.addFieldDescription(state);
qcustcdt.addFieldDescription(zipCode);
qcustcdt.addFieldDescription(creditLimit);
qcustcdt.addFieldDescription(chargeCode);
qcustcdt.addFieldDescription(balanceDue);
qcustcdt.addFieldDescription(creditDue);


// Create the sequential file object that represents the
// file on the server.We use a QSYSObjectPathName object
// to get the name of the file into the correct format.

QSYSObjectPathName fileName = new QSYSObjectPathName("QIWS",
```

```
 "QCUSTCDT",
 "FILE");

SequentialFile file = new SequentialFile(as400, fileName.getPath());


// Let the file object know the format of the records.

file.setRecordFormat(qcustcdt);


// Open the file for read-only access.Specify a blocking
// factor of 10 (the file object will get 10 records when
// it accesses the server for data).Do not use commitment
// control.

file.open(SequentialFile.READ_ONLY,
10,
SequentialFile.COMMIT_LOCK_LEVEL_NONE);


// Read the first record of the file.

Record data = file.readNext();


// Loop while there are records in the file (while we have not
// reached end-of-file).

 while (data != null)
           {

 // Display the record only if balance due is greater than
 // zero.In that case display the customer name and
 // the balance due.The following code pulls fields out
 // of the record by field name.As the field is retrieved
 // from the record it is converted from server format to
 // Java format.

 if (((BigDecimal)data.getField("BALDUE")).floatValue() > 0.0)
               {
System.out.print((String) data.getField("INIT") + "");
System.out.print((String) data.getField("LSTNAM") + "");
System.out.println((BigDecimal) data.getField("BALDUE"));
               }

 // Read the next record in the file.

 data = file.readNext();
           }


// When there are no more records to read, disconnect from the server.

as400.disconnectAllServices();
         }

catch (Exception e)
```

```
        {

// If any of the above operations failed, print an error message
// and output the exception.

System.out.println("無法讀取檔案");
System.out.println(e);
        }
      }


// Display help text when parameters are incorrect.

else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法為：");
 System.out.println("");
 System.out.println(" RLReadFile as400");
 System.out.println("");
 System.out.println("其中");
 System.out.println("");
 System.out.println(" as400 = 包含該檔案的系統");
 System.out.println("");
 System.out.println("例如：");
 System.out.println("");
 System.out.println(" RLReadFile mySystem");
 System.out.println("");
 System.out.println("");
 System.out.println("注意, 此程式讀取資料基本檔 QIWS/QCUSTCDT。");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);

   }
}
```

範例：使用記錄層次存取類別按金鑰讀取記錄

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Record-Level Access example.This program uses the record-level
// access classes to read records by key from a file on the server.
// The user will be prompted for the server name to which to run and
// the library in which to create file QCUSTCDTKY.
//
// Command syntax:
//java RLKeyedFileExample
//
// This program will copy the records from the iSeries Access for Windows
sample
// database file (QCUSTCDT in library QIWS) to file QCUSTCDTKY which has
// the same format as QIWS/QCUSTCDT but has set the CUSNUM field as the key
// for the file.
//
// This source is an example of IBM Toolbox for Java "Record-level access".
//
///////////////////////////////////////////////////////////////////////////


import java.io.*;
 import java.util.*;
import java.math.*;
import com.ibm.as400.access.*;

public class RLKeyedFileExample
{
public static void main(String[] parameters)
  {

// Created a reader to get input from the user
BufferedReader inputStream = new BufferedReader(new
InputStreamReader(System.in),1);

// Declare variables to hold the system name, library, file and member names
String systemName = "";
String library = "";

// Get the system name from the user
System.out.println();
 try
    {
System.out.print("System name: ");
systemName = inputStream.readLine();

System.out.print("建立檔案 QCUSTCDTKY 的程式庫：");
library = inputStream.readLine();
    }
catch (Exception e)
```

```
    {
System.out.println("取得使用者輸入時發生錯誤。");
e.printStackTrace();
 System.exit(0);
    }

// Create AS400 object and connect for the record level access service.
AS400 system = new AS400(systemName);
 try
    {
system.connectService(AS400.RECORDACCESS);
    }
catch (Exception e)
    {
System.out.println("無法連線供執行記錄層次存取。");
System.out.println("查看 ReadMe 檔中關於記錄層次存取的特殊指示");
e.printStackTrace();
 System.exit(0);
    }

RecordFormat qcustcdtFormat = null;
 try
    {
// Create the RecordFormat object for creating the file.The record format
for the new
// file will be the same as the record format for file QIWS/QCUSTCDT.However
we will
// make the CUSNUM field a key field.
AS400FileRecordDescription recordDescription = new
AS400FileRecordDescription(system, "/QSYS.LIB/QIWS.LIB/QCUSTCDT.FILE");

// There is only one record format for the file, so take the first (and
only) element
// of the RecordFormat array returned as the RecordFormat for the file.
System.out.println("擷取 QIWS/QCUSTCDT 的記錄格式...");
qcustcdtFormat = recordDescription.retrieveRecordFormat()[0];
// Indicate that CUSNUM is a key field
qcustcdtFormat.addKeyFieldDescription("CUSNUM");
    }
catch (Exception e)
    {
System.out.println("無法從 QIWS/QCUSTCDT 擷取記錄格式");
e.printStackTrace();
 System.exit(0);
    }

// Create the keyed file object that represents the
// file we will create on the server.We use a QSYSObectPathName object
// to get the name of the file into the correct format.
QSYSObjectPathName fileName = new QSYSObjectPathName(library,
 "QCUSTCDTKY",
 "*FILE",
 "MBR");
KeyedFile file = new KeyedFile(system, fileName.getPath());
```

```
 try
    {
System.out.println("Creating file " + library + "/QCUSTCDTKY...");
// Create the file using the qcustcdtFormat object
file.create(qcustcdtFormat, "Keyed QCUSTCDT file");

// Populate the file with the records contained in QIWS/QCUSTCDT
copyRecords(system, library);

// Open the file for read-only access.Because we will be randomly
// accessing the file, specify a blocking factor of 1.The
// commit lock level parameter will be ignored since commitment
// control has not been started.
file.open(AS400File.READ_ONLY,
1,
AS400File.COMMIT_LOCK_LEVEL_NONE);

// Assume that we want to display the information for customers
// 192837, 392859 and 938472
// The CUSNUM field is a zoned decimal field of length 6 with
// no decimal positions.Therefore, the key field value is
// represented with a BigDecimal.
BigDecimal[] keyValues = {new BigDecimal(192837), new BigDecimal(392859),
new BigDecimal(938472)};

// Create the key for reading the records.The key for a KeyedFile
// is specified with an Object[]
 Object[] key = new Object[1];

Record data = null;
for (int i = 0; i < keyValues.length; i++)
      {
// Setup the key for reading
key[0] = keyValues[i];

// Read the record for customer number keyValues[i]
data = file.read(key);
 if (data != null)
        {
// Display the record only if balance due is greater than
// zero.In that case display the customer name and
// the balance due.The following code pulls fields out
// of the record by field name.As the field is retrieved
// from the record it is converted from the server format to
// Java format.
if (((BigDecimal)data.getField("BALDUE")).floatValue() > 0.0)
          {
System.out.print((String) data.getField("INIT") + "");
System.out.print((String) data.getField("LSTNAM") + "");
System.out.println((BigDecimal) data.getField("BALDUE"));
          }
        }
      }

// All done with the file
```

```
file.close();

// Get rid of the file from the user's system
file.delete();
    }
catch (Exception e)
    {
System.out.println("無法從 QTEMP/QCUSTCDT 建立/讀取");
e.printStackTrace();
 try
      {
file.close();
// Get rid of the file from the user's system
file.delete();
      }
catch(Exception x)
      {
      }
    }

// All done with record level access; disconnect from the
// record-level access server.
system.disconnectService(AS400.RECORDACCESS);
 System.exit(0);
  }

public static void copyRecords(AS400 system, String library)
  {
// Use the CommandCall class to run the CPYF command to copy the records
// in QIWS/QCUSTCDT to QTEMP/QCUSTCDT
CommandCall c = new CommandCall(system, "CPYF FROMFILE(QIWS/QCUSTCDT)
TOFILE(" + library + "/QCUSTCDTKY) MBROPT(*REPLACE)");
 try
    {
System.out.println("從 QIWS/QCUSTCDT 將記錄複製到 " + library +
"/QCUSTCDTKY...");
c.run();
AS400Message[] msgs = c.getMessageList();
if (!msgs[0].getID().equals("CPC2955"))
      {
System.out.println("無法大量輸入資料 " + library + "/QCUSTCDTKY");
for (int i = 0; i < msgs.length; i++)
        {
System.out.println(msgs[i]);
        }
 System.exit(0);
      }
    }
catch (Exception e)
    {
System.out.println("無法大量輸入資料 " + library + "/QCUSTCDTKY");
 System.exit(0);
    }
  }
}
```

範例：使用 UserList 列出給定的群組中的所有使用者

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// User list example.This program lists all of the users in a given
// group.
//
// Command syntax:
//UserListExample system group
//
// This source is an example of IBM Toolbox for Java "UserList".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import java.util.Enumeration;

public class UserListExample
{


 public static void main(String[] args)
    {
// If a system and group were not specified, then display
// help text and exit.
if (args.length != 2)
        {
System.out.println("用法：UserListExample system group");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// The group name was passed as the second command line
// argument.
String groupName = args[1];

// Create the user list object.
UserList userList = new UserList (system);

// Get a list of the users in the given group.
userList.setUserInfo (UserList.MEMBER);
userList.setGroupInfo (groupName);
Enumeration enum = userList.getUsers ();

// Iterate through the list and print out the
// users' names and descriptions.
```

```
 while
(enum.hasMoreElements ())
            {
User u = (User) enum.nextElement ();
System.out.println ("使用者名稱： " + u.getName ());
System.out.println ("説明： " + u.getDescription ());
System.out.println ("");
            }

        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
        }

 System.exit(0);
    }


}
```

範例：JavaBeans

本節會列出 bean 主題文件中提供的程式碼範例。

- 範例：當您連線到系統及切斷連線，並執行指令時，使用接收器來列印備註
- 範例：使用小型程式與 IBM VisualAge for Java 來建立執行指令的按鈕

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

> 程式碼不保事項聲明
>
> IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
> 您可以依據這些範例，產生類似的函數，來符合您的需要。
>
> IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM
> 不擔保或默示保證這些程式的可靠性、可用性或功能。
>
> 所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
> 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：IBM Toolbox for Java Bean 程式碼

下面範例建立 AS400 物件和 CommandCall 物件，然後在物件中登記接收器。當伺服器連線或斷線以及 當 CommandCall 物件完成指令的執行時，物件上的接收程式會列印註解。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////////
//
// Beans example.  This program uses the JavaBeans support in the
// IBM Toolbox for Java classes.
//
// Command syntax:
//    BeanExample
//
///////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.CommandCall;
import com.ibm.as400.access.ConnectionListener;
import com.ibm.as400.access.ConnectionEvent;
import com.ibm.as400.access.ActionCompletedListener;
import com.ibm.as400.access.ActionCompletedEvent;

class BeanExample
{
    AS400       as400_  = new AS400();
    CommandCall cmd_    = new CommandCall( as400_ );

    BeanExample()
    {
        // Whenever the system is connected or disconnected print a
        // comment. Do this by adding a listener to the AS400 object.
        // When a system is connected or disconnected, the AS400 object
        // will call this code.

        as400_.addConnectionListener
        (new ConnectionListener()
          {
             public void connected(ConnectionEvent event)
             {
                 System.out.println( "System connected." );
             }
             public void disconnected(ConnectionEvent event)
             {
                 System.out.println( "System disconnected." );
             }
          }
        );

        // Whenever a command runs to completion print a comment. Do this
        // by adding a listener to the commandCall object. The commandCall
        // object will call this code when it runs a command.

        cmd_.addActionCompletedListener(
```

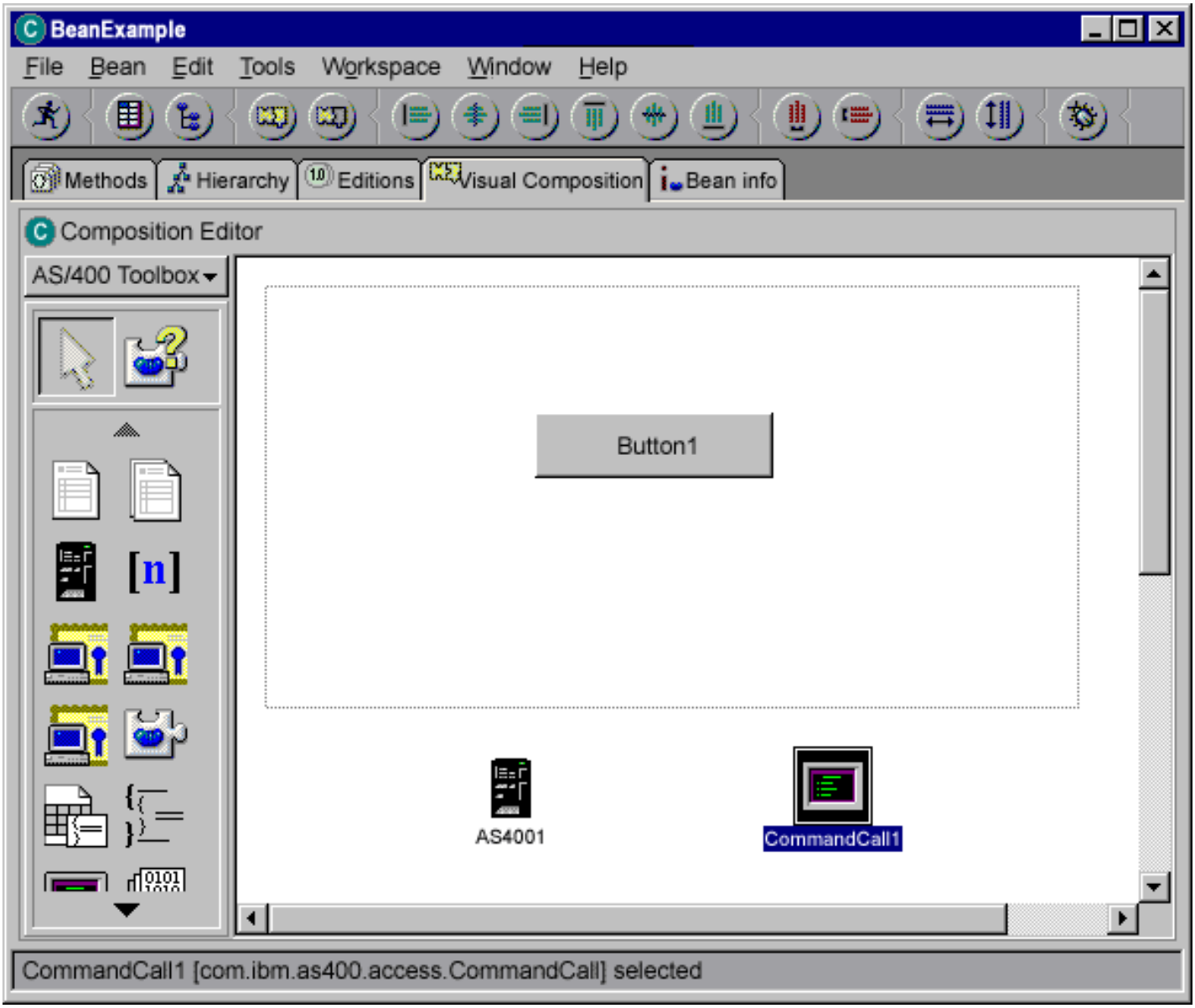
```
            new ActionCompletedListener()
            {
                public void actionCompleted(ActionCompletedEvent event)
                {
                    System.out.println( "Command completed." );
                }
            }
        );
    }

    void runCommand()
    {
     try
        {
            // Run a command. The listeners will print comments when the
            // system is connected and when the command has run to
            // completion.
            cmd_.run( "TESTCMD PARMS" );
        }
        catch (Exception ex)
        {
            System.out.println( ex );
        }
    }

    public static void main(String[] parameters)
    {
       BeanExample be = new BeanExample();

       be.runCommand();

       System.exit(0);
    }
}
```

視覺化 bean 建置器程式範例

本範例使用 IBM VisualAge for Java Enterprise Edition V2.0 組合編輯器，但其它視覺化 bean 建立器也可使用。本範例建立按鈕的 applet，一旦按下此按鈕，即會執行 iSeries 或 AS/400e 伺服器中的指令。

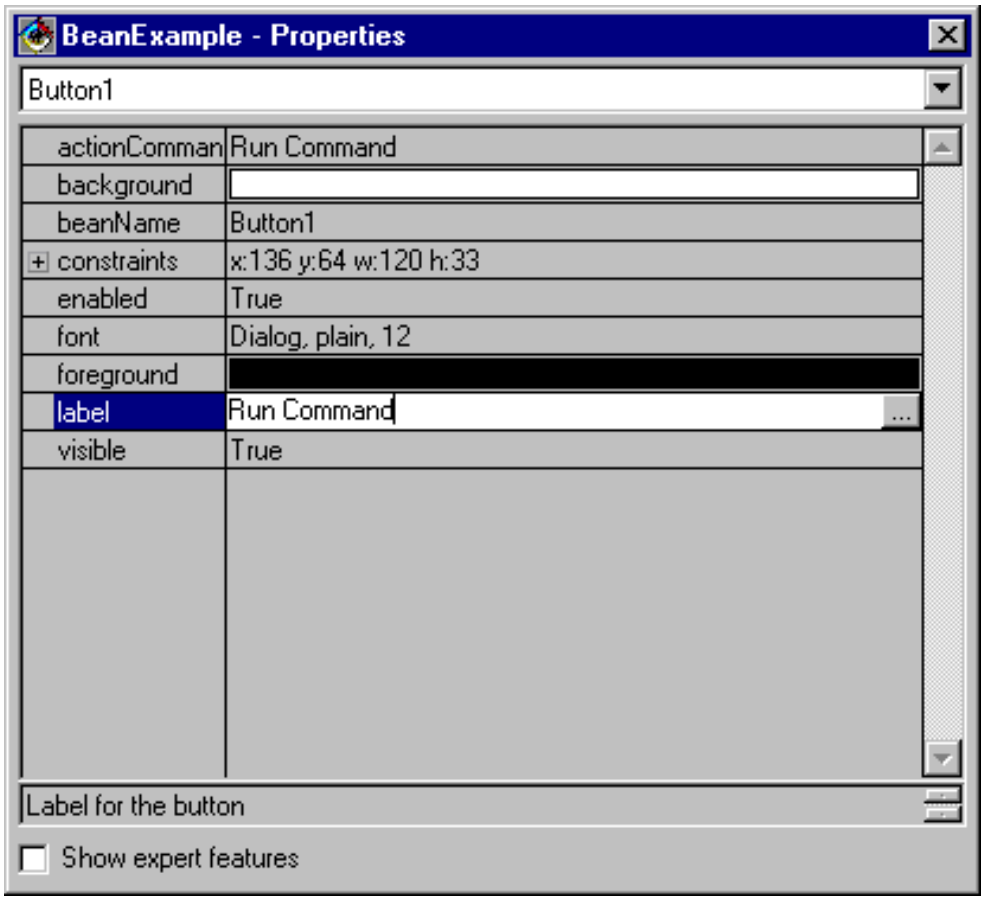
- 拖放 applet 中的按鈕。(在圖形 1 中的 Visual Composition 標籤左邊的 bean 建立器中可找到此「按鈕」。)
- 將 CommandCall bean 與 AS400 bean 拖到小型程式的外面。(在圖形 1 中的 Visual Composition 標籤左邊的 bean 建立器中可找到 bean。)

圖 1：VisualAge Visual Composition Editor 視窗 - gui.BeanExample



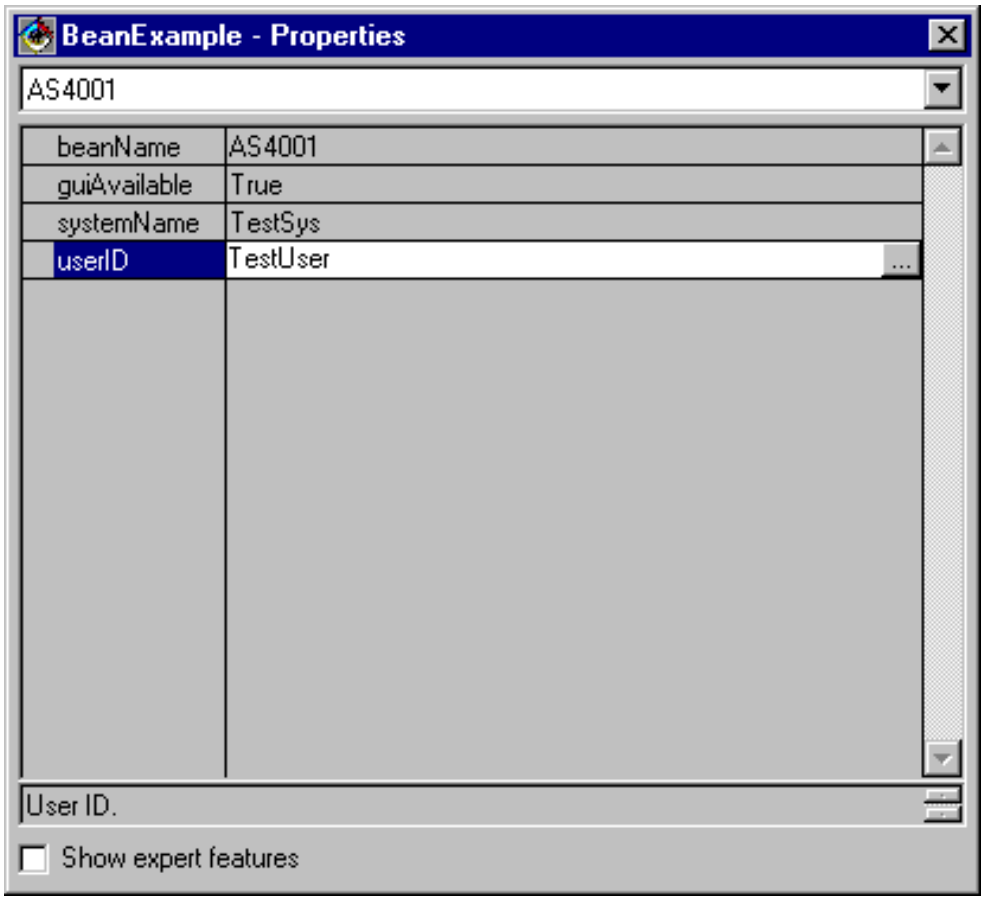
- 編輯 bean 特性。(若要編輯，請選取此 bean，然後按一下右滑鼠按鈕來顯示蹦現視窗， 該視窗有 Properties 選項。)
  - 將 Button 標籤變更為 Run command ，如圖 2 所示。

圖 2：將按鈕標籤變更為 Run command 指令



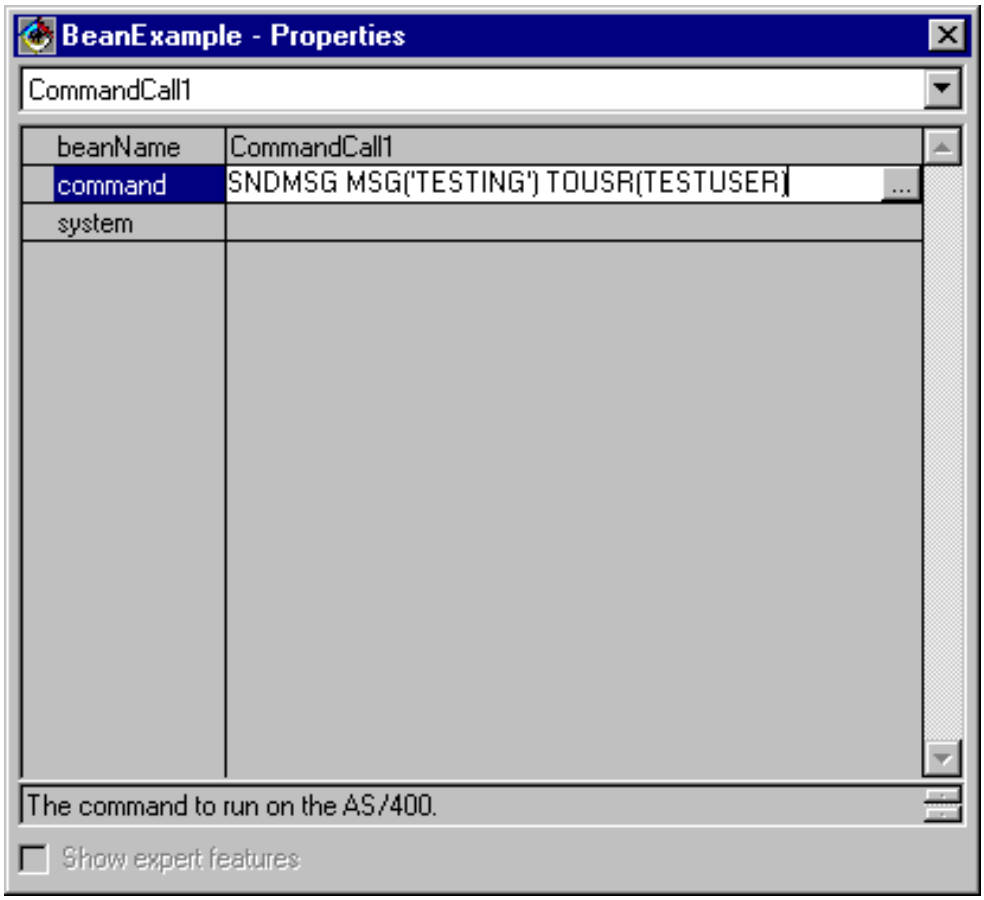
- 將 AS400 bean 的系統名稱變更為 TestSys
- 將 AS400 bean 的 user ID (使用者 ID) 變更為 TestUser, 如圖形 3 所示。

圖 3：將 user ID (使用者 ID) 變更為 TestUser



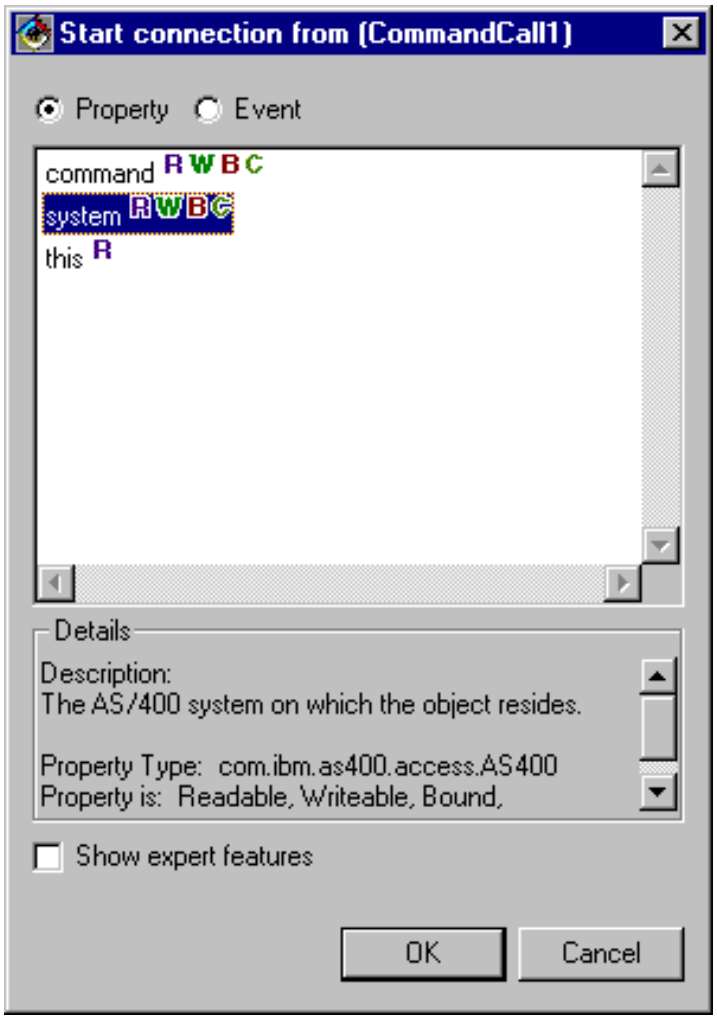
- 將 CommandCall Bean 的指令變更為 SNDMSG MSG('Testing') TOUSR('TESTUSER')　，如圖 4 所示。

圖 4：變更 CommandCall Bean 的指令



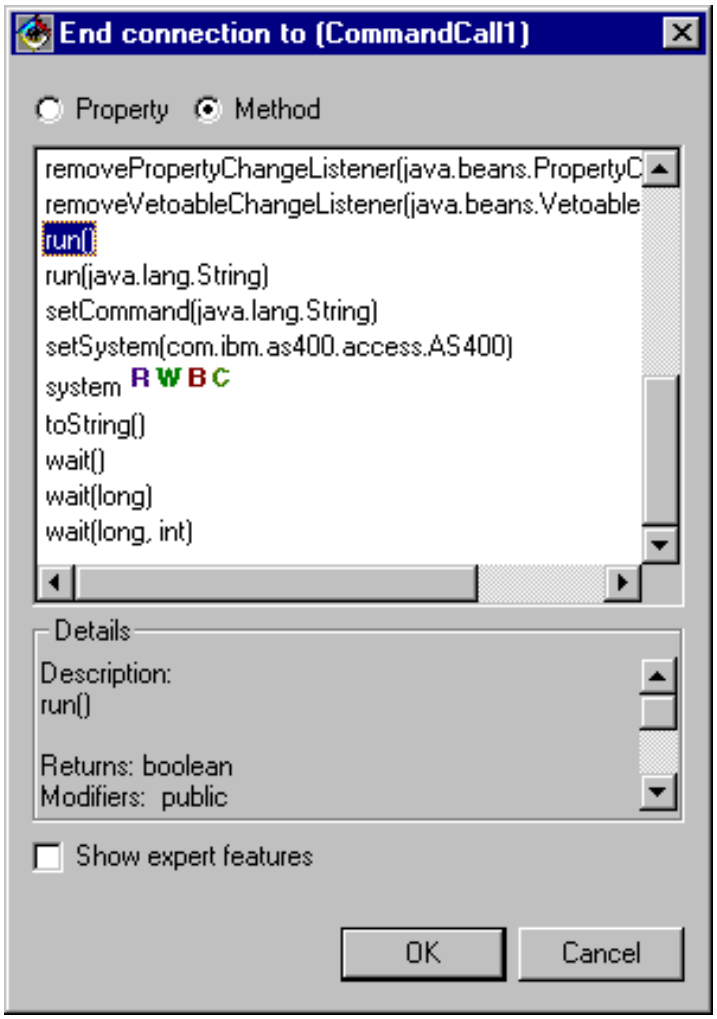
- 連接 AS400 bean 到 CommandCall bean。您用來執行此動作的方法會 因 bean 建立器而不同。在此範例中，請執行下列步驟：
  - 選取 CommandCall bean，然後按一下右滑鼠按鈕
  - 選取Connect
  - 選取Connectable Features
  - 從圖形 5 中顯示的特性列性中選取ystem。
  - 選取 AS400 bean
  - 從顯示於 AS400 Bean 上的蹦現功能表選取his

圖 5：連接 AS400 Bean 到 CommandCall Bean



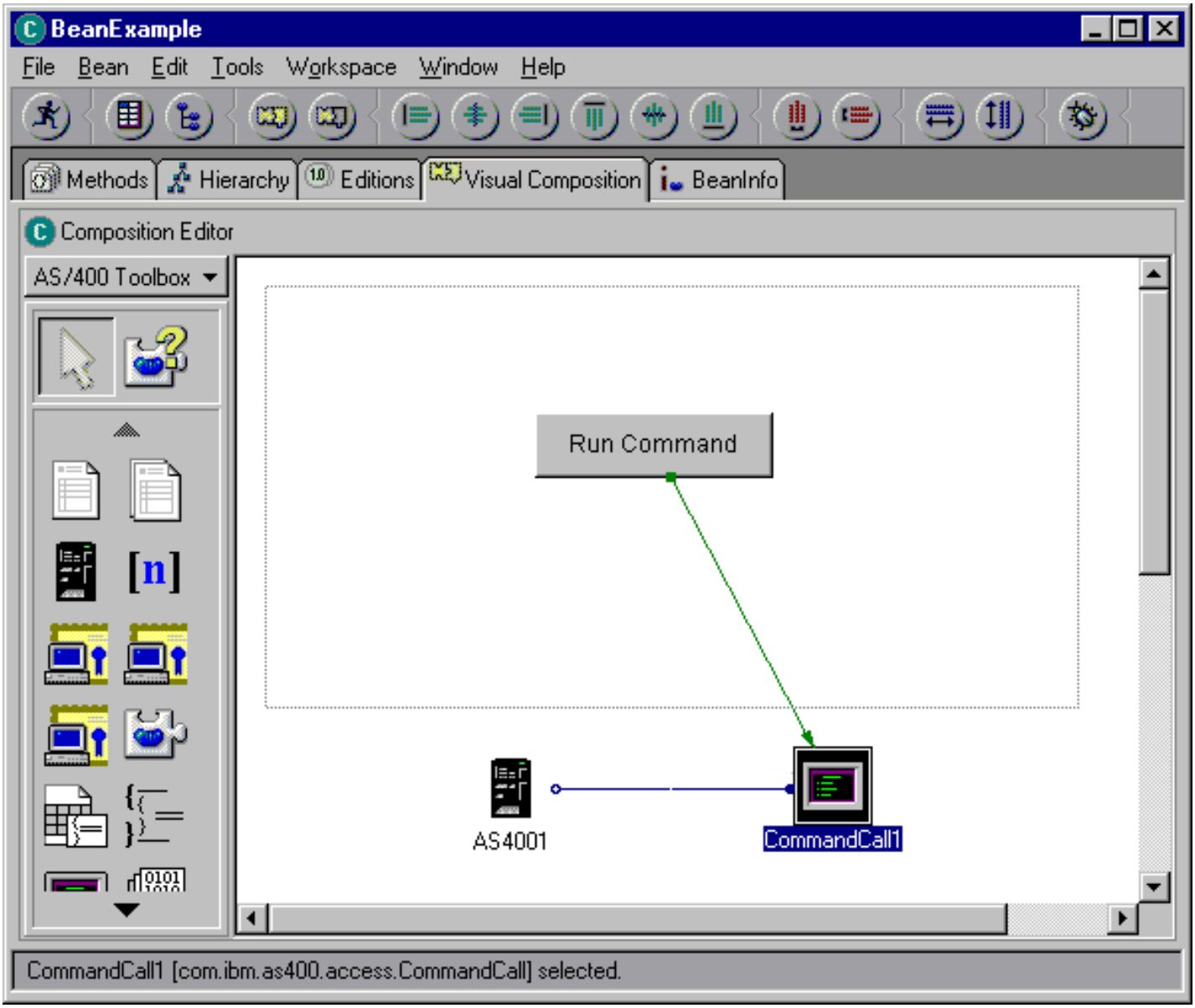
- 使按鈕連線到 CommandCall bean。
  - 選取 Button bean，然後按一下右滑鼠按鈕
  - 選取Connect
  - 選取actionPerformed
  - 選取 CommandCall bean
  - 從出現的蹦現功能表選取nnectable Features
  - 從圖形 6 顯示的方法清單中選取()。

圖 6：連線方法與按鈕



完成後，VisualAge Visual Composition Editor 視窗的外觀應該類似圖形 7。

圖 7：VisualAge Visual Composition Editor 視窗 - 完成的 Bean 範例


BeanExample
File Bean Edit Tools Workspace Window Help
Methods Hierarchy Editions Visual Composition BeanInfo
Composition Editor
AS/400 Toolbox
Run Command
AS4001
CommandCall1
CommandCall1 [com.ibm.as400.access.CommandCall] selected.

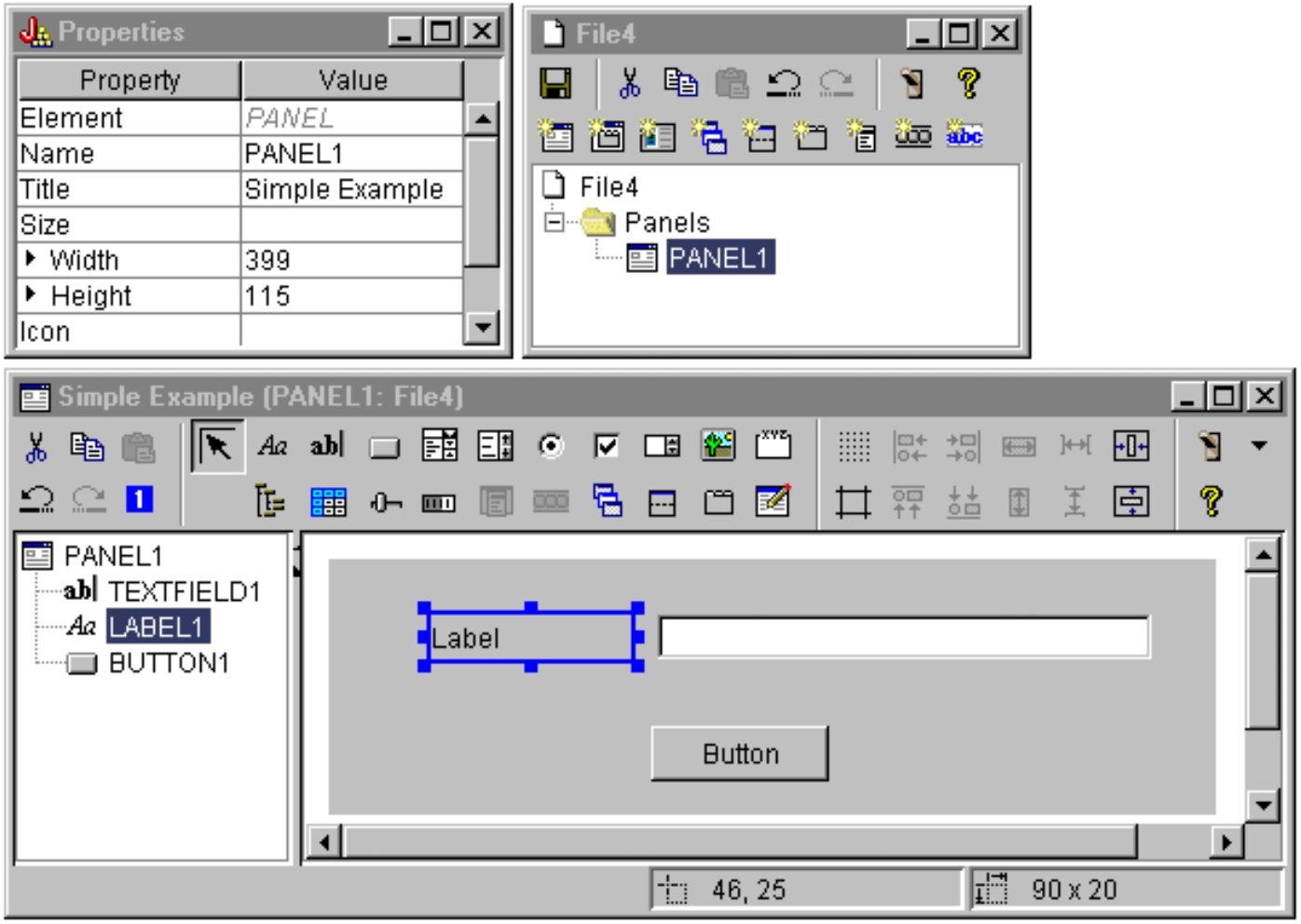
範例：用 GUI Builder 來建構畫面

此範例會藉著建構一個簡單的畫面來示範如何使用 Graphical Toolbox。 這是一個概觀，說明 Graphical Toolbox 環境的基本特性及作業。
在顯示如何建構畫面後，範例會繼續執行，告訴您如何建置一個顯示 畫面的小型 Java 應用程式。 在此範例中，使用者會在文字欄位中輸入資料 Close
按鈕。然後應用程式會將資料傳給 Java 主控台。

建構畫面

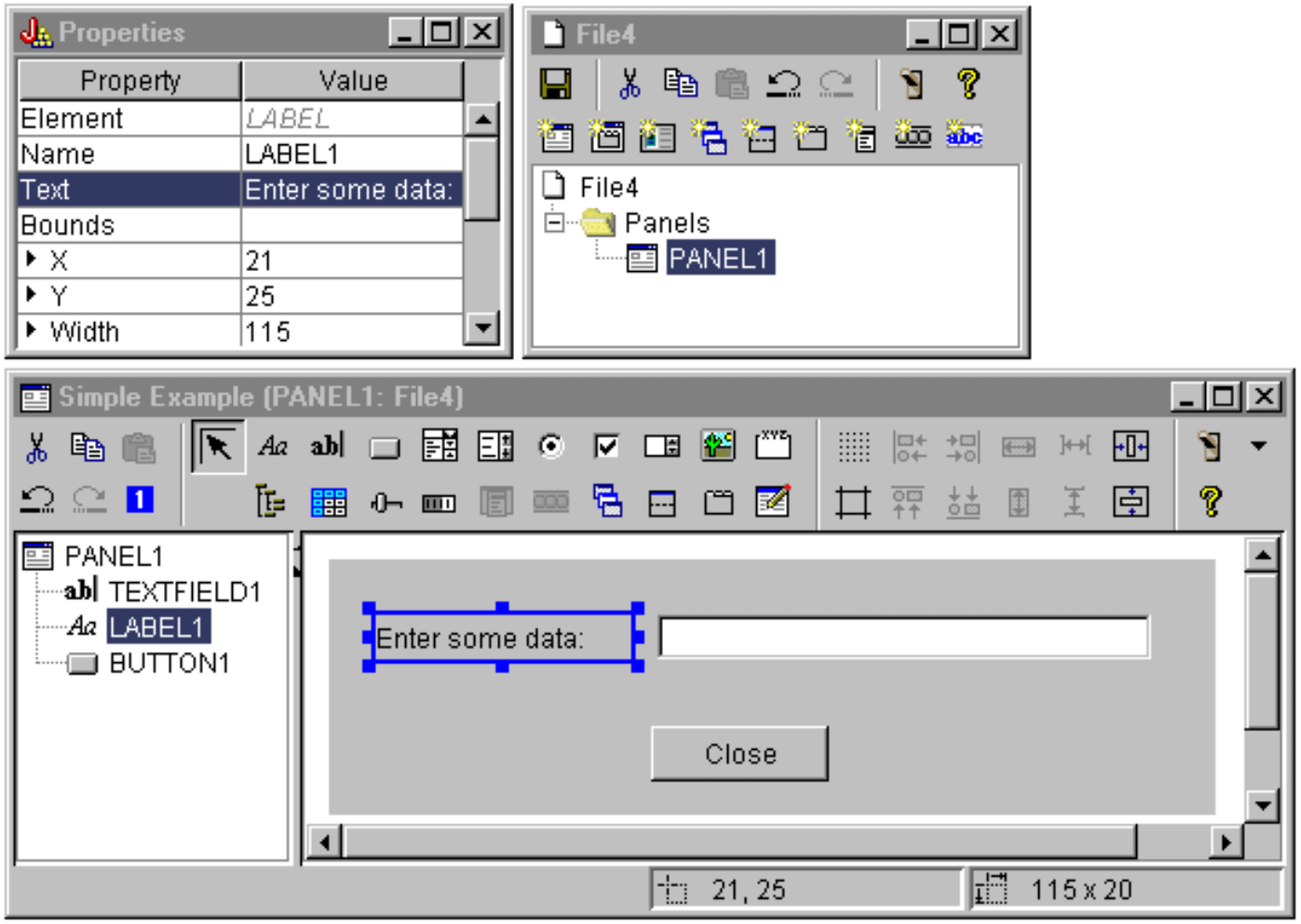
當您啟動 GUI Builder 時，會出現 Properties 及 GUI Builder 視窗。建立一個名為 MyGUI.pdml 的新檔案。 就此例而言，會插入情個新的畫
Insert Panel 圖示。 它的名稱為 "PANEL1"。 在 Properties 視窗中修改資訊，以變更標題；在 Titile 欄位中鍵入 Simple Examp
Delete 鍵即可除去三個預設按鈕。 使用 Panel Builder 視窗中的按鈕，新增圖 1 所示的三個元素：一個標籤、一個 文字欄位及一個按鈕。

圖 1：GUI Builder 視窗：開始建構畫面



選取標籤後，您可以在 Properties 視窗中變更它的文字。本範例中，也對按鈕作出變更，文字改為 Close。

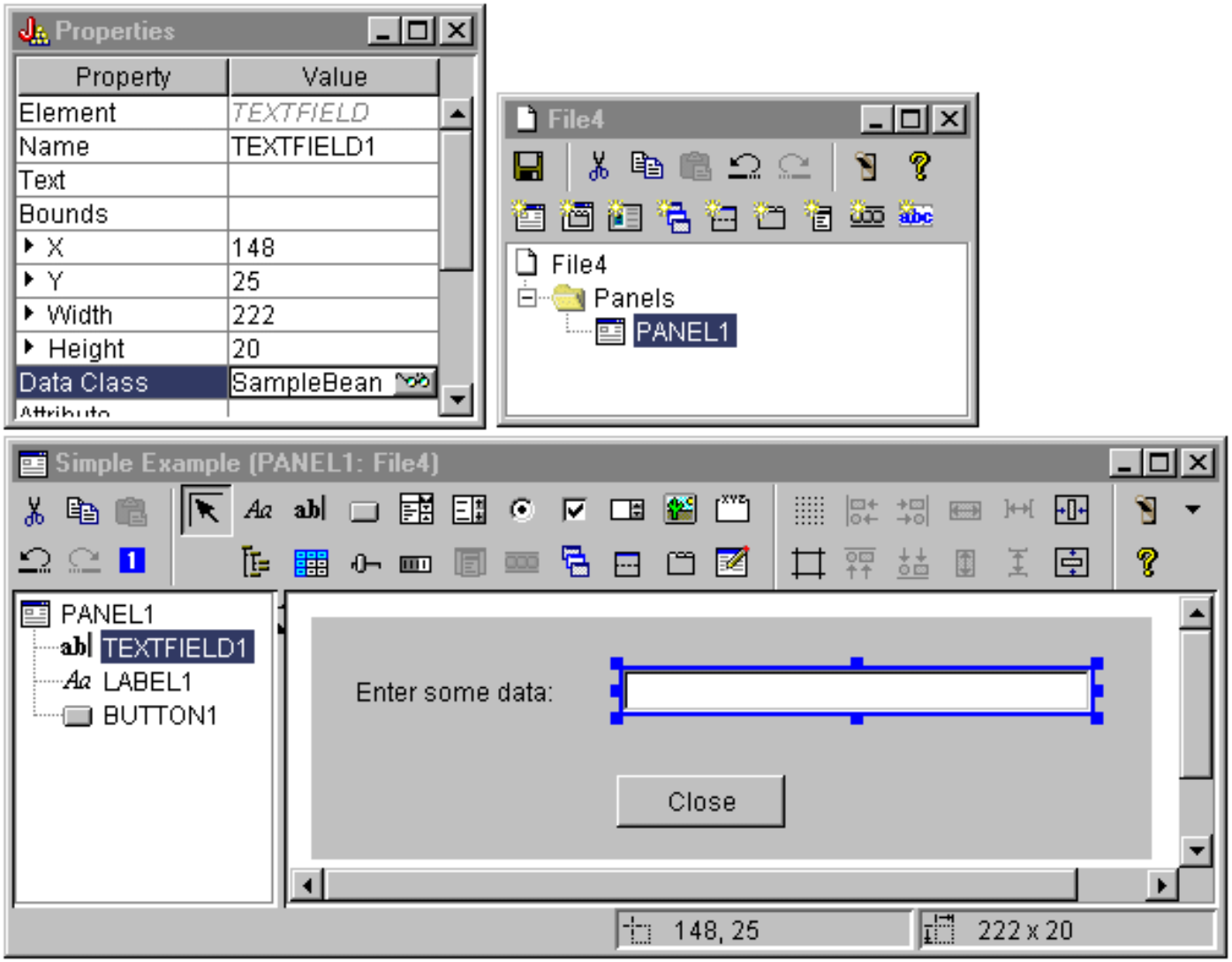
圖 2：GUI Builder 視窗：變更 Porperties 視窗中的文字



Text 欄位

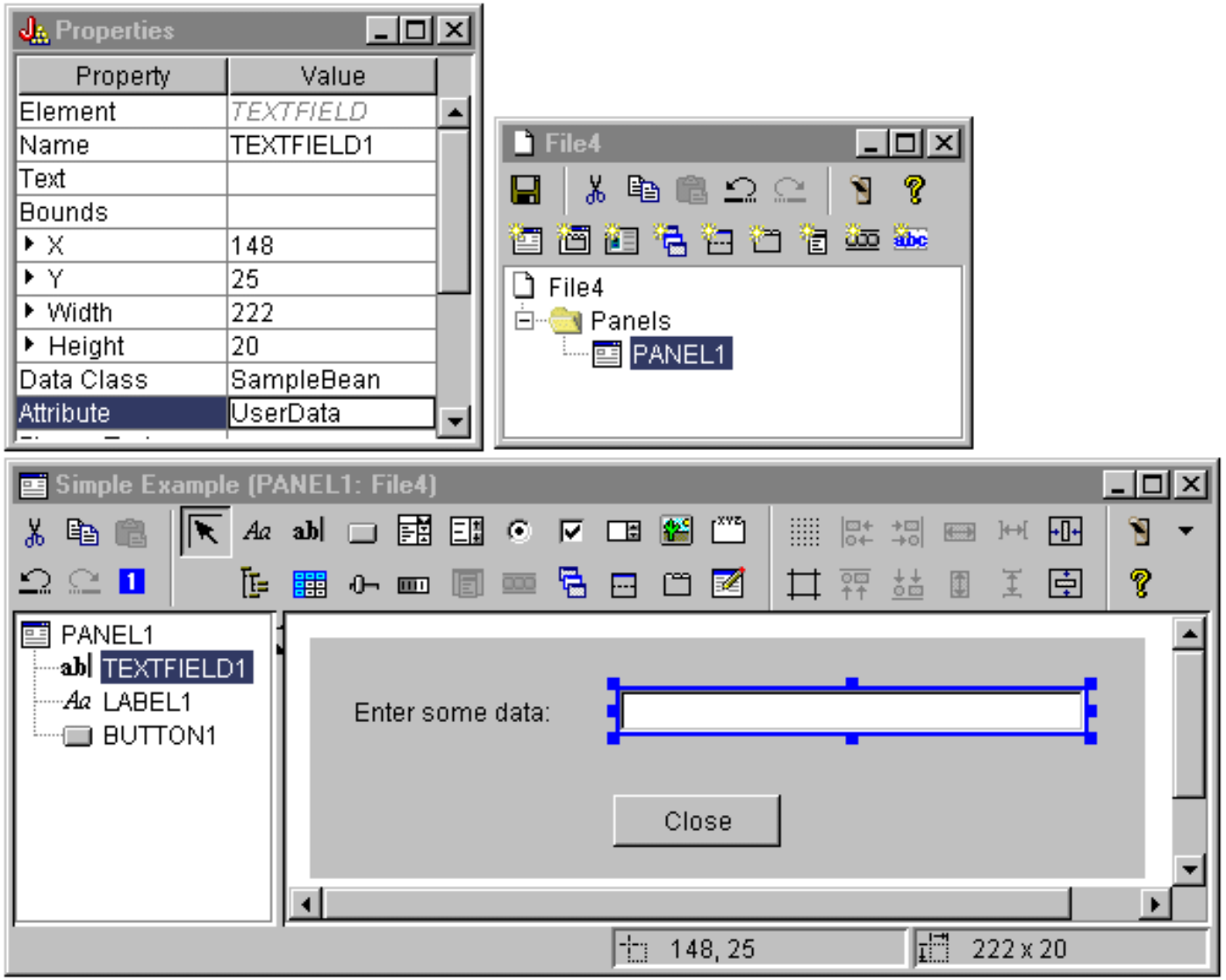
Text 欄位會包含資料，因此，您可以設定數種特性，讓 GUI Builder 執行 一些其它的工作。就本例而言，您可以將 Data Class 內容設定為類別 的名稱，命名為 SampleBean 。此 databean 會提供此文字欄位的資料。

圖 3：GUI Builder 視窗：設定 Data Class 內容



將 Attribute 特性設定為 bean 特性的名稱 (其中含有資料)。 在此情況下, 名稱為data。

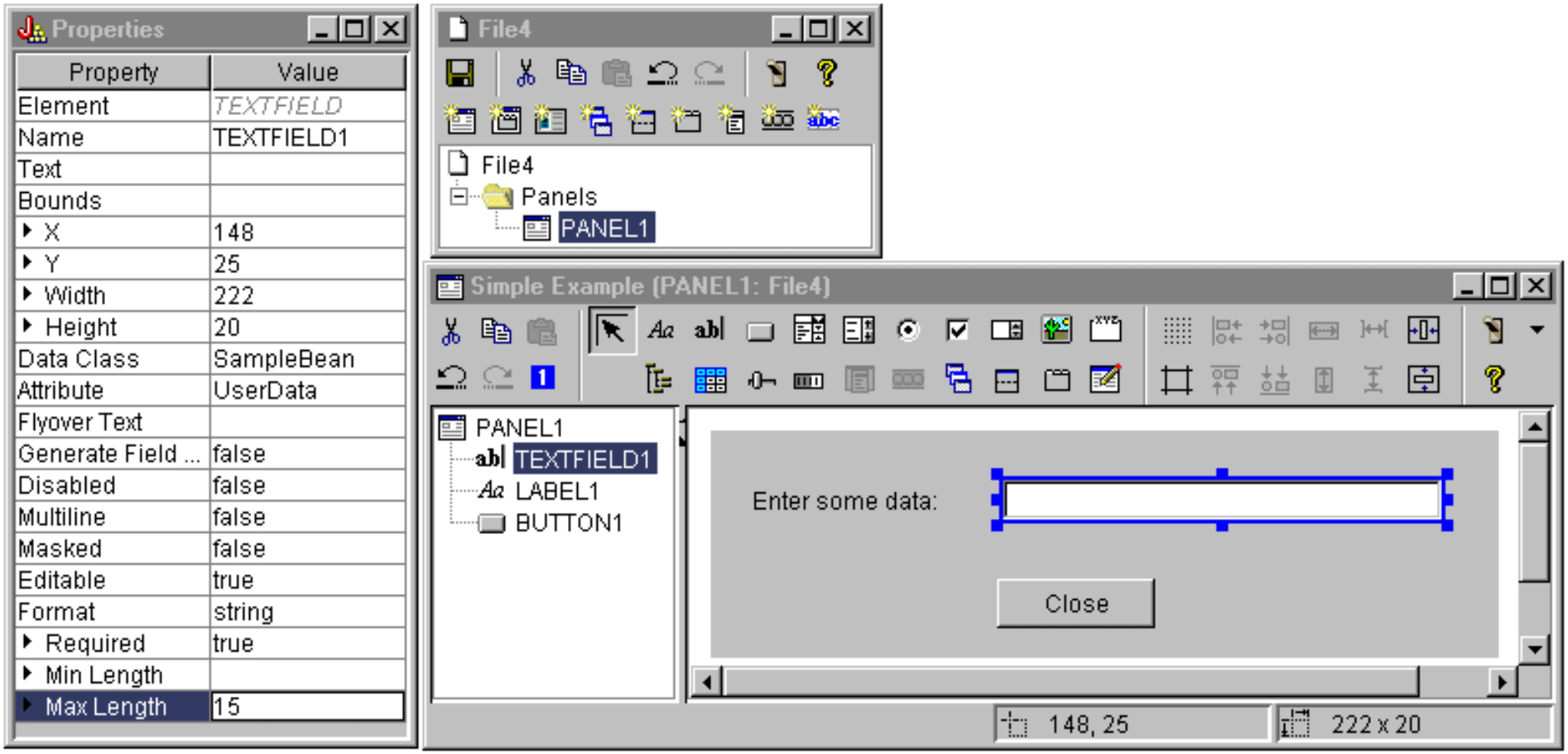
圖 4：GUI Builder 視窗：設定 Attribute 內容



請遵循上述步驟，將 UserData 連結到此 Text 欄位。 在執行時間，Graphical ToolBean 會呼叫SampleBean.getUserData ， 取得此欄位的起始值。然後，當畫面關閉時，會呼叫SampleBean.setUserData ， 將修改過的傳回應用程式。

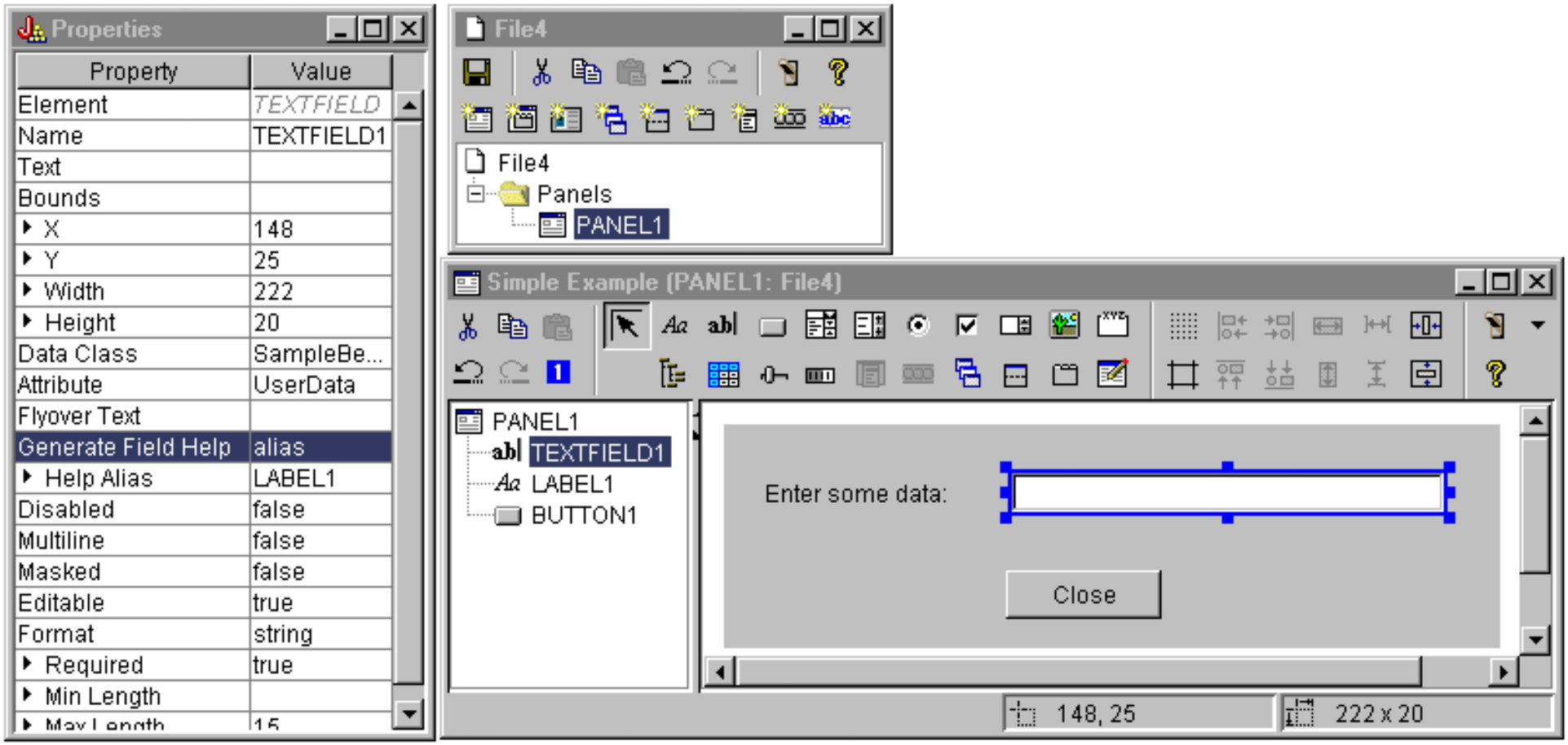
指定使用者必須提供某些資料，且該資料必須是一個字串，字串的長度上限為 15 個字元。

圖 5：GUI Builder 視窗：設定文字欄位的最大長度



表示文字欄位的上下文相關的說明是與標籤 "Enter some data" 相關的說明主題。

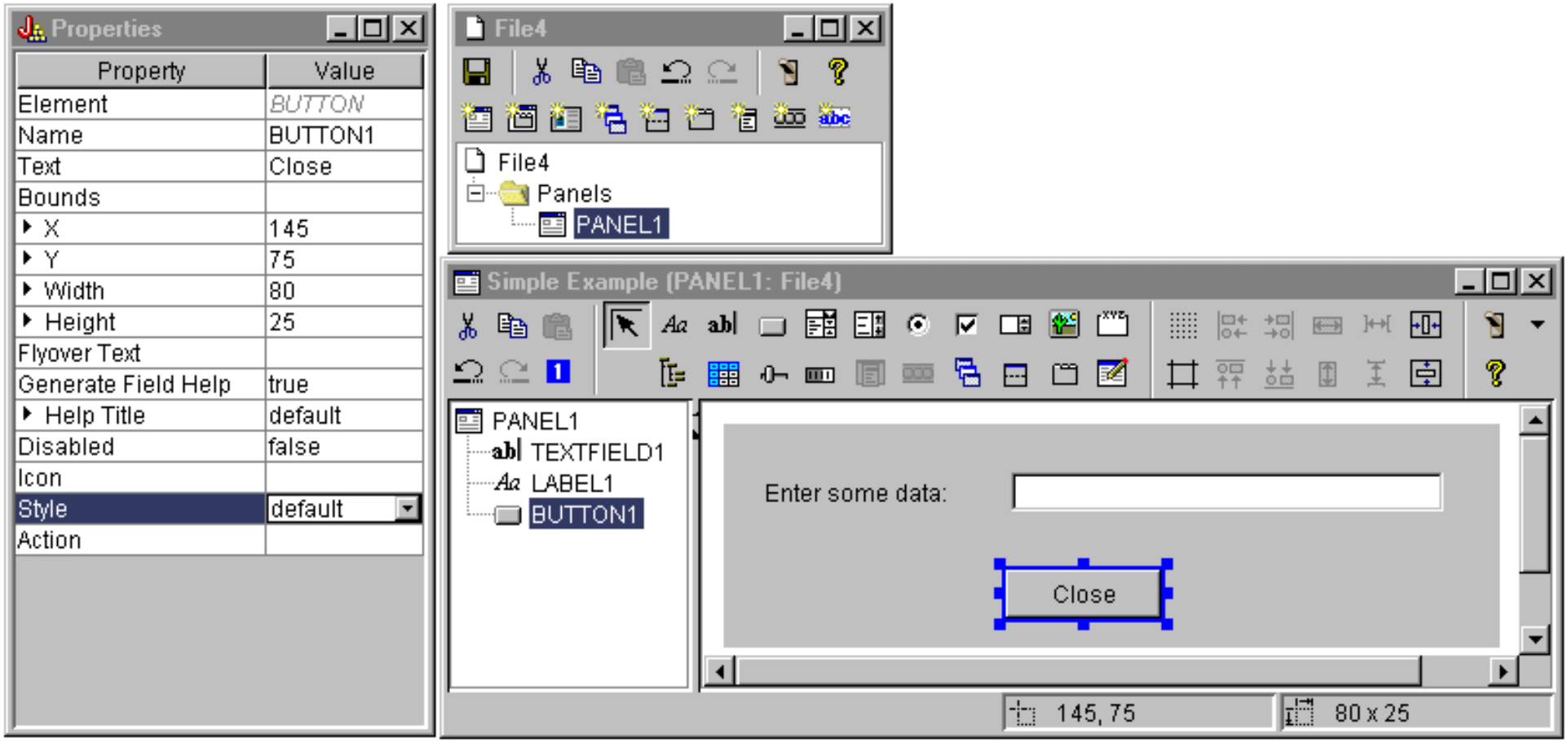
圖 6：GUI Builder 視窗：設定文字欄位的上下文相關說明



按鈕

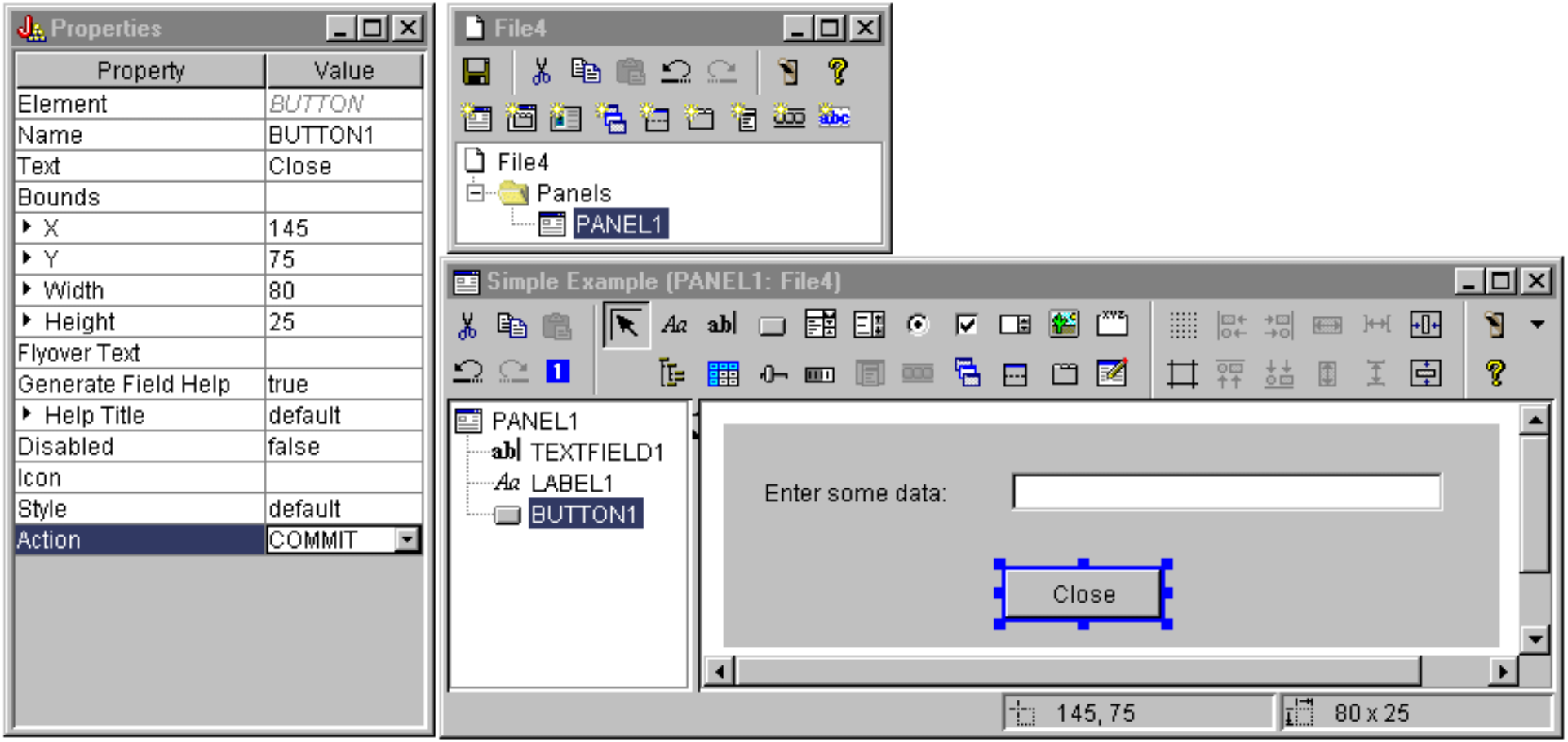
修改此樣式特性，以強調按鈕的預設值。

圖 7：GUI Builder 視窗：設定 Style 內容以強調按鈕的預設值



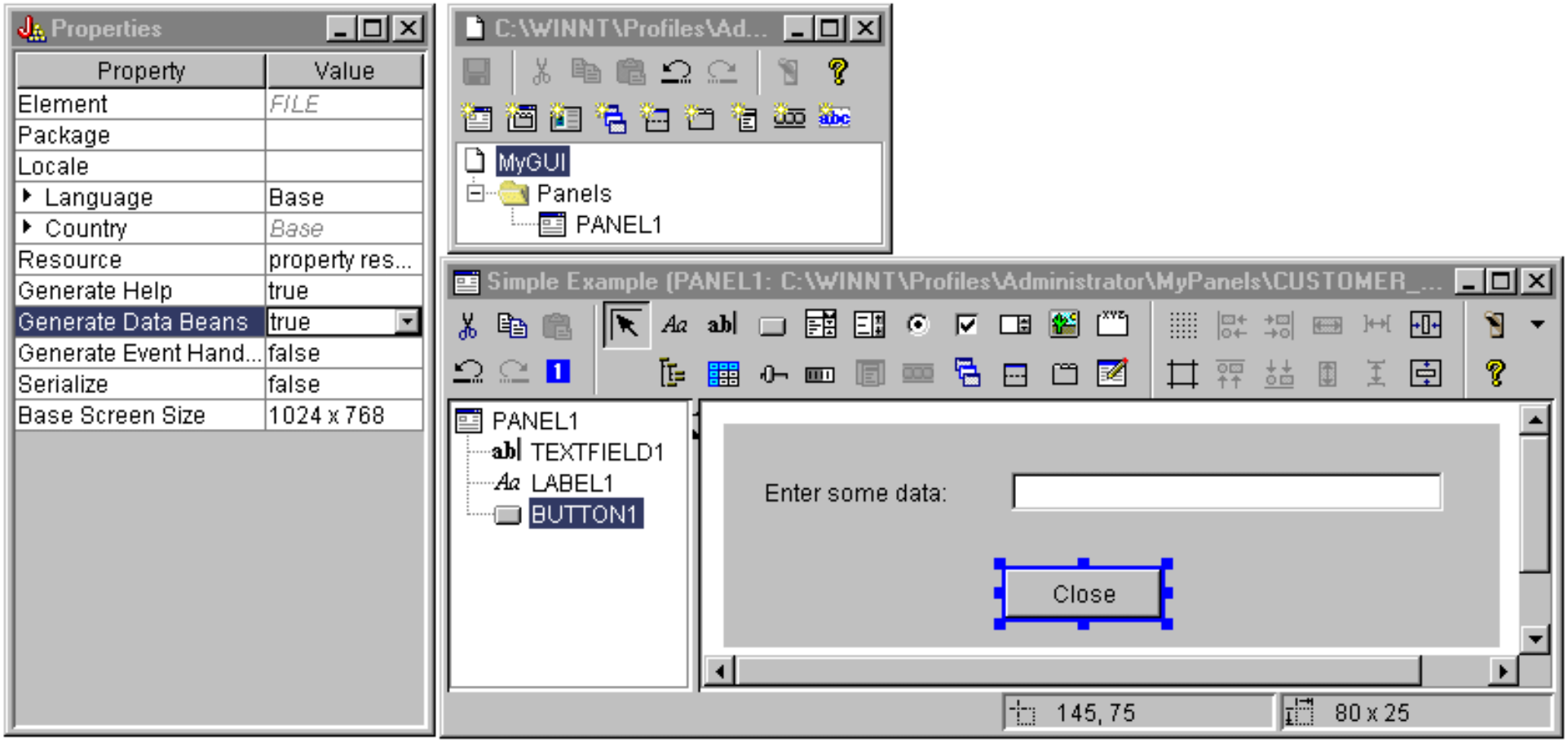
將 ACTION 特性設定為 COMMIT，這會造成在選取按鈕時，會呼叫 bean 中的 UserData 方法。

圖 8：GUI Builder 視窗：設定 Action 內容為 COMMIT



在您儲存畫面之前，請設定 PDML 檔案層次的內容，以產生線上說明架構及 Java bean。 然後您可以在 GUI Builder 視窗中按一 圖示，儲存檔案。在系統提示時，請指定 MyGUI.pdml 的檔名。

圖 9：GUI Builder 視窗：設定內容以產生線上說明架構及 Java bean



產生檔案

在您儲存畫面定義後，您可以察看由 GUI Builder 所產生的檔案。PDML 檔　這是MyGUI.pdml　的內容，可讓您瞭解「畫面定義標示語言」的運作方式。 因為您只能經由 Graphical Toolbox 所提供的工具使用 PDML，所以不需要清楚地瞭解此檔案的格式：

```
<!-- Generated by GUI Builder -->
<PDML version="2.0" source="JAVA" basescreensize="1280x1024">

 <PANEL name="PANEL1">
  <TITLE>PANEL1</TITLE>
  <SIZE>351,162</SIZE>
  <LABEL name="LABEL1"">
```

```
    <TITLE>PANEL1.LABEL1</TITLE>
    <LOCATION>18,36</LOCATION>
    <SIZE>94,18</SIZE>
    <HELPLINK>PANEL1.LABEL1</HELPLINK>
   </LABEL>
   <TEXTFIELD name="TEXTFIELD1">
    <TITLE>PANEL1.TEXTFIELD1</TITLE>
    <LOCATION>125,31</LOCATION>
    <SIZE>191,26</SIZE>
    <DATACLASS>SampleBean</DATACLASS>
    <ATTRIBUTE>UserData</ATTRIBUTE>
    <STRING minlength="0" maxlength="15"/>
    <HELPALIAS>LABEL1</HELPALIAS>
   </TEXTFIELD>
   <BUTTON name="BUTTON1">
    <TITLE>PANEL1.BUTTON1</TITLE>
    <LOCATION>125,100</LOCATION>
    <SIZE>100,26</SIZE>
    <STYLE>DEFAULT</STYLE>
    <ACTION>COMMIT</ACTION>
    <HELPLINK>PANEL1.BUTTON1</HELPLINK>
   </BUTTON>
  </PANEL>

</PDML>
```

資源組

與每一個 PDML 檔相關的是資源組。在本例中，可轉換的資源是儲存 在 PROPERTIES 檔中，稱為 MyGUI.properties。請注意，PROPERTIES 檔也含有 GUI Builder 的自定資料。

```
##Generated by GUI Builder
BUTTON_1=Close
TEXT_1=
@GenerateHelp=1
@Serialize=0
@GenerateBeans=1
LABEL_1=Enter some data:
PANEL_1.Margins=18,18,18,18,18,18
PANEL_1=Simple Example
```

JavaBean

本例也會產生 JavaBean 物件的 Java 原始程式架構。Sample.java 的內容：

```
import com.ibm.as400.ui.framework.java.*;

public class SampleBean extends Object
    implements DataBean
{
    private String m_sUserData;

    public String getUserData()
    {
        return m_sUserData;
    }

    public void setUserData(String s)
    {
        m_sUserData = s;
    }

    public Capabilities getCapabilities()
    {
        return null;
    }

    public void verifyChanges()
    {
    }

    public void save()
    {
    }

    public void load()
    {
        m_sUserData ="";
    }
}
```

請注意，架構已含有 UserData 特性的 gettor 及 settor 方法之施行方式。 其它的方法是由 介面定義，所以是必要的。

GUI Builder 已對架構呼叫了 Java 編譯器，且產生了對應的類別檔。 對於此簡單範例的目的，您不需要修改 bean 施行方式。在 真正程式中，您通常會修改 load 及 save 方法，以轉送外部資料來源的資料。其它兩個方法的預設施行方式 通常足夠了。如需其它相關資訊，請參閱有關 bean 介面的說明文件，該文件是在 PDML 執行時間組織架構的 javadocs

說明檔

GUI Builder 也會建立一個 HTML 組織架構，稱為 Help Document。 解說寫出器可以編輯此檔案，以輕易地管理解說資訊。 若需其餘相關資訊，請參閱下列主題：

- 建立 Help Document
- 編輯由 GUI Builder 產生的 Help Document

建構應用程式

一旦您已儲存畫面定義及產生的檔案，您就可以建構應用程式了。 您所需要的就是一個新的 Java 原始檔，其中含有應用程式的主進入點。例如，此檔案稱為 SampleApplication.java。它包含了下列程式：

```
import com.ibm.as400.ui.framework.java.*;
import java.awt.Frame;


public class SampleApplication
{
    public static void main(String[] args)
    {
        // Instantiate the bean object that supplies data to the panel
        SampleBean bean = new SampleBean();


        // Initialize the object
        bean.load();


        // Set up to pass the bean to the panel manager
        DataBean[] beans = { bean };


       // Create the panel manager.參數：
        // 1. PDML file as a resource name
        // 2. Name of panel to display
        // 3. List of data objects that supply panel data
        // 4. An AWT Frame to make the panel modal


        PanelManager pm = null;
```

```
        try { pm = new PanelManager("MyGUI", "PANEL_1", beans, new Frame()); }
        catch (DisplayManagerException e)
        {
            // Something didn't work, so display a message and exit
           e.displayUserMessage(null);
            System.exit(1);
        }

        // Display the panel - we give up control here
       pm.setVisible(true);

        // Echo the saved user data
        System.out.println("SAVED USER DATA: '" + bean.getUserData() + "'");

        // Exit the application
        System.exit(0);
    }
}
```

呼叫程式的責任在於呼叫load 來起始設定 bean 物件。如果畫面的資料是多個 bean 物件提供的，則在傳遞每一個物件到 Graphical Toolbox 環境之前， 應先起始設定它們

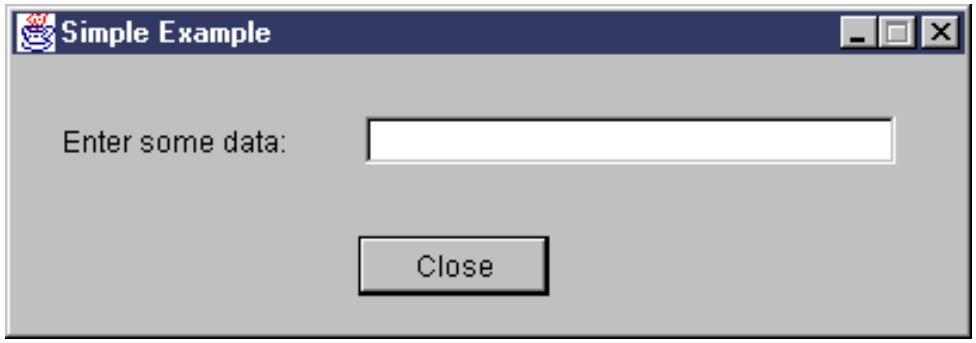
類別 com.ibm.as400.ui.framework.java.PanelManager 提 供 API 來顯示獨立式視窗及對話框。同於建構元上提供的名稱 的 PDML 檔名會被 Graphical Toolbox 視為資源名稱 - 目錄、ZIP 檔或含有 PDML 的 JAR 須 在 classpath 中識別。

因為Frame 物件是在建構元上提供的，所以視窗將以限制模式的對話框 運作。在真正的 Java 應用程式中，這個物件可能取自於適合對話框的上代視窗。
因為視窗是限制模式的，所以控制權不會傳回到應用程式，直到使用者關閉視窗為止。此時，應用程式僅回應修改過的使用者資料並結束執行。

執行應用程式

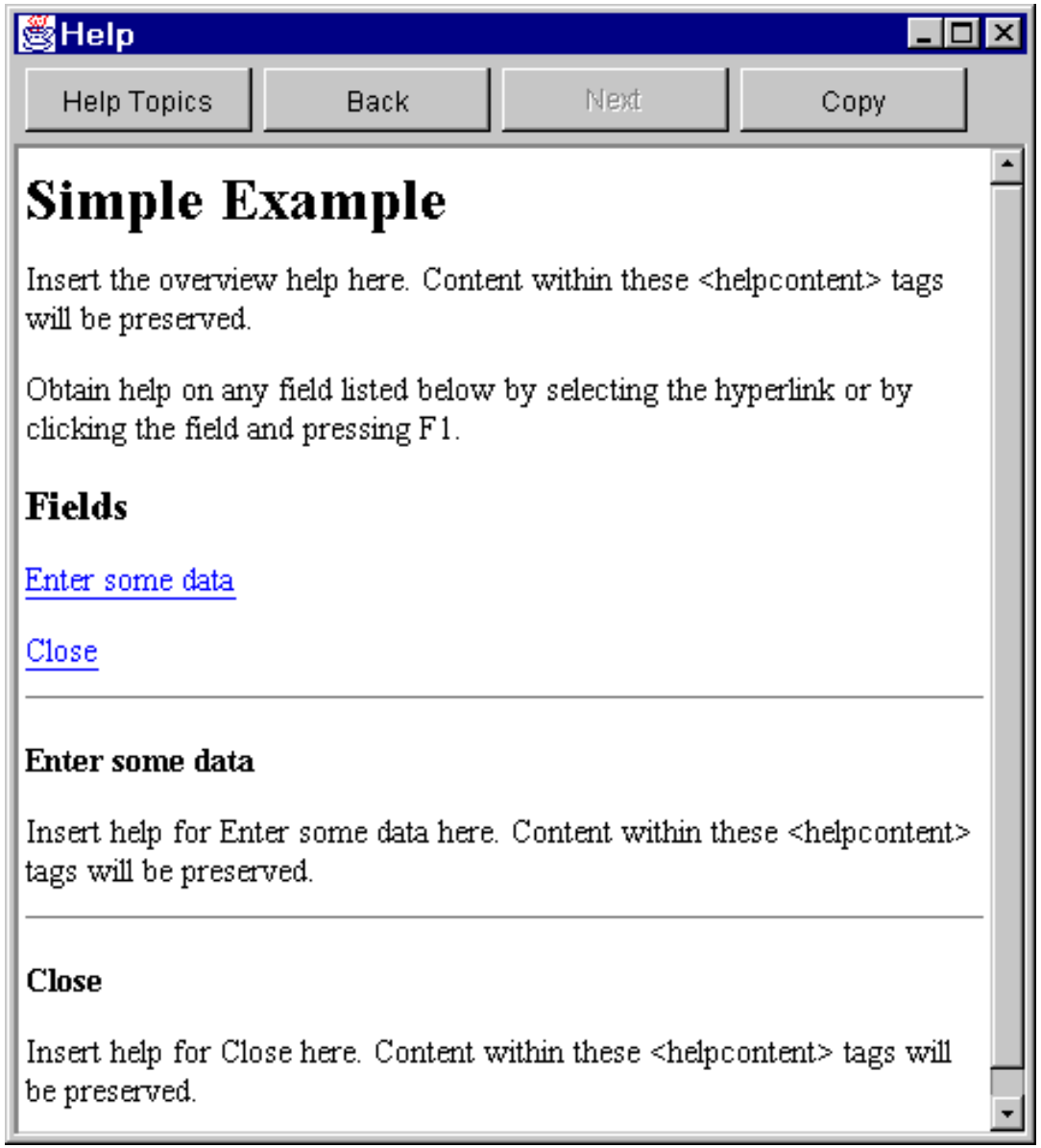
底下是編譯及執行應用程式時視窗的樣子：

圖 10：簡式範例應用程式視窗



當焦點在文字欄位上時，若使用者按下 F1，則 Graphical Toolbox 將顯示說明瀏覽器，它含有 GUI Builder 產生的線上說明架構。

圖 11：簡式範例線上說明架構



您可以編輯 HTML，並新增範例中所示的說明主題的實際說明內容。

如果文字欄位中的資料無效（例如，如果使用者未提供一個值，即按一 下lose 按鈕），則 Graphical Toolbox 將顯示一則錯誤訊息，並傳回無點給欄位，以便可以輸入資料。

圖 12：資料錯誤訊息

關於如何執行這個範例作為 applet 的資訊，請參閱[在瀏覽器中使用 Graphical Toolbox](#)。

可編輯的組合框

當控制項產生器建立「可編輯的組合框」之取出元及設定元時，就預設值而言，它會傳回設定元中的「字串」並採用取出元中的字串參數。若將設定元變更為採用「物件」類別，並將取出元變更為傳回「物件」類型， 會很有幫助。 這可讓您使用 ChoiceDescriptors 來決定使用者選項。

如果已偵測到取出元及設定元的物件 (Object) 類型，則系統會希望輸入 ChoiceDescriptor 或物件類型，而非格式化字串。

範例

假設可編輯的物件是一個可編輯的組合框 (ComboBox)，它有雙重 (Double) 值：使用系統 值 (system value)，或該值尚未設定。

```
public Object getEditable()
{
    if (m_setting == SYSTEMVALUE)
    {
       return new ChoiceDescriptor("choice1","System Value");
    }
    else if (m_setting == NOTSET)
    {
       return new ChoiceDescriptor("choice2","Value not set");
    }
        else
    {
       return m_doubleValue;
    }
}
```

同樣地，在已偵測到取出元及設定元的物件類型時，系統會傳回一 物件，該物件是含由已選取選項的 ChoiceDescriptor 或一「物件」類型。

```
public void setEditable(Object item)
{
    if (ChoiceDescriptor.class.isAssignableForm(obj.getClass()))
    {
       if (((ChoiceDescriptor)obj).getName().equalsIgnoreCase("choice1"))
           m_setting = SYSTEMVALUE;
        else
           m_setting = NOTSET;
    }
    else if (Double.class.isAssignableFrom(obj.getClass()))
    {
       m_setting = VALUE;
       m_doubleValue = (Double)obj;
    }
        else
    { /* error processing */ }
```

```
}
```

使用 GUI Builder 建立 Pane (畫面)

使用 GUI Builder 建立畫面是一件很簡單的事。請從主 GUI Builder 視窗的功能表列中，F選取 New File。

在 GUI BuilderF的le視窗的功能表列上，按一下 Insert New Panel 圖示，顯示可於其中插入畫面元件的畫面建置器。Panel 視窗的工具列按鈕代表可新增至畫面的不同元件。選取您要的元件，然後在您要放置元件的位置上按一下。

下圖顯示利用數個您可使用的選項所建立的畫面。

圖 1：使用 GUI Builder 建立 Panel Sampel

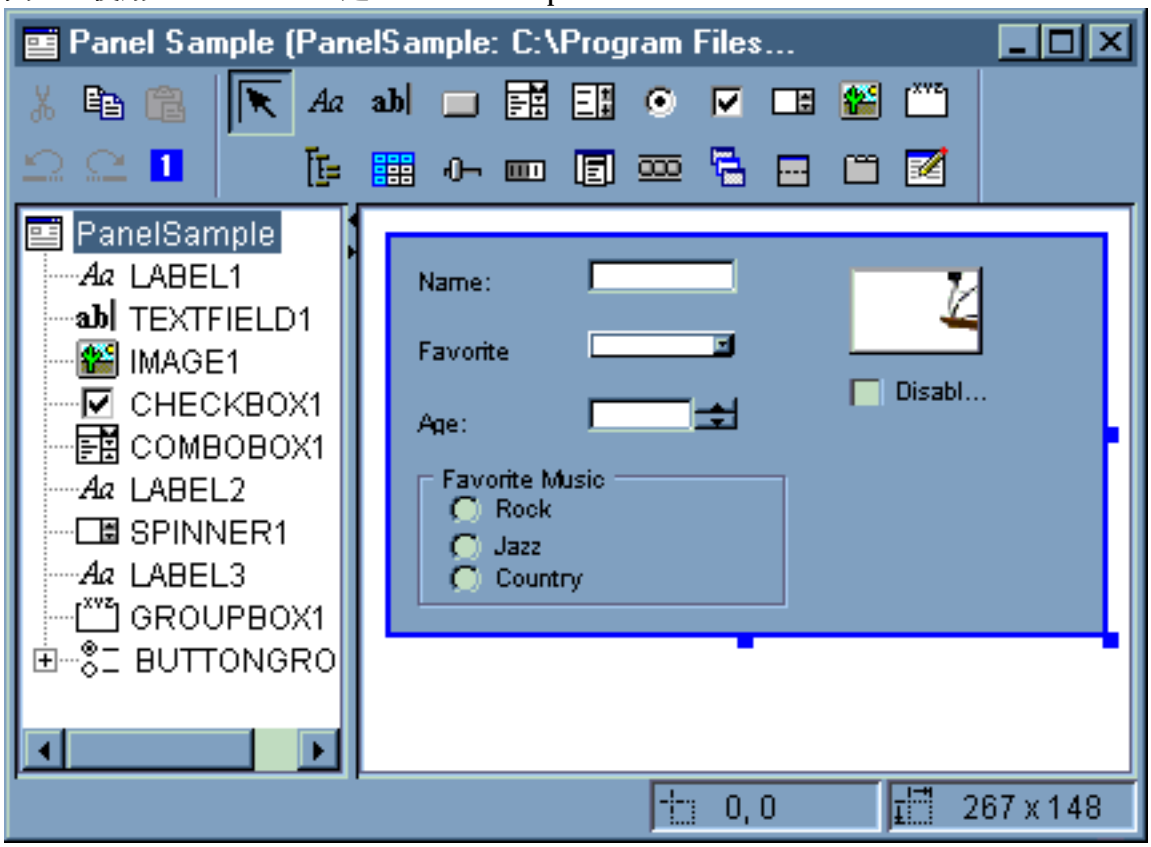


圖 1中的範例畫面使用下列 DataBean 程式碼，將不同的元件聚集在一起：

```
import com.ibm.as400.ui.framework.java.*;

public class PanelSampleDataBean extends Object
    implements DataBean
{
    private String m_sName;
    private Object m_oFavoriteFood;
    private ChoiceDescriptor[] m_cdFavoriteFood;
    private Object m_oAge;
    private String m_sFavoriteMusic;
```

```
public String getName()
{
    return m_sName;
}

public void setName(String s)
{
    m_sName = s;
}

public Object getFavoriteFood()
{
    return m_oFavoriteFood;
}

public void setFavoriteFood(Object o)
{
    m_oFavoriteFood = o;
}

public ChoiceDescriptor[] getFavoriteFoodChoices()
{
    return m_cdFavoriteFood;
}

public Object getAge()
{
    return m_oAge;
}

public void setAge(Object o)
{
    m_oAge = o;
}

public String getFavoriteMusic()
{
    return m_sFavoriteMusic;
}

public void setFavoriteMusic(String s)
{
    m_sFavoriteMusic = s;
}

public Capabilities getCapabilities()
{
     return null;
}

public void verifyChanges()
{
}

public void save()
{
```

```
        System.out.println("Name = " + m_sName);
        System.out.println("Favorite Food = " + m_oFavoriteFood);
        System.out.println("Age = " + m_oAge);
        String sMusic = "";
        if (m_sFavoriteMusic != null)
        {
            if (m_sFavoriteMusic.equals("RADIOBUTTON1"))
                sMusic = "Rock";
            else if (m_sFavoriteMusic.equals("RADIOBUTTON2"))
                sMusic = "Jazz";
            else if (m_sFavoriteMusic.equals("RADIOBUTTON3"))
                sMusic = "Country";
        }
        System.out.println("Favorite Music = " + sMusic);
    }

    public void load()
    {
        m_sName = "Sample Name";
        m_oFavoriteFood = null;
        m_cdFavoriteFood = new ChoiceDescriptor[0];
        m_oAge = new Integer(50);
        m_sFavoriteMusic = "RADIOBUTTON1";
    }
}
```

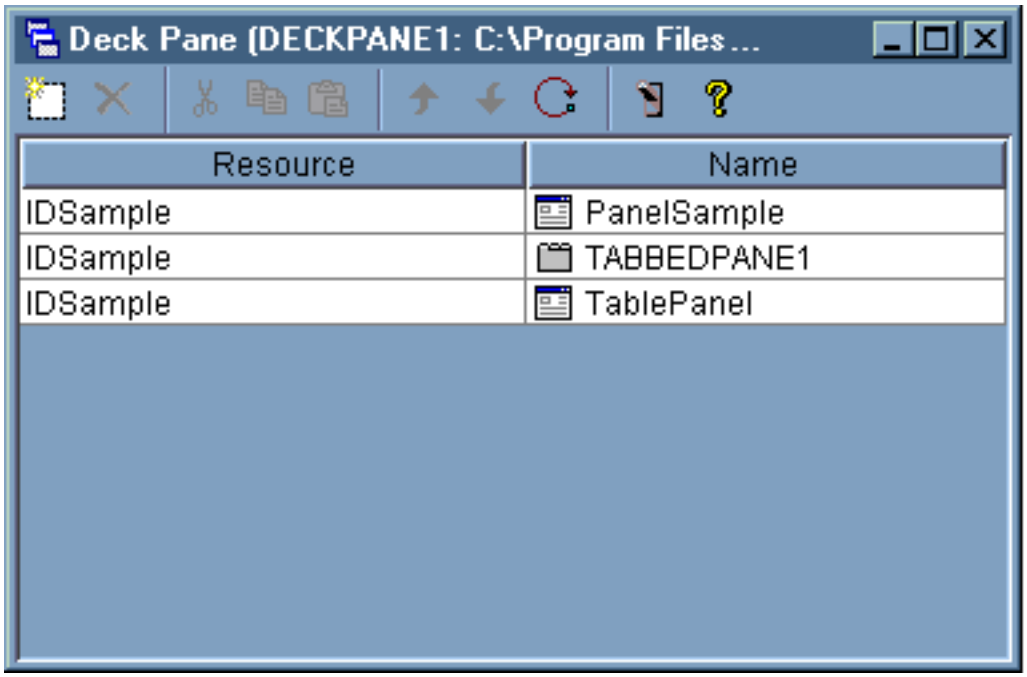
畫面是 GUI Builder 中最簡單可使用的元件，但在此簡單的畫面中，您可建置許多 UI 應用程式。

使用 GUI Builder 建立一個 Deck Pane (疊窗格)

GUI Builder 可讓建立一個疊窗格更容易。從 GUI Builder 視窗的功能表列中，選取 File --> New File。

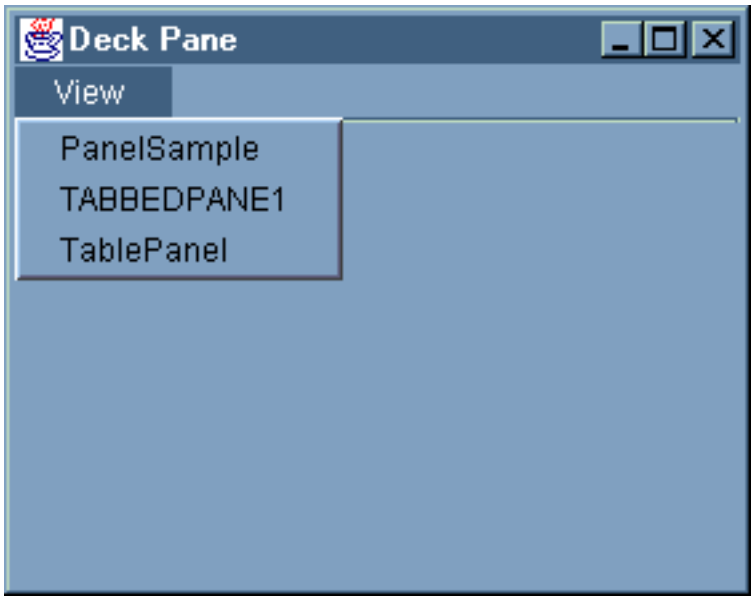
在 GUI Builder File 視窗的功能表列上，按一下 Insert Deck Pane 工具按鈕 來顯示窗格建置器，讓您在裡面插入一個疊窗格的元件。在下列範例中，新增了三個元件。

圖 1：使用 GUI Builder 建立一個疊窗格



在建立一個疊窗格之後，請按一下 Preview 工具按鈕 來預覽它。除非您選取 View 功能表，否則一個疊窗格看起來是純黑白的。

圖 2：使用 GUI Builder 預覽 Deck Pane



從 Deck PaneView 功能表，選取您要檢視的元件。 就此範例而言，您可以選取要檢視 PanelSample、TABBEDPANE1 或 TablePanel。下列圖示說明當您檢視這些元件時所看到的內容。

圖 3：使用 GUI Builder 檢視 PanelSample

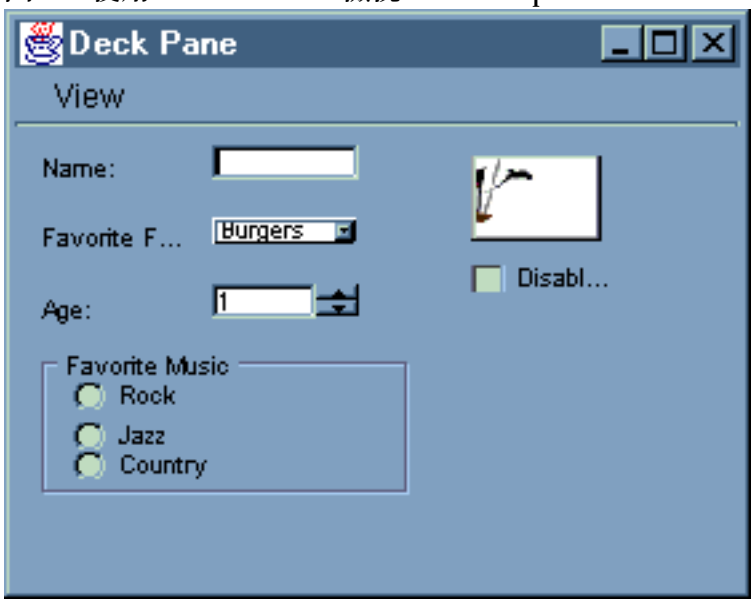


圖 4：使用 GUI Builder 檢視 TABBEDPANE1

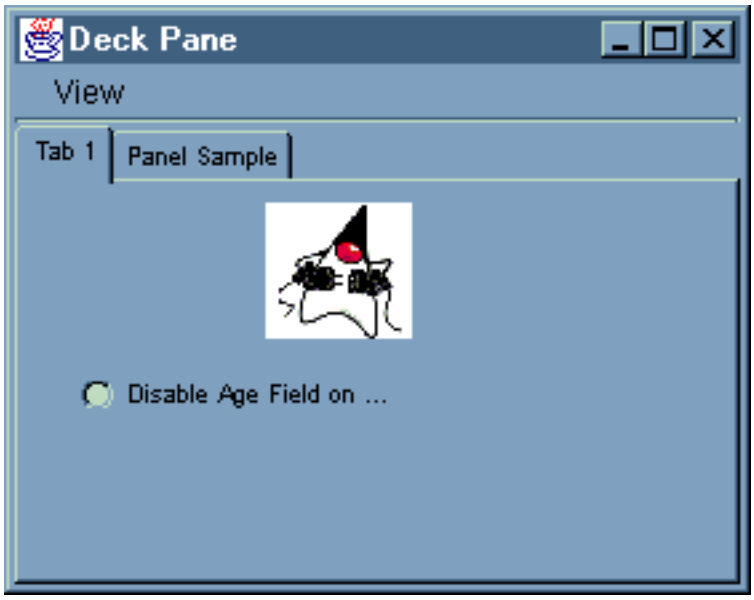
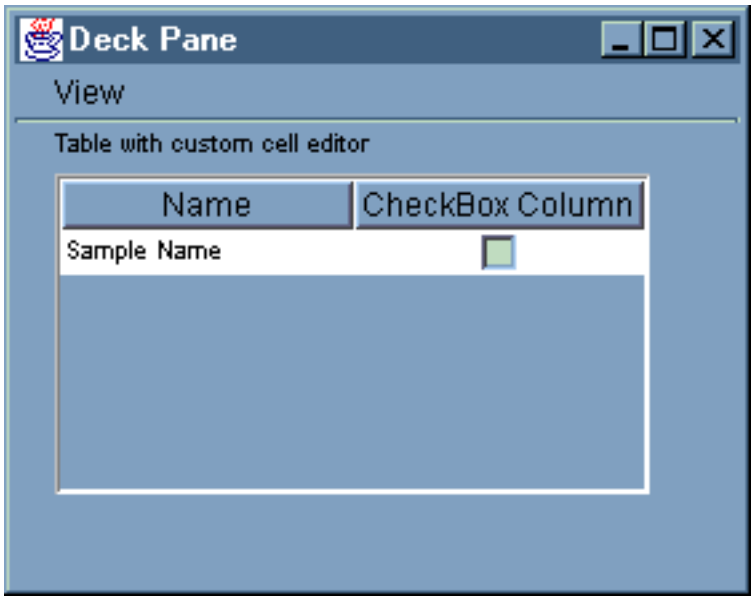


圖 5：使用 GUI Builder 檢視 TablePanel

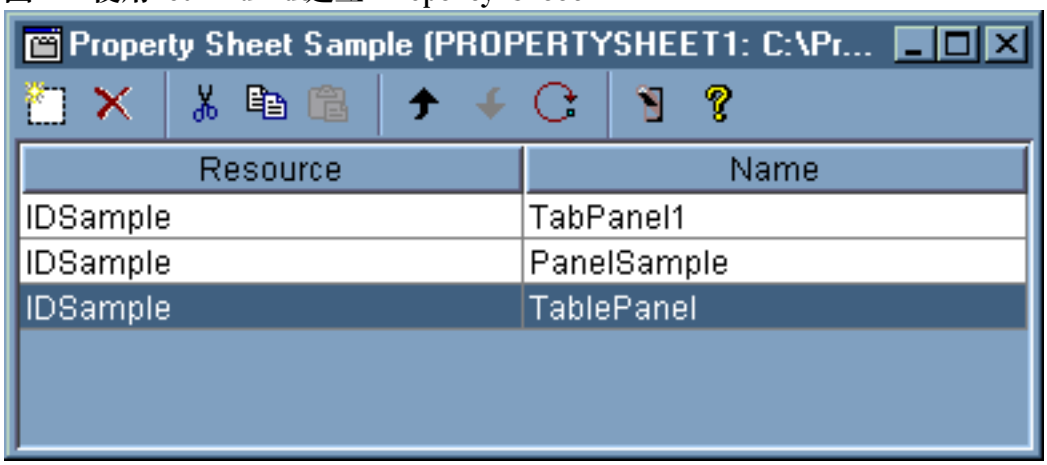

Deck Pane
View
Table with custom cell editor
Name
CheckBox Column
Sample Name

使用 GUI Builder 建立 Property Sheet (內容表)

GUI Builder 使得建立 Property Sheet 更加容易。在 GUI Builder 主 視窗的功能表列上，選取 File。

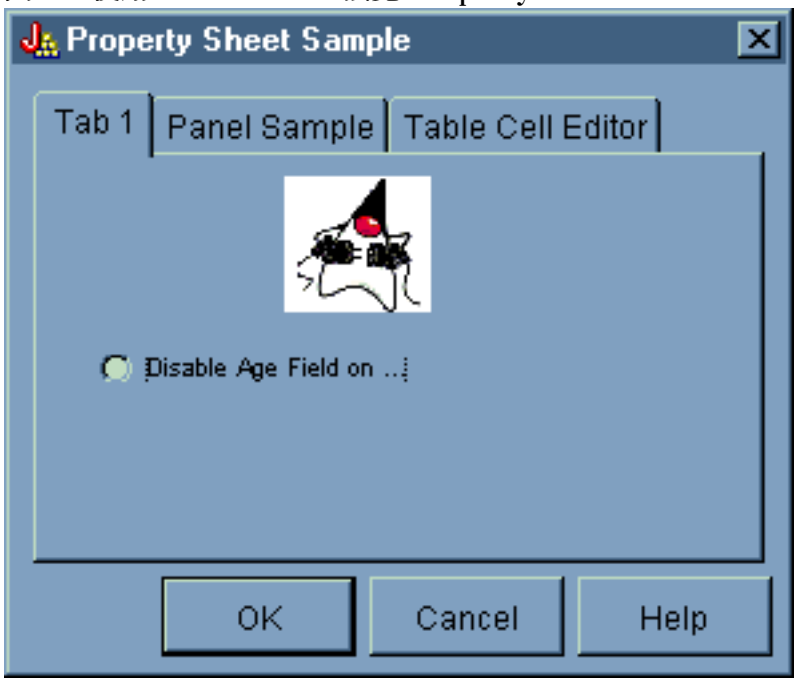
在 GUI Builder 的主視窗的功能表列上， 按一下 Insert Property Sheet 圖示 即顯示出畫面建置器，您可在其中插入內容表的元件。

圖 1：使用 GUI Builder 建立 Property Sheet



建立 Property Sheet 之後，使用 圖示 便可以預覽它。就此例而言，您可從三個標籤中選擇：

圖 2：使用 GUI Builder 預覽 Property Sheet

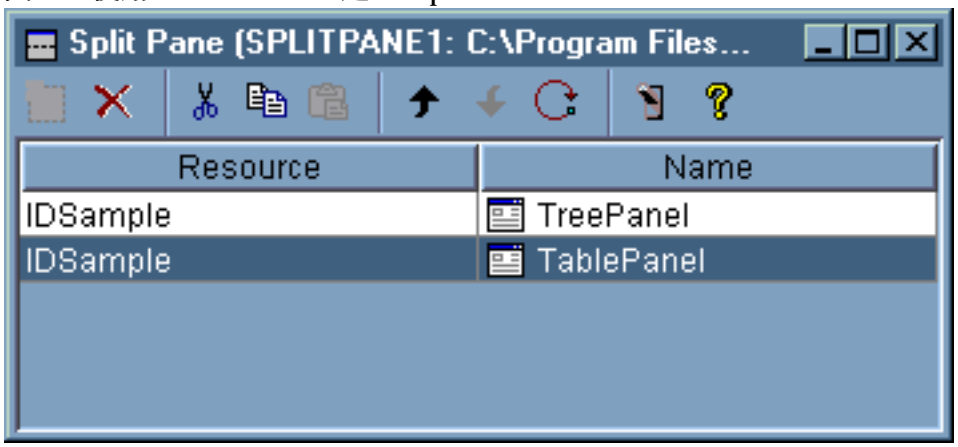


使用 GUI Builder 建立 Spilt Pane (分隔視窗)

GUI Builder 可以讓您輕鬆地建立 Spilt Pane。在 GUI Builder 主視窗的功能表列上，選取 File > New File。

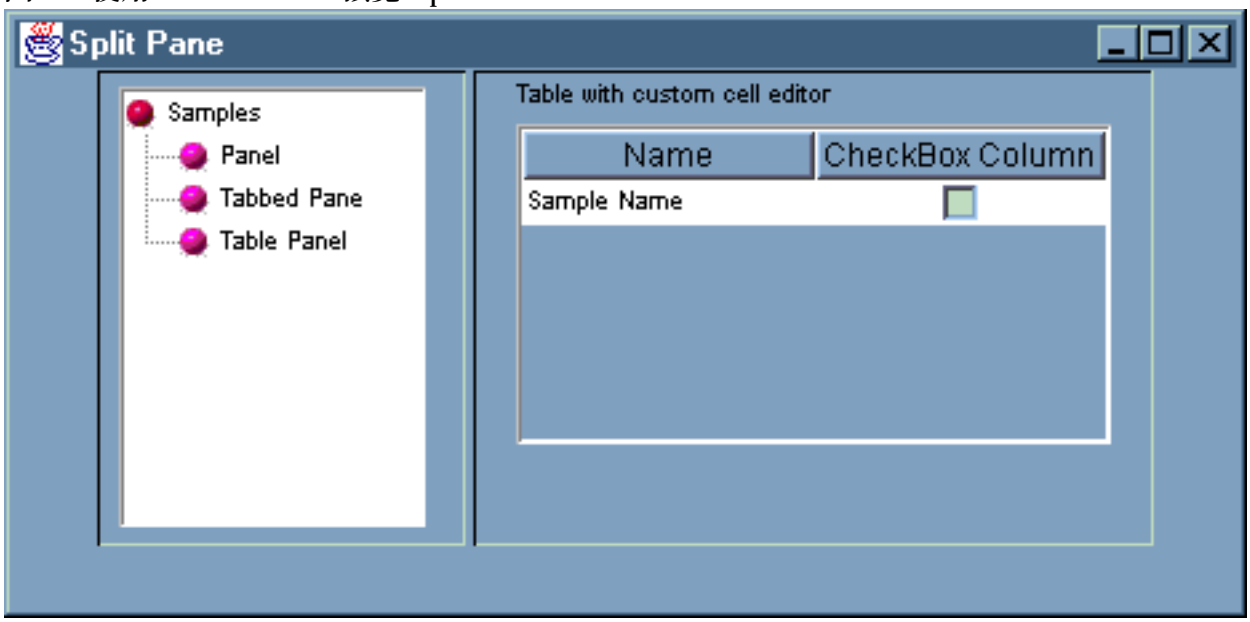
在 GUI BuilderFile視窗的功能表列上，按一下 Insert Split Pane 工具按鈕 以顯示畫面建置器，您就可以將想要的元件插入分割窗格中。在下列範例中，會新增兩個元件。

圖 1：使用 GUI Builder 建立 Split Pane



建立了分割窗格後，請按一下 Preview工具按鈕 圖示， 來預覽 Split Pane，如圖 2 中所示。

圖 2：使用 GUI Builder 預覽 Split Pane

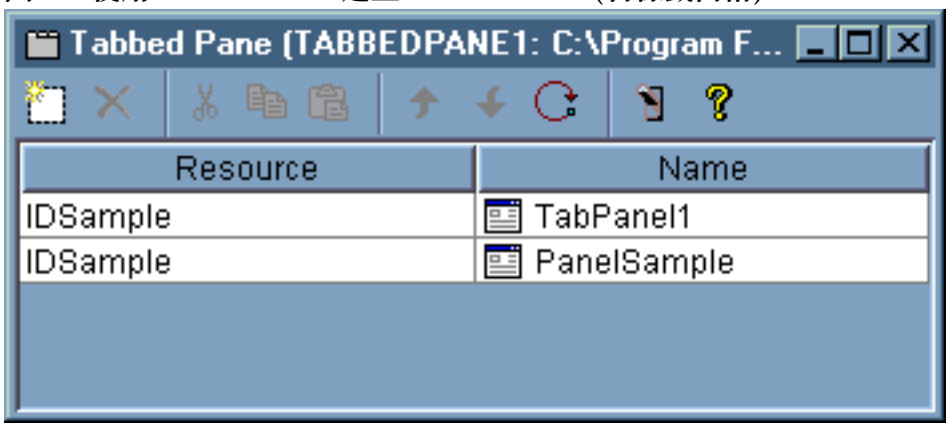


使用 GUI Builder 建立 Tabbed Pane (含標籤窗格)

GUI Builder 可以讓您輕鬆地建立一個含標籤窗格。在 GUI Builder 主視窗的功能表列上，選取File--New File。

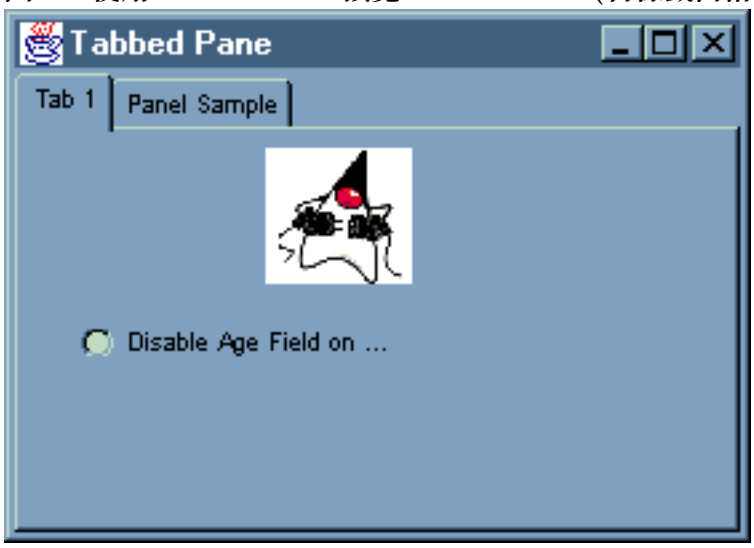
在 GUI BuilderF的ile視窗的功能表列上， 按一下 Insert Tabbed Pane 圖示，
以顯示畫面建置器，您就可以將想要的元件插入含標籤窗格中。在下列範例中，會新增兩個元件。

圖 1：使用 GUI Builder 建立 Tabbed Pane (含標籤窗格)



建立了含標籤窗格後，請按一下 Preview 工具按鈕 來預覽含標籤窗格。

圖 2：使用 GUI Builder 預覽 Tabbed Pane (含標籤窗格)

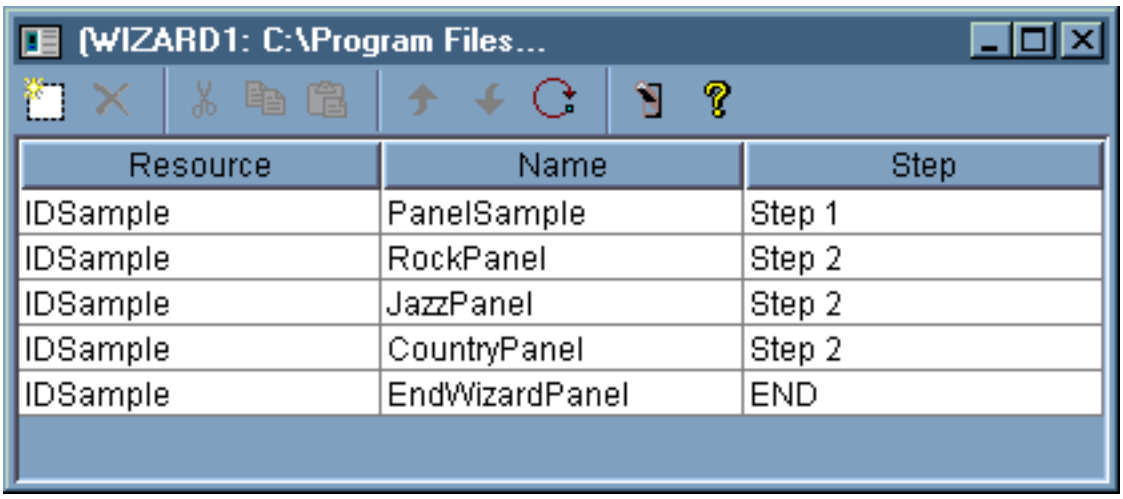


使用 GUI Builder 建立 Wizard (精靈)

GUI Builder 讓建立精靈介面變得簡單無比。從 GUI Builder 視窗的功能表列中，選取 File --> New File。

在 GUI Builder File 視窗的功能表列上，按一下 Insert Wizard 工具列按鈕  來顯示畫面建置器，您可以在該建置器上將畫面新增到精靈中。

圖 1：以 GUI Builder 建立 Wizard



建立好精靈之後，請使用 Preview 工具按鈕  來預覽它。 圖 2 顯示了這個範例中所出現的第一個畫面。

圖 2：以 GUI Builder 預覽第一個精靈畫面

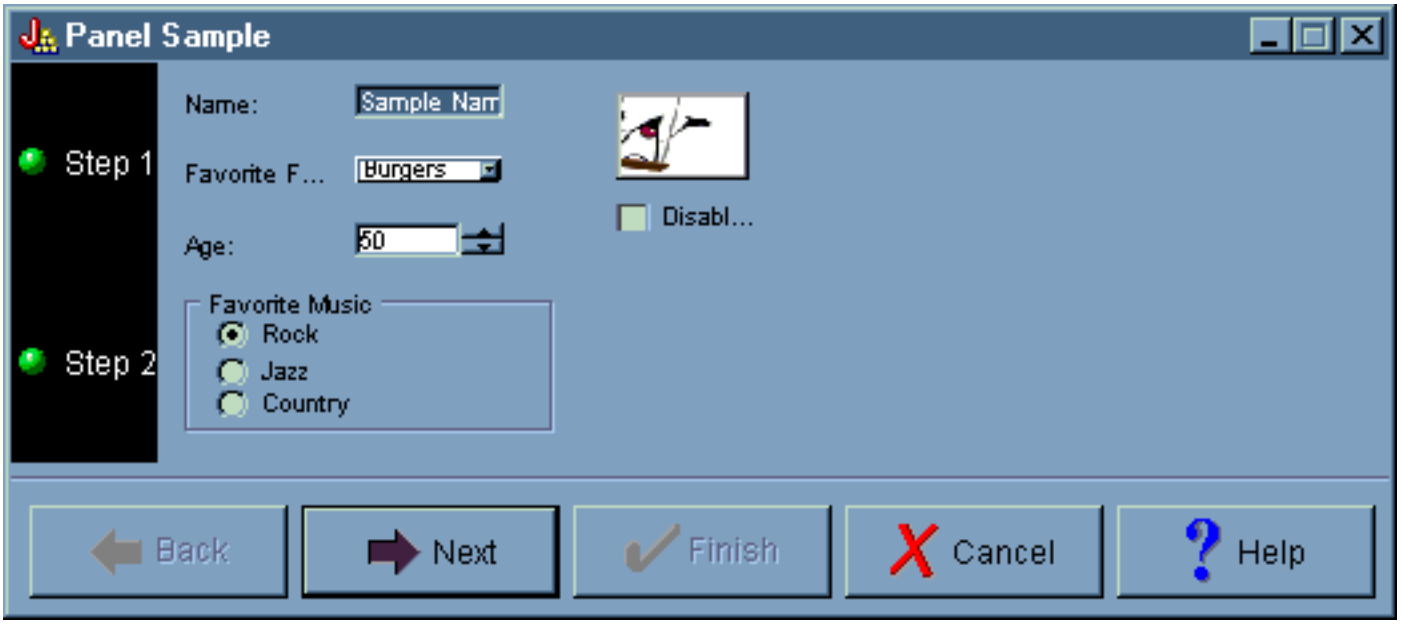
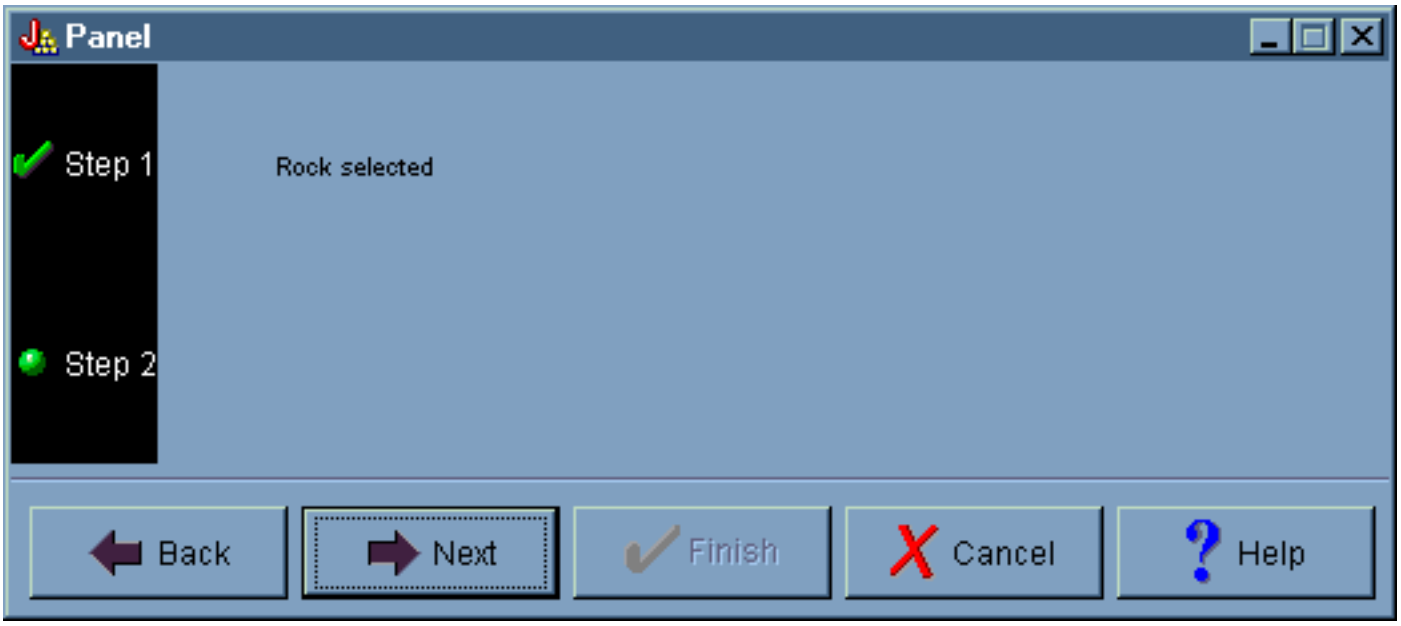


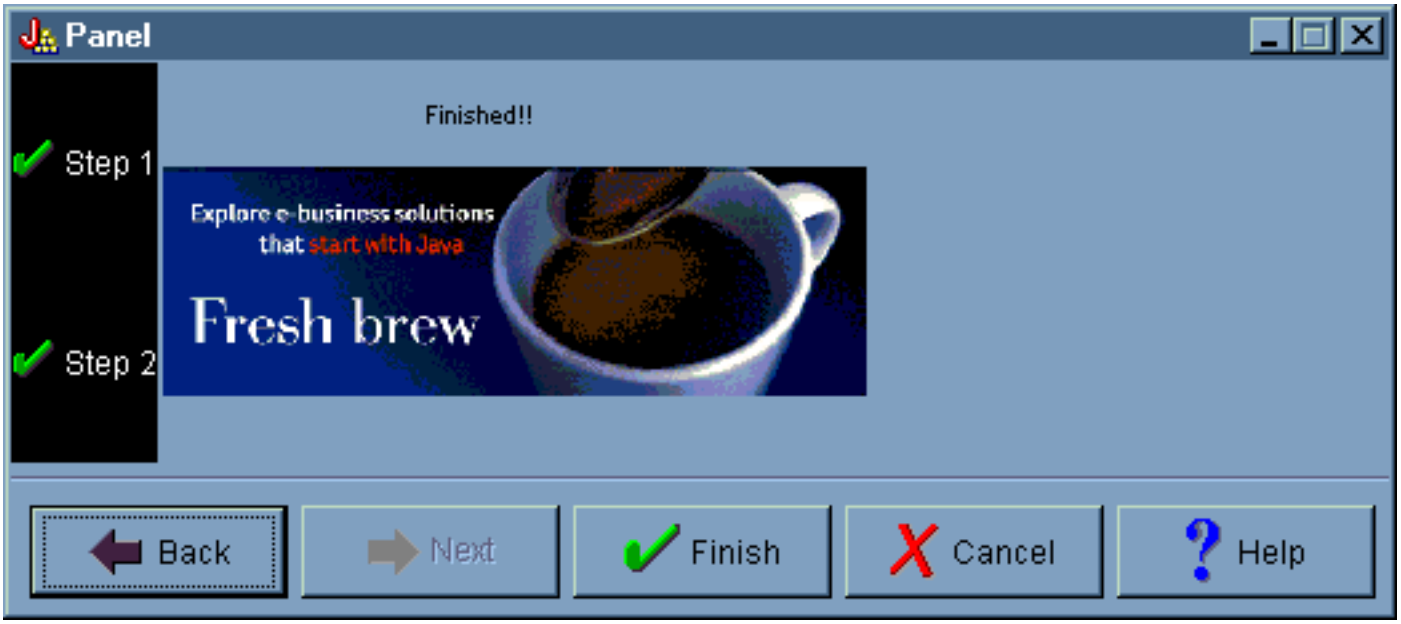
圖 2 顯示了當使用者選取 Rock 並按一下 Next 時，所顯示的第二個畫面。

圖 2：以 GUI Builder 預覽第二個精靈畫面



在第二個精靈畫面按 Next 來顯示最後一個精靈畫面，如「圖 3」所示。

圖 3：以 GUI Builder 預覽最後一個精靈畫面

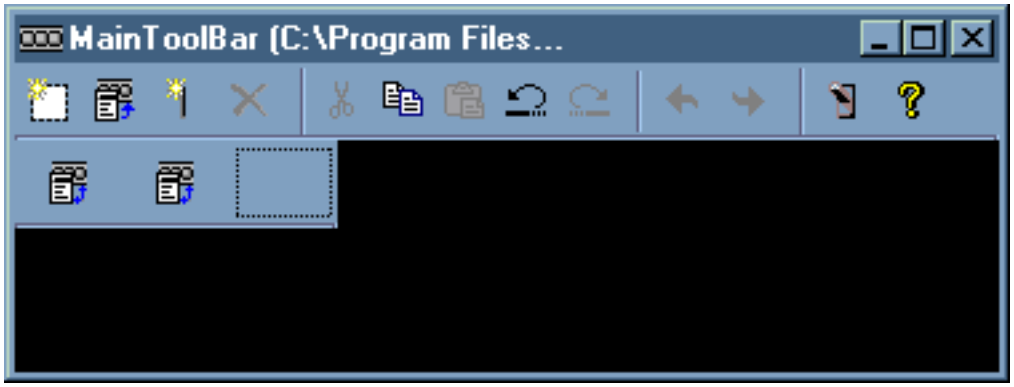


使用 GUI Builder 建立 Toolbar (工具列)

GUI Builder 可以讓您簡單地建立一個工具列。從 GUI Builder 視窗的功能表列中，選取 File > New File。

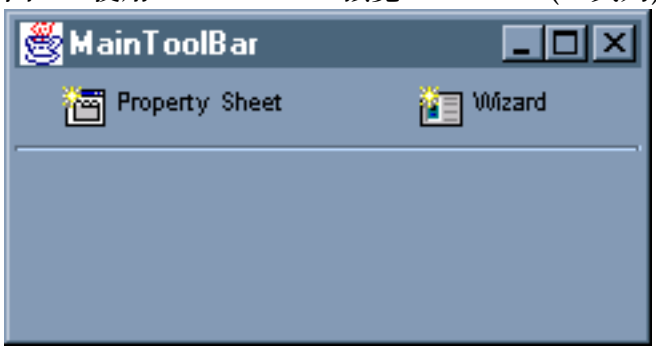
在 GUI Builder File 視窗的功能表列上，按一下 Insert Tool Bar 工具按鈕，以顯示畫面建置器，您就可以將想要的元件插入工具列中。

圖 1：使用 GUI Builder 建立工具列 (Toolbar)



建立好工具列後，請按一下 Preview 工具按鈕 來預覽工具列。 就此範例而言，您可以選擇要顯示 Propery Sheet 或 Wizard。

圖 2：使用 GUI Builder 預覽 Tool Bar (工具列)

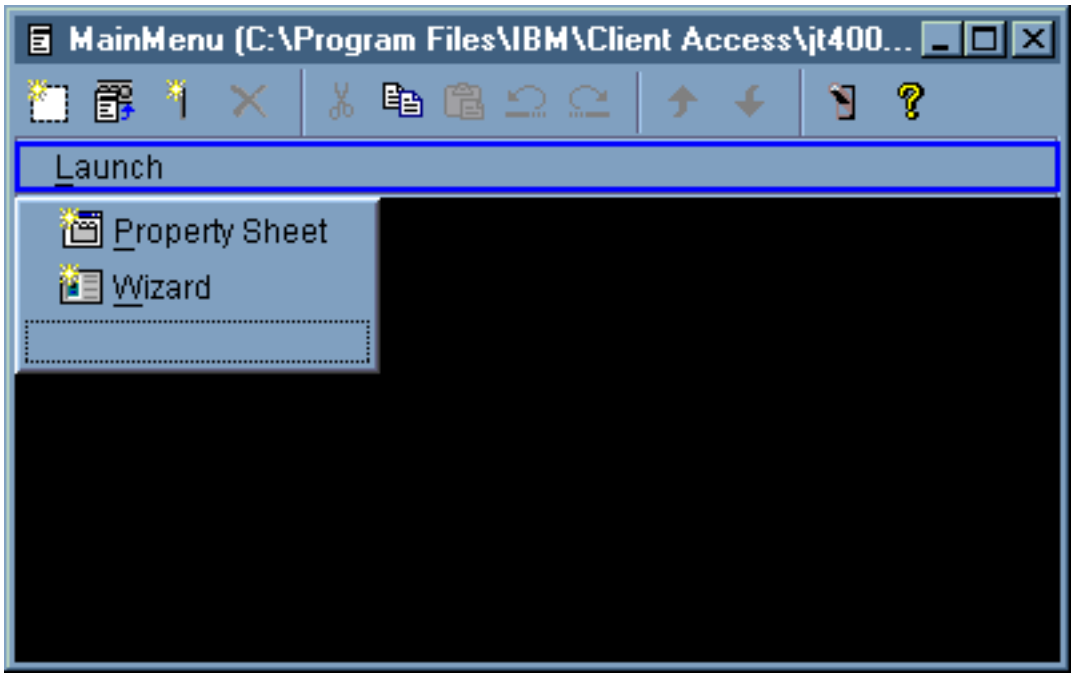


使用 GUI Builder 建立 Menubar (功能表列)

GUI Builder 可使功能表列的建立方便又容易。從 GUI Builder 視窗的功能表列中，選取 File -> New File。

在 GUI Builder File 視窗的工具列中，按一下 Insert Menu 工具按鈕，建立可於其中插入功能表元件的畫面建置器。

圖 1：GUI 建置器：建立功能表



功能表建立後，使用 Preview 工具按鈕 加以預覽。 針對此範例，您可從新建立的 Launch 功能表選取 Property Sheet 或 Wizard。 下圖說明選取這些功能表項目時您會看見的情況。

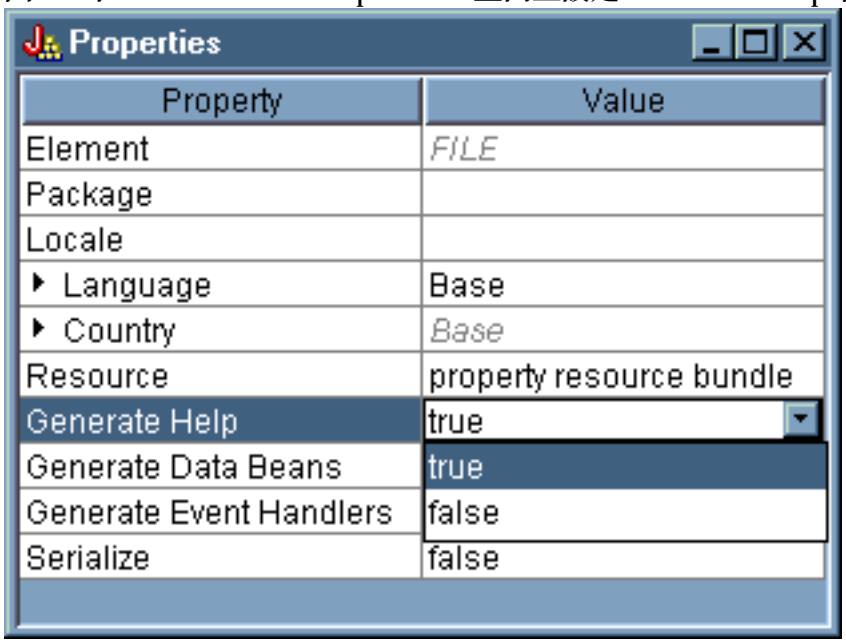
圖 2：GUI 建置器：檢視 Launch 功能表上的 Property Sheet

圖 3：GUI 建置器：檢視 Launch 功能表上的 Wizard

範例：建立 Help Document (說明檔)

使用 GUI Builder 建立說明檔是很簡單的事。在您使用的檔案內容畫面中，將 Generate help 設定為 true。

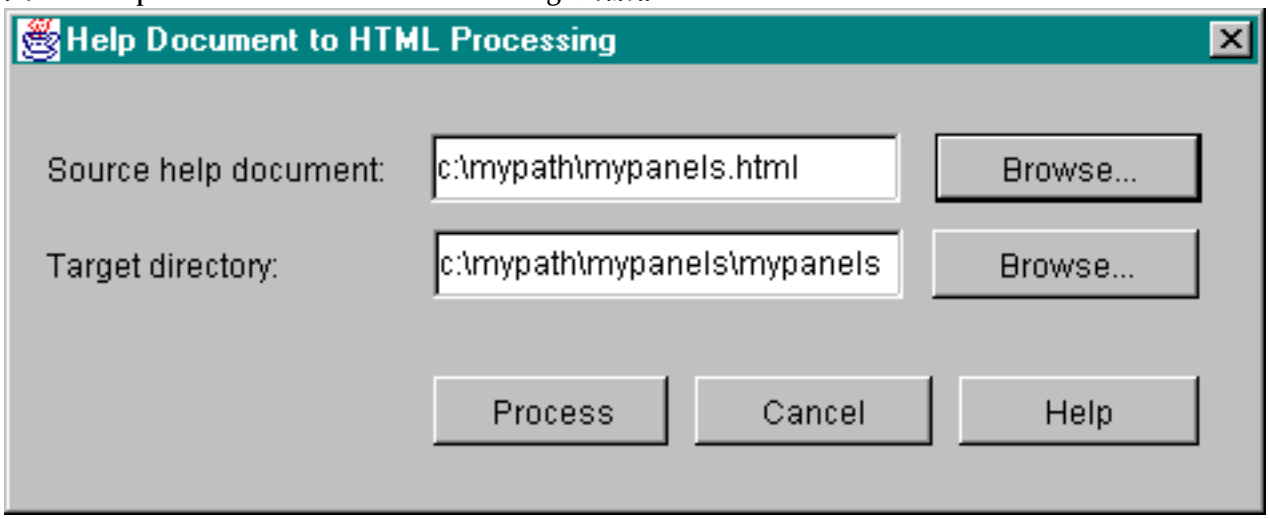
圖 1：在 GUI Builder Properties 畫面上設定 Generate Help 內容



GUI Builder 會建立一個稱為 Help Document 的 HTML 組織架構，您可以將它加編輯。

為了能在執行時使用，PDML 檔中的主題必須分成 個別的 HTML 檔。當您執行Help Document to HTML Processing時，主題會分成個別的檔案並放入在 Help Document 及 PDML 檔案後命名的子目錄中。執行環境希望個別 的 HTML 檔是在一個與 Help Document 及 PDML 檔具有相同名稱的子目錄中。Help Document to HTML Processing 對話會收集必要的資訊，並呼叫 HelpDocSplitter 程式以執行處理：

圖 2：Help Document to HTML Processing 對話框



在指令提示中鍵入下列，即會啟動 Help Document to HTML Processing：

```
jre com.ibm.as400.ui.tools.hdoc2htmViewer
```

執行此指令需要您類別路徑設定正確 。

如欲使用 Help Document to HTML Processing, 您要先選取具有與 PDML 檔相同名稱 的 Help Document。其次, 您要指定一個子目錄, 其名稱與 Help Document 及 PDML 檔 相同, 以供輸出之用。 選取 Process 以完成處理程序。

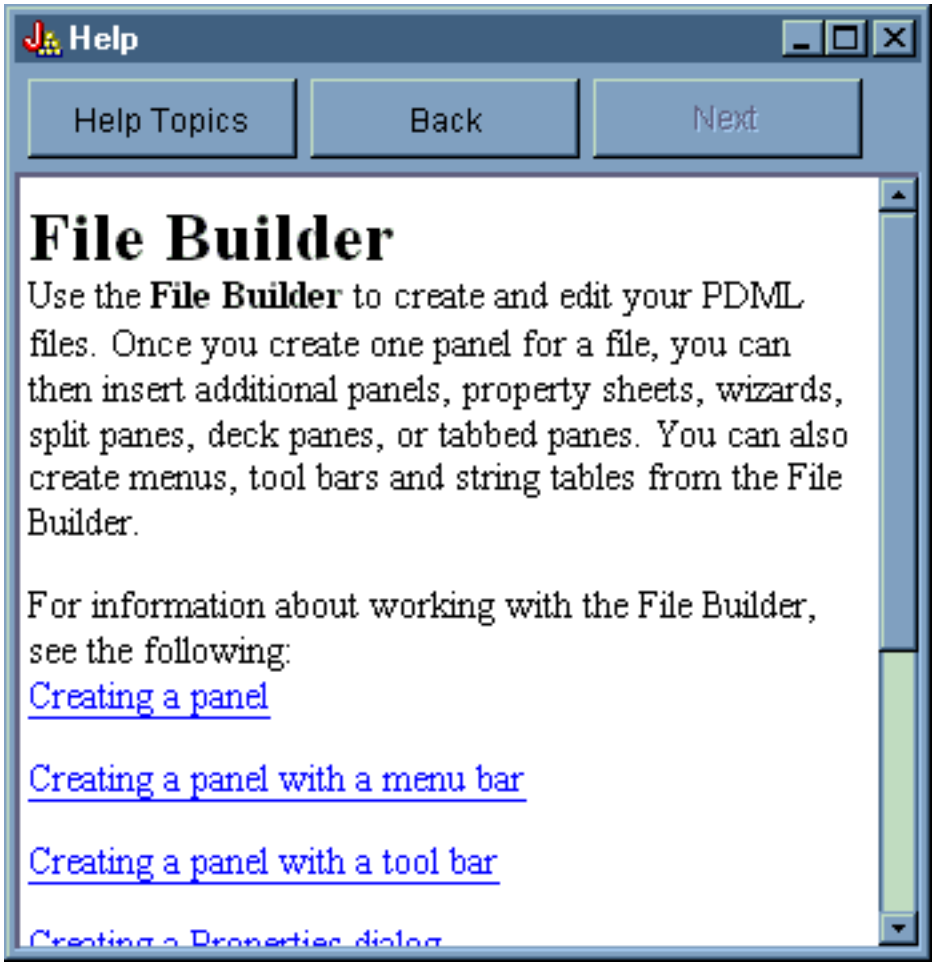
您可以在指令行中利用下列指令來分割說明文件：

```
jre com.ibm.as400.ui.tools.HelpDocSplitter "helpdocument.htm" [output
directory]
```

此指令會執行分隔檔案的處理程序。您提供作為輸入使用的 Help Document 名稱, 以及選用的輸出目錄。就預設值而言, 會建立與 Help Document 相同名稱的子目錄, 且結果檔會放入該目錄 中。

這是您看到的說明檔的範例：

圖 3：GUI Builder 說明檔範例

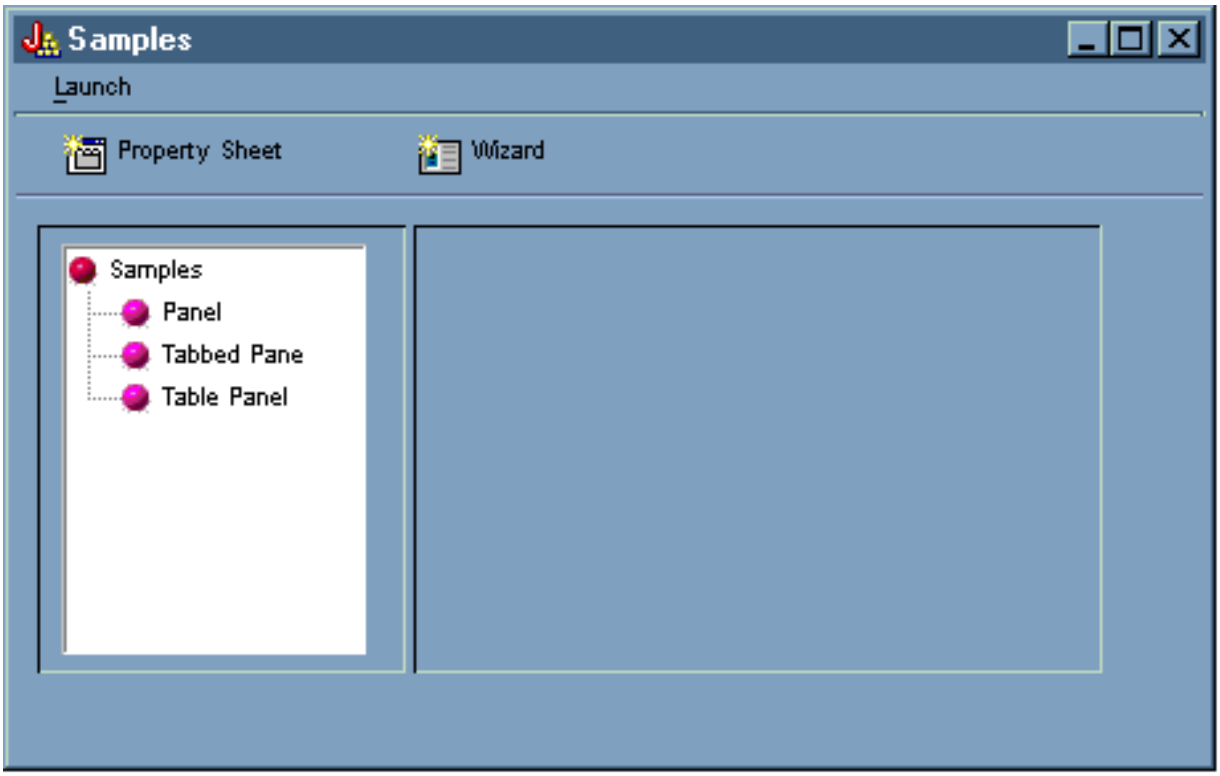


範例：使用 GUI Builder

當本區段所包含的範例與幕後運作的正確資料 bean 放在一起時， 您會取得一個完整的 GUI 應用程式。

圖 1 顯示出當您執行此範例時所顯示的第一個畫面。

圖 1：GUI Builder 範例主視窗



請注意這個螢幕可讓您使用動態畫面管理程式 圖 2 與圖 3 顯示您可以將視窗調整成較大或較小。

圖 2：調整 GUI Builder 範例主視窗的大小 (較大)

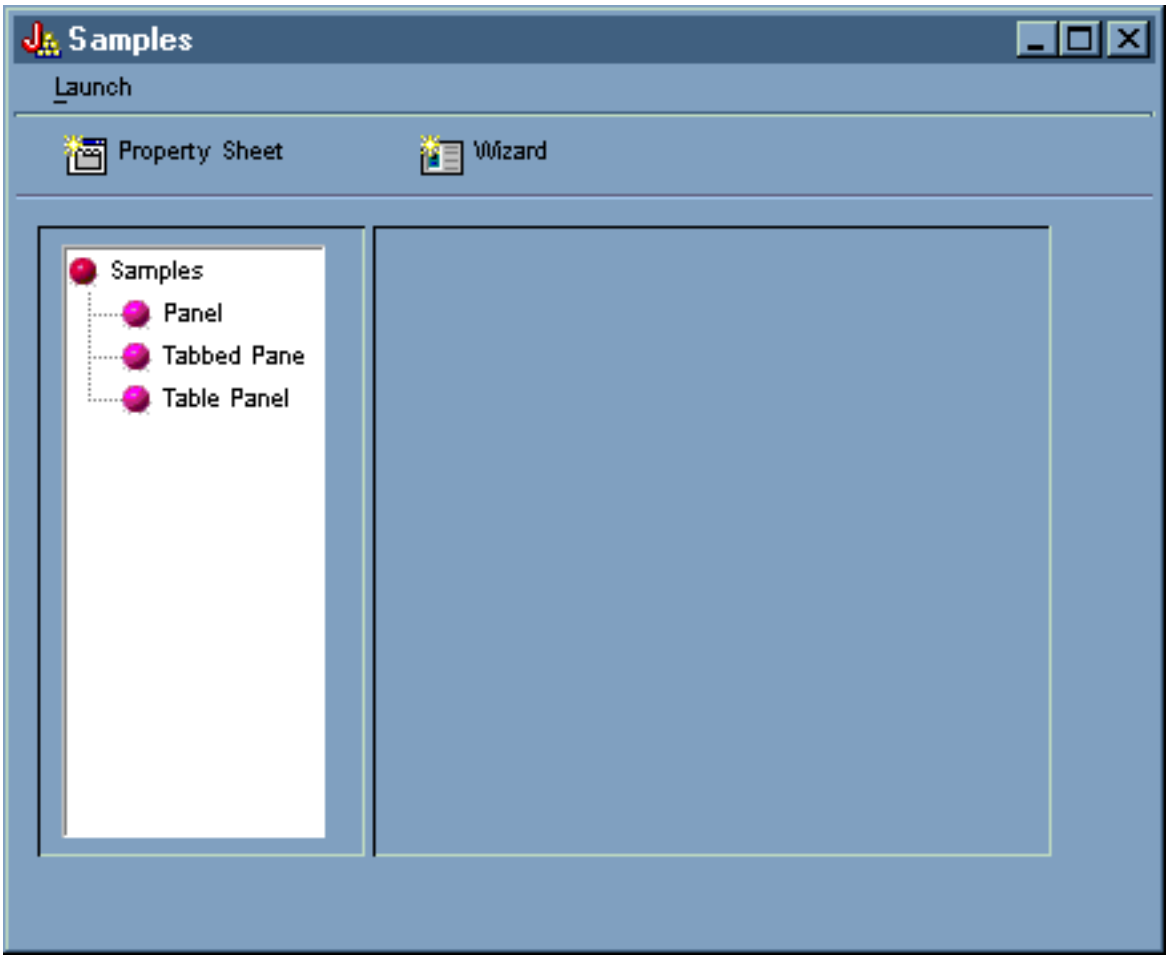
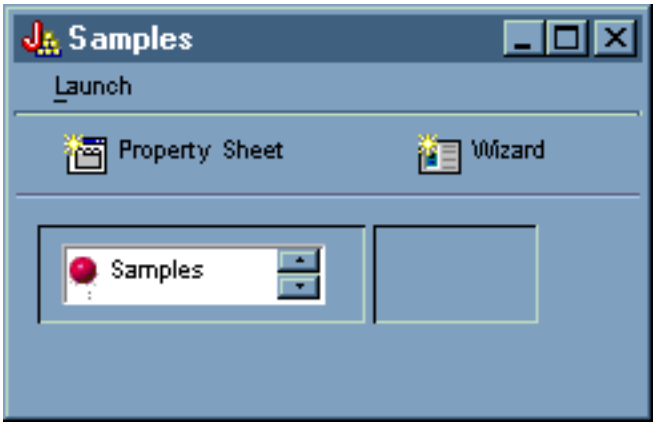


圖 3：調整 GUI Builder 範例主視窗的大小 (較小)



使用動態畫面管理程式時，文字大小並不隨著畫面以及畫面控制項大小變更。

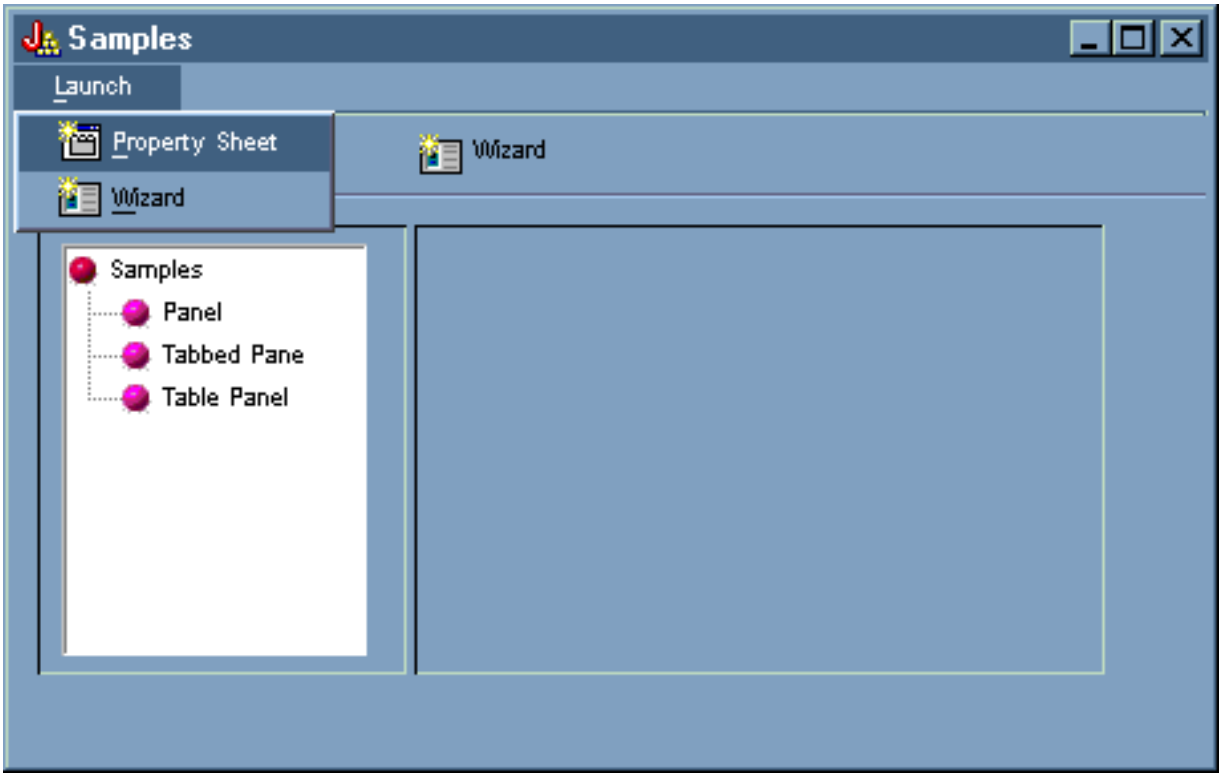
此畫面可讓您執行下列動作：

- 啟動 Property Sheet (內容表)
- 啟動 Wizard (精靈)
- 顯示左窗格中所列的範例

啟動 Property Sheet（內容表）

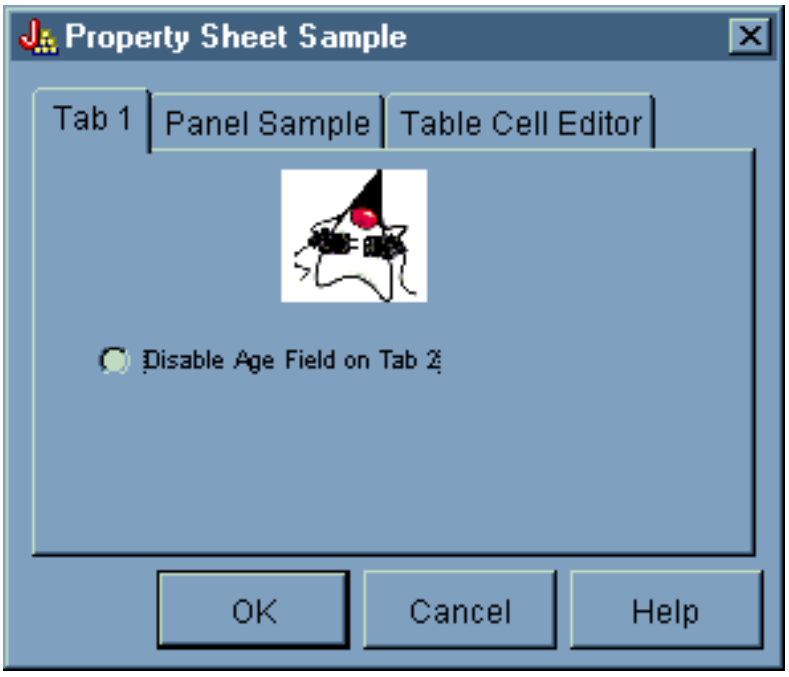
若要啟動內容表，可以按一下 Property Sheet（內容表）工具列按鈕，或使用 Launch 功能表。可以在工具列和功能表之間做選擇代表功能表項目之間的鏈結。圖 4 顯示從 GUI Builder 範例主視窗中之功能表選取 Property Sheet。

圖 4：從 Launch 功能表選取 Property Sheet



選取 Property Sheet 會顯示圖 5 中的畫面。

圖 5：Property Sheet Sample 對話框



Property Sheet Sample 第一次出現時，在預設的情況下會顯示 Tab 1。圖 6 與圖 7 顯示出當您選取其它標籤時，畫面顯示的變更情形。

圖 6：選取 Panel Sample 標籤

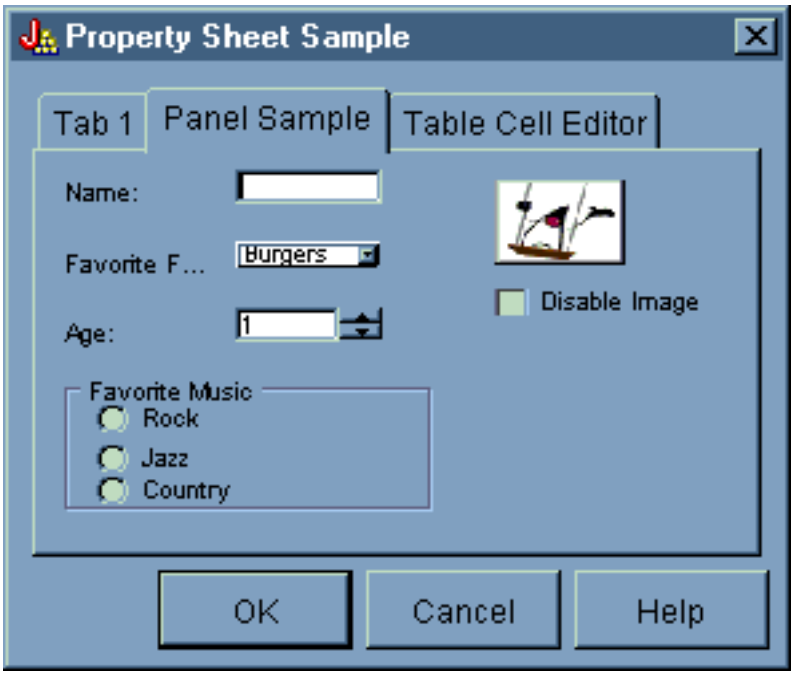
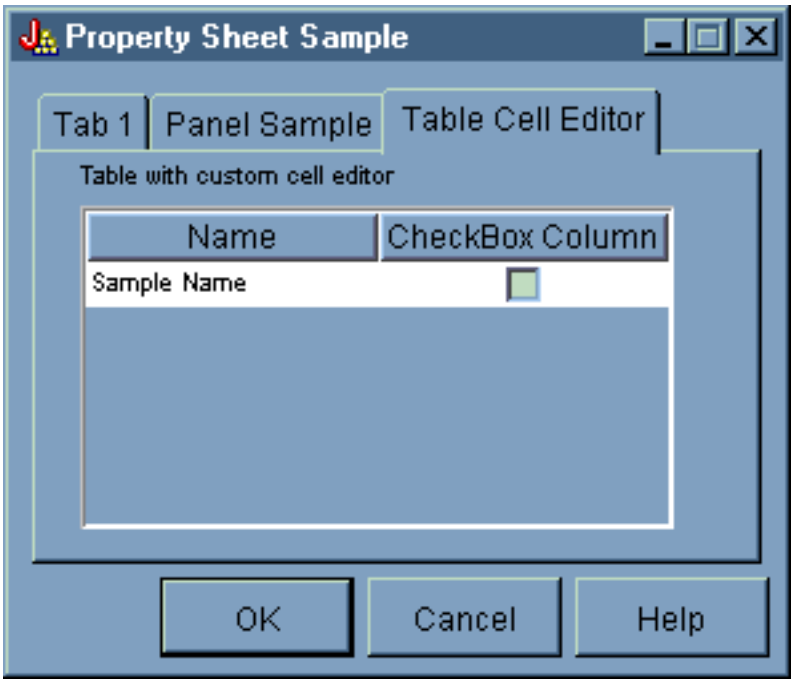


圖 7：選取 Table Cell Editor 標籤



啟動 Wizard（精靈）

若要啟動精靈，可以按一下 Wizard（精靈）工具列按鈕或使用Launch 功能表。
可以在工具列和功能表之間做選擇代表功能表項目之間的鏈結。圖 8 顯示從 GUI Builder 範例主視窗中之功能表選取Wizard。

圖 8：從 Launch 功能表選取 Wizard

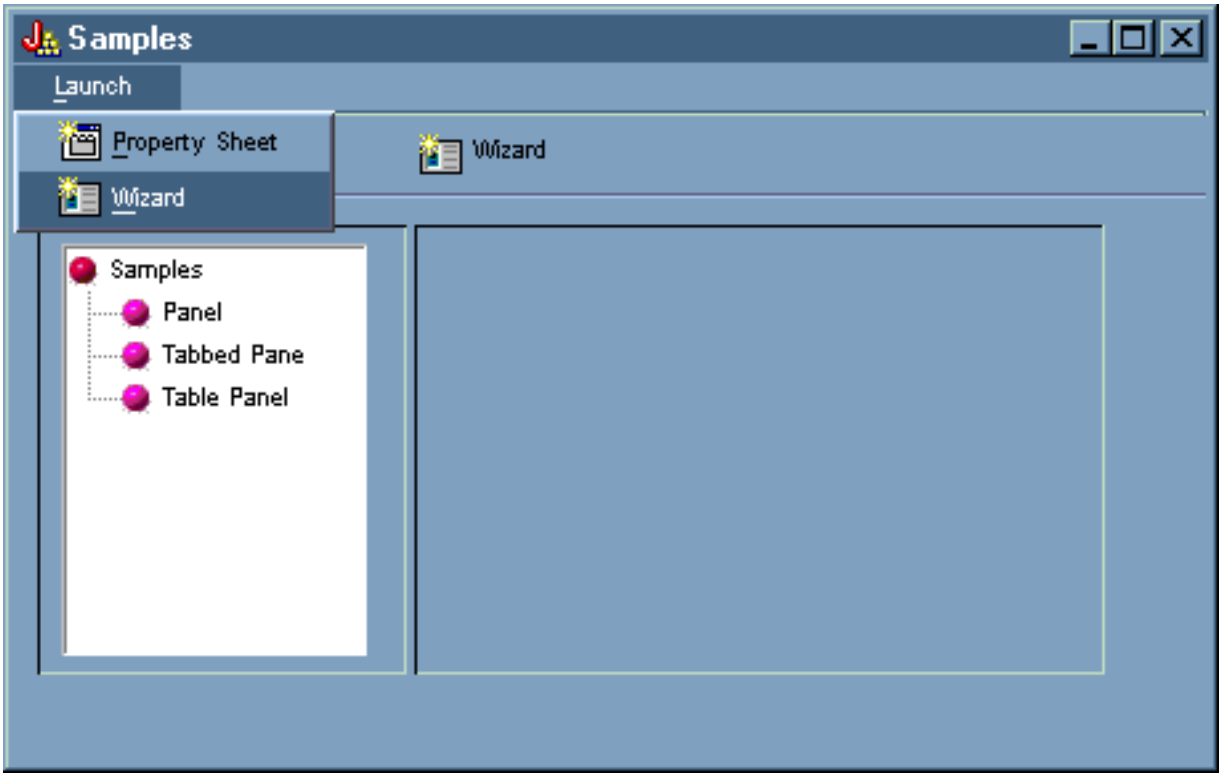
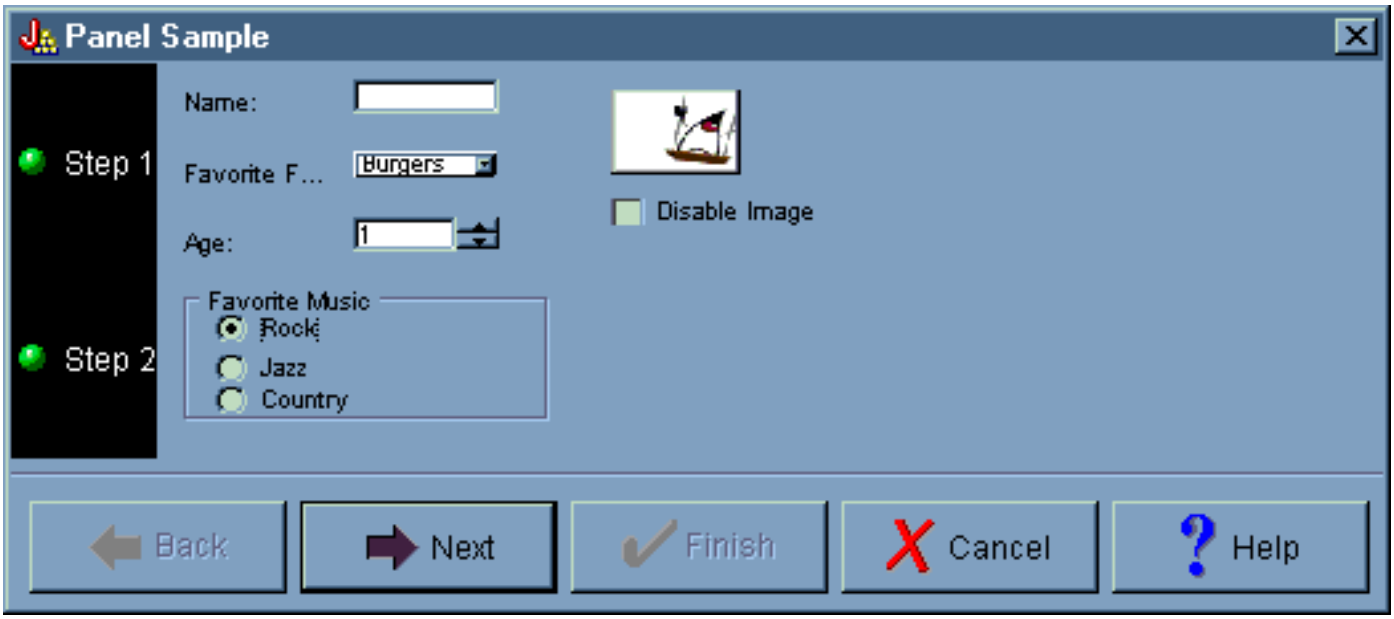


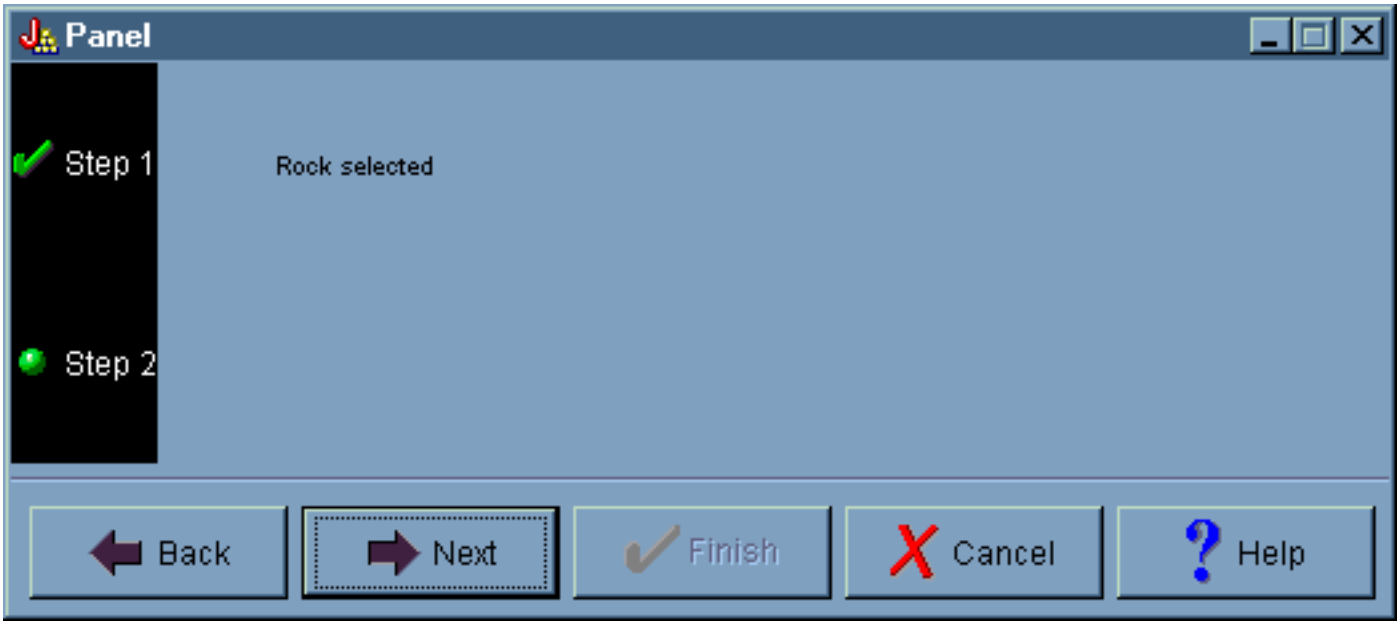
圖 9 顯示第一個精靈對話框提供的許多選項。

圖 9：選取第一個精靈對話框中的 Rock



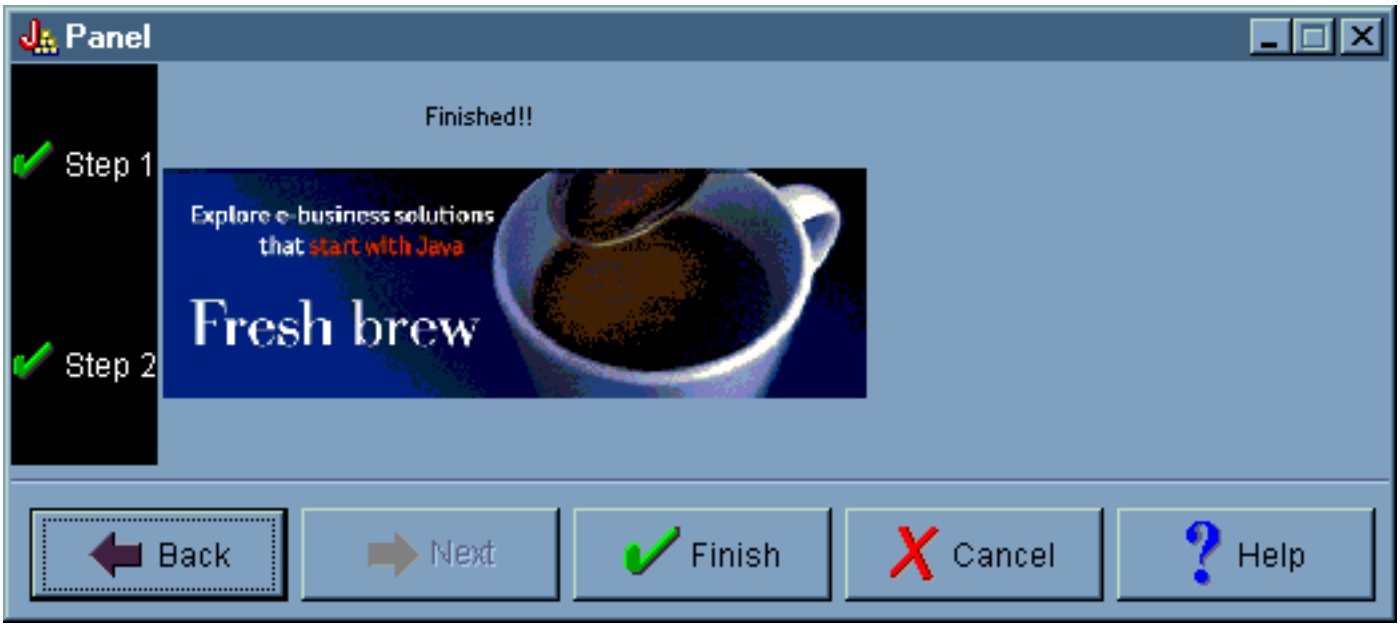
在第一個精靈對話框中，選取 Rock 並按一下 Next，以顯示如圖 10 中顯示的第二個精靈對話框。

圖 10：第二個精靈對話框（選取了 Rock 之後）



在第二個精靈對話框中，按一下 Next，以顯示如圖 11 中顯示的最後一個精靈對話框。

圖 11：最後一個精靈對話框



然而，此範例程式已被設計具迴圈功能。選取第一個精靈對話框（圖 12）中的 Country，然後按一下 Next，以顯示第二個精靈對話框（圖 13）。按一下第二個精靈對話框中的 Next，會遞迴至顯示第一個對話框（圖 14），而不是顯示最後一個精靈對話框。

圖 12：選取第一個精靈對話框中的 Country

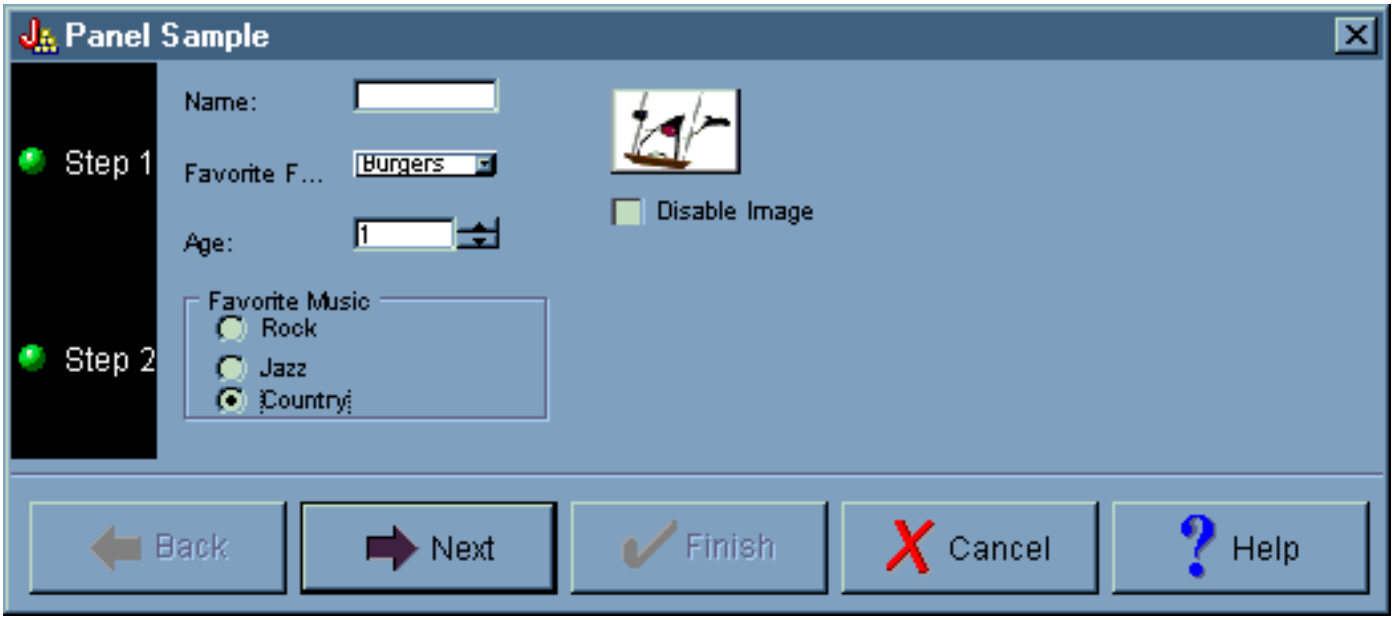


圖 13：第二個精靈對話框（選取了 Country 之後）

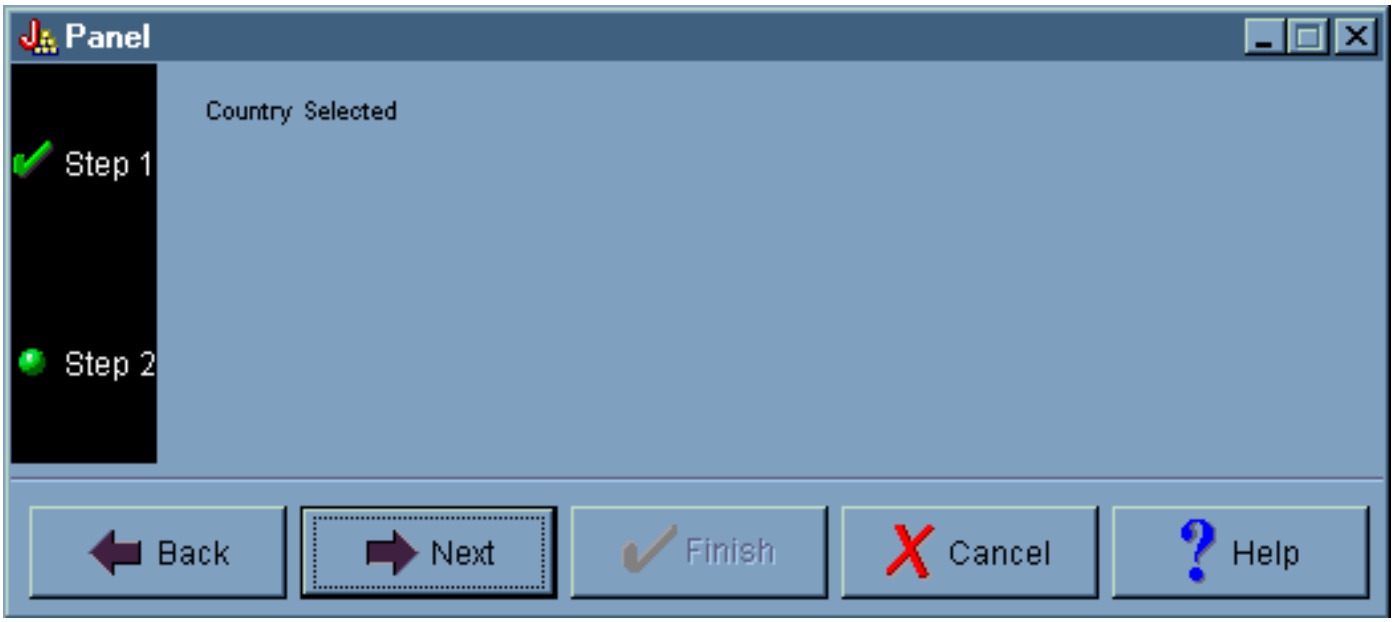
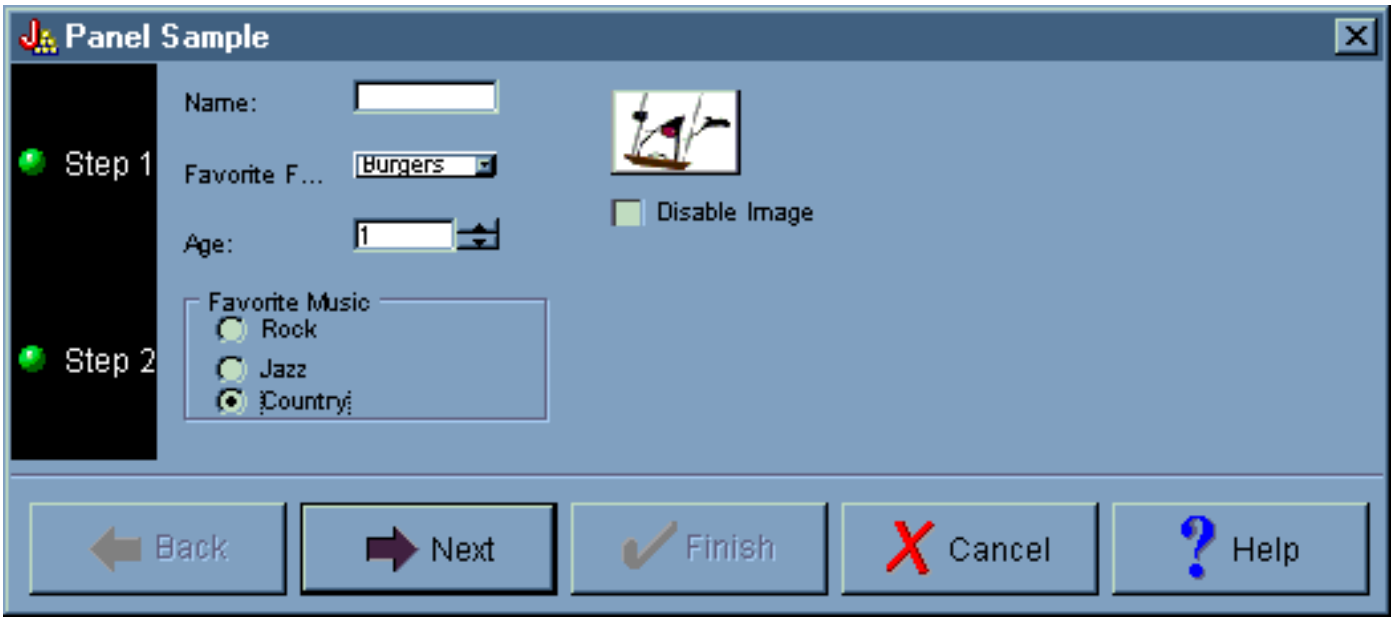


圖 14：遞迴至第一個精靈對話框

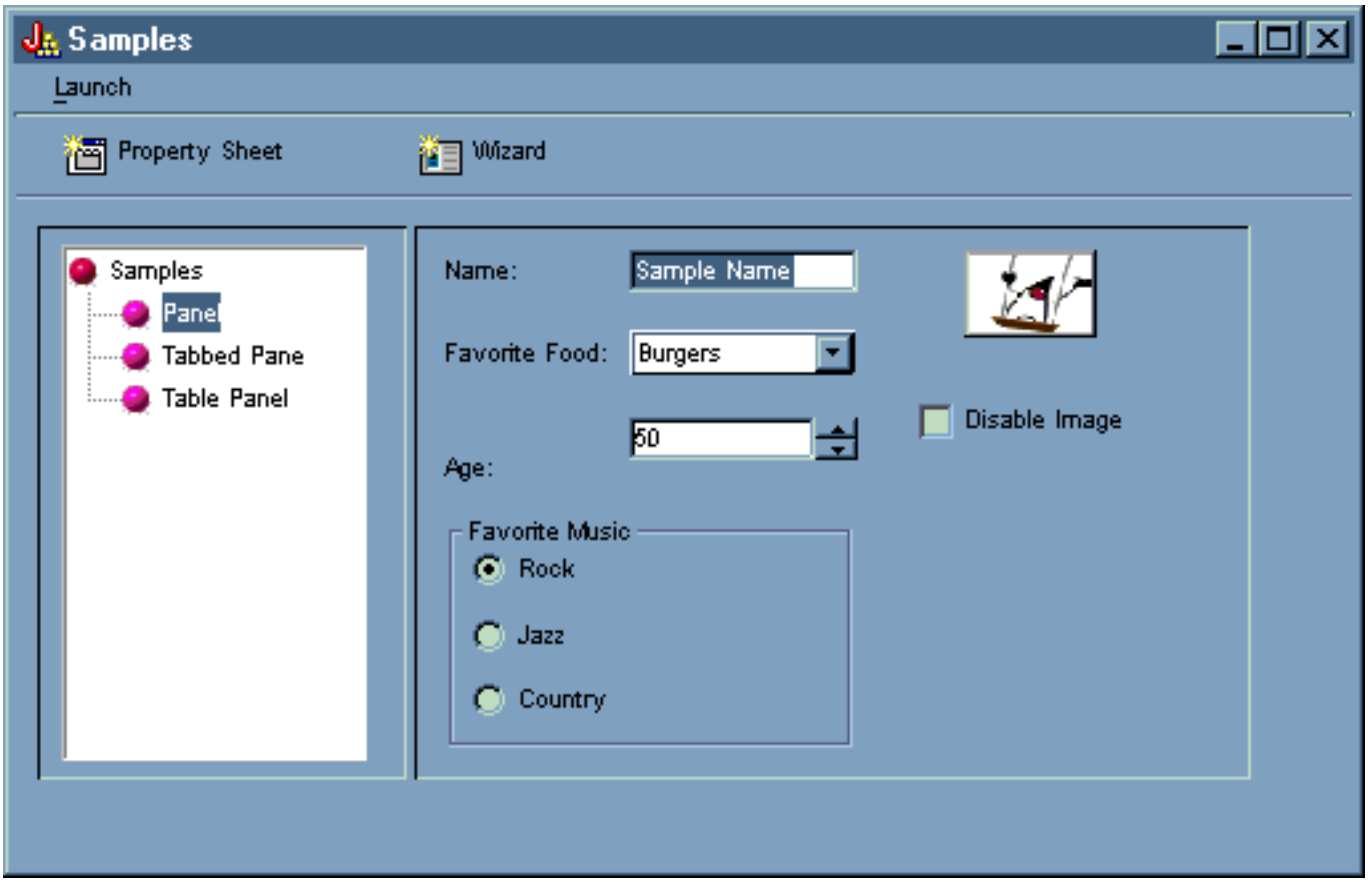


換句話說，程式設計師已經決定使用者不可選取鄉村音樂作為喜歡的音樂形態。

顯示範例

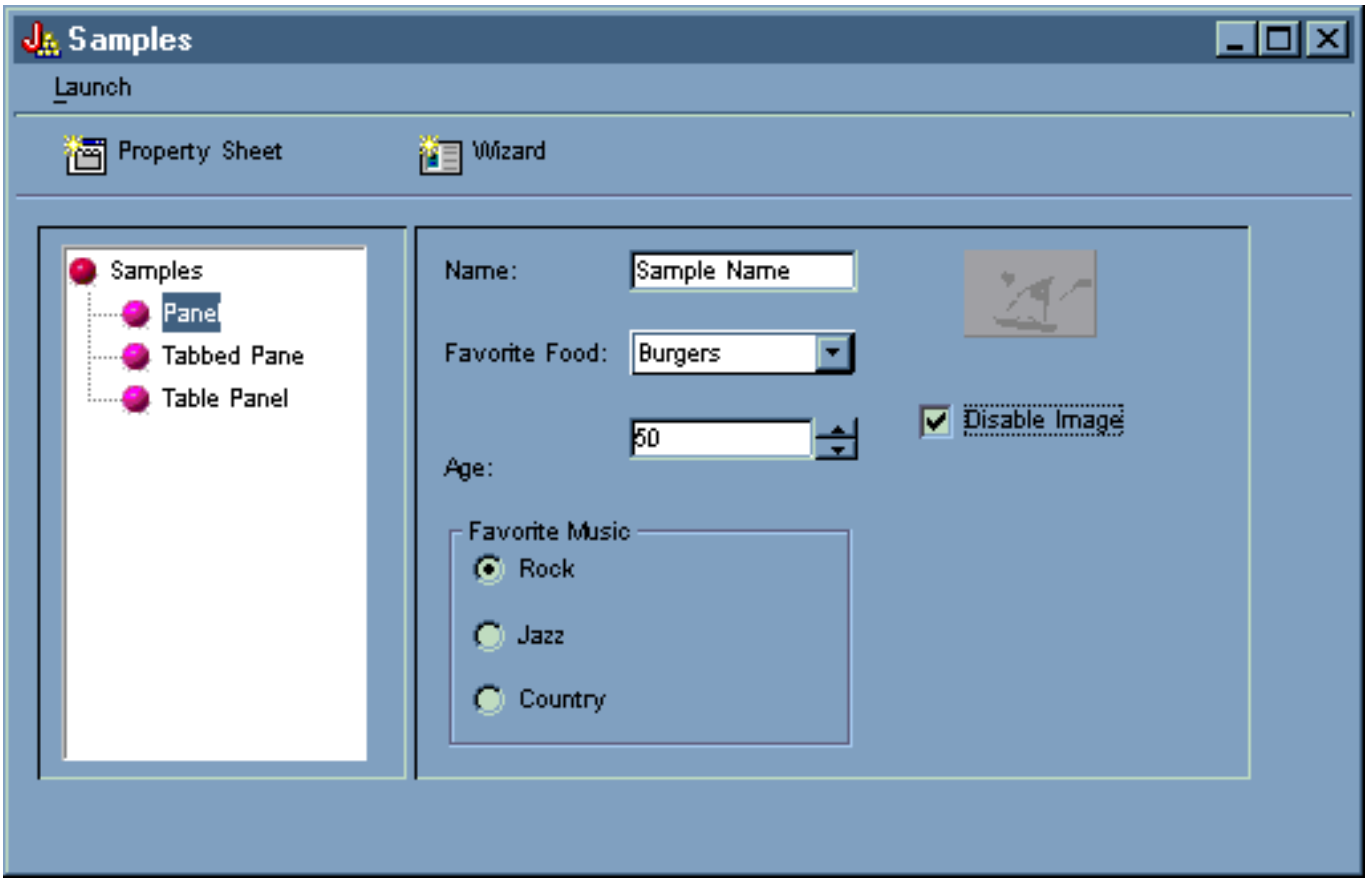
在 GUI Builder 的範例主視窗中，您也可以從工具列下面的左窗格選取其它的功能。 圖 15 顯示選取左窗格中的 會在右窗格中顯示 Panel 範例。

圖 15：選取左窗格中的 Panel



此 Panel 範例程式設計為具有停用影像的選項。選取 Image 可以顯示相同螢幕，但其影像會變成灰色，如圖 16 所示。

圖 16：選取右窗格中的 Disable Image



此 Panel 範例也說明了下拉清單方塊選項，如圖 17 中所示。

圖 17：在右窗格中，從 Favorite Food 清單中選取一個項目

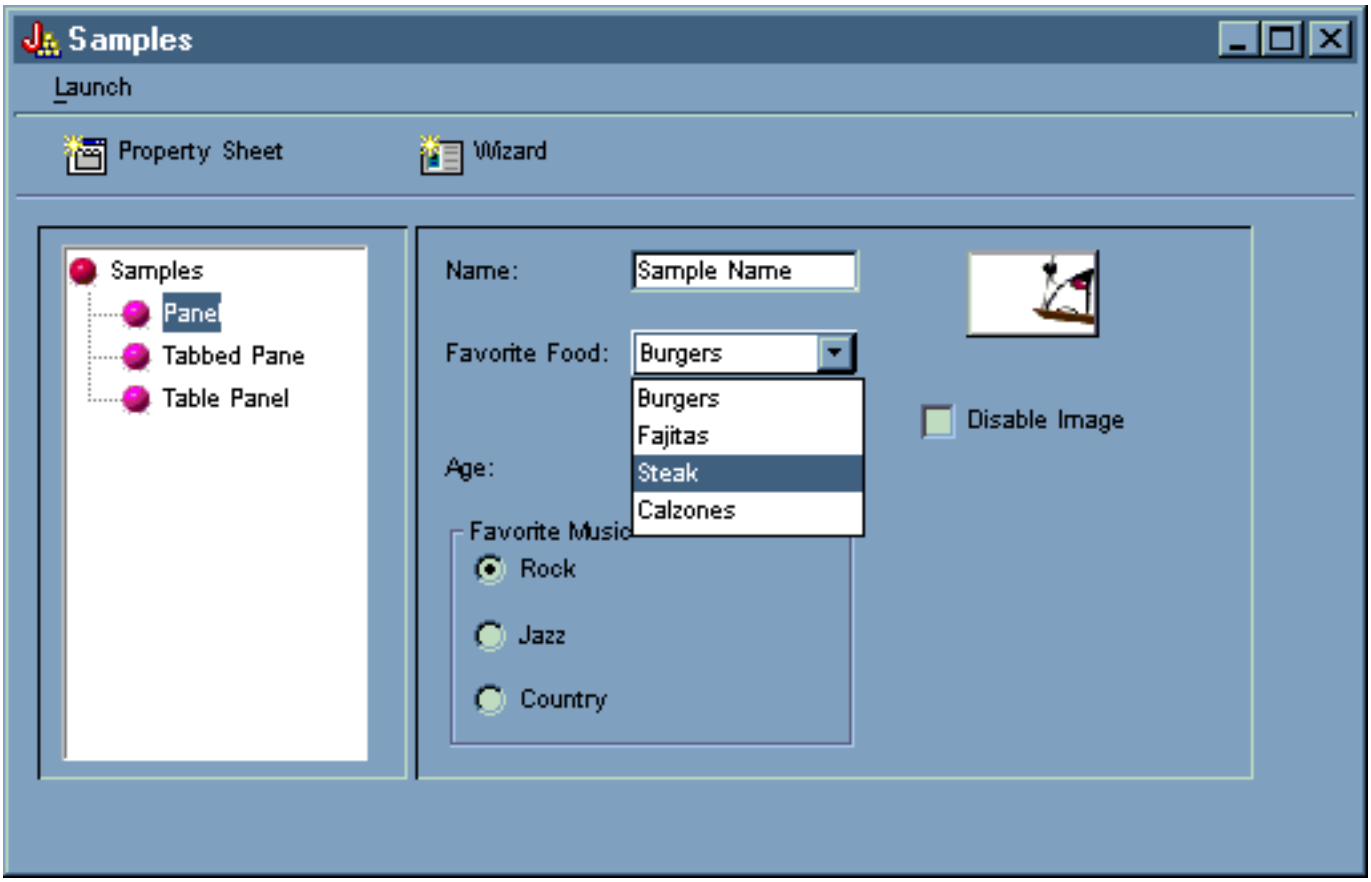


圖 18 顯示在 GUI Builder 範例主視窗的左窗格中選取 Tabbed Pane，會在右窗格中顯示 Tabbed Pane 範例。

圖 18：選取左窗格中的 Tabbed Pane

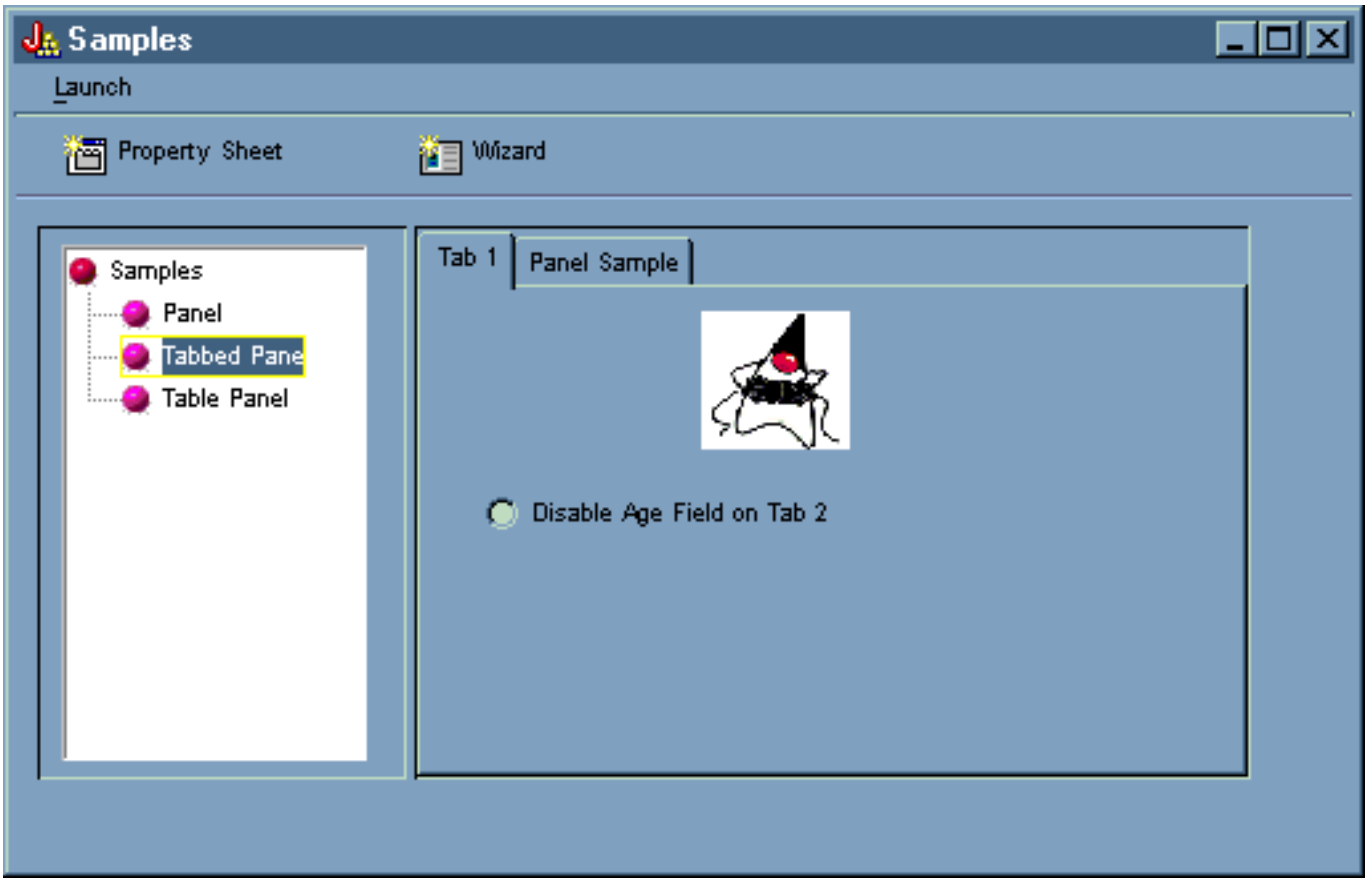
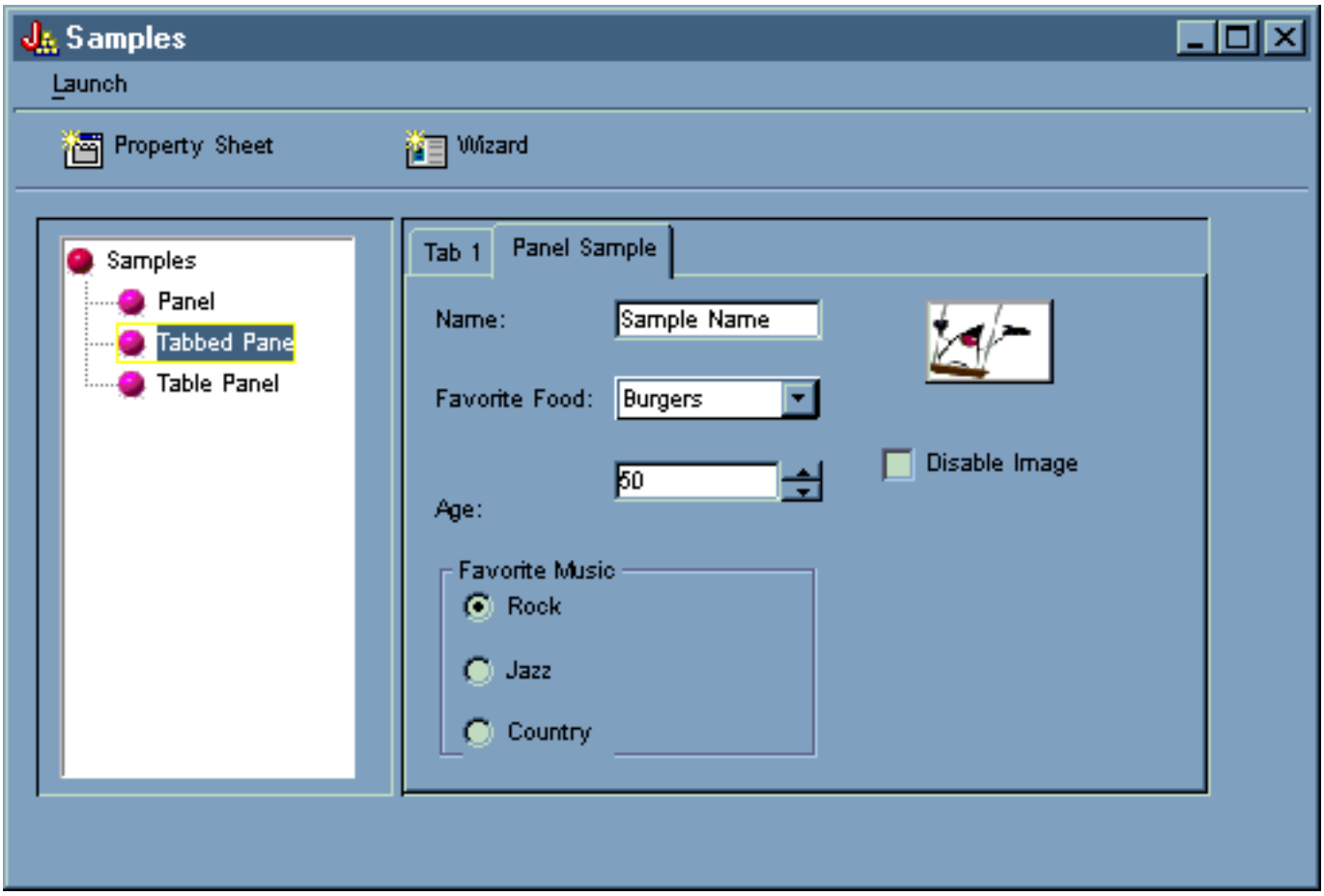


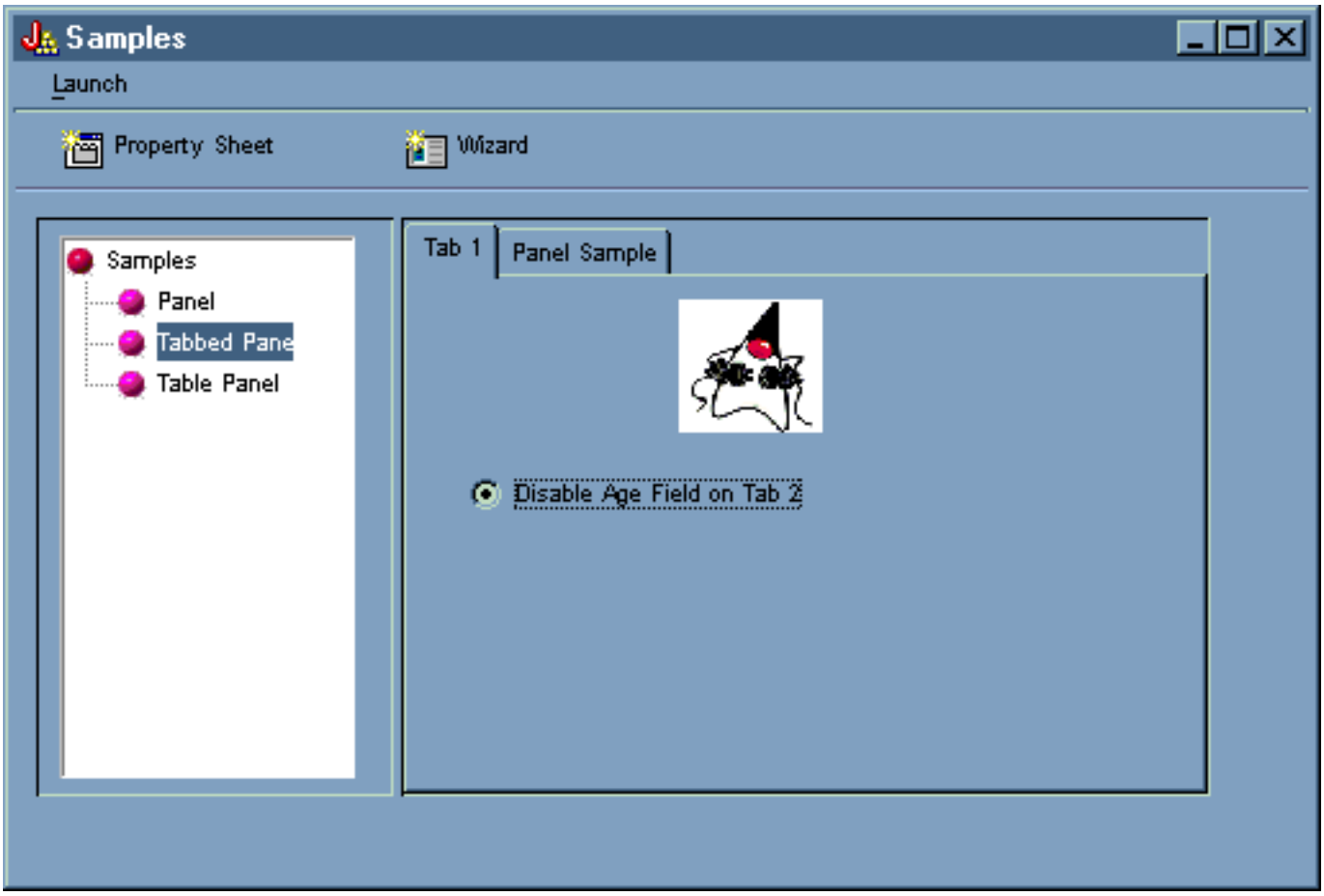
圖 19 顯示選取右窗格中的anel Sample標籤的結果。

圖 19：選取右窗格中的 Panel Sample 標籤



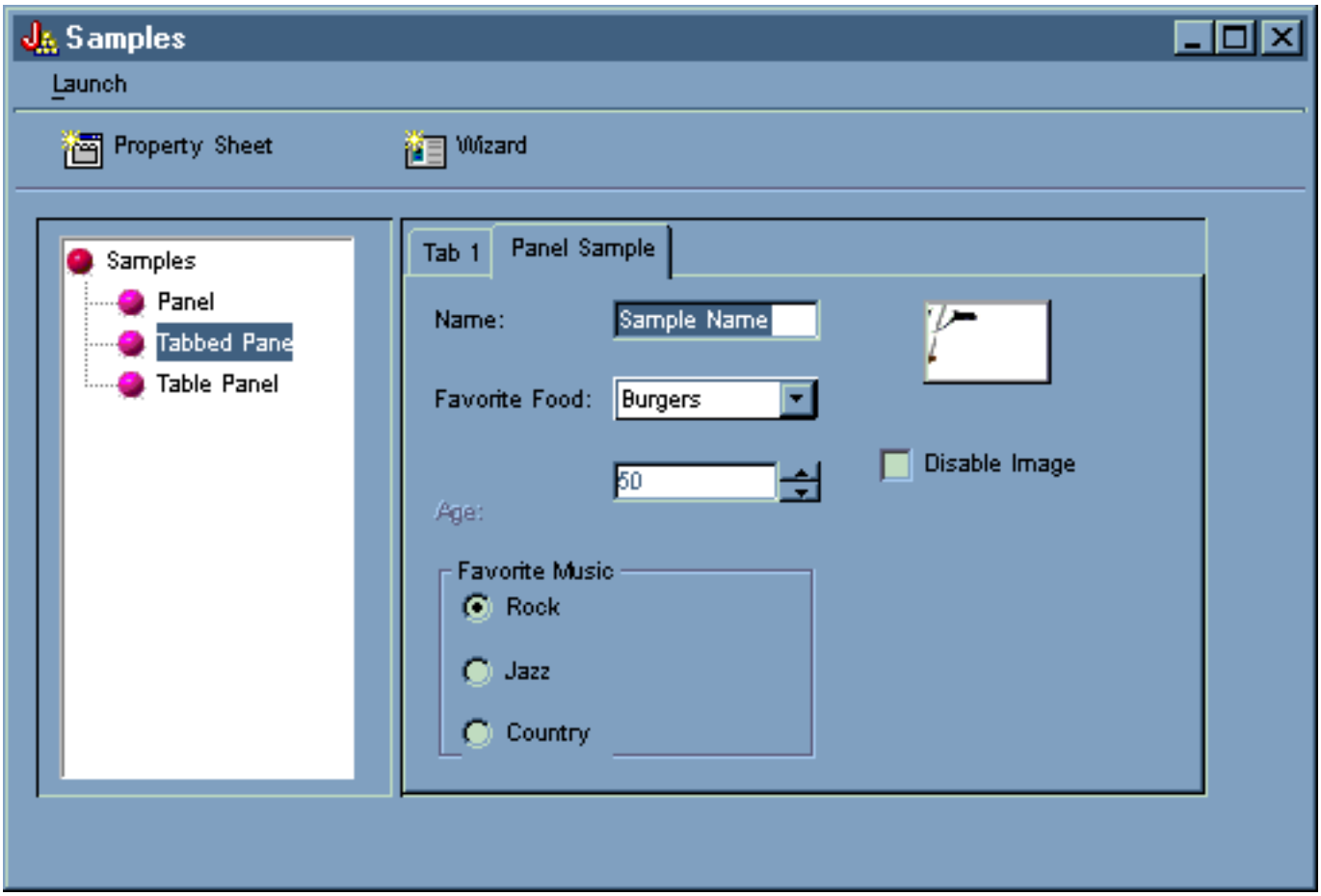
請再次選取Tab 1 (右窗格中)，然後按一下 Disable Age Field on Tab 2以取消選取 Tab 1。

圖 20：在右窗格中，選取 Disable Age Field on Tab 2



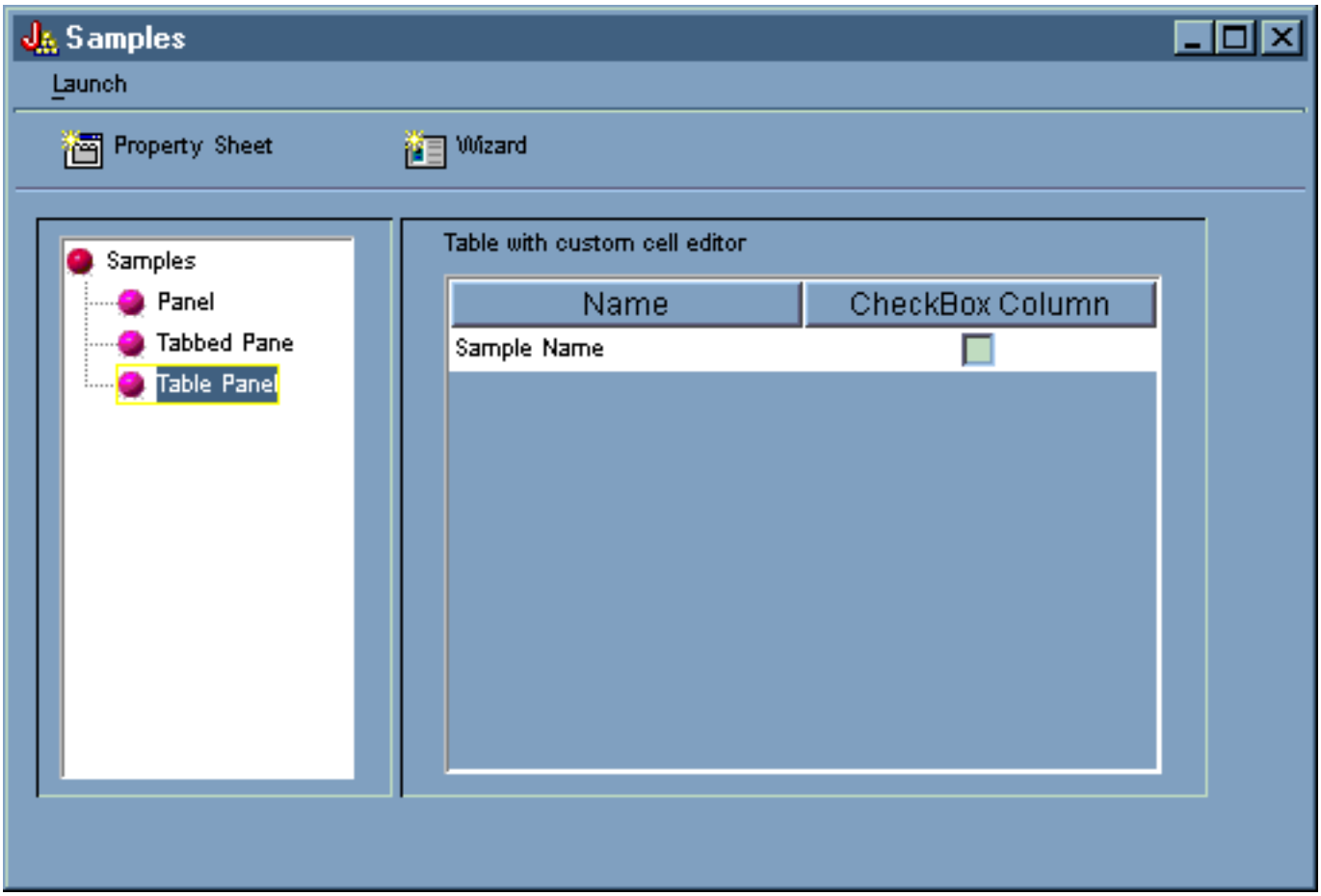
選取Disable Age Field on Ta選項，會停用anel Sample標籤中的 Age欄位，並使該選項的影像變成灰色，如圖 21 中所示。

圖 21：停用 Panel Sample 標籤中 Age 的結果



在 GUI Builder 範例主視窗的左窗格中選取able Pane，說明使用含有自訂描述器以及自訂資料格編輯器的表格畫面，如圖 22 中所示。

圖 22：選取左窗格中的 Table Panel


Samples
Launch
Property Sheet
Wizard
Samples
Panel
Tabbed Pane
Table Panel
Table with custom cell editor
Name
CheckBox Column
Sample Name

HTML 類別範例

下列範例會告訴您使用 HTML 類別的一些方法：

- 範例：使用 BidiOrdering 類別
- 範例：建立 HTMLAlign 物件
- 範例：使用 HTML 套表 類別
- 套表輸入類別範例：
  - 範例：建立 ButtonFormInput 物件
  - 範例：建立 FileFormInput 物件
  - 範例：建立 HiddenFormInput 物件
  - 範例：建立 ImageFormInput 物件
  - 範例：建立 ResetFormInput 物件
  - 範例：建立 SubmitFormInput 物件
  - 範例：建立 TextFormInput 物件
  - 範例：建立 PasswordFormInput 物件
  - 範例：建立 RadioFormInput 物件
  - 範例：建立 CheckboxFormInput 物件
- 範例：建立 HTMLHeading 物件
- 範例：使用 HTMLHyperlink 類別
- 範例：使用 HTMLImage 類別
- HTMLList 範例
  - 範例：建立排序的清單
  - 範例：建立未排序的清單
  - 範例：建立巢狀清單
- 範例：建立 HTMLMeta 標示
- 範例：建立 HTMLParameter 標示
- 範例：建立 HTMLServlet 標示
- 範例：使用 HTMLText 類別
- HTMLTree 範例
  - 範例：使用 HTMLTree 類別
  - 範例：建立一個可瀏覽的整合檔案系統樹
- 佈置套表類別：
  - 範例：使用 GridLayoutFormPanel 類別
  - 範例：使用 LineLayoutFormPanel 類別
- 範例：使用 TextAreaFormElement 類別
- 範例：使用 LabelFormOutput 類別
- 範例：使用 SelectFormElement 類別
- 範例：使用 SelectOption 類別
- 範例：使用 RadioFormInputGroup 類別
- 範例：使用 RadioFormInput 類別
- 範例：使用 HTMLTable 類別
  - 範例：使用 HTMLTableCell 類別
  - 範例：使用 HTMLTableRow 類別
  - 範例：使用 HTMLTableHeader 類別
  - 範例：使用 HTMLTableCaption 類別

您也可以一起使用 HTML 及servlet類別，就如此範例 所示。

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

程式碼不保事項聲明

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
您可以依據這些範例，產生類似的函數，來符合您的需要。

IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM
不擔保或默示保證這些程式的可靠性、可用性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 HTML 套表類別

下面範例將告訴您如何使用 HTML 套表類別。 您也可以執行此程式，以檢視範例輸出 。 以 "showHTML" 方法所使用的 HTML 類別是 粗體。

註： 請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML Forms.
//
///////////////////////////////////////////////////////////////////////////////

package customer;

import java.io.*;
import java.awt.Color;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.*;
import com.ibm.as400.util.html.*;


public class HTMLExample extends HttpServlet
{

  private static boolean found = false;         // Determines if user
already exists in
                                                // the list of registrants.

  String regPath = "c:\\registration.txt";      // Registration information
will be stored here


   public void init(ServletConfig config)
  {

     try
    {
         super.init(config);

    }
  catch (Exception e)
    {
           e.printStackTrace();
    }

  }

  /**
   *  Process the GET request.
   *  @param req The request.
   *  @param res The response.
```

```
   **/
   public void doGet (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();

      // Display the Web using the new HTML classes
      out.println(showHTML());
      out.close();
   }


    public void doPost (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
    {
        String nameStr   = req.getParameter("name");
        String emailStr = req.getParameter("email");
        String errorText= "";

        // Output stream to write to the servlet
        ServletOutputStream out = res.getOutputStream();

        res.setContentType("text/html");

        // Check name & e-mail parameters for valid values
        if (nameStr.length() == 0)
           errorText += "Customer Name not entered.  ";
        if (emailStr.length() == 0)
           errorText += "E-mail not entered.  " ;

        // If name & e-mail have both been provided, continue.
        if (errorText.length() == 0)
        {

     try
           {
              //Create the registration.txt file
              FileWriter f = new FileWriter(regPath, true);
              BufferedWriter output = new BufferedWriter(f);

              //buffered reader for searching the file
              BufferedReader in = new BufferedReader(new
FileReader(regPath));

              String line = in.readLine();

              // reset the found flag
              found = false;

              // Check to see if this customer has already registered
              //  or has already used the same e-mail address
              while (!found)
              {
                 // if file is empty or end of file reached.
                 if (line == null)
              break;
```

```
                // if customer already registered
                if ((line.equals("Customer Name: " + nameStr)) ||
(line.equals("Email address: " + emailStr)))
                {
                   // Output a message to the customer saying they have
already
                   // registered
           out.println ("<HTML> " +
                                 "<TITLE> Toolbox Registration</TITLE> " +
                      "<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> "
+
                      "<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"> "
);
                   out.println ("<P><HR>" +
                                 "<P>" + nameStr + "</B>, you have already
registered using that " +
                                 "<B>Name</B> or <B>E-mail address</B>." +
                                 "<P> Thank You!...<P><HR>");

           // Create a HTMLHyperlink object and display it
                   out.println ("<UL><LI>" + new
HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form") +
"</UL></BODY></HTML>");
                   found = true;
               break;

                }
                else    // read the next line
                   line = in.readLine();

               }

               // String object to hold data submitted from the HTML Form
               String data;

               // If the users name or e-mail aren't found in our text file,
continue.
               if (!found)
               {
                  //-------------------------------------------------------
-----------------------
                  // Insert the new customer info into a file
                        output.newLine();
                  output.write("Customer Name: " + nameStr);
                        output.newLine();
                  output.write("Email address: " + emailStr);
                        output.newLine();
                  //-------------------------------------------------------
-----------------------


                  //-------------------------------------------------------
-----------------------
                  //Getting "USE" checkbox from form
                  data = req.getParameter("use");
                  if(data != null)
                  {
```

```
                output.write("Currently Using Toolbox: " + data);
                   output.newLine();
                }
                //-------------------------------------------------------
-----------------------


                //-------------------------------------------------------
-----------------------
                //Getting "More Information" checkbox from form
                data = req.getParameter("contact");
                if (data != null)
                {
                   output.write("Requested More Information: " + data);
                output.newLine();
                }
                //-------------------------------------------------------
-----------------------


                //-------------------------------------------------------
-----------------------
                //Getting "AS400 Version" from form
                data = req.getParameter("version");
                if (data != null)
                {
                   if (data.equals("multiple versions"))
                   {
                      data = req.getParameter("MultiList");
                      output.write("Multiple Versions: " + data);
                   }
        else
                      output.write("AS400 Version: " + data);

                output.newLine();
                }
                //-------------------------------------------------------
-----------------------


                //-------------------------------------------------------
-----------------------
                //Getting "Current Projects" from form
                data = req.getParameter("interest");
                if (data != null)
                {
                   output.write("Using Java or Interested In: " + data);
                output.newLine();
                }
                //-------------------------------------------------------
-----------------------


                //-------------------------------------------------------
-----------------------
                //Getting "Platforms" from form
                data = req.getParameter("platform");
                if (data != null)
```

```
                {
                   output.write("Platforms: " + data);
                output.newLine();
                   if (data.indexOf("Other") >= 0)
                   {
                      output.write("Other Platforms: " +
req.getParameter("OtherPlatforms"));
                output.newLine();
                   }
                }
                //-----------------------------------------------------------
-----------------------


                //-----------------------------------------------------------
-----------------------
                //Getting "Number of iSeries or AS/400e servers" from form
                data = req.getParameter("list1");
                if (data != null)
                {
                   output.write("iSeries 伺服器數目：" + data);
                output.newLine();
                }
                //-----------------------------------------------------------
-----------------------


                //-----------------------------------------------------------
-----------------------
                //Getting "Comments" from form
                data = req.getParameter("comments");
                if (data != null && data.length() > 0)
                {
                   output.write("Comments: " + data);
                      output.newLine();
                }
                //-----------------------------------------------------------
-----------------------


                //-----------------------------------------------------------
-----------------------
                //Getting "Attachment"
                data = req.getParameter("myAttachment");
                if (data != null && data.length() > 0)
                {
                   output.write("Attachment File: " + data);
                output.newLine();
                }
                //-----------------------------------------------------------
-----------------------


                //-----------------------------------------------------------
-----------------------
                //Getting Hidden "Copyright" infomation
                data = req.getParameter("copyright");
```

```
                if (data != null)
                {
                   output.write(data);
                output.newLine();
                }
                //-------------------------------------------------------
-----------------------


                output.flush();
                output.close();

                // Print a thanks to the customer
                out.println("<HTML>");
                out.println("<TITLE>Thank You!</TITLE>");
             out.println("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\">
");
                out.println("<BODY BGCOLOR=\"blanchedalmond\">");
                out.println("<HR><P>Thank You for Registering, <B>" +
nameStr + "</B>!<P><HR>");

           // Create a HTMLHyperlink object and display it
              out.println("<UL><LI>" + new
HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form"));
              out.println("</UL></BODY></HTML>");

              }

            }
  catch (Exception e)
           {
              // Show error in browser
                 out.println("<HTML>");
              out.println("<TITLE>ERROR!</TITLE>");
              out.println("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\">
");
                 out.println("<BODY BGCOLOR=\"blanchedalmond\">");
              out.println("<BR><B>Error Message:</B><P>");
              out.println(e + "<P>");

           // Create a HTMLHyperlink object and display it
              out.println("<UL><LI>" + new
HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form"));
              out.println("</UL></BODY></HTML>");

            e.printStackTrace();
           }
        }
        else
        {
           // Output a message to the customer saying customer name & e-mail
not entered. Please
           // try again
           out.println ("<HTML> " +
                        "<TITLE>Invalid Registration Form</TITLE> " +
                        "<META HTTP-EQUIV=\"pragma\" content=\"no-cache\"> "
+
```

```
                    "<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"> "
);

          out.println ("<HR><B>ERROR</B> in customer data - <P><B>" +
                       errorText + "</B><P> Please Try Again... <HR>");


          // Create a HTMLHyperlink object and display it
          out.println("<UL><LI>" + new
HTMLHyperlink("./customer.HTMLExample", "Back to Registration Form") +
"</UL></BODY></HTML>");
        }
         // Close the writer
      out.close();


    }



    public void destroy(ServletConfig config)
    {
       // do nothing
    }



   public String getServletInfo()
    {
        return "My Product Registration";
    }

    private String showHTML()
    {
      // String Buffer to hold HTML Page
      StringBuffer page = new StringBuffer();

      // Create the HTML Form object
       HTMLForm form = new HTMLForm("/servlet/customer.HTMLExample");;
       HTMLText txt;

      // Build the beginning of the HTML Page and add it to the String
Buffer
      page.append("<HTML>\n");
      page.append("<TITLE> Welcome!!</TITLE>\n");
      page.append("<HEAD><SCRIPT LANGUAGE=\"JavaScript\">function
test(){alert(\"This is a sample script executed with a
ButtonFormInput.\")}</SCRIPT></HEAD>");
      page.append("<META HTTP-EQUIV=\"pragma\" content=\"no-cache\">\n");
      page.append("<BODY BGCOLOR=\"blanchedalmond\" TEXT=\"black\"><BR>\n");

     try
      {
         //-----------------------------------------------------------------
-----------------------
         // Create page title using HTML Text
          txt = new HTMLText("產品註冊");
          txt.setSize(5);
```

```
          txt.setBold(true);
        txt.setColor(new Color(199, 21, 133));
        txt.setAlignment(HTMLConstants.CENTER);

        // Add HTML Text to the String Buffer
        page.append(txt.getTag(true) + "<HR><BR>\n");
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a Line Layout
        LineLayoutFormPanel line = new LineLayoutFormPanel();
          txt = new HTMLText("Enter your name and e-mail address:");
        txt.setSize(4);
        line.addElement(txt);

        // Add the Line Layout to String Buffer
        page.append(line.toString());
        page.append("<BR>");
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Set the HTML Form METHOD
         form.setMethod(HTMLForm.METHOD_POST);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a Text input for the name.
        TextFormInput user = new TextFormInput("name");
        user.setSize(25);
        user.setMaxLength(40);

        // Create a Text input for the email address.
        TextFormInput email = new TextFormInput("email");
        email.setSize(30);
        email.setMaxLength(40);

        // Create a ImageFormInput
        ImageFormInput img = new ImageFormInput("提出套表",
"..\\images\\myPiimages/c.gif");
        img.setAlignment(HTMLConstants.RIGHT);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a LineLayoutFormPanel object for the name & e-mail
address
        LineLayoutFormPanel line2 = new LineLayoutFormPanel();

        // Add elements to the line form
        line2.addElement(new LabelFormElement("Name:"));
        line2.addElement(user);
```

```
        // Create and add a Label Element to the Line Layout
        line2.addElement(new LabelFormElement("E-mail:"));
        line2.addElement(email);
        line2.addElement(img);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create Questions line layout
        LineLayoutFormPanel line3 = new LineLayoutFormPanel();

        // Add elements to the line layout
        line3.addElement(new LineLayoutFormPanel());
        line3.addElement(new CheckboxFormInput("use", "yes", "Do you
currently use the Toolbox?", false));
        line3.addElement(new LineLayoutFormPanel());
        line3.addElement(new CheckboxFormInput("contact", "yes", "Would you
like information on future Toolbox releases?", true));
        line3.addElement(new LineLayoutFormPanel());
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create Version Radio Group
        RadioFormInputGroup group = new RadioFormInputGroup("version");

        // Add Radio Form Inputs to the Group
        group.add(new RadioFormInput("version", "v3r2", "V3R2", false));
        group.add(new RadioFormInput("version", "v4r1", "V4R1", false));
        group.add(new RadioFormInput("version", "v4r2", "V4R2", false));
        group.add(new RadioFormInput("version", "v4r3", "V4R3", false));
        group.add(new RadioFormInput("version", "v4r4", "V4R4", false));
        group.add(new RadioFormInput("version", "multiple versions",
"Multiple Versions? Which ones:", false));

        //Create a Select Form Element
        SelectFormElement mlist = new SelectFormElement("MultiList");
        mlist.setMultiple(true);
        mlist.setSize(3);

        //Create the Options for the Select Form Element
        SelectOption option1 = mlist.addOption("V3R2", "v3r2");
        SelectOption option2 = mlist.addOption("V4R1", "v4r1");
        SelectOption option3 = mlist.addOption("V4R2", "v4r2");
        SelectOption option4 = mlist.addOption("V4R3", "v4r3");
        SelectOption option5 = mlist.addOption("V4R4", "v4r4");

        // Create HTML text
        txt = new HTMLText("現行伺服器層次：");
        txt.setSize(4);

        // Create Grid Layout
        GridLayoutFormPanel grid1 = new GridLayoutFormPanel(3);

        // Add radio group & select form element to the grid
        grid1.addElement(txt);
```

```
        grid1.addElement(group);
        grid1.addElement(mlist);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create Grid Layout for interests
        GridLayoutFormPanel grid2 = new GridLayoutFormPanel(1);
          txt = new HTMLText("Current Projects or Area of Interest: (check
all that apply)");
        txt.setSize(4);

        // Add elements to Grid Layout
        grid2.addElement(new LineLayoutFormPanel());
        grid2.addElement(txt);
        // Create and add a Checkbox to the Grid Layout
        grid2.addElement(new CheckboxFormInput("interest", "applications",
"Applications", true));
        grid2.addElement(new CheckboxFormInput("interest", "applets",
"Applets", false));
        grid2.addElement(new CheckboxFormInput("interest", "servlets",
"Servlets", false));
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create Line Layout for platforms
        LineLayoutFormPanel line4 = new LineLayoutFormPanel();
          txt = new HTMLText("Client Platforms Used: (check all that
apply)");
        txt.setSize(4);

        // Add elements to Line Layout
        line4.addElement(new LineLayoutFormPanel());
        line4.addElement(txt);
        line4.addElement(new LineLayoutFormPanel());
        line4.addElement(new CheckboxFormInput("platform", "95",
"Windows95", false));
        line4.addElement(new CheckboxFormInput("platform", "98",
"Windows98", false));
        line4.addElement(new CheckboxFormInput("platform", "NT",
"WindowsNT", false));
        line4.addElement(new CheckboxFormInput("platform", "OS2", "OS/2",
false));
        line4.addElement(new CheckboxFormInput("platform", "AIX", "AIX",
false));
        line4.addElement(new CheckboxFormInput("platform", "Linux",
"Linux", false));
        line4.addElement(new CheckboxFormInput("platform", "AS400",
"iSeries", false));
        line4.addElement(new CheckboxFormInput("platform", "Other",
"Other:", false));

        TextFormInput other = new TextFormInput("OtherPlatforms");
        other.setSize(20);
```

```
        other.setMaxLength(50);

        line4.addElement(other);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a Line Layout for number of servers
        LineLayoutFormPanel grid3 = new LineLayoutFormPanel();

        txt = new HTMLText("您有多少個 iSeries 或 AS/400e 伺服器？  ");
        txt.setSize(4);

        // Create a Select Form Element for number of servers owned
        SelectFormElement list = new SelectFormElement("list1");
        // Create and add the Select Options to the Select Form Element
List
        SelectOption opt0 = list.addOption("0", "zero");
        SelectOption opt1 = list.addOption("1", "one", true);
        SelectOption opt2 = list.addOption("2", "two");
        SelectOption opt3 = list.addOption("3", "three");
        SelectOption opt4 = list.addOption("4", "four");
        SelectOption opt5 = new SelectOption("5+", "FiveOrMore", false);
        list.addOption(opt5);

        // Add Elements to the Grid Layout
        grid3.addElement(new LineLayoutFormPanel());
        grid3.addElement(txt);
        grid3.addElement(list);
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a Grid Layout for Product Comments
        GridLayoutFormPanel grid4 = new GridLayoutFormPanel(1);
          txt = new HTMLText("Product Comments:");
        txt.setSize(4);

        // Add elements to the Grid Layout
        grid4.addElement(new LineLayoutFormPanel());
        grid4.addElement(txt);
        //grid4.addElement(new LineLayoutFormPanel());
        // Create a Text Area Form
        grid4.addElement(new TextAreaFormElement("comments", 5, 75));
        grid4.addElement(new LineLayoutFormPanel());
        //-------------------------------------------------------------------
-----------------------

        //-------------------------------------------------------------------
-----------------------
        // Create a Grid Layout
        GridLayoutFormPanel grid5 = new GridLayoutFormPanel(2);
          txt = new HTMLText("您要登入伺服器嗎？");
        txt.setSize(4);

        // Create a Text input and Label for the system name.
```

```
        TextFormInput sys = new TextFormInput("system");
        LabelFormElement sysLabel = new LabelFormElement("System:");

        // Create a Text input and Label for the userid.
        TextFormInput uid = new TextFormInput("uid");
        LabelFormElement uidLabel = new LabelFormElement("UserID");

        // Create a Password input and Label for the password.
        PasswordFormInput pwd = new PasswordFormInput("pwd");
        LabelFormElement pwdLabel = new LabelFormElement("Password");

        // Add the Text inputs, password inputs, and Labels to the grid
        grid5.addElement(sysLabel);
        grid5.addElement(sys);
        grid5.addElement(uidLabel);
        grid5.addElement(uid);
        grid5.addElement(pwdLabel);
        grid5.addElement(pwd);
        //-------------------------------------------------------------
-----------------------


        //-------------------------------------------------------------
-----------------------
        // Add the various panels created to the HTML Form in the order you
wish them to appear
          form.addElement(line2);
        form.addElement(line3);
        form.addElement(grid1);
        form.addElement(grid2);
        form.addElement(line4);
        form.addElement(grid3);
        form.addElement(grid4);
        form.addElement(txt);
        form.addElement(new LineLayoutFormPanel());
        form.addElement(grid5);
        form.addElement(new LineLayoutFormPanel());
        form.addElement(new HTMLText("Submit an attachment Here: <br />"));
        // Add a File Input to the form
          form.addElement(new FileFormInput("myAttachment"));
        form.addElement(new ButtonFormInput("button", "TRY ME!",
"test()"));
        // Adds a empty Line Layout, which in turn
        // adds a line break <br /> to the form
          form.addElement(new LineLayoutFormPanel());
        form.addElement(new LineLayoutFormPanel());
        form.addElement(new SubmitFormInput("submit", "Register"));
        form.addElement(new LineLayoutFormPanel());
        form.addElement(new LineLayoutFormPanel());
        form.addElement(new ResetFormInput("reset", "Reset"));

        // Add a Hidden Input to the form
          form.addElement(new HiddenFormInput("copyright", "(C) Copyright IBM
Corp. 1999, 1999"));
        //-------------------------------------------------------------
-----------------------
```

```
            // Add the entire HTML Form to the String Buffer
            page.append(form.toString());

        }
    catch (Exception e)
        {
            e.printStackTrace();
        }

        // Add the Ending HTML tags to the Buffer
        page.append("</BODY>\n");
        page.append("</HTML>\n");

        // Return the entire HTML page string
        return page.toString();

    }
}
```

範例：使用 HTMLTree 類別

註： 請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Trees.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.File;
import java.io.PrintWriter;
import java.io.IOException;

import java.util.Vector;
import java.util.Properties;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.Trace;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLTree;
import com.ibm.as400.util.html.HTMLTreeElement;
import com.ibm.as400.util.html.URLParser;
import com.ibm.as400.util.html.DirFilter;
import com.ibm.as400.util.html.FileTreeElement;
import com.ibm.as400.util.servlet.ServletHyperlink;

/**
 *  An example of using the HTMLTree and FileTreeElement classes in a
servlet.
 **/
public class TreeNav extends HttpServlet
{
   public void init(ServletConfig config)
      throws ServletException
   {
         super.init(config);

      // The Toolbox uses a set of default icons to represents expanded,
collapsed, and documents within the HTMLTree.
      // To enhance those icons, the Toolbox ships three gifs (expanded.gif,
collapsed.gif, bullet.gif) in the
      // jt400Servlet.jar file.  Browsers do not have the ability to find
gifs in a jar or zip file, so those images
      // need to be extracted from the jar file and placed in the
appropriate webserver directory (by default it is the
      // /html directory).  Then uncomment the following lines of code and
specify the correct location in the these
      // set methods.  The location can be absolute or relative.

      HTMLTreeElement.setExpandedGif("/images/expanded.gif");
```

```
      HTMLTreeElement.setCollapsedGif("/images/collapsed.gif");
      HTMLTreeElement.setDocGif("/images/bullet.gif");
   }

   /**
   *  Process the GET request.
   *  @param req The request.
   *  @param res The response.
    **/
   public void doGet (HttpServletRequest req, HttpServletResponse resp)
   throws ServletException, IOException
   {
      HttpSession session = req.getSession(true);
      HTMLTree fileTree = (HTMLTree)session.getValue("filetree");

      // If this session does not already have a file tree, then
      // create the initial tree.
      if (fileTree == null)
         fileTree = createTree(req, resp, req.getRequestURI());

      // Set the Http servlet request on the HTMLTree.
      fileTree.setHttpServletRequest(req);

      resp.setContentType("text/html");

    PrintWriter out = resp.getWriter();
    out.println("<html>\n");
      out.println(new HTMLMeta("到期日", "星期一, 1990 年 1 月 3 日 13:00:00
GMT"));
    out.println("<body>\n");

      // Get the tag for the HTMLTree.
      out.println(fileTree.getTag());

      out.println("</body>\n");
    out.println("</html>\n");
      out.close();

      // Set the session tree value, so when entering this servlet for
      // the second time, the FileTree object will be reused.
    session.putValue("filetree", fileTree);
   }

   /**
    *  Process the POST request.
   *  @param req The request.
   *  @param res The response.
    **/
    public void doPost (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
   }


   /**
    *  This method will create the initial HTMLTree.
```

```
    **/
   private HTMLTree createTree(HttpServletRequest req, HttpServletResponse
resp, String uri)
   {
      // Create an HTMLTree object.
      HTMLTree tree = new HTMLTree(req);

     try
    {
        // Create a URLParser object.
        URLParser urlParser = new URLParser(uri);

        AS400 sys = new AS400(CPUStatus.systemName_, "javactl", "jteam1");

        // Create a File object and set the root IFS directory.
       IFSJavaFile root = new IFSJavaFile(sys, "/QIBM");

        // Create a Filter and list all of the directories.
       DirFilter filter = new DirFilter();
        //File[] dirList = root.listFiles(filter);

        // Get the list of files that satisfy the directory filter.
        String[] list = root.list(filter);

        File[] dirList = new File[list.length];

        // We don't want to require webservers to use JDK1.2 because
        // most webserver JVM's are slower to upgrade to the latest JDK
level.
        // The most efficient way to create these file objects is to use
        // the listFiles(filter) method in JDK1.2 which would be done
        // like the following, instead of using the list(filter) method
        // and then converting the returned string arrary into the
appropriate
        // File array.
        // File[] dirList = root.listFiles(filter);

        for (int j=0; j<dirList.length; ++j)
        {
           if (root instanceof IFSJavaFile)
              dirList[j] = new IFSJavaFile((IFSJavaFile)root, list[j]);
         else
              dirList[j] = new File(list[j]);
        }


        for (int i=0; i<dirList.length; i++)
       {
           // Create a FileTreeElement for each directory in the list.
           FileTreeElement node = new FileTreeElement(dirList[i]);

           // Create a ServletHyperlink for the expand/collapse icons.
           ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
           //sl.setHttpServletResponse(resp);
           node.setIconUrl(sl);


           // Create a ServletHyperlink to the TreeList servlet, which will
```

```
            // display the contents of thie FileTreeElement (directory).
         ServletHyperlink tl = new ServletHyperlink("/servlet/TreeList");
         tl.setTarget("list");

            // If the ServletHyperlink doesn't have a name, then set it to
the
            // name of the directory.
            if (tl.getText() == null)
               tl.setText(dirList[i].getName());

            // Set the TextUrl for the FileTreeElement.
            node.setTextUrl(tl);

            // Add the FileTreeElement to the HTMLTree.
         tree.addElement(node);
        }
     }
   catch (Exception e)
     {
            e.printStackTrace();
     }

     return tree;
    }


     public void destroy(ServletConfig config)
     {
        // do nothing
     }

    public String getServletInfo()
     {
         return "FileTree 導覽";
     }
}
```

範例：建立可瀏覽的整合檔案系統樹
(三個檔案之一)

此範例程式碼與另外兩個範例檔案中的程式碼一起使用， 可以顯示 HTMLTree 及 servlet 中的 FileListElement。範例中的三個檔案如下：

- FileTreeExample.java - 此檔案會產生 HTML 訊框並啟動 servlet
- TreeNav.java- 會建置及管理樹
- TreeList.java 會顯示 TreeNav.java 類別中的選項內容

註： 請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML package
// classes, which allow you to easily build HTML and File Trees.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.PrintWriter;
import java.io.IOException;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.HTMLMeta;


//
// An example of using frames to display an HTMLTree and FileListElement
// in a servlet.
//

public class FileTreeExample extends HttpServlet
{
 public void init(ServletConfig config)
throws ServletException
   {
 super.init(config);
   }

   /**
 *Process the GET request.
 *@param req The request.
 *@param res The response.
    **/

 public void doGet (HttpServletRequest req, HttpServletResponse resp)
 throws ServletException, IOException
   {
resp.setContentType("text/html");

// Set up two frames. The first, a navigation frame, will display
// the HTMLTree, which will contain FileTreeElements and allow
```

```
// navigation of the File system.The second frame will display/list
// the contents of a selected directory from the navigation frame.
PrintWriter out = resp.getWriter();
out.println("<html>\n");
out.println(new HTMLMeta("到期日","星期一, 1990 年 1 月 4 日 13:00:00
GMT"));
out.println("<frameset cols=\"25%,*\">");
out.println("<frame frameborder=\"5\" src=\"/servlet/TreeNav\"
name=\"nav\">");
out.println("<frame frameborder=\"3\" src=\"/servlet/TreeList\"
name=\"list\">");
out.println("</frameset>");
out.println("</html>\n");
out.close();
   }

   /**
*Process the POST request.
 *@param req The request.
 *@param res The response.
    **/

public void doPost (HttpServletRequest req, HttpServletResponse res)
 throws ServletException, IOException
   {
res.setContentType("text/html");
ServletOutputStream out = res.getOutputStream();
   }

public void destroy(ServletConfig config)
   {
 // do nothing
   }

 public String getServletInfo()
   {
return "FileTree Servlet";
   }
}
```

範例：建立可跨越的整合檔案系統樹 (三個檔案中的第二個檔案)

此範例程式碼與其它兩個範例檔中的程式碼相連結，可在 servlet 中顯示 HTMLTree 與 FileListElement。範例中的三個檔案為：

- FileTreeExample.java 產生 HTML 框架並啟動 servlet
- TreeNav.java - 此檔案可建置及管理樹狀結構
- TreeList.java 顯示在 TreeNav.java 類別中產生的選項內容

註：請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
///////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Trees.
//
///////////////////////////////////////////////////////////////////////////////

import java.io.File;
import java.io.PrintWriter;
import java.io.IOException;

import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLTree;
import com.ibm.as400.util.html.HTMLTreeElement;
import com.ibm.as400.util.html.URLParser;
import com.ibm.as400.util.html.DirFilter;
import com.ibm.as400.util.html.FileTreeElement;
import com.ibm.as400.util.servlet.ServletHyperlink;

//
// An example of using the HTMLTree and FileTreeElement classes
// in a servlet.
//

public class TreeNav extends HttpServlet
{
 private AS400 sys_;

 public void init(ServletConfig config)
throws ServletException
   {
 super.init(config);

 // Create an AS400 object.
sys_ = new AS400("mySystem", "myUserID", "myPassword");

// The Toolbox uses a set of default icons to represents expanded,
```

```
collapsed,
// and documents within the HTMLTree. To enhance those icons, the Toolbox
ships
// three gifs (expanded.gif, collapsed.gif, bullet.gif) in the
jt400Servlet.jar
// file.Browsers do not have the ability to find gifs in a jar or zip file,
so
// you need to extract those images from the jar file and place them in the
// appropriate webserver directory (by default it is the /html directory).
Then
// change the following lines of code to specify the correct location in the
// set methods.The location can be absolute or relative.

HTMLTreeElement.setExpandedGif("http://myServer/expanded.gif");
HTMLTreeElement.setCollapsedGif("http://myServer/collapsed.gif");
HTMLTreeElement.setDocGif("http://myServer/bullet.gif");
   }

   /**
 *Process the GET request.
 *@param req The request.
 *@param res The response.
    **/

 public void doGet (HttpServletRequest req, HttpServletResponse resp)
 throws ServletException, IOException
   {
// Use session data to remember the state of the tree.
HttpSession session = req.getSession(true);
HTMLTree fileTree = (HTMLTree)session.getValue("filetree");

// If this session does not already have a file tree, then
// create the initial tree.
if (fileTree == null)
 fileTree = createTree(req, resp, req.getRequestURI());

// Set the Http servlet request on the HTMLTree.
fileTree.setHttpServletRequest(req);

resp.setContentType("text/html");

PrintWriter out = resp.getWriter();
out.println("<html>\n");
out.println(new HTMLMeta("Expires","Mon, 03 Jan 1990 13:00:00 GMT"));
out.println("<body>\n");

// Get the tag for the HTMLTree.
out.println(fileTree.getTag());

out.println("</body>\n");
out.println("</html>\n");
out.close();

// Set the session tree value, so when entering this servlet for
// the second time, the FileTree object will be reused.
session.putValue("filetree", fileTree);
   }
```

```
   /**
*Process the POST request.
 *@param req The request.
 *@param res The response.
    **/

public void doPost (HttpServletRequest req, HttpServletResponse res)
 throws ServletException, IOException
   {
res.setContentType("text/html");
ServletOutputStream out = res.getOutputStream();
   }

   /**
*This method will create the initial HTMLTree.
    **/

 private HTMLTree createTree(HttpServletRequest req,
 HttpServletResponse resp, String uri)
   {
// Create an HTMLTree object.
HTMLTree tree = new HTMLTree(req);

 try
       {
// Create a URLParser object.
URLParser urlParser = new URLParser(uri);

// Create a File object and set the root IFS directory.
IFSJavaFile root = new IFSJavaFile(sys_, "/QIBM");

// Create a Filter.
DirFilter filter = new DirFilter();

// Get the list of files that satisfy the directory filter.
String[] list = root.list(filter);

File[] dirList = new File[list.length];

// We don't want to require webservers to use JDK1.2 because
// most webserver JVM's are slower to upgrade to the latest
// JDK level. The most efficient way to create these file objects
// is to use the listFiles(filter) method in JDK1.2 which would
// be done like the following, instead of using the list(filter)
// method and then converting the returned string arrary into the
// appropriate File array.
// File[] dirList = root.listFiles(filter);

for (int j=0; j<dirList.length; ++j)
        {
if (root instanceof IFSJavaFile)
 dirList[j] = new IFSJavaFile((IFSJavaFile)root, list[j]);
else
 dirList[j] = new File(list[j]);
        }

for (int i=0; i<dirList.length; i++)
        {
```

```
// Create a FileTreeElement for each directory in the list.
FileTreeElement node = new FileTreeElement(dirList[i]);

// Create a ServletHyperlink for the expand/collapse icons.
ServletHyperlink sl = new ServletHyperlink(urlParser.getURI());
sl.setHttpServletResponse(resp);
node.setIconUrl(sl);

// Create a ServletHyperlink to the TreeList servlet, which will
// display the contents of thie FileTreeElement (directory).
ServletHyperlink tl = new ServletHyperlink("/servlet/TreeList");
tl.setTarget("list");

// If the ServletHyperlink doesn't have a name, then set it to the
// name of the directory.
if (tl.getText() == null)
tl.setText(dirList[i].getName());

// Set the TextUrl for the FileTreeElement.
node.setTextUrl(tl);

// Add the FileTreeElement to the HTMLTree.
tree.addElement(node);
          }

sys_.disconnectAllServices();
        }

catch (Exception e)
        {
e.printStackTrace();
        }

return tree;
    }

public void destroy(ServletConfig config)
    {
 // do nothing
    }

 public String getServletInfo()
    {
return "FileTree Navigation";
    }
}
```

範例：建立可跨越的整合檔案系統樹 (三個檔案中的第三個檔案)

此範例程式碼與其它兩個範例檔中的程式碼相連結，可在 servlet 中顯示 HTMLTree 與 FileListElement。範例中的三個檔案為：

- FileTreeExample.java 產生 HTML 框架並啟動 servlet
- TreeNav.java- 建置及管理樹狀結構
- TreeList.java - 本檔案顯示在 TreeNav.java 類別中產生的選項內容

註：請閱讀 程式碼不保事項聲明 ，以取得重要的法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// This source is an example of using the IBM Toolbox for Java HTML
// package classes, which allow you to easily build HTML and File Lists.
//
////////////////////////////////////////////////////////////////////////////

import java.io.PrintWriter;
import java.io.IOException;
import java.io.File;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.Trace;
import com.ibm.as400.access.IFSJavaFile;
import com.ibm.as400.util.html.HTMLMeta;
import com.ibm.as400.util.html.HTMLHeading;
import com.ibm.as400.util.html.HTMLConstants;
import com.ibm.as400.util.html.FileListElement;
import com.ibm.as400.util.html.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 *An example of using the FileListElement class in a servlet.
 **/
public class TreeList extends HttpServlet
{

private AS400 sys_;

    /**
 *Process the GET request.
 *@param req The request.
 *@param res The response.
     **/
public void doGet(HttpServletRequest req, HttpServletResponse resp)
 throws ServletException, IOException
    {
resp.setContentType("text/html");
```

```
 try
        {

PrintWriter out = resp.getWriter();
out.println("<html>\n");
out.println(new HTMLMeta("Expires", "Mon, 02 Jan 1990 13:00:00 GMT"));
out.println("<body>\n");

// If the path parameter is not null, then the user has selected an element
from
// the FileTreeElement list in the navigation frame.
if (req.getPathInfo() != null)
            {
// Create a FileListElement passing in an AS400 system object and the Http
servlet request.
// The request will contain the necessary path information to list out the
contents of
// the FileTreeElement (directory) selected.
FileListElement fileList = new FileListElement(sys_, req);

//Alternately, create a FileListElement from a NetServer share name and
share path.
                  //
//FileListElement fileList = new FileListElement(sys_, req, "TreeShare",
"/QIBM/ProdData/HTTP/Public/jt400");


// Display the FileListElement contents.
out.println(fileList.list());
            }
else // Display this HTMLHeading if no FileTreeElement has been selected.
            {
HTMLHeading heading = new HTMLHeading(1,"An HTML File List Example");
heading.setAlign(HTMLConstants.CENTER);

out.println(heading.getTag());
            }

out.println("</body>\n");
out.println("</html>\n");
out.close();
        }
catch (Exception e)
        {
e.printStackTrace();
        }

    }

    /**
*Process the POST request.
 *@param req The request.
 *@param res The response.
    **/
public void doPost (HttpServletRequest req, HttpServletResponse res)
```

```
 throws ServletException, IOException
    {
res.setContentType("text/html");
ServletOutputStream out = res.getOutputStream();

    }

 public void init(ServletConfig config)
throws ServletException
    {
 super.init(config);

 // Create an AS400 object.
sys_ = new AS400("mySystem", "myUID", "myPWD");
    }
}
```

範例：使用 HTMLTable 類別

下列範例將告訴您 HTMLTable 類別如何運作：

```
// Create a default HTMLTable object.
HTMLTable table = new HTMLTable();

// Set the table attributes.
table.setAlignment(HTMLTable.CENTER);
table.setBorderWidth(1);

// Create a default HTMLTableCaption object and set the caption text.
HTMLTableCaption caption = new HTMLTableCaption();
caption.setElement("Customer Account Balances - January 1, 2000");

// Set the caption.
table.setCaption(caption);

// Create the table headers and add to the table.
HTMLTableHeader account_header = new HTMLTableHeader(new
HTMLText("ACCOUNT"));
HTMLTableHeader name_header = new HTMLTableHeader(new HTMLText("NAME"));
HTMLTableHeader balance_header = new HTMLTableHeader(new
HTMLText("BALANCE"));

table.addColumnHeader(account_header);
table.addColumnHeader(name_header);
table.addColumnHeader(balance_header);

// Add rows to the table.  Each customer record represents a row in the
table.
int numCols = 3;
for (int rowIndex=0; rowIndex< numCustomers; rowIndex++)
{
   HTMLTableRow row = new HTMLTableRow();
   row.setHorizontalAlignment(HTMLTableRow.CENTER);

   HTMLText account = new HTMLText(customers[rowIndex].getAccount());
   HTMLText name = new HTMLText(customers[rowIndex].getName());
   HTMLText balance = new HTMLText(customers[rowIndex].getBalance());

   row.addColumn(new HTMLTableCell(account));
   row.addColumn(new HTMLTableCell(name));
   row.addColumn(new HTMLTableCell(balance));

   // Add the row to the table.
   table.addRow(row);
}
System.out.println(table.getTag());
```

前面的 Java 程式碼範例會產生下列 HTML 程式碼：

```
<table align="center" border="1">
```

```
<caption>Customer Account Balances - January 1, 2000</caption>
<tr>
<th>ACCOUNT</th>
<th>NAME</th>
<th>BALANCE</th>
</tr>
<tr align="center">
<td>0000001</td>
<td>Customer1</td>
<td>100.00</td>
</tr>
<tr align="center">
<td>0000002</td>
<td>Customer2</td>
<td>200.00</td>
</tr>
<tr align="center">
<td>0000003</td>
<td>Customer3</td>
<td>550.00</td>
</tr>

</table>
```

下表說明上面的 HTML 程式碼如何顯示在 Web 瀏覽器中：

Customer Account Balances - January 1, 2000

ACCOUNT	NAME	BALANCE
0000001	Customer1	100.00
0000002	Customer2	200.00
0000003	Customer3	550.00

範例：程式呼叫語言 (PCML)

下列範例使用「程式呼叫語言」來呼叫 OS/400 API，每一個範例都鏈結到一個顯示 Java 程式所遵循的 PCML 來源之頁面。

- 擷取資料的簡單範例：顯示擷取伺服器中使用者設定檔的相關資訊所需之 PCML 原始程式及 Java 程式。將呼叫的 API 是 Retrieve User Information (QSYRUSRI) API。
- 擷取資訊清單：顯示擷取伺服器中授權使用者清單所需之 PCML 原始程式及 Java 程式。將呼叫的 API 是 Open List of Authorized Users (QGYOLAUS) API。此範例說明如何存取由伺服器程式傳回的結構陣列。
- 擷取多維度資料：顯示擷取伺服器的「網路檔案系統 (NFS)」所需之 PCML 原始程式及 Java 程式。將呼叫的 API 是 Retrieve NFS Exports (QZNFRTVE) API。這個範例描述如何存取結構陣列內的結構陣列。

註：每一範例的適當權限雖有所不同，但可能包括特定物件權限及特殊權限。
為了能執行這些範例，您必須以具有可執行下列作業的權限之使用者設定檔登入：

- 呼叫範例中的 OS/400 API
- 存取將要求的資訊

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

範例：擷取資料的簡單範例

本範例包含兩部份：

- 呼叫 QSYRUSRI 的 PCML 來源
- 呼叫 QSYRUSRI 的 Java 程式來源

呼叫 QSYRUSRI 的 PCML 原始程式

```
<pcml version="1.0">

<!-- PCML source for calling "Retrieve user Information" (QSYRUSRI) API -->

<!-- Format USRI0150 - Other formats are available -->
<struct name="usri0100">
<data name="bytesReturned"type="int"length="4"usage="輸出"/>
<data name="bytesAvailable" type="int"length="4"usage="輸出"/>
<data name="userProfile"type="字元" length="10" usage="輸出"/>
<data name="previousSignonDate" type="字元" length="7"usage="輸出"/>
<data name="previousSignonTime" type="字元" length="6"usage="輸出"/>
<data type="位元組" length="1"usage="輸出"/>
<data name="badSignonAttempts"type="int"length="4"usage="輸出"/>
<data name="status" type="字元" length="10" usage="輸出"/>
<data name="passwordChangeDate" type="位元組" length="8"usage="輸出"/>
<data name="noPassword" type="字元" length="1"usage="輸出"/>
<data type="位元組" length="1"usage="輸出"/>
<data name="passwordExpirationInterval" type="int"length="4"usage="輸出"/>
<data name="datePasswordExpires"type="位元組" length="8"usage="輸出"/>
<data name="daysUntilPasswordExpires" type="int"length="4"usage="輸出"/>
<data name="setPasswordToExpire"type="字元" length="1"usage="輸出"/>
<data name="displaySignonInfo"type="字元" length="10" usage="輸出"/>
</struct>

<!-- Program QSYRUSRI and its parameter list for retrieving USRI0100 format -
->
<program name="qsyrusri" path="/QSYS.lib/QSYRUSRI.pgm">
<data name="接收器" type="結構" struct="usri0100" usage="輸出"/>
<data name="receiverLength" type="int"length="4"usage="輸入" />
<data name="格式" type="字元" length="8"usage="輸入"init="USRI0100"/>
<data name="設定檔名稱"type="字元" length="10" usage="輸入"init="*CURRENT"/>
<data name="錯誤碼"type="int"length="4"usage="輸入"init="0"/>
</program>

</pcml>
```

呼叫 QSYRUSRI 的 Java 程式來源檔

```
import com.ibm.as400.data.ProgramCallDocument;
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;
```

```
// Example program to call "Retrieve User Information" (QSYRUSRI) API
public class qsyrusri {

public qsyrusri() {
    }

public static void main(String[] argv)
    {
AS400 as400System; // com.ibm.as400.access.AS400
ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
boolean rc = false;// Return code from ProgramCallDocument.callProgram()
String msgId, msgText; // Messages returned from the server
Object value;// Return value from ProgramCallDocument.getValue()

System.setErr(System.out);

// Construct AS400 without parameters, user will be prompted
as400System = new AS400();

 try
        {
// Uncomment the following to get debugging information
//com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

System.out.println("開始 PCML 範例..");
System.out.println("為 QSYRUSRI API 建構 ProgramCallDocument...");

// Construct ProgramCallDocument
// First parameter is system to connect to
// Second parameter is pcml resource name. In this example,
// serialized PCML file "qsyrusri.pcml.ser" or
// PCML source file "qsyrusri.pcml" must be found in the classpath.
pcml = new ProgramCallDocument(as400System, "qsyrusri");

// Set input parameters. Several parameters have default values
// specified in the PCML source. Do not need to set them using Java code.
System.out.println("設定輸入參數...");
pcml.setValue("qsyrusri.receiverLength", new
Integer((pcml.getOutputsize("qsyrusri.receiver"))));

// Request to call the API
// User will be prompted to sign on to the system
System.out.println("為登入使用者呼叫 QSYRUSRI API 要求資訊。");
rc = pcml.callProgram("qsyrusri");

// If return code is false, we received messages from the server
if (rc == false)
           {
// Retrieve list of server messages
AS400Message[] msgs = pcml.getMessageList("qsyrusri");

// Iterate through messages and write them to standard output
for (int m = 0; m < msgs.length; m++)
                {
msgId = msgs[m].getID();
```

```
msgText = msgs[m].getText();
System.out.println("" + msgId + " - " + msgText);
                }
System.out.println("** 呼叫 QSYRUSRI 失敗。請參閱上述訊息 **");
 System.exit(0);
            }
// Return code was true, call to QSYRUSRI succeeded
// Write some of the results to standard output
else
            {
value = pcml.getValue("qsyrusri.receiver.bytesReturned");
System.out.println("傳回的位元組：" + value);
value = pcml.getValue("qsyrusri.receiver.bytesAvailable");
System.out.println("可用的位元組： " + value);
value = pcml.getValue("qsyrusri.receiver.userProfile");
System.out.println("設定檔名稱：" + value);
value = pcml.getValue("qsyrusri.receiver.previousSignonDate");
System.out.println("前一個登入日期：" + value);
value = pcml.getValue("qsyrusri.receiver.previousSignonTime");
System.out.println("前一個登入時間：" + value);
            }
        }
catch(PcmlException e)
        {
System.out.println(e.getLocalizedMessage());
e.printStackTrace();
System.out.println("*** 呼叫 QSYRUSRI 失敗。***");
 System.exit(0);
        }

 System.exit(0);
} // End main()

}
```

範例：擷取資訊清單

本範例有兩部份：

- 呼叫 QGYOLAUS 的 PCML 原始程式
- 呼叫 QGYOLAUS 的 Java 原始程式

呼叫 QGYOLAUS 的 PCML 原始程式

```
<pcml version="1.0">

<!-- PCML source for calling "開啟授權使用者清單" (QGYOLAUS) API -->

<!-- Format AUTU0150 - Other formats are available -->
<struct name="autu0150">
<data name="name" type="char" length="10" />
<data name="userOrGroup"type="char" length="1"/>
<data name="groupMembers" type="char" length="1"/>
<data name="description"type="char" length="50" />
</struct>

<!-- List information structure (common for "Open List" type APIs) -->
<struct name="listInfo">
<data name="totalRcds"type="int"length="4" />
<data name="rcdsReturned" type="int"length="4" />
<data name="rqsHandle"type="byte" length="4" />
<data name="rcdLength"type="int"length="4" />
<data name="infoComplete" type="char" length="1" />
<data name="dateCreated"type="char" length="7" />
<data name="timeCreated"type="char" length="6" />
<data name="listStatus" type="char" length="1" />
<data type="byte" length="1" />
<data name="lengthOfInfo" type="int"length="4" />
<data name="firstRecord"type="int"length="4" />
<data type="byte" length="40" />
</struct>

<!-- Program QGYOLAUS and its parameter list for retrieving AUTU0150 format -
->
<program name="qgyolaus" path="/QSYS.lib/QGY.lib/QGYOLAUS.pgm"
parseorder="listInfo receiver">
<data name="receiver" type="struct" struct="autu0150" usage="output"
count="listInfo.rcdsReturned" outputsize="receiverLength" />
<data name="receiverLength" type="int"length="4"usage="input" init="16384"
/>
<data name="listInfo" type="struct" struct="listInfo" usage="output" />
<data name="rcdsToReturn" type="int"length="4"usage="input" init="264" />
<data name="format" type="char" length="10" usage="input" init="AUTU0150" />
<data name="selection"type="char" length="10" usage="input" init="*USER" />
<data name="member" type="char" length="10" usage="input" init="*NONE" />
<data name="errorCode"type="int"length="4"usage="input" init="0" />
```

```
</program>

<!-- Program QGYGTLE returned additional "records" from the list
 created by QGYOLAUS. -->
<program name="qgygtle" path="/QSYS.lib/QGY.lib/QGYGTLE.pgm"
parseorder="listInfo receiver">
<data name="receiver" type="struct" struct="autu0150" usage="output"
count="listInfo.rcdsReturned" outputsize="receiverLength" />
<data name="receiverLength" type="int"length="4"usage="input" init="16384"
/>
<data name="requestHandle"type="byte" length="4"usage="input" />
<data name="listInfo" type="struct" struct="listInfo" usage="output" />
<data name="rcdsToReturn" type="int"length="4"usage="input" init="264" />
<data name="startingRcd"type="int"length="4"usage="input" />
<data name="errorCode"type="int"length="4"usage="input" init="0" />
</program>

<!-- Program QGYCLST closes the list, freeing resources on the server -->
<program name="qgyclst" path="/QSYS.lib/QGY.lib/QGYCLST.pgm" >
<data name="requestHandle"type="byte" length="4"usage="input" />
<data name="errorCode"type="int"length="4"usage="input" init="0" />
</program>
</pcml>
```

呼叫 QGYOLAUS 的 Java 程式來源檔

```
import com.ibm.as400.data.ProgramCallDocument;
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;

// Example program to call "Retrieve List of Authorized Users" (QGYOLAUS)
API
public class qgyolaus
{

public static void main(String[] argv)
  {
AS400 as400System; // com.ibm.as400.access.AS400
ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
boolean rc = false;// Return code from ProgramCallDocument.callProgram()
String msgId, msgText; // Messages returned from the server
Object value;// Return value from ProgramCallDocument.getValue()

int[] indices = new int[1]; // Indices for access array value
int nbrRcds,// Number of records returned from QGYOLAUS and QGYGTLE
nbrUsers; // Total number of users retrieved
String listStatus;// Status of list on the server
byte[] requestHandle = new byte[4];

System.setErr(System.out);

// Construct AS400 without parameters, user will be prompted
as400System = new AS400();
```

```
 try
    {
// Uncomment the following to get debugging information
//com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

System.out.println("開始 PCML 範例..");
System.out.println("為 QGYOLAUS API 建構 ProgramCallDocument...");

// Construct ProgramCallDocument
// First parameter is system to connect to
// Second parameter is pcml resource name. In this example,
// serialized PCML file "qgyolaus.pcml.ser" or
// PCML source file "qgyolaus.pcml" must be found in the classpath.
pcml = new ProgramCallDocument(as400System, "qgyolaus");

// All input parameters have default values specified in the PCML source.
// Do not need to set them using Java code.

// Request to call the API
// User will be prompted to sign on to the system
System.out.println("為登入使用者呼叫 QGYOLAUS API 要求資訊。");
rc = pcml.callProgram("qgyolaus");

// If return code is false, we received messages from the server
if (rc == false)
      {
// Retrieve list of server messages
AS400Message[] msgs = pcml.getMessageList("qgyolaus");

// Iterate through messages and write them to standard output
for (int m = 0; m < msgs.length; m++)
        {
msgId = msgs[m].getID();
msgText = msgs[m].getText();
System.out.println("" + msgId + " - " + msgText);
        }
System.out.println("** Call to QGYOLAUS failed. See messages above **");
 System.exit(0);
      }
// Return code was true, call to QGYOLAUS succeeded
// Write some of the results to standard output
else
      {
boolean doneProcessingList = false;
String programName = "qgyolaus";
nbrUsers = 0;
while (!doneProcessingList)
        {
nbrRcds = pcml.getIntValue(programName + ".listInfo.rcdsReturned");
requestHandle = (byte[]) pcml.getValue(programName + ".listInfo.rqsHandle");

// Iterate through list of users
for (indices[0] = 0; indices[0] < nbrRcds; indices[0]++)
          {
 value = pcml.getValue(programName + ".receiver.name", indices);
```

```
 System.out.println("使用者： " + value);

 value = pcml.getValue(programName + ".receiver.description", indices);
 System.out.println("\t\t" + value);
          }

nbrUsers += nbrRcds;

// See if we retrieved all the users.
// If not, subsequent calls to "Get List Entries" (QGYGTLE)
// would need to be made to retrieve the remaining users in the list.
listStatus = (String) pcml.getValue(programName + ".listInfo.listStatus");
if ( listStatus.equals("2") // List is marked as "Complete"
|| listStatus.equals("3") ) // Or list is marked "Error building"
          {
doneProcessingList = true;
          }
else
          {
programName = "qgygtle";

// Set input parameters for QGYGTLE
pcml.setValue("qgygtle.requestHandle", requestHandle);
pcml.setIntValue("qgygtle.startingRcd", nbrUsers + 1);

// Call "Get List Entries" (QGYGTLE) to get more users from list
rc = pcml.callProgram("qgygtle");

// If return code is false, we received messages from the server
if (rc == false)
            {
// Retrieve list of server messages
AS400Message[] msgs = pcml.getMessageList("qgygtle");

// Iterate through messages and write them to standard output
for (int m = 0; m < msgs.length; m++)
              {
msgId = msgs[m].getID();
msgText = msgs[m].getText();
System.out.println("" + msgId + " - " + msgText);
              }
System.out.println("** 呼叫 QGYGTLE 失敗。請參閱上面的訊息 **");
 System.exit(0);
            }
// Return code was true, call to QGYGTLE succeeded

          }
        }
System.out.println("傳回的使用者數目：" + nbrUsers);

// Call the "Close List" (QGYCLST) API
pcml.setValue("qgyclst.requestHandle", requestHandle);
rc = pcml.callProgram("qgyclst");
      }
    }
catch(PcmlException e)
```

```
    {
System.out.println(e.getLocalizedMessage());
e.printStackTrace();
System.out.println("*** 呼叫 QGYOLAUS 失敗。***");
 System.exit(0);
    }

 System.exit(0);
  }
}
```

範例：擷取多維度資料

本範例有兩部份：

- 呼叫 QZNFRTVE 的 PCML 原始程式
- 呼叫 QZNFRTVE 的 Java 原始程式

呼叫 QZNFRTVE 的 PCML 原始程式

```
<pcml version="1.0">

<struct name="receiver">
<data name="lengthOfEntry"type="int"length="4" />
<data name="dispToObjectPathName" type="int"length="4" />
<data name="lengthOfObjectPathName" type="int"length="4" />
<data name="ccsidOfObjectPathName"type="int"length="4" />
<data name="readOnlyFlag" type="int"length="4" />
<data name="nosuidFlag" type="int"length="4" />
<data name="dispToReadWriteHostNames" type="int"length="4" />
<data name="nbrOfReadWriteHostNames"type="int"length="4" />
<data name="dispToRootHostNames"type="int"length="4" />
<data name="nbrOfRootHostNames" type="int"length="4" />
<data name="dispToAccessHostNames"type="int"length="4" />
<data name="nbrOfAccessHostNames" type="int"length="4" />
<data name="dispToHostOptions"type="int"length="4" />
<data name="nbrOfHostOptions" type="int"length="4" />
<data name="anonUserID" type="int"length="4" />
<data name="anonUsrPrf" type="char" length="10" />
<data name="pathName" type="char" length="lengthOfObjectPathName"
offset="dispToObjectPathName" offsetfrom="receiver" />

<struct name="rwAccessList" count="nbrOfReadWriteHostNames"
offset="dispToReadWriteHostNames" offsetfrom="receiver">
<data name="lengthOfEntry"type="int"length="4" />
<data name="lengthOfHostName" type="int"length="4" />
<data name="hostName" type="char" length="lengthOfHostName" />
<data type="byte" length="0"
offset="lengthOfEntry" />
</struct>

<struct name="rootAccessList" count="nbrOfRootHostNames"
offset="dispToRootHostNames" offsetfrom="receiver">
<data name="lengthOfEntry"type="int"length="4" />
<data name="lengthOfHostName" type="int"length="4" />
<data name="hostName" type="char" length="lengthOfHostName" />
<data type="byte" length="0"
offset="lengthOfEntry" />
</struct>

<struct name="accessHostNames" count="nbrOfAccessHostNames"
offset="dispToAccessHostNames" offsetfrom="receiver" >
<data name="lengthOfEntry"type="int"length="4" />
```

```
<data name="lengthOfHostName" type="int"length="4" />
<data name="hostName" type="char" length="lengthOfHostName" />
<data type="byte" length="0"
offset="lengthOfEntry" />
</struct>

<struct name="hostOptions" offset="dispToHostOptions" offsetfrom="receiver"
count="nbrOfHostOptions">
<data name="lengthOfEntry"type="int"length="4" />
<data name="dataFileCodepage" type="int"length="4" />
<data name="pathNameCodepage" type="int"length="4" />
<data name="writeModeFlag"type="int"length="4" />
<data name="lengthOfHostName" type="int"length="4" />
<data name="hostName" type="char" length="lengthOfHostName" />
<data type="byte" length="0"
offset="lengthOfEntry" />
</struct>

<data type="byte" length="0" offset="lengthOfEntry" />
</struct>

<struct name="returnedRcdsFdbkInfo">
<data name="bytesReturned"type="int" length="4" />
<data name="bytesAvailable" type="int" length="4" />
<data name="nbrOfNFSExportEntries"type="int" length="4" />
<data name="handle" type="int" length="4" />
</struct>

<program name="qznfrtve" path="/QSYS.lib/QZNFRTVE.pgm"
parseorder="returnedRcdsFdbkInfo receiver" >
<data name="receiver" type="struct" struct="receiver" usage="output"
count="returnedRcdsFdbkInfo.nbrOfNFSExportEntries"
outputsize="receiverLength"/>
<data name="receiverLength" type="int"length="4" usage="input" init="4096"
/>
<data name="returnedRcdsFdbkInfo" type="struct"
struct="returnedRcdsFdbkInfo" usage="output" />
<data name="formatName" type="char" length="8" usage="input" init="EXPE0100"
/>
<data name="objectPathName" type="char" length="lengthObjPathName"
usage="input" init="*FIRST" />
<data name="lengthObjPathName"type="int"length="4" usage="input" init="6" />
<data name="ccsidObjectPathName"type="int"length="4" usage="input" init="0"
/>
<data name="desiredCCSID" type="int"length="4" usage="input" init="0" />
<data name="handle" type="int"length="4" usage="input" init="0" />
<data name="errorCode"type="int"length="4" usage="input" init="0" />
</program>

</pcml>
```

呼叫 QZNFRTVE 的 Java 程式來源檔

```
import com.ibm.as400.data.ProgramCallDocument;
```

```
import com.ibm.as400.data.PcmlException;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;

// Example program to call "Retrieve NFS Exports" (QZNFRTVE) API
public class qznfrtve
{
public static void main(String[] argv)
  {
AS400 as400System; // com.ibm.as400.access.AS400
ProgramCallDocument pcml;// com.ibm.as400.data.ProgramCallDocument
boolean rc = false;// Return code from ProgramCallDocument.callProgram()
String msgId, msgText; // Messages returned from the server
Object value;// Return value from ProgramCallDocument.getValue()

System.setErr(System.out);

// Construct AS400 without parameters, user will be prompted
as400System = new AS400();

int[] indices = new int[2]; // Indices for access array value
int nbrExports; // Number of exports returned
int nbrOfReadWriteHostNames, nbrOfRWHostNames,
nbrOfRootHostNames,nbrOfAccessHostnames, nbrOfHostOpts;

 try
    {
// Uncomment the following to get debugging information
// com.ibm.as400.data.PcmlMessageLog.setTraceEnabled(true);

System.out.println("開始 PCML 範例..");
System.out.println("為 QZNFRTVE API 建構 ProgramCallDocument...");

// Construct ProgramCallDocument
// First parameter is system to connect to
// Second parameter is pcml resource name. In this example,
// serialized PCML file "qznfrtve.pcml.ser" or
// PCML source file "qznfrtve.pcml" must be found in the classpath.
pcml = new ProgramCallDocument(as400System, "qznfrtve");

// Set input parameters. Several parameters have default values
// specified in the PCML source. Do not need to set them using Java code.
System.out.println("設定輸入參數...");
pcml.setValue("qznfrtve.receiverLength", new Integer( (
pcml.getOutputsize("qznfrtve.receiver"))));

// Request to call the API
// User will be prompted to sign on to the system
System.out.println("呼叫 QZNFRTVE API 要求 NFS 匯出。");
rc = pcml.callProgram("qznfrtve");

if (rc == false)
      {
// Retrieve list of server messages
AS400Message[] msgs = pcml.getMessageList("qznfrtve");
```

```
// Iterate through messages and write them to standard output
for (int m = 0; m < msgs.length; m++)
        {
msgId = msgs[m].getID();
msgText = msgs[m].getText();
System.out.println("" + msgId + " - " + msgText);
        }
System.out.println("** 呼叫 QZNFRTVE 失敗。請參閱上面的訊息 **");
 System.exit(0);
      }
// Return code was true, call to QZNFRTVE succeeded
// Write some of the results to standard output
else
      {
nbrExports =
pcml.getIntValue("qznfrtve.returnedRcdsFdbkInfo.nbrOfNFSExportEntries");
// Iterate through list of exports
for (indices[0] = 0; indices[0] < nbrExports; indices[0]++)
        {
value = pcml.getValue("qznfrtve.receiver.pathName", indices);
System.out.println("路徑名稱 = " + value);

// Iterate and write out Read Write Host Names for this export
nbrOfReadWriteHostNames =
pcml.getIntValue("qznfrtve.receiver.nbrOfReadWriteHostNames", indices);
for(indices[1] = 0; indices[1] < nbrOfReadWriteHostNames; indices[1]++)
          {
value = pcml.getValue("qznfrtve.receiver.rwAccessList.hostName", indices);
System.out.println("讀取/寫入存取主電腦名稱 = " + value);
          }

// Iterate and write out Root Host Names for this export
nbrOfRootHostNames =
pcml.getIntValue("qznfrtve.receiver.nbrOfRootHostNames", indices);
for(indices[1] = 0; indices[1] < nbrOfRootHostNames; indices[1]++)
          {
value = pcml.getValue("qznfrtve.receiver.rootAccessList.hostName", indices);
System.out.println("Root 存取主電腦名稱 = " + value);
          }

// Iterate and write out Access Host Names for this export
nbrOfAccessHostnames =
pcml.getIntValue("qznfrtve.receiver.nbrOfAccessHostNames", indices);
for(indices[1] = 0; indices[1] < nbrOfAccessHostnames; indices[1]++)
          {
value = pcml.getValue("qznfrtve.receiver.accessHostNames.hostName",
indices);
System.out.println("存取主電腦名稱 = " + value);
          }

// Iterate and write out Host Options for this export
nbrOfHostOpts = pcml.getIntValue("qznfrtve.receiver.nbrOfHostOptions",
indices);
for(indices[1] = 0; indices[1] < nbrOfHostOpts; indices[1]++)
          {
```

```
System.out.println("主電腦選項：");
value = pcml.getValue("qznfrtve.receiver.hostOptions.dataFileCodepage",
indices);
System.out.println("資料檔字碼頁 = " + value);
value = pcml.getValue("qznfrtve.receiver.hostOptions.pathNameCodepage",
indices);
System.out.println("路徑名稱字碼頁 = " + value);
value = pcml.getValue("qznfrtve.receiver.hostOptions.writeModeFlag",
indices);
System.out.println("寫入模式旗號 = " + value);
value = pcml.getValue("qznfrtve.receiver.hostOptions.hostName", indices);
System.out.println("主電腦名稱 = " + value);
          }
} // end for loop iterating list of exports
} // end call to QZNFRTVE succeeded
    }
catch(PcmlException e)
    {
System.out.println(e.getLocalizedMessage());
e.printStackTrace();
 System.exit(-1);
    }

 System.exit(0);
} // end main()
}
```

範例：ReportWriter 類別

本節會列出 ReportWriter 類別文件中提供的程式碼範例。

JSPReportProcessor 及 PDFContext

- 範例：使用 JSPReportProcessor 與 PDFContext
- ≫範例：JSPReportProcessor範例 JSP 檔≪

XSLReportProcessor 及 PCLContext

- 範例：使用 XSLReportProcessor 與 PCLContext
- ≫範例：XSLReportProcessor 範例 XML 檔 ≪
- ≫範例：XSLReportProcessor 範例 XSL 檔≪

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

範例：使用 JSPReportProcessor 和 PDFContext

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// The following example (JSPRunReport) uses the JSPReportProcessor and the
// PDFContext classes to obtain data from a specified URL and convert the
data
// to the PDF format. The data is then streamed to a file as a PDF document.
//
// To view the contents of an example JSP source file that you can use with
// JSPRunReport, see     JSPcust_table.jsp    . You can also    download a zip file
// that contains the JSP example file. The zip file also contains XML and
XSL
// example files that you can use with the XSLReportProcessor example
// (PCLRunReport).
//
// Command syntax:
// java JSPRunReport <jsp_Url> <output_filename>
//
////////////////////////////////////////////////////////////////////////////

import java.lang.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.awt.print.*;
import java.awt.event.*;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.HashMap;

import com.ibm.xsl.composer.flo.*;
import com.ibm.xsl.composer.areas.*;
import com.ibm.xsl.composer.framework.*;
import com.ibm.xsl.composer.java2d.*;
import com.ibm.xsl.composer.prim.*;
import com.ibm.xsl.composer.properties.*;
import com.ibm.as400.util.reportwriter.processor.*;
import com.ibm.as400.util.reportwriter.pdfwriter.*;
import java.io.IOException;
import java.io.Serializable;
import org.xml.sax.SAXException;


public class JSPRunReport

{

public static void main(String args[])
    {
FileOutputStream fileout = null;

/** specify the URL that contains the data you want
```

```
to use in your report **/
String JSPurl = args[0];
URL jspurl = null;
try {
jspurl = new URL(JSPurl);
        }
catch (MalformedURLException e)
        {}

/** get output PDF file name**/
String filename = args[1];
try {
fileout = new FileOutputStream(filename);
        }
catch (FileNotFoundException e)
        {}

/** set up page format **/
Paper paper = new Paper();
paper.setSize(612,792);
paper.setImageableArea(18, 18, 576, 756);
PageFormat pf = new PageFormat();
pf.setPaper(paper);

/** create a PDFContext object and cast FileOutputStream
as an OutputStream **/
PDFContext pdfcontext = new PDFContext((OutputStream)fileout, pf);

System.out.println( Ready to parse XSL document );

/** create the JSPReportProcessor object and set the template
to the specified JSP **/
JSPReportProcessor jspprocessor = new JSPReportProcessor(pdfcontext);
try {
jspprocessor.setTemplate(jspurl);
        }

catch (NullPointerException np){
String mes = np.getMessage();
System.out.println(mes);
 System.exit(0);
            }

/** process the report **/
try {
jspprocessor.processReport();
        }
catch (IOException e) {
String mes = e.getMessage();
System.out.println(mes);
 System.exit(0);
            }
catch (SAXException se) {
String mes = se.getMessage();
System.out.println(mes);
 System.exit(0);
            }
```

```
 System.exit(0);
    }
}
```

»

範例：JSPReportProcessor 範例 JSP 檔案

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
<?xml version="1.0"?>

<!--
Copyright (c) 1999 The Apache Software Foundation.All rights
reserved.
-->

<%@ page session="false"%>
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.lang.*" %>
<%@ page import="java.util.*" %>


<%-- <jsp:useBean id='cust_table' scope='page' class='table.JSPcust_table'/>
--%>

<%!
 String[][] cust_data = new String [4][5];

 public void jspInit()
   {
 //cust_record_field [][] cust_data;
        // cust_record holds customer name, customer address, customer city,
customer state,
        // customer zip



        String [] cust_record_1 = {"IBM","3602 4th
St","Rochester","Mn","55901"};
String [] cust_record_2 = {"HP","400 2nd","Springfield","Mo","33559"};
String [] cust_record_3 = {"Wolzack","34 Hwy 52N","Lansing","Or","67895"};
String [] cust_record_4 = {"Siems","343 60th","Salem","Tx","12345"};

        cust_data[0] = cust_record_1;
        cust_data[1] = cust_record_2;
        cust_data[2] = cust_record_3;
        cust_data[3] = cust_record_4;
   }
%>


<!-- First test of parse and compose. -->
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master master-name="theMaster" >
<fo:region-body region-name="theRegion" margin-left=".2in"/>
</fo:simple-page-master>
```

```
<fo:page-sequence-master master-name="theMaster">
<fo:single-page-master-reference master-name="thePage"/>
</fo:page-sequence-master>
</fo:layout-master-set>
<fo:page-sequence master-name="theMaster">
 <fo:flow flow-name="theRegion">
 <fo:block>
<fo:block text-align="center"> NORCAP </fo:block>
<fo:block space-before=".2in" text-align="center"> PAN PACIFIC HOTEL IN SAN
FRANCISCO </fo:block>
<fo:block text-align="center"> FRIDAY, DECEMBER 8-9, 2000 </fo:block>
 </fo:block>
 <fo:block space-before=".5in" font-size="8pt">
 <fo:table table-layout="fixed">
 <fo:table-column column-width="3in"/>
 <fo:table-column column-width="3in"/>
 <fo:table-column column-width="3in"/>
 <fo:table-column column-width="3in"/>
 <fo:table-column column-width="3in"/>
 <fo:table-body>
 <fo:table-row>
<fo:table-cell column-number="1">
 <fo:block border-bottom-style="solid">NAME
 </fo:block>
</fo:table-cell>
<fo:table-cell column-number="2">
 <fo:block border-bottom-style="solid">ADDRESS
 </fo:block>
</fo:table-cell>
<fo:table-cell column-number="3">
 <fo:block border-bottom-style="solid">CITY
 </fo:block>
</fo:table-cell>
<fo:table-cell column-number="4">
 <fo:block border-bottom-style="solid">STATE
 </fo:block>
</fo:table-cell>
<fo:table-cell column-number="5">
 <fo:block border-bottom-style="solid">ZIP CODE
 </fo:block>
</fo:table-cell>
</fo:table-row>

            <%
// add row to table
for(int i = 0; i <= 3; i++)
              {
String[] _array = cust_data[i];
            %>

<fo:table-row>
<fo:table-cell column-number="1">
<fo:block space-before=".1in">
<% if(_array[0].equals("IBM")) { %>
 <fo:inline background-color="blue">
```

```
 <% out.print(_array[0]); %>
 </fo:inline>
<% } else { %>
 <% out.print(_array[0]); %>
                <% } %>
</fo:block>
</fo:table-cell>
<fo:table-cell column-number="2">
<fo:block space-before=".1in">
<% out.print(_array[1]); %>
</fo:block>
</fo:table-cell>
<fo:table-cell column-number="3">
<fo:block space-before=".1in">
<% out.print(_array[2]); %>
</fo:block>
</fo:table-cell>
<fo:table-cell column-number="4">
<fo:block space-before=".1in">
 <% out.print(_array[3]); %>
</fo:block>
</fo:table-cell>
<fo:table-cell column-number="5">
 <fo:block space-before=".1in">
<% out.print(_array[4]); %>
 </fo:block>
 </fo:table-cell>
</fo:table-row>

          <%
 } // end row while
          %>

 </fo:table-body>
 </fo:table>
 </fo:block>
 </fo:flow>
</fo:page-sequence>
</fo:root>
```

«

範例：使用 XSLReportProcessor 和 PCLContext

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// The following example (PCLRunReport) uses the XSLPReportProcessor and the
// PCLContext classes to obtain XML data and convert the data to the PCL
format.
// The data is then streamed to a printer OutputQueue.
//
// To view the contents of example XML and XSL source files that you can use
// with PCLRunReport, see      realestate.xml    and  realestate.xsl    . You can also
//  download a zip file     that contains the XML and XSL example files. The zip
// file also contains a JSP example file that you can use with the
// JSPReportProcessor example (JSPRunReport).
//
// Command syntax:
// java PCLRunReport <xml_file> <xsl_file>
//
////////////////////////////////////////////////////////////////////////////

import java.lang.*;
import java.awt.*;
import java.io.*;
import java.awt.print.*;
import java.awt.event.*;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.HashMap;

import com.ibm.xsl.composer.flo.*;
import com.ibm.xsl.composer.areas.*;
import com.ibm.xsl.composer.framework.*;
import com.ibm.xsl.composer.java2d.*;
import com.ibm.xsl.composer.prim.*;
import com.ibm.xsl.composer.properties.*;
import com.ibm.as400.util.reportwriter.processor.*;
import com.ibm.as400.util.reportwriter.pclwriter.*;
import java.io.IOException;
import java.io.Serializable;
import org.xml.sax.SAXException;
import com.ibm.as400.access.*;

public class PCLRunReport

{

public static void main(String args[])
    {
SpooledFileOutputStream fileout = null;
String xmldocumentName = args[0];
String xsldocumentName = args[1];

String sys = "<系統>";/* Insert ISeries system name */
```

```
String user = "<使用者>"; /* Insert ISeries user profile name */
String pass = "<密碼>"; /* Insert ISeries password*/

AS400 system = new AS400(sys, user, pass);

/* Insert ISeries output queue */
String outqname = "/QSYS.LIB/qusrsys.LIB/<outq>.OUTQ";
OutputQueue outq = new OutputQueue(system, outqname);
PrintParameterList parms = new PrintParameterList();
parms.setParameter(PrintObject.ATTR_OUTPUT_QUEUE, outq.getPath());

try{
fileout = new SpooledFileOutputStream(system, parms, null, null);
        }
catch (Exception e)
        {}

/** set up page format **/
Paper paper = new Paper();
paper.setSize(612,792);
paper.setImageableArea(18, 36, 576, 720);
PageFormat pf = new PageFormat();
pf.setPaper(paper);

/** create a PCLContext object and case FileOutputStream
as an OutputStream **/
PCLContext pclcontext = new PCLContext((OutputStream)fileout, pf);

System.out.println("準備剖析 XSL 文件");

/** create the XSLReportProcessor object **/
XSLReportProcessor xslprocessor = new XSLReportProcessor(pclcontext);
try {
xslprocessor.setXMLDataSource(xmldocumentName);
        }
catch (SAXException se) {
String mes = se.getMessage();
System.out.println(mes);
 System.exit(0);
            }
catch (IOException ioe) {
String mes = ioe.getMessage();
System.out.println(mes);
 System.exit(0);
            }
catch (NullPointerException np){
String mes = np.getMessage();
System.out.println(mes);
 System.exit(0);
            }
/** set the template to the specified XML data source **/
try {
xslprocessor.setTemplate(xsldocumentName);
        }
catch (NullPointerException np){
String mes = np.getMessage();
System.out.println(mes);
```

```
 System.exit(0);
            }
catch (IOException e) {
String mes = e.getMessage();
System.out.println(mes);
 System.exit(0);
            }
catch (SAXException se) {
String mes = se.getMessage();
System.out.println(mes);
 System.exit(0);
            }

/** process the report **/
try {
xslprocessor.processReport();
        }
catch (IOException e) {
String mes = e.getMessage();
System.out.println(mes);
 System.exit(0);
            }
catch (SAXException se) {
String mes = se.getMessage();
System.out.println(mes);
 System.exit(0);
            }

 System.exit(0);
    }

}
```

»

範例：XSLReportProcessor 範例 XML 檔案

附註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
<?xml version="1.0"?>

<RESIDENTIAL-LISTINGS VERSION="061698">

<RESIDENTIAL-LISTING ID="ID1287" VERSION="061698">
<GENERAL>
        <TYPE>Apartment</TYPE>
        <PRICE>$110,000</PRICE>
        <STRUCTURE><NUM-BEDS>3</NUM-BEDS><NUM-BATHS>1</NUM-
BATHS></STRUCTURE>
<AGE UNITS="YEARS">15</AGE>
<LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
        <ADDRESS>13 Some Avenue</ADDRESS>
<CITY>Dorchester</CITY><ZIP>02121</ZIP>
        </LOCATION>
<IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house1.jpg"/>
        <MLS>
<MLS-CODE SECURITY="Restricted">
                        30224877
                </MLS-CODE>
<MLS-SOURCE SECURITY="Public">
                        <NAME>Bob the Realtor</NAME>
                        <PHONE>1-617-555-1212</PHONE>
                        <FAX>1-617-555-1313</FAX>
                        <WEB>
                                <EMAIL>Bob@bigbucks.com</EMAIL>
                                <SITE>www.bigbucks.com</SITE>
                        </WEB>
                </MLS-SOURCE>
        </MLS>
        <DATES><LISTING-DATE>3/5/98</LISTING-DATE></DATES>
        <LAND-AREA UNITS="ACRES">0.01</LAND-AREA>

</GENERAL>

<FEATURES>
 <DISCLOSURES>
                In your dreams.
 </DISCLOSURES>
         <UTILITIES>
        Yes
         </UTILITIES>
         <EXTRAS>
        Pest control included.
         </EXTRAS>
         <CONSTRUCTION>
        Wallboard and glue
```

```
        </CONSTRUCTION>
        <ACCESS>
        Front door.
        </ACCESS>
</FEATURES>

<FINANCIAL>
        <ASSUMABLE>
        I assume so.
        </ASSUMABLE>
        <OWNER-CARRY>
        Too heavy.
        </OWNER-CARRY>
        <ASSESMENTS>
        $150,000
        </ASSESMENTS>
        <DUES>
        $100
        </DUES>
        <TAXES>
        $2,000
        </TAXES>
        <LENDER>
        Fly by nite mortgage co.
        </LENDER>
        <EARNEST>
        Burt
        </EARNEST>
        <DIRECTIONS>
        North, south, east, west
        </DIRECTIONS>
</FINANCIAL>

<REMARKS>
</REMARKS>

<CONTACTS>
        <COMPANY>
                <NAME>
                Noplace Realty
                </NAME>
                <ADDRESS>
                12 Main Street
                </ADDRESS>
                <CITY>
                Lowell, MA
                </CITY>
                <ZIP>
                34567
                </ZIP>
        </COMPANY>
        <AGENT>
                <NAME>
                Mary Jones
                </NAME>
```

```
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </AGENT>
        <OWNER>
                <NAME>
                </NAME>
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </OWNER>
<TENANT>
         Yes.
        </TENANT>
<COMMISION>
         15%
        </COMMISION>
</CONTACTS>

</RESIDENTIAL-LISTING>

<RESIDENTIAL-LISTING VERSION="061698" ID="ID1289">
<GENERAL>

<IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house2.jpg">
        </IMAGE>

        <MLS>
<MLS-CODE SECURITY="Restricted">
                                                30298877
                </MLS-CODE>
<MLS-SOURCE SECURITY="Public">
                        <NAME>
                                Mary the Realtor
                        </NAME>
                        <PHONE>
                                1-617-555-3333
                        </PHONE>
                        <FAX>
                                1-617-555-4444
                        </FAX>
                        <WEB>
                                <EMAIL>
                                Mary@somebucks.com
                                </EMAIL>
                                <SITE>
                                www.bigbucks.com
                                </SITE>
                        </WEB>
```

```
                </MLS-SOURCE>
        </MLS>

        <TYPE>
                Home
        </TYPE>

        <PRICE>
                $200,000
        </PRICE>

<AGE UNITS="MONTHS">
                3
        </AGE>

<LOCATION COUNTRY="USA" STATE="CO" COUNTY="MIDDLESEX" SECURITY="Public">
                <ADDRESS>
                        1 Main Street
                </ADDRESS>
                <CITY>
                Boulder
                </CITY>
                <ZIP>
                        11111
                </ZIP>
        </LOCATION>

        <STRUCTURE>
        <NUM-BEDS>
        2
</NUM-BEDS>
        <NUM-BATHS>
        2
</NUM-BATHS>
        </STRUCTURE>

        <DATES>
                <LISTING-DATE>
                4/3/98
</LISTING-DATE>
        </DATES>

<LAND-AREA UNITS="ACRES">
                0.01
        </LAND-AREA>

</GENERAL>

<FEATURES>
 <DISCLOSURES>
                In your dreams.
 </DISCLOSURES>
         <UTILITIES>
        Yes
         </UTILITIES>
         <EXTRAS>
```

```
        Pest control included.
         </EXTRAS>
         <CONSTRUCTION>
        Wallboard and glue
         </CONSTRUCTION>
         <ACCESS>
        Front door.
         </ACCESS>
</FEATURES>

<FINANCIAL>
         <ASSUMABLE>
        I assume so.
         </ASSUMABLE>
         <OWNER-CARRY>
        Too heavy.
         </OWNER-CARRY>
         <ASSESMENTS>
        $150,000
         </ASSESMENTS>
         <DUES>
        $100
         </DUES>
         <TAXES>
        $2,000
         </TAXES>
         <LENDER>
        Fly by nite mortgage co.
         </LENDER>
         <EARNEST>
        Burt
         </EARNEST>
         <DIRECTIONS>
        North, south, east, west
         </DIRECTIONS>
</FINANCIAL>

<REMARKS>
</REMARKS>

<CONTACTS>
        <COMPANY>
                <NAME>
                 Noplace Realty
                </NAME>
                <ADDRESS>
                 12 Main Street
                </ADDRESS>
                <CITY>
                 Lowell, MA
                </CITY>
                <ZIP>
                 34567
                </ZIP>
        </COMPANY>
```

```
        <AGENT>
                <NAME>
                 Mary Jones
                </NAME>
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </AGENT>
        <OWNER>
                <NAME>
                </NAME>
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </OWNER>
<TENANT>
         Yes.
        </TENANT>
<COMMISION>
         15%
        </COMMISION>
</CONTACTS>

</RESIDENTIAL-LISTING>
<RESIDENTIAL-LISTING VERSION="061698" ID="ID1290">
<GENERAL>

<IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house3.jpg">
        </IMAGE>

        <MLS>
<MLS-CODE SECURITY="Restricted">
                        20079877
                </MLS-CODE>
<MLS-SOURCE SECURITY="Public">
                        <NAME>
                                Bob the Realtor
                        </NAME>
                        <PHONE>
                                1-617-555-1212
                        </PHONE>
                        <FAX>
                                1-617-555-1313
                        </FAX>
                        <WEB>
                                <EMAIL>
                                Bob@bigbucks.com
                                </EMAIL>
                                <SITE>
```

```
                                www.bigbucks.com
                                </SITE>
                        </WEB>
                </MLS-SOURCE>
        </MLS>

        <TYPE>
                Apartment
        </TYPE>

        <PRICE>
                $65,000
        </PRICE>

<AGE UNITS="YEARS">
                30
        </AGE>

<LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
                <ADDRESS>
                        25 Which Ave.
                </ADDRESS>
                <CITY>
                Cambridge
                </CITY>
                <ZIP>
                        02139
                </ZIP>
        </LOCATION>

        <STRUCTURE>
        <NUM-BEDS>
        3
</NUM-BEDS>
        <NUM-BATHS>
        1
</NUM-BATHS>
        </STRUCTURE>

        <DATES>
                <LISTING-DATE>
                3/5/97
</LISTING-DATE>
        </DATES>

<LAND-AREA UNITS="ACRES">
                0.05
        </LAND-AREA>

</GENERAL>

<FEATURES>
 <DISCLOSURES>
                In your dreams.
 </DISCLOSURES>
         <UTILITIES>
```

```
        Yes
         </UTILITIES>
         <EXTRAS>
        Pest control included.
         </EXTRAS>
         <CONSTRUCTION>
        Wallboard and glue
         </CONSTRUCTION>
         <ACCESS>
        Front door.
         </ACCESS>
</FEATURES>

<FINANCIAL>
         <ASSUMABLE>
        I assume so.
         </ASSUMABLE>
         <OWNER-CARRY>
        Too heavy.
         </OWNER-CARRY>
         <ASSESMENTS>
        $150,000
         </ASSESMENTS>
         <DUES>
        $100
         </DUES>
         <TAXES>
        $2,000
         </TAXES>
         <LENDER>
        Fly by nite mortgage co.
         </LENDER>
         <EARNEST>
        Burt
         </EARNEST>
         <DIRECTIONS>
        North, south, east, west
         </DIRECTIONS>
</FINANCIAL>

<REMARKS>
</REMARKS>

<CONTACTS>
        <COMPANY>
                <NAME>
                 Noplace Realty
                </NAME>
                <ADDRESS>
                 12 Main Street
                </ADDRESS>
                <CITY>
                 Lowell, MA
                </CITY>
                <ZIP>
```

```
                34567
                </ZIP>
        </COMPANY>
        <AGENT>
                <NAME>
                 Mary Jones
                </NAME>
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </AGENT>
        <OWNER>
                <NAME>
                </NAME>
                <ADDRESS>
                </ADDRESS>
                <CITY>
                </CITY>
                <ZIP>
                </ZIP>
        </OWNER>
<TENANT>
         Yes.
        </TENANT>
<COMMISION>
         15%
        </COMMISION>
</CONTACTS>

</RESIDENTIAL-LISTING>
<RESIDENTIAL-LISTING VERSION="061698"ID="ID1291">
<GENERAL>

<IMAGE FORMAT="JPG" WIDTH="300" HEIGHT="150" SRC="house4.jpg">
        </IMAGE>

        <MLS>
<MLS-CODE SECURITY="Restricted">
                        29389877
                </MLS-CODE>
<MLS-SOURCE SECURITY="Public">
                        <NAME>
                                Mary the Realtor
                        </NAME>
                        <PHONE>
                                1-617-555-3333
                        </PHONE>
                        <FAX>
                                1-617-555-4444
                        </FAX>
                        <WEB>
                                <EMAIL>
```

```
                                        Mary@somebucks.com
                                        </EMAIL>
                                        <SITE>
                                        www.bigbucks.com
                                        </SITE>
                                </WEB>
                        </MLS-SOURCE>
                </MLS>

                <TYPE>
Home
                </TYPE>

                <PRICE>
                        $449,000
                </PRICE>

<AGE UNITS="YEARS">
                        7
                </AGE>

<LOCATION COUNTRY="USA" STATE="MA" COUNTY="MIDDLESEX" SECURITY="Public">
                        <ADDRESS>
                                100 Any Road
                        </ADDRESS>
                        <CITY>
                        Lexington
                        </CITY>
                        <ZIP>
                                02421
                        </ZIP>
                </LOCATION>

                <STRUCTURE>
                <NUM-BEDS>
                7
</NUM-BEDS>
                <NUM-BATHS>
                3
</NUM-BATHS>
                </STRUCTURE>

                <DATES>
                        <LISTING-DATE>
                        6/8/98
</LISTING-DATE>
                </DATES>

<LAND-AREA UNITS="ACRES">
                        2.0
                </LAND-AREA>

</GENERAL>

<FEATURES>
 <DISCLOSURES>
```

```
                In your dreams.
 </DISCLOSURES>
          <UTILITIES>
         Yes
          </UTILITIES>
          <EXTRAS>
         Pest control included.
          </EXTRAS>
          <CONSTRUCTION>
         Wallboard and glue
          </CONSTRUCTION>
          <ACCESS>
         Front door.
          </ACCESS>
</FEATURES>

<FINANCIAL>
          <ASSUMABLE>
         I assume so.
          </ASSUMABLE>
          <OWNER-CARRY>
         Too heavy.
          </OWNER-CARRY>
          <ASSESMENTS>
         $300,000
          </ASSESMENTS>
          <DUES>
         $100
          </DUES>
          <TAXES>
         $2,000
          </TAXES>
          <LENDER>
         Fly by nite mortgage co.
          </LENDER>
          <EARNEST>
         Burt
          </EARNEST>
          <DIRECTIONS>
         North, south, east, west
          </DIRECTIONS>
</FINANCIAL>

<REMARKS>
</REMARKS>

<CONTACTS>
         <COMPANY>
                  <NAME>
                   Noplace Realty
                  </NAME>
                  <ADDRESS>
                   12 Main Street
                  </ADDRESS>
                  <CITY>
```

```
                Lowell, MA
               </CITY>
               <ZIP>
                34567
               </ZIP>
          </COMPANY>
          <AGENT>
               <NAME>
                Mary Jones
               </NAME>
               <ADDRESS>
               </ADDRESS>
               <CITY>
               </CITY>
               <ZIP>
               </ZIP>
          </AGENT>
          <OWNER>
               <NAME>
               </NAME>
               <ADDRESS>
               </ADDRESS>
               <CITY>
               </CITY>
               <ZIP>
               </ZIP>
          </OWNER>
<TENANT>
           Yes.
          </TENANT>
<COMMISION>
           15%
          </COMMISION>
</CONTACTS>

</RESIDENTIAL-LISTING>

</RESIDENTIAL-LISTINGS>
```

«

»

範例：XSLReportProcessor 範例 XSL 檔案

附註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
<?xml version="1.0"?>

<!-- Sample of styling an imagined real estate document. -->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:fo="http://www.w3.org/1999/XSL/Format" >

<xsl:template match="RESIDENTIAL-LISTINGS">
<fo:root>
<fo:layout-master-set>
        <fo:simple-page-master master-name="theMaster">
 <fo:region-body region-name="theRegion"/>
        </fo:simple-page-master>
        <fo:page-sequence-master master-name="theMaster">
         <fo:single-page-master-reference master-name="thePage" />
</fo:page-sequence-master>
        </fo:layout-master-set>
         <fo:page-sequence master-name="theMaster">
         <fo:flow flow-name="theRegion">
 <xsl:apply-templates/>
        </fo:flow>
        </fo:page-sequence>
</fo:root>
</xsl:template>
<xsl:template match="RESIDENTIAL-LISTING">


<fo:block font-family="Times New Roman" font-weight="normal" font-
size="24pt"
 background-color="silver" padding-before="5px" padding-after="5px"
 padding-start="5px" padding-end="5px" border-before-style="solid"
 border-before-color="blue" border-after-style="solid" border-after-
color="blue"
 border-start-style="solid" border-start-color="blue" border-end-
style="solid"
 border-end-color="blue">

        <fo:character character="y" background-color="blue" border-before-
style="solid"
 border-before-color="yellow" border-after-style="solid" border-after-
color="yellow"
 border-start-style="solid" border-start-color="yellow" border-end-
style="solid"
 border-end-color="yellow" />
        </fo:block>
```

```
</xsl:template>
</xsl:stylesheet>
```

«

範例：資源類別

本節會列出資源類別文件中提供的程式碼範例。

資源與 ChangeableResource

- 範例：從 RUser（資源的具體子類別）擷取屬性
- 範例：設定 RJob（ChangeableResource 的具體子類別）的屬性值
- 範例：使用同屬程式碼來存取資源

ResourceList

- 範例：取得及列印 ResourceList 的內容
- 範例：使用同屬程式碼來存取 ResourceList
- 範例：在 servlet 中顯示資源清單

簡報

- 範例：使用簡報

範例：從 Resource 擷取屬性值

Resource 的具體次類別有 com.ibm.as400.resource.RUser，它代表 iSeries 使用者。RUser 支援許多 屬性，每個 ID 都可以用來取得屬性值。

以下是從 RUser 擷取屬性值的範例：

```
   // Create an RUser object to refer to a specific user.
   AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
   RUser user = new RUser(system, "AUSERID");

   // Get the text description attribute value.
   String textDescription =
(String)user.getAttributeValue(RUser.TEXT_DESCRIPTION);
```

範例：為 ChangeableResource 設定屬性值

ChangeableResource 的具體次類別是 com.ibm.as400.resource.RJob，它代表一項 iSeries 工作。RJob 支援許多屬性 ID，每個 ID 都可以用來存取屬性值。以下範例為 RJob 設定兩個屬性值：

```
// Create an RJob object to refer to a specific job.
AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
RJob job = new RJob(system, "AJOBNAME", "AUSERID", "AJOBNUMBER");

// Set the date format attribute value.
job.setAttributeValue(RJob.DATE_FORMAT, RJob.DATE_FORMAT_JULIAN);

// Set the country or region ID attribute value.
job.setAttributeValue(RJob.COUNTRY_ID, RJob.USER_PROFILE);

// Commit both attribute changes.
job.commitAttributeChanges();
```

範例：使用同屬程式碼來存取資源

您可以撰寫同屬程式碼來使用任何 Resource 或 ChangeableResource 次類別。
這類程式碼可增進重新使用和維護的能力，而且不必修改即可使用於未來的 Resource 或 ChangeableResource 次類別。

每個屬性都有相關的屬性 meta 資料物件（com.ibm.as400.resource.ResourceMetaData），可說明屬性的各種內容。這些內容包括屬性是否為唯讀，以及預設和可能的值有哪些。

以下同屬程式碼範例可列印出資源支援的每一屬性的值。

```
void printAllAttributeValues(Resource resource) throws ResourceException
{
// Get the attribute meta data.
ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();


// Loop through all attributes and print the values.
for(int i = 0; i < attributeMetaData.length; ++i)
    {
Object attributeID = attributeMetaData[i].getID();
Object value = resource.getAttributeValue(attributeID);
System.out.println("Attribute " + attributeID + " = " + value);
    }
}
```

以下同屬程式碼範例可將 ChangeableResource 的所有屬性重設成為預設值：

```
void resetAttributeValues(ChangeableResource resource) throws
ResourceException
{
// Get the attribute meta data.
ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();


// Loop through all attributes.
for(int i = 0; i < attributeMetaData.length; ++i)
    {
// If the attribute is changeable (not read only), then
// reset its value to the default.
if (! attributeMetaData[i].isReadOnly())
        {
Object attributeID = attributeMetaData[i].getID();
Object defaultValue = attributeMetaData[i].getDefaultValue();
resource.setAttributeValue(attributeID, defaultValue);
        }
    }

// Commit all of the attribute changes.
resource.commitAttributeChanges();
}
```

範例：資源清單

以下範例說明使用資源清單的各種方式：

- 範例： 取得和列印 ResourceList 的內容
- 範例： 使用同屬程式碼來存取 ResourceList
- 範例 在 servlet 中顯示資源清單

範例：取得和列印 ResourceList 的內容

ResourceList 具體次類別的一個範例就是com.ibm.as400.resource.RJobList，代表 iSeries 工作的清單。RJobList 支援許選項 ID和 排序 ID，每一個都可以用來將清單加以過濾或排序。本範例列印出 RJobList 的內容：

```
// Create an RJobList object to represent a list of jobs.
AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
RJobList jobList = new RJobList(system);

// Filter the list to include only interactive jobs.
jobList.setSelectionValue(RJobList.JOB_TYPE, RJob.JOB_TYPE_INTERACTIVE);

// Sort the list by user name, then job name.
Object[] sortValue = new Object[] { RJob.USER_NAME, RJob.JOB_NAME };
jobList.setSortValue(sortValue);

// Open the list and wait for it to complete.
jobList.open();
jobList.waitForComplete();

// Read and print the contents of the list.
long length = jobList.getListLength();
for(long i = 0; i < length; ++i)
{
    System.out.println(jobList.resourceAt(i));
}

// Close the list.
jobList.close();
```

範例：使用同屬程式碼來使用資源

除了直接使用具體次類別以外，您還可以撰寫同屬程式碼來使用 任何 ResourceList 次類別。這類程式碼可增進重新使用和維護的能力， 而且不必修改即可使用於未來的 ResourceList 次類別。

範例：列印 ResourceList 的內容

以下的同屬程式碼範例可列印出 ResourceList 的部份內容：

```
void printContents(ResourceList resourceList, long numberOfItems) throws
```

```
ResourceException
   {
       // Open the list and wait for the requested number of items
       // to become available.
       resourceList.open();
       resourceList.waitForResource(numberOfItems);

       for(long i = 0; i < numberOfItems; ++i)
       {
           System.out.println(resourceList.resourceAt(i));
       }
   }
```

範例：使用 ResourceMetaData 來存取資源支援的每一個屬性

每個屬性都有相關的屬性 meta 資料物件 (com.ibm.as400.resource.ResourceMetaData)，可說明屬性的各種內容。這些內容包括屬性是否為唯讀，以及預設和可能的值有哪些。

以下同屬程式碼範例可列印出資源支援的每一屬性的值。

```
void printAllAttributeValues(Resource resource) throws ResourceException
{
    // Get the attribute meta data.
    ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();


    // Loop through all attributes and print the values.
    for(int i = 0; i < attributeMetaData.length; ++i)
    {
        Object attributeID = attributeMetaData[i].getID();
        Object value = resource.getAttributeValue(attributeID);
        System.out.println("Attribute " + attributeID + " = " + value);
    }
}
```

範例：使用 ResourceMetaData 來重設 ChangeableResource 的每一個屬性

以下同屬程式碼範例可將 ChangeableResource 的所有屬性重設成為預設值：

```
void resetAttributeValues(ChangeableResource resource) throws
ResourceException
{
    // Get the attribute meta data.
    ResourceMetaData[] attributeMetaData = resource.getAttributeMetaData();


    // Loop through all attributes.
    for(int i = 0; i < attributeMetaData.length; ++i)
    {
        // If the attribute is changeable (not read only), then
        // reset its value to the default.
```

```
        if (! attributeMetaData[i].isReadOnly())
        {
            Object attributeID = attributeMetaData[i].getID();
            Object defaultValue = attributeMetaData[i].getDefaultValue();
            resource.setAttributeValue(attributeID, defaultValue);
        }
    }

    // Commit all of the attribute changes.
    resource.commitAttributeChanges();
}
```

範例：在 servlet 中顯示資源清單

請使用 ResourceListRowData 類別搭配 HTMLFormConverter 或 HTMLTableConverter 類別，即可在 servlet 中顯示資源清單。

- HTMLFormConverter 會把資源清單的內容顯示成一系列的格式，其中每份格式各包含有資源清單中的資源屬性值。
- HTMLTableConverter 會把資源清單的內容顯示成表格，每一列各包含有資源清單中的資源的相關資訊。

ResourceListRowData 物件用的直欄指定為直欄屬性 ID 的陣列，每一列各代表資源物件。

```
   // Create the resource list.  This example creates
   // a list of all messages in the current user's message
   // queue.
   AS400 system = new AS400("MYSYSTEM", "MYUSERID", "MYPASSWORD");
   RMessageQueue messageQueue = new RMessageQueue(system,
RMessageQueue.CURRENT);

   // Create the ResourceListRowData object.  In this example,
   // there are four columns in the table.  The first column
   // contains the icons and names for each message in the
   // message queue.  The remaining columns contain the text,
   // severity, and type for each message.
   ResourceListRowData rowdata = new ResourceListRowData(messageQueue,
       new Object[] { null, RQueuedMessage.MESSAGE_TEXT,
RQueuedMessage.MESSAGE_SEVERITY,
                      RQueuedMessage.MESSAGE_TYPE } );

   // Create HTMLTable and HTMLTableConverter objects to
   // use for generating and customizing the HTML tables.
         HTMLTable table = new HTMLTable();
   table.setCellSpacing(6);
   table.setBorderWidth(8);

      HTMLTableConverter converter = new HTMLTableConverter();
      converter.setTable(table);
   converter.setUseMetaData(true);

   // Generate the HTML table.
   String[] html = converter.convert(rowdata);
         System.out.println(html[0]);
```

範例：RFML

本節會列出 RFML 文件中提供的程式碼範例。

- 範例：使用 RFML 相對於使用 Toolbox for Java Record 類別
- 範例：RFML 來源檔

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

> 程式碼不保事項聲明
>
> IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
> 您可以依據這些範例，產生類似的函數，來符合您的需要。
>
> IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM 不擔保或默示保證這些程式的可靠性、可用性或功能。
>
> 所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
> 明確聲明不提供有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用設定檔記號認證來交換 OS/400 緒識別

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

下列程式碼範例告訴您如何使用設定檔記號認證， 來交換 OS/400 緒識別，並以特定使用者身份執行工作：

```
// Prepare to work with the local AS/400 system.
AS400 system = new AS400("localhost", "*CURRENT", "*CURRENT");

// Create a single-use ProfileTokenCredential with a 60 second timeout.
// A valid user ID and password must be substituted.
ProfileTokenCredential pt = new ProfileTokenCredential();
pt.setSystem(system);
pt.setTimeoutInterval(60);
pt.setTokenType(ProfileTokenCredential.TYPE_SINGLE_USE);

pt.setToken("USERID", "PASSWORD");

// Swap the OS/400 thread identity, retrieving a credential to
// swap back to the original identity later.
AS400Credential cr = pt.swap(true);

// Perform work under the swapped identity at this point.

// Swap back to the original OS/400 thread identity.
cr.swap();

// Clean up the credentials.
cr.destroy();
pt.destroy();
```

Servlet 類別範例

下面範例會顯示使用 Servlet 類別的一些方式：

- 範例：使用 ListRowData 類別
- 範例：使用 RecordListRowData 類別
- 範例：使用 SQLResultSetRowData 類別
- 範例：使用 HTMLFormConverter 類別
- 範例：使用 ListMetaData 類別
- 範例：使用 SQLResultSetMetaData 類別
- 範例：在 servlet 中顯示資源清單

您也可以將 Servlet 和HTML 類別一起使用，如同本範例所示。

以下不保聲明適用於所有的 IBM Toolbox for Java 範例：

範例：使用 ListRowData

本範例包含三個部份：

- 說明 ListRowData 類別如何運作的JAVA 原始程式
- 使用HTMLTableConverter 從 JAVA 原始程式產生的HTML 來源檔
- 瀏覽器如何顯示產生的 HTML

說明 ListRowData 類別如何運作的 JAVA 原始程式

```
        // Access an existing non-empty data queue
        KeyedDataQueue dq = new KeyedDataQueue(systemObject_,
"/QSYS.LIB/MYLIB.LIB/MYDQ.DTAQ");

        // Create a metadata object.
        ListMetaData metaData = new ListMetaData(2);

        // Set first column to be the customer ID.
        metaData.setColumnName(0, "客戶 ID");
        metaData.setColumnLabel(0, "客戶 ID");
        metaData.setColumnType(0, RowMetaDataType.STRING_DATA_TYPE);

        // Set second column to be the order to be processed.
        metaData.setColumnName(1, "排序號碼");
        metaData.setColumnLabel(1, "排序號碼");
        metaData.setColumnType(1, RowMetaDataType.STRING_DATA_TYPE);

        // Create a ListRowData object.
        ListRowData rowData = new ListRowData();
        rowData.setMetaData(metaData);

        // Get the entries off the data queue.
        KeyedDataQueueEntry data = dq.read(key, 0, "EQ");
        while (data != null)
        {
           // Add queue entry to row data object.
           Object[] row = new Object[2];
           row[0] = new String(key);
           row[1] = new String(data.getData());
           rowData.addRow(row);

           // Get another entry from the queue.
           data = dq.read(key, 0, "EQ");
        }

        // Create an HTML converter object and convert the rowData to HTML.
        HTMLTableConverter conv = new HTMLTableConverter();
        conv.setUseMetaData(true);
        HTMLTable[] html = conv.convertToTables(rowData);

        // Display the output from the converter.
        System.out.println(html[0]);
```

使用 HTMLTableConverter 從 JAVA 原始程式產生的 HTML 來源檔

在以上的 JAVA 原始程式範例中使用 HTMLTableConverter 類別會產生以下 HTML 程式碼。

```
<table>
<tr>
<th>Customer ID</th>
<th>Order Number</th>
</tr>
<tr>
<td>777-53-4444</td>
<td>12345-XYZ</td>
</tr>
<tr>
<td>777-53-4444</td>
<td>56789-ABC</td>
</tr>
</table>
```

瀏覽器如何顯示產生的 HTML

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

Customer ID	Order Number
777-53-4444	12345-XYZ
777-53-4444	56789-ABC

範例：使用 RecordListRowData

本範例包含三個部份：

- Java 原始程式它說明 RecordListRowData 類別如何運作
- HTML 原始程式，使用HTMLTableConverter 從 Java 原始程式產生
- 瀏覽器如何顯示產生的 HTML

說明 RecordListRowData 類別如何運作的 Java 原始程式

```
        // Create a server object.
        AS400 mySystem = new AS400 ("mySystem.myComp.com", "使用者 ID",
"密碼");

        // Get the path name for the file.
        QSYSObjectPathName file = new QSYSObjectPathName(myLibrary, myFile,
"%first%", "mbr");
        String ifspath = file.getPath();

        // Create a file object that represents the file.
        SequentialFile sf = new SequentialFile(mySystem, ifspath);

        // Retrieve the record format from the file.
        AS400FileRecordDescription recordDescription = new
AS400FileRecordDescription(mySystem, ifspath);
        RecordFormat recordFormat =
recordDescription.retrieveRecordFormat()[0];

        // Set the record format for the file.
        sf.setRecordFormat(recordFormat);

        // Get the records in the file.
        Record[] records = sf.readAll();

        // Create a RecordListRowData object and add the records.
        RecordListRowData rowData = new RecordListRowData(recordFormat);

        for (int i=0; i < records.length; i++)
        {
           rowData.addRow(records[i]);
        }

        // Create an HTML converter object and convert the rowData to HTML.
        HTMLTableConverter conv = new HTMLTableConverter();
        conv.setMaximumTableSize(3);
        HTMLTable[] html = conv.convertToTables(rowData);

        // Display the first HTML table generated by the converter.
        System.out.println(html[0]);
```

使用 HTMLTableConverter 從 JAVA 原始程式產生的 HTML 來源檔

在以上的 JAVA 原始程式範例中使用HTMLTableConverter 類別會產生以下 HTML 程式碼。

```
<table>
<tr>
<th>CUSNUM</th>
<th>LSTNAM</th>
<th>INIT</th>
<th>STREET</th>
<th>CITY</th>
<th>STATE</th>
<th>ZIPCOD</th>
<th>CDTLMT</th>
<th>CHGCOD</th>
<th>BALDUE</th>
<th>CDTDUE</th>
</tr>
<tr>
<td>938472</td>
<td>Henning </td>
<td>G K</td>
<td>4859 Elm Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75217</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">37.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones   </td>
<td>B D</td>
<td>21B NW 135 St</td>
<td>Clay  </td>
<td>NY</td>
<td align="right">13041</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">100.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine    </td>
<td>S S</td>
<td>PO Box 79    </td>
<td>Broton</td>
<td>VT</td>
<td align="right">5046</td>
<td align="right">700</td>
<td align="right">1</td>
<td align="right">439.00</td>
```

```
<td align="right">0.00</td>
</tr>
</table>
```

瀏覽器如何顯示產生的 HTML

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

CUSNUM	LSTNAM	INIT	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
938472	Henning	G K	4859 Elm Ave	Dallas	TX	75217	5000	3	37.00	0.00
839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
392859	Vine	S S	PO Box 79	Broton	VT	5046	700	1	439.00	0.00

範例：使用 SQLResultSetRowData

本範例包含三個部份：

- JAVA 原始程式, 顯示 SQLResultSetRowData 類別運作的方式
- HTML 來源檔 , 使用HTMLTableConverter 從 JAVA 原始程式產生
- 瀏覽器顯示產生之 HTML 的方法

JAVA 原始程式, 顯示 SQLResultSetRowData 類別運作的方式

```
 // Create a server object.
 AS400 mySystem = new AS400 ("mySystem.myComp.com", "UserId", "Password");

 // Register and get a connection to the database.

DriverManager.registerDriver (new com.ibm.as400.access.AS400JDBCDriver
());
 Connection connection = DriverManager.getConnection("jdbc:as400://" +
mySystem.getSystemName());

 // Execute an SQL statement and get the result set.
 Statement statement = connection.createStatement();
 statement.execute("select * from qiws.qcustcdt");
 ResultSet resultSet = statement.getResultSet();

 // Create the SQLResultSetRowData object and initialize to the result set.
 SQLResultSetRowData rowData = new SQLResultSetRowData(resultSet);

 // Create an HTML table object to be used by the converter.
 HTMLTable table = new HTMLTable();

 // Set descriptive column headers.
 String[] headers = {"Customer Number", "Last Name", "Initials",
 "Street Address", "City", "State", "Zip Code",
 "Credit Limit", "Charge Code", "Balance Due",
 "Credit Due"};
 table.setHeader(headers);

 // Set several formatting options within the table.
 table.setBorderWidth(2);
 table.setCellSpacing(1);
 table.setCellPadding(1);

 // Create an HTML converter object and convert the rowData to HTML.
 HTMLTableConverter conv = new HTMLTableConverter();
 conv.setTable(table);
 HTMLTable[] html = conv.convertToTables(rowData);

 // Display the HTML table generated by the converter.
 System.out.println(html[0]);
```

使用 HTMLTableConverter 從 JAVA 原始程式產生的 HTML 來源檔

在以上的 JAVA 原始程式範例中使用HTMLTableConverter 類別會產生下列 HTML 程式碼。

```
<table border="2" cellpadding="1" cellspacing="1">
```

```
<tr>
<th>客戶編號</th>
<th>姓氏</th>
<th>姓名縮寫</th>
<th>街道地址</th>
<th>城市</th>
<th>州</th>
<th>郵遞區號</th>
<th>信用額度</th>
<th>收費碼</th>
<th>待繳總額</th>
<th>待繳循環信用利息</th>
</tr>
<tr>
<td>938472</td>
<td>Henning </td>
<td>G K</td>
<td>4859 Elm Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75217</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">37.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>839283</td>
<td>Jones </td>
<td>B D</td>
<td>21B NW 135 St</td>
<td>Clay</td>
<td>NY</td>
<td align="right">13041</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">100.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>392859</td>
<td>Vine</td>
<td>S S</td>
<td>PO Box 79</td>
<td>Broton</td>
<td>VT</td>
<td align="right">5046</td>
<td align="right">700</td>
<td align="right">1</td>
<td align="right">439.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>938485</td>
<td>Johnson </td>
<td>J A</td>
<td>3 Alpine Way </td>
<td>Helen </td>
<td>GA</td>
```

```
<td align="right">30545</td>
<td align="right">9999</td>
<td align="right">2</td>
<td align="right">3987.50</td>
<td align="right">33.50</td>
</tr>
<tr>
<td>397267</td>
<td>Tyron </td>
<td>W E</td>
<td>13 Myrtle Dr </td>
<td>Hector</td>
<td>NY</td>
<td align="right">14841</td>
<td align="right">1000</td>
<td align="right">1</td>
<td align="right">0.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>389572</td>
<td>Stevens </td>
<td>K L</td>
<td>208 Snow Pass</td>
<td>Denver</td>
<td>CO</td>
<td align="right">80226</td>
<td align="right">400</td>
<td align="right">1</td>
<td align="right">58.75</td>
<td align="right">1.50</td>
</tr>
<tr>
<td>846283</td>
<td>Alison</td>
<td>J S</td>
<td>787 Lake Dr</td>
<td>Isle</td>
<td>MN</td>
<td align="right">56342</td>
<td align="right">5000</td>
<td align="right">3</td>
<td align="right">10.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>475938</td>
<td>Doe </td>
<td>J W</td>
<td>59 Archer Rd </td>
<td>Sutter</td>
<td>CA</td>
<td align="right">95685</td>
<td align="right">700</td>
<td align="right">2</td>
<td align="right">250.00</td>
<td align="right">100.00</td>
</tr>
<tr>
```

```
<td>693829</td>
<td>Thomas</td>
<td>A N</td>
<td>3 Dove Circle</td>
<td>Casper</td>
<td>WY</td>
<td align="right">82609</td>
<td align="right">9999</td>
<td align="right">2</td>
<td align="right">0.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>593029</td>
<td>Williams</td>
<td>E D</td>
<td>485 SE 2 Ave </td>
<td>Dallas</td>
<td>TX</td>
<td align="right">75218</td>
<td align="right">200</td>
<td align="right">1</td>
<td align="right">25.00</td>
<td align="right">0.00</td>
</tr>
<tr>
<td>192837</td>
<td>Lee </td>
<td>F L</td>
<td>5963 Oak St</td>
<td>Hector</td>
<td>NY</td>
<td align="right">14841</td>
<td align="right">700</td>
<td align="right">2</td>
<td align="right">489.50</td>
<td align="right">0.50</td>
</tr>
<tr>
<td>583990</td>
<td>Abraham </td>
<td>M T</td>
<td>392 Mill St</td>
<td>Isle</td>
<td>MN</td>
<td align="right">56342</td>
<td align="right">9999</td>
<td align="right">3</td>
<td align="right">500.00</td>
<td align="right">0.00</td>
</tr>
</table>
```

瀏覽器顯示產生之 HTML 的方法

下表說明 HTML 原始程式碼在瀏覽器中看起來的樣子。

客戶編號	姓氏	姓名縮寫	街道地址	城市	州	郵遞區號	信用額度	收費碼	待繳總額	待繳循環信用利息
938472	Henning	G K	4859 Elm Ave	Dallas	TX	75217	5000	3	37.00	0.00
839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
392859	Vine	S S	PO Box 79	Broton	VT	5046	700	1	439.00	0.00
938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
397267	Tyron	W E	13 Myrtle Dr	Hector	NY	14841	1000	1	0.00	0.00
389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	1.50
846283	Alison	J S	787 Lake Dr	Isle	MN	56342	5000	3	10.00	0.00
475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
693829	Thomas	A N	3 Dove Circle	Casper	WY	82609	9999	2	0.00	0.00
593029	Williams	E D	485 SE 2 Ave	Dallas	TX	75218	200	1	25.00	0.00
192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
583990	Abraham	M T	392 Mill St	Isle	MN	56342	9999	3	500.00	0.00

範例：使用 HTMLFormConverter

執行具有 servlet 支援的 web 伺服器時，請編譯及執行下列範例以瞭解 HTMLFormConverter 如何運作：

```
import java.awt.Color;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.GridLayoutFormPanel;
import com.ibm.as400.util.html.HTMLConstants;
import com.ibm.as400.util.html.HTMLForm;
import com.ibm.as400.util.html.HTMLTable;
import com.ibm.as400.util.html.HTMLTableCaption;
import com.ibm.as400.util.html.HTMLText;
import com.ibm.as400.util.html.LabelFormElement;
import com.ibm.as400.util.html.LineLayoutFormPanel;
import com.ibm.as400.util.html.SubmitFormInput;
import com.ibm.as400.util.html.TextFormInput;

import com.ibm.as400.util.servlet.HTMLFormConverter;
import com.ibm.as400.util.servlet.SQLResultSetRowData;

import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400JDBCDriver;

/**
 * 在 Servlet 中使用 HTMLFormConverter 類別的範例。
 */
public class HTMLFormConverterExample extends HttpServlet

{
 private String userId_ = "myUserId";
 private String password_ = "myPwd";
 private AS400system_;

 private Connection databaseConnection_;



 // Perform cleanup before returning to the main HTML form.
 public void cleanup()
   {
 try
     {
 // Close the database connection.
if (databaseConnection_ != null)
```

```
        {
databaseConnection_.close();
databaseConnection_ = null;
        }
      }
catch (Exception e)
      {
e.printStackTrace();
      }
   }


// Convert the row data to formatted HTML.
 private HTMLTable[] convertRowData(SQLResultSetRowData rowData)
   {
 try
      {
 // Create the converter, which will generate HTML from
 // the result set that comes back from the database query.
 HTMLFormConverter converter = new HTMLFormConverter();

 // Set the form attributes.
 converter.setBorderWidth(3);
 converter.setCellPadding(2);
 converter.setCellSpacing(4);

 // Convert the row data to HTML.
 HTMLTable[] htmlTable = converter.convertToForms(rowData);
 return htmlTable;
      }
catch (Exception e)
      {
e.printStackTrace();
 return null;
      }

   }


 // Return the response to the client.
 public void doGet(HttpServletRequest request,
 HttpServletResponse response)
 throws IOException, ServletException
   {
response.setContentType("text/html");
PrintWriter out = response.getWriter();

out.println(showHtmlMain());
out.close();
   }


 // Handle the data posted to the form.
 public void doPost (HttpServletRequest request, HttpServletResponse
response)
 throws ServletException, IOException
```

```
   {
SQLResultSetRowData rowData = new SQLResultSetRowData();
HTMLTable[] htmlTable = null;

// Get the current session object or create one if needed.
HttpSession session = request.getSession(true);

ServletOutputStream out = response.getOutputStream();

response.setContentType("text/html");

Hashtable parameters = getRequestParameters (request);

// Retrieve the row data and HTML table values for this session.
rowData = (SQLResultSetRowData) session.getValue("sessionRowData");
htmlTable = (HTMLTable[]) session.getValue("sessionHtmlTable");

// if this is the first time through, show first record
if (parameters.containsKey("getRecords"))
      {
 rowData = getAllRecords(parameters, out);

 if (rowData != null)
        {
// Set the row data value for this session.
session.putValue("sessionRowData", rowData);

// Position to the first record.
 rowData.first();

// Convert the row data to formatted HTML.
htmlTable = convertRowData(rowData);

if (htmlTable != null)
            {
 rowData.first();
 session.putValue("sessionHtmlTable", htmlTable);
 out.println(showHtmlForRecord(htmlTable, 0));
            }
         }
      }
// if the "Return To Main" button was pressed, go back to the main HTML form
else if (parameters.containsKey("returnToMain"))
      {
 session.invalidate();
 cleanup();
out.println(showHtmlMain());
      }
// if the "First" button was pressed, show the first record
else if (parameters.containsKey("getFirstRecord"))
      {
 rowData.first();
 out.println(showHtmlForRecord(htmlTable, 0));
      }
// if the "Previous" button was pressed, show the previous record
else if (parameters.containsKey("getPreviousRecord"))
```

```
      {
 if (!rowData.previous())
         {
 rowData.first();
         }
 out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
      }
// if the "Next" button was pressed, show the next record
else if (parameters.containsKey("getNextRecord"))
      {
 if (!rowData.next())
         {
 rowData.last();
         }
 out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
      }
// if the "Last" button was pressed, show the last record
else if (parameters.containsKey("getLastRecord"))
      {
 rowData.last();
 out.println(showHtmlForRecord(htmlTable, rowData.getCurrentPosition()));
      }
// if none of the above, there must have been an error
else
      {
 out.println(showHtmlForError("發生內部錯誤。非預期的參數。"));
      }

// Save the row data value for this session so the current position
// is updated in the object associated with this session.
session.putValue("sessionRowData", rowData);

// Close the output stream
out.close();
   }


 // Get all the records from the file input by the user.
 private SQLResultSetRowData getAllRecords(Hashtable parameters,
ServletOutputStream out)
 throws IOException
   {
SQLResultSetRowData records = null;

 try
      {
 // Get the system, library and file name from the parameter list.
 String sys = ((String) parameters.get("System")).toUpperCase();
 String lib = ((String) parameters.get("Library")).toUpperCase();
 String file = ((String) parameters.get("File")).toUpperCase();
 if ((sys == null || sys.equals("")) ||
 (lib == null || lib.equals("")) ||
 (file == null || file.equals("")))
         {
out.println(showHtmlForError("系統、檔案或檔案庫名稱無效。"));
```

```
        }
else
        {
// Get the connection to the server.
getDatabaseConnection (sys, out);
if (databaseConnection_ != null)
          {
 Statement sqlStatement = databaseConnection_.createStatement();

 // Query the database to get the result set.
 String query = "SELECT * FROM " + lib + "." + file;
 ResultSet rs = sqlStatement.executeQuery (query);

 boolean rsHasRows = rs.next();// position cursor to first row

 // Show error message if the file contains no record;
 // otherwise, set row data to result set data.
 if (!rsHasRows)
              {
out.println(showHtmlForError("檔案中沒有任何記錄。"));
              }
else
              {
records = new SQLResultSetRowData (rs);
              }

 // Don't close the Statement before we're done using the ResultSet
 // or bad things may happen.
 sqlStatement.close();
          }
        }
      }
catch (Exception e)
      {
e.printStackTrace();
out.println(showHtmlForError(e.toString()));
      }

return records;
   }


 // Establish a database connection.
 private void getDatabaseConnection (String sysName, ServletOutputStream
out)
 throws IOException
   {
if (databaseConnection_ == null)
      {
 try
        {
databaseConnection_ = DriverManager.getConnection("jdbc:as400://" + sysName,
userId_, password_ );
        }
catch (Exception e)
```

```
        {
e.printStackTrace();
out.println(showHtmlForError(e.toString()));
        }
      }
   }


 // Gets the parameters from an HTTP servlet request.
 private static Hashtable getRequestParameters (HttpServletRequest request)
   {
Hashtable parameters = new Hashtable ();
Enumeration enum = request.getParameterNames();
 while
(enum.hasMoreElements ())
      {
 String key = (String) enum.nextElement();
 String value = request.getParameter (key);
 parameters.put (key, value);
      }
return parameters;
   }


 // Get the servlet information.
 public String getServletInfo()
   {
return "HTMLFormConverterExample";
   }


 // Perform initialization steps.
 public void init(ServletConfig config)
   {
 try
      {
 super.init(config);

 // Register the JDBC driver
 try
        {
DriverManager.registerDriver(new AS400JDBCDriver());
        }
catch (Exception e)
        {
System.out.println("找不到 JDBC 驅動程式");
        }
      }
catch (Exception e)
      {
e.printStackTrace();
      }
   }


 // Set the page header info.
```

```
 private String showHeader(String title)
   {
StringBuffer page = new StringBuffer();
page.append("<html><head><title>" + title + "</title>");
page.append("</head><body bgcolor=\"blanchedalmond\">");
return page.toString ();
   }


 // Show the HTML page with the appropriate error information.
 private String showHtmlForError(String message)
   {
String title = "錯誤";
StringBuffer page = new StringBuffer();
page.append (showHeader (title));
 try
      {
// Create the HTML Form object
 HTMLForm errorForm = new HTMLForm("HTMLFormConverterExample");

 // Set up so that doPost() gets called when the form is submitted.
 errorForm.setMethod(HTMLForm.METHOD_POST);

 // Create a single-column panel to which the HTML elements will be added.
 GridLayoutFormPanel grid = new GridLayoutFormPanel();

 // Create the text element for the error and add it to the panel.
 HTMLText text = new HTMLText(message);
 text.setBold(true);
 text.setColor(Color.red);
 grid.addElement(text);

 // Create the button to return to main and add it to the panel.
 grid.addElement(new SubmitFormInput("returnToMain", "返回主畫面"));

 // Add the panel to the HTML form.
 errorForm.addElement(grid);

 page.append(errorForm.toString());
      }
catch (Exception e)
      {
e.printStackTrace();
      }
page.append("</body></html>");
return page.toString();
   }


 // Show the HTML form for an individual record.
 private String showHtmlForRecord(HTMLTable[] htmlTable, int position)
   {
String title = "HTMLFormConverter 範例";
StringBuffer page = new StringBuffer();
page.append (showHeader (title));
```

```
page.append("<h1>" + title + "</h1>");

 try
      {
// Create the HTML Form object
 HTMLForm recForm = new HTMLForm("HTMLFormConverterExample");

 // Set up so that doPost() gets called when the form is submitted.
 recForm.setMethod(HTMLForm.METHOD_POST);

 // Set up a single-column panel layout, within which to arrange
 // the generated HTML elements.
 GridLayoutFormPanel grid = new GridLayoutFormPanel();

 // Create and add a table caption that keeps track of the current record.
 HTMLText recNumText = new HTMLText("記錄編號：" + (position + 1));
 recNumText.setBold(true);
 grid.addElement(recNumText);

 // Set up a two-column panel layout, within which to arrange
 // the table and text to comment on the converter output.
 GridLayoutFormPanel tableGrid = new GridLayoutFormPanel(2);
 tableGrid.addElement(htmlTable[position]);
 HTMLText comment = new HTMLText(" <---- 從 HTMLFormConverter 類別輸出");
 comment.setBold(true);
 comment.setColor(Color.blue);
 tableGrid.addElement(comment);

 // Add the table line to the panel.
 grid.addElement(tableGrid);

 // Set up a single-row panel layout, within which to arrange
 // the buttons for moving through the record list.
 LineLayoutFormPanel buttonLine = new LineLayoutFormPanel();
 buttonLine.addElement(new SubmitFormInput("getFirstRecord", "第一個"));
 buttonLine.addElement(new SubmitFormInput("getPreviousRecord", "上一個"));
 buttonLine.addElement(new SubmitFormInput("getNextRecord", "下一個"));
 buttonLine.addElement(new SubmitFormInput("getLastRecord", "最後一個"));

 // Set up another single-row panel layout for the Return To Main button.
 LineLayoutFormPanel returnToMainLine = new LineLayoutFormPanel();
 returnToMainLine.addElement(new SubmitFormInput("returnToMain",
"返回主畫面"));

 // Add the lines containing the buttons to the grid panel.
 grid.addElement(buttonLine);
 grid.addElement(returnToMainLine);

 // Add the panel to the form.
 recForm.addElement(grid);

 // Add the form to the HTML page.
 page.append(recForm.toString());
      }
catch (Exception e)
      {
```

```
e.printStackTrace();
      }

page.append("</body></html>");
return page.toString();
   }


 // Show the main HTML form (request input for system, file, and library
name).
 private String showHtmlMain()
   {
String title = "HTMLFormConverter Example";
StringBuffer page = new StringBuffer();
page.append (showHeader (title));

page.append("<h1>" + title + "</h1>");


// Create the HTML Form object
HTMLForm mainForm = new HTMLForm("HTMLFormConverterExample");

 try
     {
 // Set up so that doPost() gets called when the form is submitted.
 mainForm.setMethod(HTMLForm.METHOD_POST);

 // Add a brief description to the form.
 HTMLText desc = new HTMLText("<P>此範例使用 HTMLFormConverter 類別, " +
"以轉換從伺服器檔案中擷取的資料。" +
"轉換器會產生 HTML 檔案的陣列。" +
"陣列中的每一個登錄都是檔案中的一個" +
"記錄。" +
"一次會顯示一個記錄, " +
"且您可以使用按鈕在記錄清單中" +
"向前或向後移動。</P>");
 mainForm.addElement(desc);

 // Add instructions to the form.
 HTMLText instr = new HTMLText("<P>請輸入伺服器名稱, " +
 "及您要存取的檔案與該檔案的檔案庫名稱。" +
 "然後按下「顯示記錄」按鈕" +
 "以繼續。</P>");
 mainForm.addElement(instr);


 // Create a grid layout panel and add the system, file and library input
fields.
 GridLayoutFormPanel panel = new GridLayoutFormPanel(2);

 LabelFormElement sysPrompt = new LabelFormElement("伺服器：");
 TextFormInput system = new TextFormInput("系統");
 system.setSize(10);

 LabelFormElement filePrompt = new LabelFormElement("檔名：");
 TextFormInput file = new TextFormInput("檔案");
```

```
 file.setSize(10);

 LabelFormElement libPrompt = new LabelFormElement("檔案庫名稱：");
 TextFormInput library = new TextFormInput("檔案庫");
 library.setSize(10);

 panel.addElement(sysPrompt);
 panel.addElement(system);
 panel.addElement(filePrompt);
 panel.addElement(file);
 panel.addElement(libPrompt);
 panel.addElement(library);

 // Add the panel to the form.
 mainForm.addElement(panel);

 // Create the submit button and add it to the form.
 mainForm.addElement(new SubmitFormInput("getRecords", "顯示記錄"));
      }
catch (Exception e)
      {
e.printStackTrace();
      }

page.append(mainForm.toString());
page.append("</body></html>");

return page.toString();
   }

}
```

上述範例所產生的 HTML 看起來如下：

```
<table border="0">
<tr>
<td><b>Record number: 1</b></td>
</tr>
<tr>
<td><table border="0">
<tr>
<td><table border="3" cellpadding="2" cellspacing="4">
<tr>
<th>CUSNUM</th>
<td>839283</td>
</tr>
<tr>
<th>LSTNAM</th>
<td>Jones </td>
</tr>
<tr>
<th>INIT</th>
<td>B D</td>
</tr>
```

```
<tr>
<th>STREET</th>
<td>21B NW 135 St</td>
</tr>
<tr>
<th>CITY</th>
<td>Clay</td>
</tr>
<tr>
<th>STATE</th>
<td>NY</td>
</tr>
<tr>
<th>ZIPCOD</th>
<td>13041</td>
</tr>
<tr>
<th>CDTLMT</th>
<td>400</td>
</tr>
<tr>
<th>CHGCOD</th>
<td>1</td>
</tr>
<tr>
<th>BALDUE</th>
<td>100.00</td>
</tr>
<tr>
<th>CDTDUE</th>
<td>0.00</td>
</tr>
</table>
</td>
<td><font color="#0000ff"> <b><!-- 從 HTMLFormConverter 類別輸出-->
</b></font></td>
</tr>
</table>
</td>
</tr>
<tr>
<form>
<td><input type="submit" name="getFirstRecord" value="第一個" />
<input type="submit" name="getPreviousRecord" value="上一個" />
<input type="submit" name="getNextRecord" value="下一個" />
<input type="submit" name="getLastRecord" value="最後一個" />
<br />
</td>
</tr>
<tr>
<td><input type="submit" name="returnToMain" value="返回主畫面" />
<br />
</td>
</tr>
```

```
</table>
</form>
```

HTML 及 Servlet 類別的 Lights On 範例

此範例說明 HTML 與 servlet 類別如何運作。這是一般的概觀。如欲檢視此範例, 請使用 Web 伺服器並執行瀏覽器以編譯與編譯本範例。

```
import java.io.IOException;
import java.io.CharArrayWriter;
import java.io.PrintWriter;
import java.sql.*;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.servlet.*;
import javax.servlet.http.*;

import com.ibm.as400.util.html.*;
import com.ibm.as400.util.servlet.*;
import com.ibm.as400.access.*;

/*
在 Servlet 中使用 Toolbox 類別的範例。

伺服器中 SQL 資料庫綱目：

檔案 . . . . . LICENSES
檔案庫 . . . LIGHTSON

 欄位              類型            長度   空字元
 LICENSE           CHARACTER         10   NOT NULL
 USER_ID           CHARACTER         10   NOT NULL WITH DEFAULT
 E_MAIL            CHARACTER         20   NOT NULL
 WHEN_ADDED        DATE                   NOT NULL WITH DEFAULT
 TIME_STAMP        TIMESTAMP              NOT NULL WITH DEFAULT


檔案 . . . . . REPORTS
檔案庫 . . . LIGHTSON

 欄位              類型            長度   空字元
 LICENSE           CHARACTER         10   NOT NULL
 REPORTER          CHARACTER         10   NOT NULL WITH DEFAULT
 DATE_ADDED        DATE                   NOT NULL WITH DEFAULT
 TIME_ADDED        TIME                   NOT NULL WITH DEFAULT
 TIME_STAMP        TIMESTAMP              NOT NULL WITH DEFAULT
 LOCATION          CHARACTER         10   NOT NULL
 COLOR             CHARACTER         10   NOT NULL
 CATEGORY          CHARACTER         10   NOT NULL
*/

public class LightsOn extends javax.servlet.http.HttpServlet

{
   private AS400  system_;
  private String password_;  // password for the server and for the SQL
database
```

```
  private java.sql.Connection databaseConnection_;


  public void destroy (ServletConfig config)
  {
    try {
      if (databaseConnection_ != null) {
            databaseConnection_.close();
      }
    }
    catch (Exception e) { e.printStackTrace (); }
  }

   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
   throws IOException, ServletException
  {
     HttpSession session = request.getSession();

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

         out.println(showHtmlMain());

      out.close();
  }

   public void doPost (HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException
  {
      HttpSession session = request.getSession(true);
      ServletOutputStream out = response.getOutputStream();
      response.setContentType("text/html");

      Hashtable parameters = getRequestParameters (request);

    if (parameters.containsKey("askingToReport"))
      out.println (showHtmlForReporting ());
    else if (parameters.containsKey("askingToRegister"))
      out.println (showHtmlForRegistering ());
    else if (parameters.containsKey("askingToUnregister"))
      out.println(showHtmlForUnregistering());
    else if (parameters.containsKey("askingToListRegistered"))
      out.println (showHtmlForListingAllRegistered ());
    else if (parameters.containsKey("askingToListReported"))
      out.println (showHtmlForListingAllReported ());
    else if (parameters.containsKey("returningToMain"))
         out.println(showHtmlMain());


    else { // None of the above, so assume the user has filled out a form
           // and is submitting information.  Grab the incoming info
           // and do the requested action.
```

```
      if (parameters.containsKey("submittingReport")) {
        String acknowledgement = reportLightsOn (parameters, out);
        out.println (showAcknowledgement(acknowledgement));
      }

      else if (parameters.containsKey("submittingRegistration")) {
        String acknowledgement = registerLicense (parameters, out);
        out.println (showAcknowledgement(acknowledgement));
      }

      else if (parameters.containsKey("submittingUnregistration")) {
        String acknowledgement = unregisterLicense (parameters, out);
        out.println (showAcknowledgement(acknowledgement));
      }

      else {
        out.println (showAcknowledgement("Error (internal): " +
                           "Neither Report, Register, " +
                           "Unregister, ListRegistered, or ListReported."));
      }
    }

    out.close(); // Close the output stream.

  }

  // Gets the parameters from an HTTP servlet request, and packages them
  // into a hashtable for convenience.
   private static Hashtable getRequestParameters (HttpServletRequest
request)
  {
      Hashtable parameters = new Hashtable ();
      Enumeration enum = request.getParameterNames();
    while (enum.hasMoreElements()) {
        String key = (String) enum.nextElement();
        String value = request.getParameter (key);
        parameters.put (key, value);
    }
      return parameters;
  }


  // Removes blanks and hyphens from a String, and sets it to all uppercase.
  private static String normalize (String oldString)
  {
    if (oldString == null || oldString.length() == 0) return null;
    StringBuffer newString = new StringBuffer ();
    for (int i=0; i<oldString.length(); i++) {
      if (oldString.charAt(i) != ' ' && oldString.charAt(i) != '-')
        newString.append (oldString.charAt(i));
    }
    return newString.toString().toUpperCase();
  }

  // Composes a list of single-quoted strings.
  private static String quoteList (String[] inList)
```

```
{
  StringBuffer outList = new StringBuffer();
  for (int i=0; i<inList.length; i++)
  {
    outList.append ("'" + inList[i] + "'");
    if (i<inList.length-1)
      outList.append (",");
  }
  return outList.toString();
}


 public String getServletInfo()
{
  return "Lights-On Servlet";
}

private AS400 getSystem ()
{
  try
  {
    if (system_ == null)
    {
      system_ = new AS400();

      // Note: It would be better to get these values
      // from a properties file.
      String sysName  = "MYSYSTEM";    // TBD
      String userId   = "MYUSERID";    // TBD
      String password = "MYPASSWD";    // TBD

      system_.setSystemName(sysName);
      system_.setUserId(userId);
      system_.setPassword(password);
      password_ = password;

      system_.connectService(AS400.DATABASE);
      system_.connectService(AS400.FILE);
      system_.addPasswordCacheEntry(sysName, userId, password_);
    }
  }
  catch (Exception e) { e.printStackTrace (); system_ = null; }

  return system_;
}


 public void init(ServletConfig config)
{
  boolean rc;

  try {
       super.init(config);

    // Register the JDBC driver.
      try {
```

```
        java.sql.DriverManager.registerDriver(new
com.ibm.as400.access.AS400JDBCDriver());
      }
      catch (Exception e)
      {
        System.out.println("JDBC Driver not found");
      }


    }
    catch (Exception e) { e.printStackTrace(); }
  }

  private void getDatabaseConnection ()
  {
    if (databaseConnection_ == null) {
    try {
        databaseConnection_ = java.sql.DriverManager.getConnection(
                  "jdbc:as400://" + getSystem().getSystemName() + "/" +
                  "LIGHTSON", getSystem().getUserId(), password_ );

      }
    catch (Exception e) { e.printStackTrace (); }
    }
  }


  private String registerLicense (Hashtable parameters, ServletOutputStream
out)
  {
    String licenseNum = normalize ((String)parameters.get("licenseNum"));
    String eMailAddress = (String)parameters.get("eMailAddress");
    StringBuffer acknowledgement = new StringBuffer();

    if (licenseNum == null || licenseNum.length() == 0)
      acknowledgement.append ("Error: License number not specified.\n");

    if (eMailAddress == null || eMailAddress.length() == 0)
      acknowledgement.append ("Error: Notification e-mail address not
specified.\n");

    if (acknowledgement.length() == 0)
    {
    try
      {
        // Insert the new license number and e-mail address into the
database.
      getDatabaseConnection ();
              Statement sqlStatement =
databaseConnection_.createStatement();

        // Issue the request.
        String cmd = "INSERT INTO LICENSES " +
          "(LICENSE, E_MAIL) VALUES (" +
          quoteList (new String[] {licenseNum, eMailAddress} ) +
          ")";
```

```
        sqlStatement.executeUpdate(cmd);
              sqlStatement.close();

        // Acknowledge the request.
        acknowledgement.append ("License number " + licenseNum + " has been
registered.");
        acknowledgement.append ("Notification e-mail address is: " +
eMailAddress);
      }
    catch (Exception e) { e.printStackTrace (); }
    }
    return acknowledgement.toString();
  }


  private String unregisterLicense (Hashtable parameters,
ServletOutputStream out)
  {
    String licenseNum = normalize ((String)parameters.get("licenseNum"));
    StringBuffer acknowledgement = new StringBuffer();

    if (licenseNum == null || licenseNum.length() == 0)
      acknowledgement.append ("Error: License number not specified.\n");

    if (acknowledgement.length() == 0)
    {
    try
      {
        // Remove the specified license number and e-mail address from
database.
      getDatabaseConnection ();
              Statement sqlStatement =
databaseConnection_.createStatement();

        // Delete the row(s) from the LICENSES database.
        String cmd = "DELETE FROM LICENSES WHERE LICENSE = '" + licenseNum +
"'";
        sqlStatement.executeUpdate(cmd);
              sqlStatement.close();

        // Acknowledge the request.
        acknowledgement.append ("License number " + licenseNum + " has been
unregistered.");
      }
    catch (Exception e) { e.printStackTrace (); }
    }
    return acknowledgement.toString();
  }


  private String reportLightsOn (Hashtable parameters, ServletOutputStream
out)
  {
    String licenseNum = normalize ((String)parameters.get("licenseNum"));
    String location = (String)parameters.get("location");
    String color = (String)parameters.get("color");
```

```
    String category = (String)parameters.get("category");
    StringBuffer acknowledgement = new StringBuffer();
    if (licenseNum == null || licenseNum.length() == 0)
      acknowledgement.append ("Error: License number not specified.");

    if (acknowledgement.length() == 0)
    {
    try
      {
        // Report "lights on" for a specified vehicle.
      getDatabaseConnection ();
              Statement sqlStatement =
databaseConnection_.createStatement();

        // Add an entry to the REPORTS database.
        String cmd = "INSERT INTO REPORTS " +
          "(LICENSE, LOCATION, COLOR, CATEGORY) VALUES (" +
          quoteList (new String[] {licenseNum, location, color, category} )
+
          ")";
        sqlStatement.executeUpdate(cmd);
              sqlStatement.close();

        // Acknowledge the request.
        acknowledgement.append ("License number " + licenseNum +
                                " has been reported.  Thanks!");
      }
    catch (Exception e) { e.printStackTrace (); }
    }
    return acknowledgement.toString();
  }


   private String showHeader(String title)
  {
      StringBuffer page = new StringBuffer();
      page.append("<html><head><title>" + title + "</title>");
      page.append("</head><body bgcolor=\"blanchedalmond\">");
      return page.toString();
  }


  private String showAcknowledgement (String acknowledgement)
  {
    String title = "Acknowledgement";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

    try {
      HTMLForm form = new HTMLForm("LightsOn");
    GridLayoutFormPanel grid = new GridLayoutFormPanel();
      HTMLText text = new HTMLText(acknowledgement);
      if (acknowledgement.startsWith("Error")) text.setBold(true);
         grid.addElement(text);
      grid.addElement(new SubmitFormInput("returningToMain", "Home"));
      form.addElement(grid);
```

```
      page.append(form.toString());
    }
    catch (Exception e) { e.printStackTrace (); }
      page.append("</body></html>");
      return page.toString();
  }


   private String showHtmlMain()
  {
    String title = "Lights-On tool";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

      page.append("<h1>" + title + "</h1>");


      // Create the HTML Form object.
      HTMLForm mainForm = new HTMLForm("LightsOn");
    GridLayoutFormPanel grid = new GridLayoutFormPanel();

    try {
      // Set up so that doPost() gets called when the form is submitted.
      mainForm.setMethod(HTMLForm.METHOD_POST);

      // Create some buttons.
      grid.addElement(new SubmitFormInput("askingToReport",
                                          "Report a vehicle with lights
on"));
      grid.addElement(new SubmitFormInput("askingToRegister",
                                          "Register my license number"));
      grid.addElement(new SubmitFormInput("askingToUnregister",
                                          "Unregister my license number"));
      grid.addElement(new SubmitFormInput("askingToListRegistered",
                                          "List all registered licenses"));
      grid.addElement(new SubmitFormInput("askingToListReported",
                                          "List all vehicles with lights
on"));

      mainForm.addElement(grid);
    }
    catch (Exception e) { e.printStackTrace (); }

      page.append(mainForm.toString());
      page.append("</body></html>");

      return page.toString();
  }


  private String showHtmlForReporting ()
  {
    String title = "Report a vehicle with lights on";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));
```

```
    page.append("<h1>" + title + "</h1>");

    // Create the HTML Form object.
  HTMLForm reportForm = new HTMLForm("LightsOn");
  GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

  try {
    // Set up so that doPost() gets called when the form is submitted.
    reportForm.setMethod(HTMLForm.METHOD_POST);

    TextFormInput licenseNum = new TextFormInput("licenseNum");
    licenseNum.setSize(10);
    licenseNum.setMaxLength(10);

    // Add elements to the line form
    grid.addElement(new LabelFormElement("Vehicle license number:"));
    grid.addElement(licenseNum);

    // Create a radio button group and add the radio buttons.
    RadioFormInputGroup colorGroup = new RadioFormInputGroup("color");

    colorGroup.add("color", "white", "white", true);
    colorGroup.add("color", "black", "black", false);
    colorGroup.add("color", "gray", "gray", false);
    colorGroup.add("color", "red", "red", false);
    colorGroup.add("color", "yellow", "yellow", false);
    colorGroup.add("color", "green", "green", false);
    colorGroup.add("color", "blue", "blue", false);
    colorGroup.add("color", "brown", "brown", false);

    // Create a selection list for category of vehicle.
    SelectFormElement category = new SelectFormElement("category");
    category.addOption("sedan", "sedan", true);
    category.addOption("convertible", "convertibl"); // 10-char field in
DB
    category.addOption("truck", "truck");
    category.addOption("van", "van");
    category.addOption("SUV", "SUV");
    category.addOption("motorcycle", "motorcycle");
    category.addOption("other", "other");

    // Create a selection list for vehicle location (building number).
    SelectFormElement location = new SelectFormElement("location");
    location.addOption("001", "001", true);
    location.addOption("002", "002");
    location.addOption("003", "003");
    location.addOption("005", "005");
    location.addOption("006", "006");
    location.addOption("015", "015");

    grid.addElement(new LabelFormElement("Color:"));
    grid.addElement(colorGroup);

    grid.addElement(new LabelFormElement("Vehicle type:"));
    grid.addElement(category);
```

```
      grid.addElement(new LabelFormElement("Building:"));
      grid.addElement(location);

      grid.addElement(new SubmitFormInput("submittingReport", "Submit
report"));
      grid.addElement(new SubmitFormInput("returningToMain", "Home"));

      reportForm.addElement(grid);
    }
    catch (Exception e) { e.printStackTrace (); }

    page.append(reportForm.toString());

      page.append("</body></html>");

      return page.toString();
  }


  private String showHtmlForRegistering ()
  {
    String title = "Register my license number";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

      page.append("<h1>" + title + "</h1>");

      // Create the HTML Form object.
    HTMLForm registrationForm = new HTMLForm("LightsOn");

    // Set up a two-column panel layout, within which to arrange
      // the generated HTML elements.
    GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

    try {
      // Set up so that doPost() gets called when the form is submitted.
      registrationForm.setMethod(HTMLForm.METHOD_POST);

      TextFormInput licenseNum = new TextFormInput("licenseNum");
      licenseNum.setSize(10);
      licenseNum.setMaxLength(10);

      TextFormInput eMailAddress = new TextFormInput("eMailAddress");
      eMailAddress.setMaxLength(20);

      grid.addElement(new LabelFormElement("License number:"));
      grid.addElement(licenseNum);

      grid.addElement(new LabelFormElement("E-mail notification address:"));
      grid.addElement(eMailAddress);

      grid.addElement(new SubmitFormInput("submittingRegistration",
"Register"));
      grid.addElement(new SubmitFormInput("returningToMain", "Home"));

      registrationForm.addElement(grid);
    }
```

```
    catch (Exception e) { e.printStackTrace (); }

    page.append(registrationForm.toString());

      page.append("</body></html>");

      return page.toString();
  }


  private String showHtmlForUnregistering ()
  {
    String title = "Unregister my license number";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

      page.append("<h1>" + title + "</h1>");

      // Create the HTML Form object.
    HTMLForm unregistrationForm = new HTMLForm("LightsOn");
    GridLayoutFormPanel grid = new GridLayoutFormPanel(2);

    try {
      // Set up so that doPost() gets called when the form is submitted.
      unregistrationForm.setMethod(HTMLForm.METHOD_POST);

      // Create the LineLayoutFormPanel object.
      TextFormInput licenseNum = new TextFormInput("licenseNum");
      licenseNum.setSize(10);
      licenseNum.setMaxLength(10);

      grid.addElement(new LabelFormElement("Vehicle license number:"));
      grid.addElement(licenseNum);

      grid.addElement(new SubmitFormInput("submittingUnregistration",
"Unregister"));
      grid.addElement(new SubmitFormInput("returningToMain", "Home"));

      unregistrationForm.addElement(grid);
    }
    catch (Exception e) {
            e.printStackTrace();
      CharArrayWriter cWriter = new CharArrayWriter();
      PrintWriter pWriter = new PrintWriter (cWriter, true);
      e.printStackTrace (pWriter);
      page.append (cWriter.toString());
    }

    page.append(unregistrationForm.toString());

      page.append("</body></html>");

      return page.toString();
  }


  private String showHtmlForListingAllRegistered ()
```

```
  {
    String title = "All registered licenses";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

    try
    {
      // Create the HTML Form object.
      HTMLForm mainForm = new HTMLForm("LightsOn");

      // Set up a single-column panel layout, within which to arrange
      // the generated HTML elements.
    GridLayoutFormPanel grid = new GridLayoutFormPanel();

      // Specify the layout for the generated table.
      HTMLTable table = new HTMLTable();
      table.setAlignment(HTMLConstants.LEFT);
      table.setBorderWidth(3);

      // Create and add the table caption and header.
      HTMLTableCaption caption = new HTMLTableCaption();
      caption.setAlignment(HTMLConstants.TOP);
      caption.setElement(title);
      table.setCaption(caption);
      table.setHeader(new String[] { "License", "Date added" } );

      // Create the converter, which will generate table HTML from
      // the result set that comes back from the database query.
      HTMLTableConverter converter = new HTMLTableConverter();
      converter.setTable(table);

      getDatabaseConnection ();
               Statement sqlStatement =
databaseConnection_.createStatement();

      // First pre-query the database to verify that it's not empty.
      String query = "SELECT COUNT(*) FROM LICENSES";
               ResultSet rs = sqlStatement.executeQuery (query);
      rs.next();  // position cursor to first row
      int rowCount = rs.getInt(1);

      if (rowCount == 0) {
        page.append ("<font size=4 color=red>No vehicles have been
reported.</font>");
      }
      else {
        query = "SELECT LICENSE,WHEN_ADDED FROM LICENSES";
        rs = sqlStatement.executeQuery (query);
        SQLResultSetRowData rowData = new SQLResultSetRowData (rs);
        HTMLTable[] generatedHtml = converter.convertToTables(rowData);
        grid.addElement(generatedHtml[0]);
      }
               sqlStatement.close();
      // Note: Mustn't close statement before we're done using result set.

      grid.addElement(new SubmitFormInput("returningToMain", "Home"));
```

```
      mainForm.addElement(grid);
      page.append(mainForm.toString());
    }
    catch (Exception e) { e.printStackTrace (); }
      page.append("</body></html>");
      return page.toString();
  }


  private String showHtmlForListingAllReported ()
  {
    String title = "All vehicles with lights on";
      StringBuffer page = new StringBuffer();
      page.append (showHeader (title));

    try
    {
      // Create the HTML Form object.
      HTMLForm form = new HTMLForm("LightsOn");

      // Set up a single-column panel layout, within which to arrange
      // the generated HTML elements.
    GridLayoutFormPanel grid = new GridLayoutFormPanel();

      // Specify the layout for the generated table.
      HTMLTable table = new HTMLTable();
      table.setAlignment(HTMLConstants.LEFT);
      table.setBorderWidth(3);

      // Create and add the table caption and header.
      HTMLTableCaption caption = new HTMLTableCaption();
      caption.setAlignment(HTMLConstants.TOP);
      caption.setElement(title);
      table.setCaption(caption);
      table.setHeader(new String[] { "License", "Color", "Category", "Date",
"Time" } );

      // Create the converter, which will generate table HTML from
      // the result set that comes back from the database query.
      HTMLTableConverter converter = new HTMLTableConverter();
      converter.setTable(table);

      getDatabaseConnection ();
              Statement sqlStatement =
databaseConnection_.createStatement();

      // First pre-query the database to verify that it's not empty.
      String query = "SELECT COUNT(*) FROM REPORTS";
              ResultSet rs = sqlStatement.executeQuery (query);
      rs.next();  // position cursor to first row
      int rowCount = rs.getInt(1);

      if (rowCount == 0) {
        page.append ("<font size=4 color=red>No vehicles have been
reported.</font>");
```

```
      }
      else {
        query = "SELECT LICENSE,COLOR,CATEGORY,DATE_ADDED,TIME_ADDED FROM
REPORTS";
        rs = sqlStatement.executeQuery (query);
        SQLResultSetRowData rowData = new SQLResultSetRowData (rs);
        HTMLTable[] generatedHtml = converter.convertToTables(rowData);
        grid.addElement(generatedHtml[0]);
      }
              sqlStatement.close();
      // Note: Mustn't close statement before we're done using result set.

      grid.addElement(new SubmitFormInput("returningToMain", "Home"));
      form.addElement(grid);
      page.append(form.toString());
    }
    catch (Exception e) { e.printStackTrace (); }
      page.append("</body></html>");
      return page.toString();
  }

}
```

撰寫第一支 Toolbox for Java 程式

若要開始這個簡單的習題，您的工作站上必須先安裝好 Java。您可以複查Java 應用程式的執行基本要求，決定需要安裝哪一個版本。

將 Java 安裝到從屬站之後，請完成下列作業：

1. 將 jt400.jar 複製到工作站
2. 將 JAR 檔的完整路徑附加到 CLASSPATH，以便將 jt400.jar 新增到 CLASSPATH。 例如，當 jt400.jar 檔位在您的工作站（執行 Windows）上的 c:\lib 目錄時，請將下一行新增到 CLASSPATH 陳述式的尾端：

   ```
   ;c:\lib\jt400.jar
   ```

3. 開啟文字編輯器並鍵入第一支簡式程式設計範例

   註： 請務必略過參照「附註」的文字（例如，附註 1、附註 2等）。將新文件儲存為 CmdCall.java。

4. 在您的工作站上啟動一個指令階段作業，並使用下列指令編譯簡式程式設計範例：

   ```
   javac CmdCall.java
   ```

5. 在指令階段作業中，鍵入下列指令來執行簡式程式設計範例：

   ```
   java CmdCall
   ```

«

[簡式程式設計範例

範例：使用 CommandCall

請使用下列內容作為程式範例。該範例包括程式碼中相關關鍵行的詳細說明。
您可以使用下列方式檢視詳細說明：

- 按一下  影像，以在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，以查看位於範例底端的詳細說明。

註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
//////////////////////////////////////////////////////////////////////////////
//
// Example using the IBM Toolbox for Java's access class, CommandCall.
//
// This source is an example of IBM Toolbox for Java "Job List".
//
//////////////////////////////////////////////////////////////////////////////
//
// The access classes of the Toolbox are in the
com.ibm.as400.access.package.
// Import this package to use the Toolbox classes.
//
//////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;

public class CmdCall
{
 public static void main(String[] args)
   {
// Like other Java classes the Toolbox classes throw
// exceptions when something goes wrong.These must be
// caught by programs that use the Toolbox.
try 附註 1 
      {
AS400 system = new AS400();

 CommandCall cc = new CommandCall(system);        附註 2 

 cc.run("CRTLIB MYLIB");     附註 3 

 AS400Message[] ml = cc.getMessageList();        附註 4 

 for (int i=0; i<ml.length; i++)
        {
System.out.println(ml[i].getText());       附註 5 
        }
      }
```

```
catch (Exception e)
      {
e.printStackTrace();
      }

 System.exit(0);
   }
}
```

1. Toolbox for Java 使用 "AS400" 物件識別目標伺服器。 若未使用參數建構 AS400 物件，Toolbox for Java 將提示系統名稱、使用者 ID 與密碼。 AS400 類別也包括具有系統名稱、使用者 ID 與密碼的建構元。
2. 使用 Toolbox for Java CommandCall 物件傳送指令到伺服器。建立 CommandCall 物件時， 您可以傳送 AS400 物件給它，如此 CommondCall 物件可知道傳送指令的目標伺服器。
3. 在指令呼叫物件上使用 run() 方法，以執行指令。
4. 執行指令的結果為產生 OS/400 訊息清單。Toolbox 以 AS400Message 表示這些訊息。 完成指令時，將從 CommandCall 物件得到結果訊息。
5. 列印訊息文字。亦可使用訊息 ID、訊息嚴重性以及其它資訊。 此程式僅列印訊息文字。

[簡式程式設計範例]

範例：使用訊息佇列 (3 之 1)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
//////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
//////////////////////////////////////////////////////////////////////////////

package examples;     附註 1  


import java.io.*;
 import java.util.*;

import com.ibm.as400.access.*;        附註 2  

public class displayMessages extends Object
{

 public static void main(String[] parameters)          附註 3  
   {
displayMessages me = new displayMessages();
me.Main(parameters);      附註 4  

System.exit(0);     附註 5  
   }


 void displayMessage()
   {
   }


 void Main(String[] parms)
   {
try  附註 6  
     {

 // IBM Toolbox for Java code goes here
```

```
      }
catch (Exception e)
      {
 e.printStackTrace();      附註 7
      }
   }
}
```

1. 此類別位於 examples 套件中。Java 使用套件來避免 Java 類別檔案之間的名稱衝突。
2. 此指令行使得 access 套件中的所有 IBM Toolbox for Java 類別都可以供本程式使用。access 套件中的類別都具有共同的字首 com.ibm.as400。 藉由使用 import 陳述式，程式即可以它的名稱 (而非其完整的名稱) 來參照類別。 例如，您可以 AS400 來參照 AS400 類別，而不需要用 com.ibm.as400.AS400。
3. 此類別具有 main 方法；因此，它可以當作應用程式執行。 若要呼叫程式，請執行 examples.displayMessages。注意，執行程式時大小寫必須相符。 由於您使用了 IBM Toolbox for Java 類別，因此 jt400.zip 必須位於 classpath 環境變數中。
4. 附註 3 指出的 main 方法是靜態的。靜態方法的其中一項限制是它只能在其類別中呼叫其它靜態方法。為避免這項限制，許多的 Java 程式會建立物件，然後在名為 main() 的方法中進行起始設定處理程序。Main() 方法可以呼叫位於 displayMessages 物件中的其它方法。
5. IBM Toolbox for Java 可以代表應用程式建立一些緒， 來執行 IBM Toolbox for Java 活動。如果程式沒有在終止時發出 System.exit(0) 就可能無法正常終止。例如，假設此程式是從 Windows 95 DOS 提示執行。若沒有此行，當程式完成時，就不會返回指令提示。 所以使用者必須輸入 Ctrl-C 來取得指令提示。
6. IBM Toolbox for Java 程式碼丟出您的程式必須捕捉的異常。
7. 此程式會在程式執行錯誤處理程序時，顯示異常文字。IBM Toolbox for Java 丟出的異常文字都已翻譯經過，因此，異常文字會與工作站的語言相同。

[下一部份 | 簡式程式設計範例]

範例：使用訊息佇列 (3 之 2)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
////////////////////////////////////////////////////////////////////////////////

package examples;


import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class displayMessages extends Object
{

public static void main(String[] parameters)
   {
displayMessages me = new displayMessages();

me.Main(parameters);

 System.exit(0);
   }


 void displayMessage()
   {
   }


 void Main(String[] parms)
   {
 try
      {

AS400 system = new AS400();       附註 1


 if (parms.length > 0)
```

```
system.setSystemName(parms[0]);        附註 2
      }
catch (Exception e)
      {
e.printStackTrace();
      }
   }
}
```

1. 程式將使用AS400 物件來指定要連接的伺服器。 但是有一個例外，即需要伺服器資源的所有程式都必須擁有 AS/400 物件。此例外是 JDBC。 如果您的程式使用 JDBC，則 IBM Toolbox for Java JDBC 驅動程式會為該程式建立 AS400 物件。
2. 此程式假設第一行指令行參數是伺服器的名稱。 如果有傳遞參數給程式，則會使用 AS400 物件的 setSystemName 方法來設定系統名稱。AS400 物件也需要伺服器登入資訊：

   - 如果是在工作站上執行此程式，則 IBM Toolbox for Java 程式將提示使用者輸入使用者 ID 和密碼。 註：如果未將系統名稱指定為指令行參數，AS400 物件也會提示您提供系統名稱。
   - 如果是在 iSeries JVM 上執行此程式，則會使用執行 Java 程式的使用者之 ID 和密碼。在此情況下，使用者不需指定系統名稱，但將系統名稱預設為執行此程式所在的系統名稱。

[上一部份 |下一部份 |簡式程式設計範例]

範例：使用訊息佇列 (3 之 3)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
//////////////////////////////////////////////////////////////////////////////
//
// Example using the Message Queue function of the IBM Toolbox for Java
//
// This source is an example of IBM Toolbox for Java "Message Queue".
//
//////////////////////////////////////////////////////////////////////////////

package examples;


import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class displayMessages extends Object
{

public static void main(String[] parameters)
   {
displayMessages me = new displayMessages();

me.Main(parameters);

 System.exit(0);
   }


 void displayMessage()
   {
   }


 void Main(String[] parms)
   {
 try
      {
AS400 system = new AS400();

 if (parms.length > 0)
system.setSystemName(parms[0]);
```

```
 MessageQueue queue = new MessageQueue(system, MessageQueue.CURRENT);
```
附註 1 

```
Enumeration e = queue.getMessages();
```
附註 2 

```
while (e.hasMoreElements())
                {

QueuedMessage message = (QueuedMessage) e.nextElement();
```
附註 3 

```
System.out.println(message.getText());
```
附註 4 

```
                }
            }
catch (Exception e)
            {
e.printStackTrace();
            }
        }
}
```

1. 此程式的目的是將訊息顯示在伺服器訊息佇列中。 這項作業會使用 IBM Toolbox for Java 的 MessageQueue 物件。 建構了訊息佇列物件之後，參數即為 AS400 物件和訊息佇列名稱。AS400 物件會指出哪一部伺服器有包含資源，而訊息佇列名稱會指定伺服器上的哪一個訊息佇列。 在此狀況下將會使用常數，以便通知訊息佇列物件來存取登入使用者的佇列。
2. 訊息佇列物件將從伺服器中取得訊息的清單。此時會建立與伺服器的連線。
3. 從清單移除訊息。此訊息位於 IBM Toolbox for Java 程式的 QueuedMessage 物件中。
4. 列印訊息文字。

[上一部份 | 簡式程式設計範例]

範例：使用記錄層次存取 (2 之 1)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。 您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Record level access example.This program will prompt the user
// for the name of the server and the file to display.The file must exist
// and should contain records.Each record in the file will be displayed
// to System.out.
//
// Calling syntax: java RLSequentialAccessExample
//
// This source is an example of IBM Toolbox for Java "RecordLevelAccess"
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLSequentialAccessExample
{
public static void main(String[] parameters)
   {
BufferedReader inputStream = new BufferedReader(new
InputStreamReader(System.in),1);

String systemName = "";
String library = "";
String file = "";
String member = "";

System.out.println();
 try
      {
System.out.print("系統名稱：");
systemName = inputStream.readLine();

System.out.print("檔案所在的檔案庫：");
library = inputStream.readLine();

System.out.print("檔案名稱：");
file = inputStream.readLine();

System.out.print("成員名稱 (按 Enter 鍵取得第一個成員)：");
member = inputStream.readLine();
if (member.equals(""))
        {
member = "*FIRST";
```

```
        }
System.out.println();
      }
catch (Exception e)
      {
System.out.println("取得使用者輸入時發生錯誤。");
e.printStackTrace();
 System.exit(0);
      }


AS400 system = new AS400(systemName);          附註 1
 try
      {
system.connectService(AS400.RECORDACCESS);
      }
catch (Exception e)
      {
System.out.println("無法連線供執行記錄層次存取。");
System.out.println("請察看程式設計師手冊中的設定檔案，取得有關記錄層次存取的特殊指示");
e.printStackTrace();
 System.exit(0);
      }


QSYSObjectPathName filePathName = new QSYSObjectPathName(library, file,
member, "MBR");     附註 2


SequentialFile theFile = new SequentialFile(system, filePathName.getPath());
附註 3


AS400FileRecordDescription recordDescription = new
AS400FileRecordDescription(system, filePathName.getPath());
 try
      {
RecordFormat[] format = recordDescription.retrieveRecordFormat();              附註 4


theFile.setRecordFormat(format[0]);          附註 5


theFile.open(AS400File.READ_ONLY, 100, AS400File.COMMIT_LOCK_LEVEL_NONE);
附註 6

System.out.println("顯示檔案 " + library.toUpperCase() + "/" +
file.toUpperCase() + "(" + theFile.getMemberName().trim() + "):");

Record record = theFile.readNext();          附註 7
 while(record != null)
        {
System.out.println(record);
record = theFile.readNext();
        }
System.out.println();
```

```
theFile.close();     附註 8
```

```
system.disconnectService(AS400.RECORDACCESS);          附註 9
      }
catch (Exception e)
      {
System.out.println("嘗試顯示檔案時發生錯誤。");
e.printStackTrace();

 try
       {
// Close the file
theFile.close();
       }
catch(Exception x)
       {
       }

system.disconnectService(AS400.RECORDACCESS);
 System.exit(0);
      }

// Make sure that the application ends; see readme for details
 System.exit(0);
    }
  }
```

1. 這一行程式碼會建立一個 AS400 物件並連接到記錄層次存取服務。
2. 這個指令行會建立一個 QSYSObjectPathName 物件，用來取得要顯示的物件之整合檔案系統路徑名稱格式。
3. 此陳述式會建立一個物件，代表您連接到的伺服器上的現有循序檔。 此循序檔即為要顯示的檔案。
4. 這些指令行會擷取檔案的記錄格式。
5. 此指令行會設定檔案的記錄格式。
6. 此指令行會開啟選取的檔案供讀取。可能的話，它一次會讀取 100 筆記錄。
7. 這行程式碼會循序讀取每一筆記錄。
8. 此指令行會關閉檔案。
9. 此指令行會切斷與記錄層次存取服務的連線。

[下一部份 簡式程式設計範例

範例：使用記錄層次存取 (2 之 2)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Record level access example.
//
// Calling syntax: java RLACreateExample
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import com.ibm.as400.access.*;

public class RLACreateExample
{
 public static void main(String[] args)
  {
AS400 system = new AS400 (args[0]);

String filePathName = "/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MBR1.MBR";              附註 1

 try
    {
SequentialFile theFile = new SequentialFile(system, filePathName);

// Begin Note Two
CharacterFieldDescription lastNameField = new CharacterFieldDescription(new
AS400Text(20), "LNAME");
CharacterFieldDescription firstNameField = new CharacterFieldDescription(new
AS400Text(20), "FNAME");
BinaryFieldDescription yearsOld = new BinaryFieldDescription(new
AS400Bin4(), "AGE");

RecordFormat fileFormat = new RecordFormat("RF");
fileFormat.addFieldDescription(lastNameField);
fileFormat.addFieldDescription(firstNameField);
fileFormat.addFieldDescription(yearsOld);

theFile.create(fileFormat, "包含名稱與與經歷時間的檔案");            附註 2
// End Note Two


theFile.open(AS400File.READ_WRITE, 1, AS400File.COMMIT_LOCK_LEVEL_NONE);
```

```
// Begin Note Three
Record newData = fileFormat.getNewRecord();
newData.setField("LNAME", "Doe");
newData.setField("FNAME", "John");
newData.setField("AGE", new Integer(63));

theFile.write(newData);      附註 3
// End Note Three

theFile.close();
    }
catch (Exception e)
    {
System.out.println("發生錯誤：");
e.printStackTrace();
    }

system.disconnectService(AS400.RECORDACCESS);

 System.exit(0);
  }
}
```

1. 上一行的 (args[0]) 和 MYFILE.FILE 是程式碼的片段，
   它們是其餘範例賴以執行的先決條件。該程式假設檔案庫 MYLIB 存在伺服器中，且使用者擁有它的存取權。
2. Java 說明中標示著 "Begin Note Two" 和 "End Note Two"
   的文字示範如何自行建立記錄格式，而非從現有的檔案取得記錄格式。此區塊的最後一行會在伺服器上建立檔案。
3. Java 說明中標示著 "Begin Note Three" 和 "End Note Three"
   的文字示範一個方法，它可以產生記錄再將它寫入檔案中。

[上一部份 | 簡式程式設計範例

範例：使用 JDBC 類別建立及輸入表格資料 (2 之 1)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
//////////////////////////////////////////////////////////////////////////////
//
// JDBCPopulate example.This program uses the IBM Toolbox for Java JDBC
driver
// to create and populate a table.
//
// Command syntax:
//JDBCPopulate system collectionName tableName
//
// For example,
//JDBCPopulate MySystem MyLibrary MyTable
//
// This source is an example of IBM Toolbox for Java JDBC driver.
//
//////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCPopulate
{

private static final String words[]
= { "一","二","三","四", "五",
"六","七","八","九", "十",
"十一", "十二", "十三", "十四", "十五",
"十六","十七","十八", "十九", "二十" };



public static void main(String[] parameters)
    {

if (parameters.length != 3) {
System.out.println("");
System.out.println("用法：");
System.out.println("");
System.out.println(" JDBCPopulate system collectionName tableName");
System.out.println("");
System.out.println("");
System.out.println("例如：");
System.out.println("");
System.out.println("");
```

```
System.out.println(" JDBCPopulate MySystem MyLibrary MyTable");
System.out.println("");
return;
        }

String system = parameters[0];
String collectionName = parameters[1];
String tableName= parameters[2];

Connection connection = null;

try {


DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());
```
附註 1 

```
connection = DriverManager.getConnection ("jdbc:as400://"
+ system + "/" + collectionName);        附註 2 


try {
Statement dropTable = connection.createStatement ();
dropTable.executeUpdate ("DROP TABLE " + tableName);            附註 3 
            }
catch (SQLException e) {
            }


Statement createTable = connection.createStatement ();
createTable.executeUpdate ("CREATE TABLE " + tableName
+ " (I INTEGER, WORD VARCHAR(20), SQUARE INTEGER, "
+ " SQUAREROOT DOUBLE)");      附註 4 


PreparedStatement insert = connection.prepareStatement ("INSERT INTO "
+ tableName + " (I, WORD, SQUARE, SQUAREROOT) "
+ " VALUES (?, ?, ?, ?)");      附註 5 


for (int i = 1; i <= words.length; ++i) {
insert.setInt (1, i);
insert.setString (2, words[i-1]);
insert.setInt (3, i*i);
insert.setDouble (4, Math.sqrt(i));
insert.executeUpdate ();      附註 6 
            }

System.out.println ("表格 " + collectionName + "." + tableName
+ " 已輸入完成。");
        }

catch (Exception e) {
System.out.println ();
```

```
System.out.println ("錯誤：" + e.getMessage());
        }

finally {

try {
if (connection != null)
connection.close ();      附註 7
            }
catch (SQLException e) {
// Ignore.
            }
        }

 System.exit(0);
    }



}
```

1. 這一行會載入 IBM Toolbox for Java JDBC 驅動程式。JDBC 驅動程式是 JDBC 和您使用的資料庫之間的必要通訊管道。
2. 此陳述式會連線到資料庫。會出現提示請您提供使用者 ID 和密碼。系統會提供一個預設綱目，如此您就不需在 SQL 陳述式中定義表格名稱。
3. 這些指令行會刪除已存在的表格。
4. 這些指令行會建立表格。
5. 這個指令行會備妥一個陳述式，用來在表格中插入列。由於您要執行這個陳述式數次，所以您應該使用 PreparedStatement 和參數標記。
6. 此程式碼區塊會為您輸入表格資料；每當執行迴圈時，它會將列插入表格中。
7. 現在表格已建立好並填入資料，此陳述式會關閉與資料庫的連線。

[下一部份 | 簡式程式設計範例

範例：使用 JDBC 類別建立及輸入表格資料 (2 之 2)

請使用下列內容作為您的程式範例。該範例包括程式碼中的關鍵行的詳細說明。
您可以使用下列方式來檢視詳細說明：

- 按一下  影像，以便在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，來察看位於範例底端的詳細說明。

註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
//////////////////////////////////////////////////////////////////////////////
//
// JDBCQuery example.This program uses the IBM Toolbox for Java JDBC driver
to
// query a table and output its contents.
//
// Command syntax:
//JDBCQuery system collectionName tableName
//
// For example,
//JDBCQuery MySystem qiws qcustcdt
//
// This source is an example of IBM Toolbox for Java JDBC driver.
//
//////////////////////////////////////////////////////////////////////////////

import java.sql.*;

public class JDBCQuery
{



// Format a string so that it has the specified width.
private static String format (String s, int width)
    {
String formattedString;

// The string is shorter than specified width,
// so we need to pad with blanks.
if (s.length() < width) {
StringBuffer buffer = new StringBuffer (s);
for (int i = s.length(); i < width; ++i)
buffer.append (" ");
formattedString = buffer.toString();
        }

// Otherwise, we need to truncate the string.
else
formattedString = s.substring (0, width);

return formattedString;
```

```
    }


public static void main(String[] parameters)
    {
// Check the input parameters.
if (parameters.length != 3) {
System.out.println("");
System.out.println("用法：");
System.out.println("");
System.out.println(" JDBCQuery system collectionName tableName");
System.out.println("");
System.out.println("");
System.out.println("例如：");
System.out.println("");
System.out.println("");
System.out.println(" JDBCQuery mySystem qiws qcustcdt");
System.out.println("");
return;
        }

String system = parameters[0];
String collectionName = parameters[1];
String tableName= parameters[2];

Connection connection = null;

try {


DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver());
```
附註 1 

```
// Get a connection to the database.Since we do not
// provide a user id or password, a prompt will appear.
connection = DriverManager.getConnection ("jdbc:as400://" + system);
DatabaseMetaData dmd = connection.getMetaData ();
```
附註 2 

```
// Execute the query.
Statement select = connection.createStatement ();
ResultSet rs = select.executeQuery ("SELECT * FROM "
+ collectionName + dmd.getCatalogSeparator() + tableName);
```
附註 3 

```
// Get information about the result set.Set the column
// width to whichever is longer: the length of the label
// or the length of the data.
ResultSetMetaData rsmd = rs.getMetaData ();
int columnCount = rsmd.getColumnCount ();
```
附註 4 

```
String[] columnLabels = new String[columnCount];
int[] columnWidths = new int[columnCount];
for (int i = 1; i <= columnCount; ++i) {
columnLabels[i-1] = rsmd.getColumnLabel (i);
columnWidths[i-1] = Math.max (columnLabels[i-1].length(),
rsmd.getColumnDisplaySize (i));
```
附註 5 

```
            }

// Output the column headings.
for (int i = 1; i <= columnCount; ++i) {
System.out.print (format (rsmd.getColumnLabel(i), columnWidths[i-1]));
System.out.print (" ");
            }
System.out.println ();

// Output a dashed line.
StringBuffer dashedLine;
for (int i = 1; i <= columnCount; ++i) {
for (int j = 1; j <= columnWidths[i-1]; ++j)
System.out.print ("-");
System.out.print (" ");
            }
System.out.println ();

// Iterate throught the rows in the result set and output
// the columns for each row.
while (rs.next ()) {
for (int i = 1; i <= columnCount; ++i) {
String value = rs.getString (i);
if (rs.wasNull ())
value = "<null>";     附註 6
System.out.print (format (value, columnWidths[i-1]));
System.out.print (" ");
                }
System.out.println ();
            }

        }

catch (Exception e) {
System.out.println ();
System.out.println ("錯誤：" + e.getMessage());
        }

finally {

// Clean up.
try {
if (connection != null)
connection.close ();
            }
catch (SQLException e) {
// Ignore.
            }
        }

 System.exit(0);
    }


}
```

1. 這一行會載入 IBM Toolbox for Java JDBC 驅動程式。JDBC 驅動程式扮演 JDBC 與您使用的資料庫之間的溝通管道。
2. 此指令行會擷取連線的 meta 資料，該資料是說明資料庫諸多性質的一個物件。
3. 此陳述式會執行指定表格的查詢。
4. 這些指令行會擷取表格的相關資訊。
5. 這些指令行會將直欄寬度設定為標籤長度或資料長度 (端視何者較長)。
6. 此程式碼區塊會疊代表格中的所有列，並顯示每一列的每一直欄內容。

[上一部份 | 簡式程式設計範例]

範例：以 GUI 形式顯示伺服器工作清單

請使用下列內容作為程式範例。該範例包括程式碼中相關關鍵行的詳細說明。 您可以使用下列方式檢視詳細說明：

- 按一下  影像，以在蹦現視窗中顯示詳細說明。
- 按一下文字鏈結，以查看位於範例底端的詳細說明。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Example using the IBM Toolbox for Java's vaccess
// class, VJobList.
//
// This source is an example of IBM Toolbox for Java "Job List".
//
////////////////////////////////////////////////////////////////////////////

package examples;     附註 1

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;       附註 2

import javax.swing.*;      附註 3
import java.awt.*;
import java.awt.event.*;

public class GUIExample
{

public static void main(String[] parameters)          附註 4
  {
 GUIExample example = new GUIExample(parameters);

  }

public GUIExample(String[] parameters)
  {
 try  附註 5
     {
// Create an AS400 object.
 //The system name was passed as the first command line argument.
AS400 system = new AS400 (parameters[0]);          附註 6

VJobList jobList = new VJobList (system);          附註 7

// Create a frame.
 JFrame frame = new JFrame ("Job List Example");          附註 8

 // Create an error dialog adapter.This will display any errors to the user.
 ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (frame);              附註 9
```

```
 // Create an explorer pane to present the job list.
 AS400ExplorerPane explorerPane = new AS400ExplorerPane (jobList);              附註 10
```

```
 explorerPane.addErrorListener (errorHandler);          附註 11
```

```
// Use load to load the information from the system.
 explorerPane.load();      附註 12
```

```
 // When the frame closes, exit the program.
 frame.addWindowListener (new WindowAdapter ()          附註 13
     {
public void windowClosing (WindowEvent event)
          {
 System.exit(0);
          }
     } );
```

```
// Layout the frame with the explorer pane.
 frame.getContentPane().setLayout(new BorderLayout() );
 frame.getContentPane().add("Center", explorerPane);            附註 14
```

```
frame.pack();
 frame.show();    附註 15
     }
```

```
catch (Exception e)
     {
e.printStackTrace();      附註 16
```

```
 System.exit(0);     附註 17
     }
     }
}
```

1. 此類別在範例套件中。Java 使用套件以避免 Java 類別檔案之間的名稱衝突。
2. 這一行使得 vaccess 套件中的所有 IBM Toolbox for Java 類別可供此程式使用。Vaccess 套件中的類別具有通用字首 com.ibm.as400.vaccess。 使用 import 陳述式，程式可呼叫名稱，而非套件與名稱。例如，您可使用 AS400ExplorerPane，而非 com.ibm.as400.AS400ExplorerPane 參照 AS400ExplorerPane 類別。
3. 這一行使得 Swing 套件中的所有 Java Foundation Classes (JFC) 可供此程式使用。 使用 IBM Toolbox for Java vacce 類別的 Java 程式，需要 Sun Microsystems, Inc. 的 JDK 1.1.2 與 Java Swing 1.0.3， 而 Swing 可從 Sun 的 JFC 1.
4. 此類別具有主要方法，因此可當作應用程式執行。若要呼叫程式， 請執行 "java examples.GUIExample serverName"，其中 serverName 是您伺服器的名稱。jt400.zip 或 jt400.jar 必須在類別路徑中，才可呼叫此程式。
5. IBM Toolbox for Java 程式碼拋出程式必須捕捉的異常。
6. AS400 類別由 IBM Toolbox for Java 所用。此類別管理登入資訊、建立及維護 Socket 連線，並傳送及接收資料。在此範例中，程式將傳送伺服器名稱給 AS400 物件。
7. VJobList 類別由 IBM Toolbox for Java 使用，以呈現可在 vaccess (GUI) 元件中顯示的伺服器工作清單。請注意，使用 AS400 物件可指定清單所在的伺服器。
8. 此行可建構頁框或頂層視窗，以使用於顯示工作清單。
9. ErrorDialogAdapter 是 IBM Toolbox for Java 圖形式使用者介面 (GUI) 元件，建立此元件可當每次應用程式發生錯誤事件時，自動顯示對話視窗。
10. 此行可建立 AS400ExplorerPane，此為圖形式使用者介面 (GUI)，可表示伺服器資源中的中的物件階層。AS400ExplorerPan VJobList 的左邊根部顯示樹狀結構，而在右邊顯示資源的明細。如此僅將窗格起始設定為預設狀態，而不載入 VJobList 的內容到窗格中。
11. 此行會新增您在步驟九所建立的錯誤處理程式， 作為 VJobList 圖形式使用者介面 (GUI) 元件上的接收程式。
12. 此行會將 JobList 的內容載入 ExplorerPane 中。必須明確呼叫此方法，以與伺服器通信並從伺服器載入資訊。如此可讓應用程式控制與伺服器通信的時間。使用此方法您可：

- 將窗格新增到頁框之前先載入內容。載入所有資訊之前將不顯示頁框，如本範例。
- 將窗格新增到頁框且顯示該頁框之後再載入內容。頁框出現「等待游標」，而在載入資訊時將填入資訊。

13. 此行可新增視窗接收程式，以在關閉框架時結束應用程式。
14. 此行將工作清單圖形式使用者介面 GUI 元件新增到控制框架中央。
15. 此行呼叫 show 方法，可讓使用者看見視窗。
16. IBM Toolbox for Java 異常已轉換，因此將以工作站的語言顯示文字。 例如，此程式可在錯誤處理程序時，顯示異常文字
17. IBM Toolbox for Java 建立緒，以執 行 IBM Toolbox for Java 活動。若程式在終止時沒有執行 System.exit(0)，可能無法正常結束。例如， 若從 Windows 95 DOS 提示執行此程式而不使用此行，程式完成時將不會返回指令提示。

[簡式程式設計範例]

範例：程式設計祕訣

本節會列出管理連線主題文件中提供的程式碼範例。

管理連線

- 範例：以 CommandCall 物件建立 iSeries 伺服器連線
- 範例：以 CommandCall 物件建立兩個 iSeries 伺服器連線
- 範例：以 AS/400 物件建立 CommandCall 與 IFSFileInputStream 物件
- 範例：使用 AS400ConnectionPool 事先連線 iSeries 伺服器
- 範例：使用 AS400ConnectionPool 事先連線 iSeries 伺服器上的特定服務程式，然後重新使用連線

啟動與結束連線

- 範例：Java 程式如何事先連線到 iSeries 伺服器
- 範例：Java 程式如何切斷 iSeries 伺服器連線
- 範例：Java 程式如何以 disconnectService() 與 run() 切斷 iSeries 伺服器連線並重新連接它
- 範例：Java 程式如何切斷 iSeries 伺服器連線且無法重新連線

異常情況

- 範例：使用異常情況

錯誤事件

- 範例：處理錯誤事件
- 範例：定義錯誤接收器
- 範例：使用自訂的處理程式來處理錯誤事件

追蹤

- 範例：使用追蹤
- 範例：使用 setTraceOn()
- 範例：使用元件追蹤

最佳化

- 範例：建立兩個 AS400 物件
- 範例：使用 AS400 物件來代表第二個伺服器

安裝與更新

- 範例：使用 AS400ToolboxInstaller 類別

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

程式碼不保事項聲明

**IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
您可以依據這些範例，產生類似的函數，來符合您的需要。**

IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM 不擔保或默示保證這些程式的可靠性、可用性或功能。

**所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。**

»範例：ToolboxME for iSeries

本節會列出 ToolboxME for iSeries 文件中提供的程式碼範例。

- 範例：建立 ToolboxME for iSeries 範例 - JdbcDemo.java
- 範例：使用 ToolboxME for iSeries、MIDP 及 JDBC
- 範例：使用 ToolboxME for iSeries、MIDP 及 Toolbox for Java

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

»

範例：使用 ToolboxME for iSeries、MIDP 和 JDBC

下列原始程式說明 ToolboxME for iSeries 應用程式可以使用 Information Device Profile (MIDP) 和 JDBC 來離線存取資料庫和儲存資訊的一種方式。

此範例示範一家房屋仲介商如何能夠檢視目前銷售的房屋以及出價。 仲介商使用 Tier0 裝置存取房屋資訊，它儲存在 iSeries 伺服器資料庫中。

當下列程式碼範例是建置成工作程式時，它連接到基於此用途而建立的資料庫。

要建立原始程式碼的工作版本及取得原始程式來建立必要的資料庫及大量輸入資料，您必須下載此範例。您也可以複查 建立和執行此程式範例的指示。

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// ToolboxME for iSeries example. This program is an example MIDlet that
shows how
// you might code a JdbcMe application for the MIDP profile. Refer to the
// startApp, pauseApp, destroyApp and commandAction methods to see how it
handles
// each requested transition.
//
////////////////////////////////////////////////////////////////////////////

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.sql.*;
import javax.microedition.rms.*;

import com.ibm.as400.micro.*;


public class JdbcMidpBid extends MIDlet implements CommandListener
{
    private static int BID_PROPERTY = 0;
    private Display   display;

    private TextField urlText = new TextField("urltext",
"jdbc:as400://mySystem;user=myUid;password=myPwd;", 65, TextField.ANY);
    private TextField jdbcmeText = new TextField("jdbcmetext",
"meserver=myMEServer", 40, TextField.ANY);
    private TextField jdbcmeTraceText = new TextField("jdbcmetracetext",
"0", 10, TextField.ANY);
    private final static String GETBIDS = "沒有投標，請選取這裡下載投標";
    private List      main = new List("JdbcMe 投標示範程式",
Choice.IMPLICIT);
    private List      listings = null;
    private Form      aboutBox;
```

```
    private Form      bidForm;
    private Form      settingsForm;
    private int       bidRow = 0;
    private String    bidTarget = null;
    private String    bidTargetKey = null;
    private TextField bidText = new TextField("bidtext", "", 10,
TextField.NUMERIC);
    private Form      errorForm = null;

    private Command exitCommand   = new Command("結束", Command.SCREEN, 0);
    private Command backCommand   = new Command("上一步", Command.SCREEN,
0);
    private Command cancelCommand = new Command("取消", Command.SCREEN, 0);
    private Command goCommand     = new Command("到", Command.SCREEN, 1);
    private Displayable   onErrorGoBackTo = null;

    /*
     * Construct a new JdbcMidpBid.
     */
    public JdbcMidpBid()
    {
        display = Display.getDisplay(this);
    }

    /**
     * Show the main screen
     */
    public void startApp()
    {
        main.append("顯示投標", null);
        main.append("取得新投標", null);
        main.append("設定值", null);
        main.append("關於", null);
        main.addCommand(exitCommand);
        main.setCommandListener(this);

        display.setCurrent(main);
    }

    public void commandAction(Command c, Displayable s)
    {
        // All exitCommand processing is the same.
        if (c == exitCommand)
        {
            destroyApp(false);
            notifyDestroyed();
    return;
        }
        if (s instanceof List)
        {
            List     current = (List)s;

            // An action occurred on the main page
            if (current == main)
            {
                int   idx = current.getSelectedIndex();
                switch (idx)
```

```
            {
            case 0:     // Show current bids
                showBids();
              break;
            case 1:     // Get New Bids
                getNewBids();
              break;
            case 2:     // Settings
                doSettings();
              break;
            case 3:     // About
                aboutBox();
              break;
           default:
              break;
            }
return;
        } // current == main

        // An action occurred on the listings page
        if (current == listings)
        {
            if (c == backCommand)
            {
                display.setCurrent(main);
return;
            }
            if (c == List.SELECT_COMMAND)
            {
                int idx = listings.getSelectedIndex();
                String stext = listings.getString(idx);
                if (stext.equals(GETBIDS))
                {
                    getNewBids();
return;
                }
                int   commaIdx = stext.indexOf(',');
                bidTargetKey = stext.substring(0, commaIdx);
                bidTarget = stext.substring(commaIdx+1) + "\n";
                // Also keep track of which offline result set row
                // This is. It happens to be the same as the index
                // in the list.
                bidRow = idx;

                bidOnProperty();
            }
        } // current == listings
return;
    } // instanceof List
    if (s instanceof Form)
    {
        Form     current = (Form)s;
        if (current == errorForm)
        {
            if (c == backCommand)
                display.setCurrent(onErrorGoBackTo);
```

```
    return;
            } // errorForm
            if (current == settingsForm)
            {
                if (c == backCommand)
                {
                    // Done with settings.
                    display.setCurrent(main);
                    settingsForm = null;
    return;
                }
            } // settingsForm
            if (current == aboutBox)
            {
                if (c == backCommand)
                {
                    // Done with about box.
                    display.setCurrent(main);
                    aboutBox = null;
    return;
                }
            }
            if (current == bidForm)
            {
                if (c == cancelCommand)
                {
                    display.setCurrent(listings);
                    bidForm = null;
    return;
                }
                if (c == goCommand)
                {
                    submitBid();
                    if (display.getCurrent() != bidForm)
                    {
                        // If we're no longer positioned at the
                        // bidForm, we will get rid of it.
                        bidForm = null;
                    }
    return;
                }
    return;
            } // current == bidForm
        } // instanceof Form
    }

    public void aboutBox()
    {
        aboutBox = new Form("aboutbox");
        aboutBox.setTitle("About");
        aboutBox.append(new StringItem("", "JdbcMe 的 Midp RealEstate 範例
"));
        aboutBox.addCommand(backCommand);
        aboutBox.setCommandListener(this);
        display.setCurrent(aboutBox);
```

```
}

/**
 *  The settings form.
 */
public void doSettings()
{
    settingsForm = new Form("settingsform");
    settingsForm.setTitle("Settings");
    settingsForm.append(new StringItem("", "DB URL"));
    settingsForm.append(urlText);
    settingsForm.append(new StringItem("", "JdbcMe server"));
    settingsForm.append(jdbcmeText);
    settingsForm.append(new StringItem("", "Trace"));

    settingsForm.addCommand(backCommand);
    settingsForm.setCommandListener(this);
    display.setCurrent(settingsForm);
}


/**
 * Show the bid screen for the bid target
 * that we selected.
 */
public void bidOnProperty()
{
    StringItem item = new StringItem("", bidTarget);

    bidText = new TextField("bidtext", "", 10, TextField.NUMERIC);
    bidText.setString("");

    bidForm = new Form("bidform");
    bidForm.setTitle("出價：");
    BID_PROPERTY = 0;
    bidForm.append(item);
    bidForm.append(new StringItem("", "您的出價："));
    bidForm.append(bidText);
    bidForm.addCommand(cancelCommand);
    bidForm.addCommand(goCommand);
    bidForm.setCommandListener(this);
    display.setCurrent(bidForm);
}

/**
 * Update the listings card with the
 * current list of bids that we're interested in.
 */
public void getNewBids()
{
    // Reset the old listing
    listings = null;
    listings = new List("JdbcMe Bids", Choice.IMPLICIT);
    java.sql.Connection     conn = null;
    Statement               stmt = null;
try
    {
```

```
            conn = DriverManager.getConnection(urlText.getString() + ";" +
jdbcmeText.getString());

            stmt = conn.createStatement();

            // Since we don't want the prepared statement to persist,
            // a normal statement is really better in this environemnt.
            String sql = "select mls, address, currentbid from
qjdbcme.realestate where currentbid <> 0";

            boolean results
=((JdbcMeStatement)stmt).executeToOfflineData(sql,"JdbcMidpBidListings",0,
0);
            if (results)
            {
                setupListingsFromOfflineData();
            }
        else
            {
                listings.append("找不到出價", null);
                listings.addCommand(backCommand);
                listings.setCommandListener(this);
            }
        }
      catch (Exception e)
        {
            // Currently no valid listings retrieved, so lets
            // reset it to empty.
            listings = new List("JdbcMe Bids", Choice.IMPLICIT);
            listings.append(GETBIDS, null);
            listings.addCommand(backCommand);
            listings.setCommandListener(this);

            // Return to main after showing the error.
            showError(main, e);
    return;
        }
        finally
        {
            if (conn != null)
            {
    try
                {
                    conn.close();
                }
      catch (Exception e)
                {
                }
            }
            conn = null;
            stmt = null;
        }
        showBids();
    }

    public void setupListingsFromOfflineData()
    {
```

```
    // Skip the first four rows in the record store
    // (eyecatcher, version, num columns, sql column
    // types)
    // and each subsequent row in the record store is
    // a single column. Our query returns 3 columns which
    // we'll return concatenated as a single string.
    ResultSet         rs = null;
    listings.addCommand(backCommand);
    listings.setCommandListener(this);
try
    {
        int                     i = 5;
        int                     max = 0;
        StringBuffer   buf  = new StringBuffer(20);

        // Creator and dbtype unused in MIDP
        rs = new JdbcMeOfflineResultSet("JdbcMidpBidListings", 0, 0);
        if (rs == null)
        {
            // New listings...
            listings = new List("JdbcMe Bids", Choice.IMPLICIT);
            listings.append(GETBIDS, null);
            listings.addCommand(backCommand);
            listings.setCommandListener(this);
return;
        }

        i = 0;
        String      s = null;
 while (rs.next ())
        {
            ++i;

            s = rs.getString(1);
            buf.append(s);

            buf.append(",");
            s = rs.getString(2);
            buf.append(s);

            buf.append(", $");
            s = rs.getString(3);
            buf.append(s);

            listings.append(buf.toString(), null);
            buf.setLength(0);
        }

        if (i == 0)
        {
            listings.append("找不到出價", null);
return;
        }
    }
  catch (Exception e)
    {
        // Currently no valid listings retrieved, so lets
```

```
            // reset it to empty.
            listings = new List("JdbcMe 投標", Choice.IMPLICIT);
            listings.append(GETBIDS, null);
            listings.addCommand(backCommand);
            listings.setCommandListener(this);

            // Return to main after showing the error.
            showError(main, e);
    return;
        }
        finally
        {
            if (rs != null)
            {
    try
                {
                    rs.close();
                }
      catch (Exception e)
                {
                }
                rs = null;
            }
            System.gc();
        }
    }

    /**
     * Update the listings card with the
     * current list of bids that we're interested in.
     */
    public void submitBid()
    {
        java.sql.Connection     conn = null;
        Statement               stmt = null;
    try
        {
            conn = DriverManager.getConnection(urlText.getString() + ";" +
jdbcmeText.getString());

            stmt = conn.createStatement();

            // Since we don't want the prepared statement to persist,
            // a normal statement is really better in this environemnt.
            StringBuffer   buf = new StringBuffer(100);
            buf.append("Update QJdbcMe.RealEstate Set CurrentBid = ");
            buf.append(bidText.getString());
            buf.append(" Where MLS = '");
            buf.append(bidTargetKey);
            buf.append("' and CurrentBid < ");
            buf.append(bidText.getString());
            String      sql = buf.toString();

            int   updated = stmt.executeUpdate(sql);
            if (updated == 1)
            {
                // BID Accepted.
```

```
                String oldS = listings.getString(bidRow);
                int   commaIdx = bidTarget.indexOf(',');
                String bidAddr = bidTarget.substring(0, commaIdx);

                String newS = bidTargetKey + "," + bidAddr + ", $" +
bidText.getString();

                ResultSet         rs = null;
    try
                {
                    // Creator and dbtype unused in MIDP
                    rs = new JdbcMeOfflineResultSet("JdbcMidpBidListings",
0, 0);
                    rs.absolute(bidRow+1);
                    rs.updateString(3, bidText.getString());
                    rs.close();
                }
      catch (Exception e)
                {
                    if (rs != null)
                        rs.close();
                }

                // Also update our live list of that result set.
                listings.set(bidRow, newS, null);
                display.setCurrent(listings);
                conn.commit();
            }
        else
            {
                conn.rollback();
                throw new SQLException("出價失敗, 有人勝過你");
            }
        }
        catch (SQLException e)
        {
            // Return to the bid form after showing the error.
            showError(bidForm, e);
    return;
        }
        finally
        {
            if (conn != null)
            {
    try
                {
                    conn.close();
                }
      catch (Exception e)
                {
                }
            }
            conn = null;
            stmt = null;
        }

        // Exit without exception, then show the current bids
```

```
        showBids();
    }

    /**
     * Show an error condition.
     */
    public void showError(Displayable d, Exception e)
    {
        String s = e.toString();

        onErrorGoBackTo = d;
        errorForm = new Form("錯誤");
        errorForm.setTitle("SQL 錯誤");
        errorForm.append(new StringItem("", s));
        errorForm.addCommand(backCommand);
        errorForm.setCommandListener(this);
        display.setCurrent(errorForm);
    }

    /**
     * Show the current bids.
     */
    public void showBids()
    {
        if (listings == null)
        {
            // If we have no current listings, lets set
            // them up.
            listings = new List("JdbcMe Bids", Choice.IMPLICIT);
            setupListingsFromOfflineData();
        }
        display.setCurrent(listings);
    }

    /**
     * Time to pause, free any space we don't need right now.
     */
    public void pauseApp()
    {
        display.setCurrent(null);
    }

    /**
     * Destroy must cleanup everything.
     */
    public void destroyApp(boolean unconditional)
    {
    }
}«
```

»

範例：使用 ToolboxME for iSeries、MIDP 和 Toolbox for Java

下列原始程式說明 ToolboxME for iSeries 應用程式可以使用 Java Information Device Profile (MIDP) 和 IBM Toolbox for Java 在 iSeries 伺服器上存取資料和服務的一種方式。

此範例示範內建於 IBM Toolbox for Java 2 Micro Edition 支援的每一個功能。此應用程式的特性是一些不一樣的頁面或畫面，說明 Tier0 裝置可使用這些功能的一些方式。

當下列程式碼範例是建置成工作程式時，它使用 Program Call Markup Language (PCML) 檔案在 iSeries 伺服器上執行指令。

要建立原始程式碼的工作版本及取得必要的 PCML 原始程式來執行伺服器指令，您必須下載此範例。您也可以複查 建立和執行此程式範例的指示。

註：請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// ToolboxME for iSeries example. This program is an example that shows how
// ToolboxME for iSeries can use PCML to access data and services on an
// iSeries server.
//
// This application requires that the qsyrusri.pcml file is present in the
// CLASSPATH of the MEServer.
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
import java.sql.*;
import java.util.Hashtable;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

import com.ibm.as400.micro.*;


public class ToolboxMidpDemo extends MIDlet implements CommandListener
{
    private Display   display_;

    // A ToolboxME system object.
   private AS400  system_;

    private List      main_ = new List("Toolbox MIDP Demo",
Choice.IMPLICIT);

    // Create a form for each component.
```

```
    private Form      signonForm_;
    private Form      cmdcallForm_;
    private Form      pgmcallForm_;
    private Form      dataqueueForm_;
    private Form      aboutForm_;

    // Visable Text for each component.
    static final String SIGN_ON       = "SignOn";
    static final String COMMAND_CALL  = "CommandCall";
    static final String PROGRAM_CALL  = "ProgramCall";
    static final String DATA_QUEUE    = "DataQueue";
    static final String ABOUT         = "About";

    static final String NOT_SIGNED_ON = "Not signed on.";
    static final String DQ_READ       = "Read";
    static final String DQ_WRITE      = "Write";

    // A ticker to display the signon status.
    private Ticker    ticker_ = new Ticker(NOT_SIGNED_ON);

    // Commands that can be performed.
    private static final Command actionExit_   = new Command("Exit",
Command.SCREEN, 0);
    private static final Command actionBack_   = new Command("Back",
Command.SCREEN, 0);
    private static final Command actionGo_     = new Command("Go",
Command.SCREEN, 1);
    private static final Command actionClear_  = new Command("Clear",
Command.SCREEN, 1);
    private static final Command actionRun_    = new Command("Run",
Command.SCREEN, 1);
    private static final Command actionSignon_ = new Command(SIGN_ON,
Command.SCREEN, 1);
    private static final Command actionSignoff_= new Command("SignOff",
Command.SCREEN, 1);

    private Displayable   onErrorGoBackTo_;  // the form to return to when
done displaying the error form

    // TextFields for the SignOn form.
    private TextField signonSystemText_ = new TextField("System",
"rchasdm3", 20, TextField.ANY);
    private TextField signonUidText_ = new TextField("UserId", "JAVA", 10,
TextField.ANY);
    private TextField signonPwdText_ = new TextField("Password", "JTEAM1",
10, TextField.PASSWORD);  // TBD temporary
    private TextField signonServerText_ = new TextField("MEServer",
"localhost", 10, TextField.ANY);
    private StringItem signonStatusText_ = new StringItem("Status",
NOT_SIGNED_ON);

    // TextFields for the CommandCall form.
    private TextField cmdText_ = new TextField("Command", "CRTLIB FRED",
256, TextField.ANY);  // TBD: max size; TBD: TextBox???
    private StringItem cmdMsgText_ = new StringItem("Messages", null);
    private StringItem cmdStatusText_ = new StringItem("Status", null);
```

```
    // TextFields for the ProgramCall form.
    private StringItem pgmMsgDescription_ = new StringItem("Messages",
null);
    private StringItem pgmMsgText_ = new StringItem("訊息", null);

    // TextFields for the DataQueue form.
    private TextField dqInputText_ = new TextField("Data to write", "Hi
there", 30, TextField.ANY);
    private StringItem dqOutputText_ = new StringItem("DQ contents", null);
    private ChoiceGroup dqReadOrWrite_ = new ChoiceGroup("Action",
Choice.EXCLUSIVE, new String[] { DQ_WRITE, DQ_READ}, null);
    private StringItem dqStatusText_ = new StringItem("Status", null);


    /**
     * Creates a new ToolboxMidpDemo.
     **/
    public ToolboxMidpDemo()
    {
        display_ = Display.getDisplay(this);
        // Note: The KVM-based demo used TabbedPane for the main panel.
MIDP has no similar class, so we use a List instead.
    }

    /**
     * Show the main screen.
     * Implements abstract method of class Midlet.
     **/
    protected void startApp()
    {
        main_.append(SIGN_ON, null);
        main_.append(COMMAND_CALL, null);
        main_.append(PROGRAM_CALL, null);
        main_.append(DATA_QUEUE, null);
        main_.append(ABOUT, null);

        main_.addCommand(actionExit_);
        main_.setCommandListener(this);

        display_.setCurrent(main_);
    }

    // Implements method of interface CommandListener.
    public void commandAction(Command action, Displayable dsp)
    {
        // All 'exit' and 'back' processing is the same.
        if (action == actionExit_)
        {
            destroyApp(false);

            notifyDestroyed();
        }
        else if (action == actionBack_)
        {
            // Return to main menu.
            display_.setCurrent(main_);
        }
```

```
        else if (dsp instanceof List)
        {
            List current = (List)dsp;

            // An action occurred on the main page
            if (current == main_)
            {
                int   idx = current.getSelectedIndex();

                switch (idx)
                {
                case 0:     // SignOn
                    showSignonForm();
              break;
                case 1:     // CommandCall
                    showCmdForm();
              break;
                case 2:     // ProgramCall
                    showPgmForm();
              break;
                case 3:     // DataQueue
                    showDqForm();
              break;
                case 4:     // About
                    showAboutForm();
              break;
                default:    // None of the above
                    feedback("內部錯誤：無法處理在 main 選取的索引：" + idx,
AlertType.ERROR);
              break;
                }
            } // current == main
        else
                feedback("內部錯誤：Displayable 物件是一個清單但不是
main_。", AlertType.ERROR);
        } // instanceof List
        else if (dsp instanceof Form)
        {
            Form current = (Form)dsp;

            if (current == signonForm_)
            {
                if (action == actionSignon_)
                {
                    // Create a ToolboxME system object.
                    system_ = new AS400(signonSystemText_.getString(),
signonUidText_.getString(), signonPwdText_.getString(),
signonServerText_.getString());

    try
                    {
                        // Connect to the iSeries.
                        system_.connect();

                        // Set the signon status text.
                        signonStatusText_.setText("已登入。");
```

```
                        // Display a confirmation dialog that the user is
signed on.
                        feedback("已順利登入。", AlertType.INFO, main_);

                        // Replace the SignOn button with SignOff.
                        signonForm_.removeCommand(actionSignon_);
                        signonForm_.addCommand(actionSignoff_);

                        // Update the ticker.
                        ticker_.setString("... Signed on to '" +
signonSystemText_.getString() + "' as '" + signonUidText_.getString() + "'
via '" + signonServerText_.getString() + "' ... ");
                    }
      catch (Exception e)
                    {
            e.printStackTrace();

                        // Set the signon status text.
                        signonStatusText_.setText(NOT_SIGNED_ON);

                        feedback("Signon failed.  " + e.getMessage(),
AlertType.ERROR);
                    }
                }
                else if (action == actionSignoff_)
                {
      if (system_ == null)
                        feedback("內部錯誤：系統是空值。", AlertType.ERROR);
        else
                    {
    try
                        {
                            // Disconnect from the iSeries.
                            system_.disconnect();
                            system_ = null;

                            // Set the signon status text.
                            signonStatusText_.setText(NOT_SIGNED_ON);

                            // Display a confirmation dialog that the user
is no longer signed on.
                            feedback("已順利登出。", AlertType.INFO, main_);

                            // Replace the SignOff button with SignOn.
                            signonForm_.removeCommand(actionSignoff_);
                            signonForm_.addCommand(actionSignon_);

                            // Update the ticker.
                            ticker_.setString(NOT_SIGNED_ON);
                        }
      catch (Exception e)
                        {
                            feedback(e.toString(), AlertType.ERROR);

            e.printStackTrace();

                            signonStatusText_.setText("錯誤。");
```

```
                        feedback("登出期間發生錯誤。", AlertType.ERROR);
                    }
                }
            }
            else  // None of the above.
            {
                feedback("內部錯誤：動作無法被辨識。", AlertType.INFO);
            }
        } // signonForm_
        else if (current == cmdcallForm_)
        {
            if (action == actionRun_)
            {
                // If the user has not signed on, display an alert.
  if (system_ == null)
                {
                    feedback(NOT_SIGNED_ON, AlertType.ERROR);
    return;
                }

                // Get the command the user entered in the wireless
device.
                String cmdString = cmdText_.getString();

                // If the command was not specified, display an alert.
                if (cmdString == null || cmdString.length() == 0)
                    feedback("指定指令。", AlertType.ERROR);
        else
                {
    try
                    {
                   // Run the command.
                        String[] messages = CommandCall.run(system_,
cmdString);

                        StringBuffer status = new
StringBuffer("指令已完成 ");

                        // Check to see if their are any messages.
                        if (messages.length == 0)
                        {
                            status.append("無傳回的訊息。");

                            cmdMsgText_.setText(null);

                            cmdStatusText_.setText("指令已順利完成。");
                        }
        else
                        {
                            if (messages.length == 1)
                                status.append("1 個傳回的訊息。");
        else
                                status.append(messages.length + "
傳回的訊息。");

                            // If there are messages, display only the
first message.
```

```
                            cmdMsgText_.setText(messages[0]);

                            cmdStatusText_.setText(status.toString());
                        }

                        repaint();
                    }
      catch (Exception e)
                    {
                        feedback(e.toString(), AlertType.ERROR);

            e.printStackTrace();

                        feedback("執行指令時發生錯誤。",
AlertType.ERROR);
                    }
                }
            }
            else if (action == actionClear_)
            {
                // Clear the command text and messages.
                cmdText_.setString("");

                cmdMsgText_.setText(null);

                cmdStatusText_.setText(null);

                repaint();
            }
            else  // None of the above.
            {
                feedback("內部錯誤：動作無法被辨識。", AlertType.INFO);
            }
        } // cmdcallForm_
        else if (current == pgmcallForm_)
        {
            if (action == actionRun_)
            {
                // If the user is not signed on before doing a program
call, display an alert.
      if (system_ == null)
                {
                    feedback(NOT_SIGNED_ON, AlertType.ERROR);
    return;
                }

                pgmMsgText_.setText(null);

                // See the PCML example in the Toolbox programmer's
guide.
                String pcmlName = "qsyrusri.pcml"; // The PCML file we
want to use.
                String apiName = "qsyrusri";

                // Create a hashtable that contains the input parameters
for the program call.
                Hashtable parmsToSet = new Hashtable(2);
                parmsToSet.put("qsyrusri.receiverLength", "2048");
```

```
                    parmsToSet.put("qsyrusri.profileName",
signonUidText_.getString().toUpperCase());

                    // Create a string array that contains the output
parameters to retrieve.
                    String[] parmsToGet = { "qsyrusri.receiver.userProfile",
                        "qsyrusri.receiver.previousSignonDate",
                        "qsyrusri.receiver.previousSignonTime",
                        "qsyrusri.receiver.daysUntilPasswordExpires"};

                    // A string array containing the descriptions of the
parameters to display.
                    String[] displayParm = { "設定檔", "前次登入日期",
"前次登入時間", "密碼過期 (天數)"};

    try
                    {
                       // Run the program.
                        String[] valuesToGet = ProgramCall.run(system_,
pcmlName, apiName, parmsToSet, parmsToGet);

                        // Create a StringBuffer and add each of the
parameters we retreived.
                        StringBuffer txt = new StringBuffer();
                        txt.append(displayParm[0] + ": " + valuesToGet[0] +
"\n");

                        char[] c = valuesToGet[1].toCharArray();
                        txt.append(displayParm[1] + ": " +
c[3]+c[4]+"/"+c[5]+c[6]+"/"+c[1]+c[2]  + "\n");

                        char[] d = valuesToGet[2].toCharArray();
                        txt.append(displayParm[2] + ": " +
d[0]+d[1]+":"+d[2]+d[3] + "\n");
                        txt.append(displayParm[3] + ": " + valuesToGet[3] +
"\n");

                        // Set the displayable text of the program call
results.
                        pgmMsgText_.setText(txt.toString());

                        StringBuffer status = new StringBuffer("程式已完成
");

                        if (valuesToGet.length == 0)
                        {
                            status.append("無回覆值。");

                            feedback(status.toString(), AlertType.INFO);
                        }
        else
                        {
                            if (valuesToGet.length == 1)
                                status.append("1 個回覆值。");
        else
                                status.append(valuesToGet.length + "
回覆值。");
```

```
                        feedback(status.toString(), AlertType.INFO);
                    }
                }
      catch (Exception e)
                {
                    feedback(e.toString(), AlertType.ERROR);

            e.printStackTrace();

                    feedback("執行程式時發生錯誤。", AlertType.ERROR);
                }
            }
            else if (action == actionClear_)
            {
                // Clear the program call results.
                pgmMsgText_.setText(null);

                repaint();
            }
        }  // pgmcallForm_
        else if (current == dataqueueForm_)  // DataQueue
        {
            if (action == actionGo_)
            {
                // If the user has not signed on before performing Data
Queue actions, display an alert.
      if (system_ == null)
                {
                    feedback(NOT_SIGNED_ON, AlertType.ERROR);

    return;
                }

                // Create a library to create the data queue in.
    try
                {
                    CommandCall.run(system_, "CRTLIB FRED");
                }
      catch (Exception e)
                {
                }

                // Run a command to create a data queue.
    try
                {
                    CommandCall.run(system_, "CRTDTAQ FRED/MYDTAQ
MAXLEN(2000)");
                }
      catch (Exception e)
                {
                    feedback("建立資料佇列時發生錯誤。   " +
e.getMessage(), AlertType.WARNING);
                }

    try
                {
                    // See which action was selected (Read or Write).
```

```
                        if
(dqReadOrWrite_.getString(dqReadOrWrite_.getSelectedIndex()).equals(DQ_WRITE))
                        {
                            // Write
                            dqOutputText_.setText(null);

                            // Get the text from the wireless device input
to be written to the data queue.
                            if (dqInputText_.getString().length() == 0)
                                dqStatusText_.setText("未指定資料。");
        else
                            {
                                // Write to the data queue.
                                DataQueue.write(system_,
"/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ", dqInputText_.getString().getBytes() );

                                dqInputText_.setString(null);

                                // Display the status.
                                dqStatusText_.setText("'write'
作業已完成。");
                            }
                        }
                        else  // Read
                        {
                            // Read from the data queue.
                            byte[] b = DataQueue.readBytes(system_,
"/QSYS.LIB/FRED.LIB/MYDTAQ.DTAQ");

                            // Determine if the data queue contained entries
or not and display the appropriate message.
                            if (b == null)
                            {

dqStatusText_.setText("沒有資料佇列登錄可用。");

                                dqOutputText_.setText(null);
                            }
                            else if (b.length == 0)
                            {

dqStatusText_.setText("資料佇列登錄不含資料。");

                                dqOutputText_.setText(null);
                            }
        else
                            {
                                dqStatusText_.setText("'read'
作業已完成。");

                                dqOutputText_.setText(new String(b));
                            }
                        }

                        repaint();
                    }
      catch (Exception e)
                    {
```

```
            e.printStackTrace();

                    feedback(e.toString(), AlertType.ERROR);

                    feedback("Error when running command.  " +
e.getMessage(), AlertType.ERROR);
                }
            }  // actionGo_
            else if (action == actionClear_)
            {
                // Clear the data queue form.
                dqInputText_.setString("");

                dqOutputText_.setText(null);

                dqReadOrWrite_.setSelectedFlags(new boolean[] { true,
false});

                dqStatusText_.setText(null);

                repaint();
            }
            else  // None of the above.
            {
                feedback("內部錯誤：動作無法被辨識。", AlertType.INFO);
            }
        } // dataqueueForm_
        else if (current == aboutForm_)  // "關於".
        {
            // Should never reach here, since the only button is "Back".
        }  // None of the above.
    else
            feedback("內部錯誤：套表無法被辨識。", AlertType.ERROR);
    } // instanceof Form
    else
        feedback("內部錯誤：Displayable 物件無法被辨識。",
AlertType.ERROR);
    }



    /**
     *  Displays the "About" form.
     **/
    private void showAboutForm()
    {
        // If the about form is null, create and append it.
        if (aboutForm_ == null)
        {
            aboutForm_ = new Form(ABOUT);
            aboutForm_.append(new StringItem(null, "這是一個 MIDP
應用程式範例, 它使用 Toolbox Micro Edition (ToolboxME)。"));

            aboutForm_.addCommand(actionBack_);
            aboutForm_.setCommandListener(this);
        }

        display_.setCurrent(aboutForm_);
```

```
}


/**
 *  Displays the "SignOn" form.
 **/
private void showSignonForm()
{
    // Create the signon form.
    if (signonForm_ == null)
    {
        signonForm_ = new Form(SIGN_ON);
        signonForm_.append(signonSystemText_);
        signonForm_.append(signonUidText_);
        signonForm_.append(signonPwdText_);
        signonForm_.append(signonServerText_);
        signonForm_.append(signonStatusText_);
        signonForm_.addCommand(actionBack_);
        signonForm_.addCommand(actionSignon_);
        signonForm_.setCommandListener(this);
        signonForm_.setTicker(ticker_);
    }

    display_.setCurrent(signonForm_);
}


/**
 *  Displays the "CommandCall" form.
 **/
private void showCmdForm()
{
    // Create the command call form.
    if (cmdcallForm_ == null)
    {
        cmdcallForm_ = new Form(COMMAND_CALL);
        cmdcallForm_.append(cmdText_);
        cmdcallForm_.append(cmdMsgText_);
        cmdcallForm_.append(cmdStatusText_);
        cmdcallForm_.addCommand(actionBack_);
        cmdcallForm_.addCommand(actionClear_);
        cmdcallForm_.addCommand(actionRun_);
        cmdcallForm_.setCommandListener(this);
        cmdcallForm_.setTicker(ticker_);
    }

    display_.setCurrent(cmdcallForm_);
}


/**
 *  Displays the "ProgramCall" form.
 **/
private void showPgmForm()
{
```

```
        // Create the program call form.
        if (pgmcallForm_ == null)
        {
            pgmcallForm_ = new Form(PROGRAM_CALL);
            pgmcallForm_.append(new StringItem(null,
"這會呼叫「擷取使用者資訊 (QSYRUSRI)
API」並傳回關於現行使用者設定檔的資訊。"));
            pgmcallForm_.append(pgmMsgText_);
            pgmcallForm_.addCommand(actionBack_);
            pgmcallForm_.addCommand(actionClear_);
            pgmcallForm_.addCommand(actionRun_);
            pgmcallForm_.setCommandListener(this);
            pgmcallForm_.setTicker(ticker_);
        }

        display_.setCurrent(pgmcallForm_);
    }


    /**
     *  Displays the "DataQueue" form.
     **/
    private void showDqForm()
    {
        // Create the data queue form.
        if (dataqueueForm_ == null)
        {
            dataqueueForm_ = new Form(DATA_QUEUE);
            dataqueueForm_.append(dqInputText_);
            dataqueueForm_.append(dqOutputText_);
            dataqueueForm_.append(dqReadOrWrite_);
            dataqueueForm_.append(dqStatusText_);
            dataqueueForm_.addCommand(actionBack_);
            dataqueueForm_.addCommand(actionClear_);
            dataqueueForm_.addCommand(actionGo_);
            dataqueueForm_.setCommandListener(this);
            dataqueueForm_.setTicker(ticker_);
        }

        display_.setCurrent(dataqueueForm_);
    }


    private void feedback(String text, AlertType type)
    {
        feedback(text, type, display_.getCurrent());
    }

    /**
     *  This method is used to create a dialog and display feedback
information using an Alert to the user.
     **/
    private void feedback(String text, AlertType type, Displayable
returnToForm)
    {
        System.err.flush();
        System.out.flush();
```

```
        Alert alert = new Alert("警示", text, null, type);

        if (type == AlertType.INFO)
            alert.setTimeout(3000);  // milliseconds
        else
            alert.setTimeout(Alert.FOREVER);  // Require user to dismiss the
alert.

        display_.setCurrent(alert, returnToForm);
    }


    // Force a repaint of the current form.
    private void repaint()
    {
        Alert alert = new Alert("更新顯示 ...", null, null, AlertType.INFO);
        alert.setTimeout(1000);  // milliseconds

        display_.setCurrent(alert, display_.getCurrent());
    }



    /**
     * Time to pause, free any space we don't need right now.
     * Implements abstract method of class Midlet.
     **/
    protected void pauseApp()
    {
        display_.setCurrent(null);
    }



    /**
     * Destroy must cleanup everything.
     * Implements abstract method of class Midlet.
     **/
    protected void destroyApp(boolean unconditional)
    {
        // Disconnect from the iSeries if the Midlet is being destroyed or
exited.
        if (system_ != null)
        {
    try
            {
                system_.disconnect();
            }
      catch (Exception e)
            {
            }
        }
    }
}
```

«

範例：公用程式類別

本節會列出公用程式類別文件中提供的程式碼範例。

IBM Toolbox Installer

- 範例：使用 AS400ToolboxInstaller 類別
- 範例：經由使用AS400ToolboxInstaller 來安裝 IBM Toolbox for Java
- 範例：從指令行安裝 ACCESS 套件
- 範例：從指令行使用 Graphical Toolbox 類別

JarMaker

- 範例：從 jt400.jar 取出 AS400.class 及其所有相依類別
- 範例：將 jt400.jar 分成一組檔案，每一個檔案大小均為 300 個位元組
- 範例：從 JAR 檔案除去未使用的檔案
- 範例：例：經由 -ccsid 參數略過轉換表，來建立 一個較小的 JAR 檔案，其大小為 400 個位元組

CommandPrompter

- 範例：使用 CommandPrompter 來提示及執行指令

下列不保聲明適用於所有 IBM Toolbox for Java 範例：

> 程式碼不保事項聲明
>
> IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
> 您可以依據這些範例，產生類似的函數，來符合您的需要。
>
> IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM
> 不擔保或默示保證這些程式的可靠性、可用性或功能。
>
> 所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
> 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

範例：使用 AS400ToolboxInstaller 以安裝及更新 IBM Toolbox for Java

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// Install/Update example.This program uses the AS400ToolboxInstaller class
// to install and update the IBM Toolbox for Java package on the
workstation.
//
// The program checks the target path for the IBM Toolbox for Java package.
// If the package is not found, it installs the package on the workstation.
// If the package is found it checks the source path for updates.If
// updates are found they are copied to the workstation.
//
// Command syntax:
//checkToolbox source target
//
// Where
//source = location of the source files.This name is in URL format.
//target = location of the target files.
//
////////////////////////////////////////////////////////////////////////////

import java.io.*;
 import java.util.*;
import java.net.*;
import utilities.*;

public class checkToolbox extends Object
{
public static void main(String[] parameters)
   {
System.out.println( " " );



// Continue with the install/update only if both source and target
// names were specified.

if (parameters.length >= 2)
      {

 // The first parameter is the source for the files, the second is the
target.

 String sourcePath = parameters[0];
 String targetPath = parameters[1];

 boolean installIt = false;
 boolean updateIt= false;



 // Created a reader to get input from the user.

 BufferedReader inputStream = new BufferedReader(new
```

```
InputStreamReader(System.in),1);

 try
        {
// Point at the source package.AS400ToolboxInstaller uses the URL
// class to access the files.

URL sourceURL= new URL(sourcePath);



// See if the package is installed on the client.If not, ask the user
// if install should be performed at this time.

if (AS400ToolboxInstaller.isInstalled("ACCESS", targetPath) == false)
           {
 System.out.print("未安裝 IBM Toolbox for Java。立即安裝 (Y/N)：");

 String userInput = inputStream.readLine();

 if ((userInput.charAt(0) == 'y') ||
 (userInput.charAt(0) == 'Y'))
installIt = true;
           }



// The package is installed.See if updates need to be copied from the
// server.If the target is out of data ask the user if update should
// be performed at this time.

else
           {
 if (AS400ToolboxInstaller.isUpdateNeeded("ACCESS", targetPath, sourceURL)
== true)
              {
System.out.print("IBM Toolbox for Java 已過期。安裝修訂程式 (Y/N)：");

String userInput = inputStream.readLine();

if ((userInput.charAt(0) == 'y') ||
(userInput.charAt(0) == 'Y'))
 updateIt = true;
              }
else
System.out.println("目標目錄是最新的, 不需要更新。");
           }



// If the package needs to be installed or updated.

if (updateIt || installIt)
           {
 // Copy the files from the server to the target.

 AS400ToolboxInstaller.install("ACCESS", targetPath, sourceURL);
```

```
 // Report that the install/update was successful.

 System.out.println(" ");

 if (installIt)
System.out.println("安裝順利完成！");
else
System.out.println("更新順利完成！");



 // Tell the user what must be added to the CLASSPATH environment
 // variable.

 Vector classpathAdditions = AS400ToolboxInstaller.getClasspathAdditions();

 if (classpathAdditions.size() > 0)
                {
System.out.println("");
System.out.println("新增下列到 CLASSPATH 環境變數：");

for (int i = 0; i < classpathAdditions.size(); i++)
                  {
 System.out.print("");
 System.out.println((String)classpathAdditions.elementAt(i));
                  }
               }



 // Tell the user what can be revmoed from the CLASSPATH environment
 // variable.

 Vector classpathRemovals = AS400ToolboxInstaller.getClasspathRemovals();

 if (classpathRemovals.size() > 0)
                {
System.out.println("");
System.out.println("從 CLASSPATH 環境變數中移除下列：");

for (int i = 0; i < classpathRemovals.size(); i++)
                  {
 System.out.print("");
 System.out.println((String)classpathRemovals.elementAt(i));
                  }
               }
            }
         }

catch (Exception e)
         {
// If any of the above operations failed say the operation failed
// and output the exception.

System.out.println("安裝/更新失敗");
System.out.println(e);
         }
      }
```

```
// Display help text when parameters are incorrect.

else
      {
 System.out.println("");
 System.out.println("");
 System.out.println("");
 System.out.println("參數不正確。指令語法為："
 System.out.println("");
 System.out.println("checkToolbox sourcePath targetPath");
 System.out.println("");
 System.out.println("其中"
 System.out.println("");
 System.out.println(" sourcePath = IBM Toolbox for Java 檔案的來源");
 System.out.println(" targetPath = IBM Toolbox for Java 檔案的目標");
 System.out.println("");
 System.out.println("例如："
 System.out.println("");
 System.out.println(" checkToolbox
http://mySystem/QIBM/ProdData/HTTP/Public/jt400/ d:\\jt400");
 System.out.println("");
 System.out.println("");
      }

 System.exit(0);
   }
}
```

»

範例：使用 CommandPrompter

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////////
//
// CommandPrompter example. This program uses CommandPrompter, CommandCall,
and
// AS400Message to prompt for a command, run the command, and display any
// messages returned if the command does not run.
//
// Command syntax:
//    Prompter commandString
//
////////////////////////////////////////////////////////////////////////////

import com.ibm.as400.ui.util.CommandPrompter;
import com.ibm.as400.access.AS400;
import com.ibm.as400.access.AS400Message;
import com.ibm.as400.access.CommandCall;
import javax.swing.JFrame;
import java.awt.FlowLayout;
public class Prompter
{
public static void main ( String args[] ) throws Exception
  {
  JFrame frame = new JFrame();
  frame.getContentPane().setLayout(new FlowLayout());
  AS400 system = new AS400("mySystem", "myUserId", "myPasswd");
  String cmdName = args[0];

  // Launch the CommandPrompter
  CommandPrompter cp = new CommandPrompter(frame, system, cmdName);
  if (cp.showDialog() == CommandPrompter.OK)
    {
    String cmdString = cp.getCommandString();
    System.out.println("Command string: " + cmdString);

    // Run the command that was built in the prompter.
    CommandCall cmd = new CommandCall(system, cmdString);
    if (!cmd.run())
      {
      AS400Message[] msgList = cmd.getMessageList();
      for (int i = 0; i < msgList.length; ++i)
        {
          System.out.println(msgList[i].getText());
        }
      }
    }
       System.exit(0);
  }
}
```

«

範例：Vaccess 類別

本節會列出 Vaccess 類別文件中提供的程式碼範例。

AS400Panes

- 範例：建立 AS400DetailsPane 來呈現 systemAS400DetailsPane 上使用者定義的清單
- 範例：下面範例會在將明細窗格的內容新增到訊框之前先載入該內容
- 範例：使用 AS400ListPane 來呈現使用者清單
- 範例：使用 AS400DetailsPane 來顯示從指令呼叫傳回的訊息
- 範例：使用 AS400TreePane 來顯示目錄的樹狀檢視畫面
- 範例：使用 AS400ExplorerPane 來呈現不同的列印資源

指令呼叫

- 範例：建立 CommandCallButton
- 範例：新增 ActionCompletedListener 來處理指令所產生的全部 iSeries 訊息
- 範例：使用 CommandCallMenuItem

資料佇列

- 範例：建立 DataQueueDocument
- 範例：使用 DataQueueDocument

錯誤事件

- 範例：處理錯誤事件
- 範例：定義錯誤接收器
- 範例：使用自訂的處理程式來處理錯誤事件

JDBC

- 範例：使用 JDBC 驅動程式來建立及擴大表格。
- 範例：使用 JDBC 驅動程式來查詢表格並輸出它的內容。
- 範例：建立 AS400JDBCDataSourcePane

工作

- 範例：建立 VJobList, 並以 AS400ExplorerPane 呈現清單
- 範例：在探索器窗格中呈現工作的清單

程式呼叫

- 範例：建立 ProgramCallMenuItem
- 範例：處理所有程式產生的 iSeries 訊息
- 範例：新增兩個參數
- 範例：在應用程式中使用 ProgramCallButton

記錄層次存取

- 範例：建立一個 RecordListTablePane 物件，來顯示小於或等於 某個索引的所有記錄

SpooledFileViewer

- 範例：建立一個排存檔檢視器來檢視先前在 iSeries 上建立的排存檔

系統值

- 範例：使用 AS400Explorer 窗格建立一個系統值 GUI

使用者和群組

- 範例：透過 AS400DetailsPane 建立 VUserList
- 範例：使用 AS400ListPane，建立可供選擇的使用者清單

範例：使用 VUserList

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// VUserList example.This program presents a list of users on
// a system in a list pane, and allows selection of one or more
// users.
//
// Command syntax:
//VUserListExample system
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VUserListExample
{


private static AS400ListPane listPane;


 public static void main(String[] args)
    {
// If a system is not specified, display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：VUserListExample system");
return;
        }

 try
        {
// Create an AS400 object.The system name is passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a VUserList.This represents a list of users
// displayed in the list pane.
 VUserList
userList = new VUserList (system);

// Create a frame.
JFrame f = new JFrame ("VUserList 範例");

// Create an error dialog adapter.This displays
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);
```

```
// Create a list pane to display the user list.
// Use load to get the information from the server.
listPane = new AS400ListPane (userList);
listPane.addErrorListener (errorHandler);

listPane.load ();

// When the frame closes, report the selected
// users and exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
reportSelectedUsers ();
 System.exit(0);
                }
            });

// Layout the frame with the list pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", listPane);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
 System.exit(0);
        }
    }


private static void reportSelectedUsers ()
    {
VObject[] selectedUsers = listPane.getSelectedObjects ();

if (selectedUsers.length == 0)
System.out.println ("未選取使用者。");
else
        {
System.out.println ("選定的使用者為：");
for (int i = 0; i < selectedUsers.length; ++i)
System.out.println (selectedUsers[i]);
        }
    }


}
```

範例：使用 VMessageList

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VMessageList example.This program presents a details
// view of messages returned from a command call.
//
// Command syntax:
//VMessageListExample system
//
// This source is an example of IBM Toolbox for Java "VMessageList".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VMessageListExample
{


 public static void main(String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：VMessageListExample system");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a CommandCall object a run the command.
CommandCall command = new CommandCall (system);
command.run ("CRTLIB FRED");

// Create a VMessageList object with the messages
// returned from the command call.
VMessageList messageList = new VMessageList (command.getMessageList ());

// Create a frame.
JFrame f = new JFrame ("VMessageList 範例");

// Create an error dialog adapter.This will display
```

```
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

// Create a details pane to display the message list.
// Use load to load the information.
AS400DetailsPane detailsPane = new AS400DetailsPane (messageList);
detailsPane.addErrorListener (errorHandler);
detailsPane.load ();

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame with the details pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", detailsPane);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
 System.exit(0);
        }
    }


}
```

範例：使用 VIFSDirectory

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// VIFSDirectory example.This program presents a tree view of
// some directories in the integrated file system.
//
// Command syntax:
//VIFSDirectoryExample system
//
// This source is an example of IBM Toolbox for Java "VIFSDirectory".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VIFSDirectoryExample
{


 public static void main(String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：VIFSDirectoryExample system");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a VIFSDirectory object which represents the root
// of the directory tree that we are going to show.
VIFSDirectory directory = new VIFSDirectory (system, "/QIBM/ProdData");

// Create a frame.
JFrame f = new JFrame ("VIFSDirectory 範例");

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

// Create a tree pane to present the directories hierarchically.
```

```
// Load the information from the system.
AS400TreePane treePane = new AS400TreePane (directory);
treePane.addErrorListener (errorHandler);

treePane.load ();

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame with the tree pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", treePane);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤：" + e.getMessage ());
 System.exit(0);
        }
    }


}
```

範例：使用 VPrinters

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// VPrinters example.This program presents various network
// print resources with an explorer pane.
//
// Command syntax:
//VPrintersExample system
//
// This source is an example of IBM Toolbox for Java "VPrinters".
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VPrintersExample
{


 public static void main(String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：VPrintersExample system");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a VPrinters object which represents the list
// of printers attached to the system.
VPrinters printers = new VPrinters (system);


// Create a frame.
JFrame f = new JFrame ("VPrinters 範例");

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);

// Create an explorer pane to present the network print resources.
```

```
// Use load to load the information from the system.
AS400ExplorerPane explorerPane = new AS400ExplorerPane (printers);
explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame with the explorer pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", explorerPane);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
 System.exit(0);
        }
    }


}
```

範例：使用 CommandCallMenuItem

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Command call menu item example.This program demonstrates how to
// use a menu item that calls a server command.It will display
// any messages that are returned in a dialog.
//
// Command syntax:
//CommandCallMenuItemExample system
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class CommandCallMenuItemExample
{


private static JFrame f;


 public static void main(String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法： CommandCallMenuItemExample 系統");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a frame.
f = new JFrame ("指令呼叫功能表項目範例"

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);


// Create a CommandCallMenuItem object to run the command.
```

```
CommandCallMenuItem menuItem = new CommandCallMenuItem ("Clear library
FRED",
null, system, "CLRLIB FRED"
menuItem.addErrorListener (errorHandler);

// Add an action completed listener to display any
// returned messages in a dialog.
 menuItem.addActionCompletedListener (new ActionCompletedListener ()
            {
public void actionCompleted (ActionCompletedEvent event)
                {
// Get the message list from the event source.
CommandCallMenuItem item = (CommandCallMenuItem) event.getSource ();
AS400Message[] messageList = item.getMessageList ();

// Use an AS400DetailsPane to display the messages.
VMessageList vmessageList = new VMessageList (messageList);
AS400DetailsPane messageDetails = new AS400DetailsPane (vmessageList);
messageDetails.load ();

// Show the details in a dialog.
JDialog dialog = new JDialog(f);
dialog.getContentPane().setLayout(new BorderLayout());
dialog.getContentPane().add("Center"messageDetails);
dialog.pack();
dialog.setVisible(true);
                }
            });

// Create a menu with the item.
JMenu menu = new JMenu ("伺服器指令呼叫");
 menu.add (menuItem);

JMenuBar menuBar = new JMenuBar ();
menuBar.add (menu);

f.getRootPane ().setJMenuBar (menuBar);

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame with the details pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.setSize (300, 400);
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤：" + e.getMessage ());
 System.exit(0);
        }
```

```
    }

}
```

範例：使用 DataQueueDocument

註： 請閱讀 程式碼不保事項聲明 ，以取得重要法律資訊。

```
////////////////////////////////////////////////////////////////////////
//
// Data queue document example.This program demonstrates how to
// use a document that is associated with a server data queue.
//
// Command syntax:
//DataQueueDocumentExample system read|write
//
////////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class DataQueueDocumentExample
{
private static DataQueueDocumentdqDocument;
private static JTextField text;
private static booleanrw;

 public static void main(String[] args)
    {
// If a system or read|write was not specified, then display
// help text and exit.
if (args.length != 2)
        {
System.out.println("用法：DataQueueDocumentExample system read|write");
return;
        }

rw = args[1].equalsIgnoreCase ("read");
String mode = rw ? "Read" : "Write";

 try
        {
// Create two frames.
JFrame f = new JFrame ("資料佇列文件範例 - " + mode);

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);


// Create a working cursor adapter.This will adjust
// the cursor whenever a data queue is read or written.
WorkingCursorAdapter cursorAdapter = new WorkingCursorAdapter (f);

// Create an AS400 object.The system name was passed
```

```
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create the data queue path name.
QSYSObjectPathName dqName = new QSYSObjectPathName ("QGPL",
"JAVATALK", "DTAQ");

// Make sure the the data queue exists.
DataQueue dq = new DataQueue (system, dqName.getPath ());
 try
            {
dq.create (200);
            }
catch (Exception e)
            {
// Ignore exceptions.Most likely, the data queue
// already exists.
            }

// Create a DataQueueDocument object.
dqDocument = new DataQueueDocument (system, dqName.getPath ());
dqDocument.addErrorListener (errorHandler);
dqDocument.addWorkingListener (cursorAdapter);

// Create a text field used to present the document.
text = new JTextField (dqDocument, "", 40);
text.setEditable (! rw);

// When the program runs, we need a way to control when
// the reads and writes take place.We will let the
// use control this with a button.
Button button = new Button (mode);
button.addActionListener (new ActionListener ()
            {
public void actionPerformed (ActionEvent event)
                {
if (rw)
dqDocument.read ();
else {
dqDocument.write ();
text.setText ("");
                    }
                }
            });

// When the the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame.
f.getContentPane ().setLayout (new FlowLayout ());
```

```
f.getContentPane ().add (text);
f.getContentPane ().add (button);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤：" + e.getMessage ());
 System.exit(0);
        }
    }
}
```

AS400 JDBCDataSourcePane

AS400JDBCDataSourcePane 類別表示 AS400JDBCDataSource 物件的內容值。 並可選擇性地對 AS400JDBCDataSource 物件進行變更。

AS400JDBCDataSourcePane 延伸 JComponent。若要使用 AS400JDBCDataSourcePane 來顯示資料來源的內容，可在 AS400JDBCDataSourcePane 建構元上指定資料來源，或是於 使用 setDataSource() 建立 AS400JDBCDataSourcePane 之後指定資料來源。 使用 applyChanges()，可將對圖形式使用者介面 (GUI) 所做的變更套用到資料來源物件。

下列範例建立 AS400JDBCDataSourcePane 及 確定 按鈕，並將之新增到框架。當您按一下 確定 時，便可將對 GUI 的變更套用到資料來源。

範例： 使用 AS400JDBCDataSourcePane

```
// Create a data source.
myDataSource = new AS400JDBCDataSource();

// Create a window to hold the pane and an OK button.
JFrame frame = new JFrame ("JDBC Data Source Properties");

// Create a data source pane.
dataSourcePane = new AS400JDBCDataSourcePane(myDataSource);

// Create an OK button
JButton okButton = new JButton("OK");

// Add an ActionListener to the OK button.  When OK is
// pressed, applyChanges() will be called to commit any
// changes to the data source.
okButton.addActionListener(new ActionListener()
   {
      public void actionPerformed(ActionEvent ev)
      {
         // Apply all changes made on the data source pane
         // to the data source.  If all changes are applied
         // successfully, get the data source from the pane.
         if (dataSourcePane.applyChanges())
         {
            System.out.println("ok pressed");
            myDataSource = dataSourcePane.getDataSource();
            System.out.println(myDataSource.getServerName());
         }
      }
   }
);

// Setup the frame to show the pane and OK button.
frame.getContentPane ().setLayout (new BorderLayout ());
frame.getContentPane ().add ("Center", dataSourcePane);
frame.getContentPane ().add ("South", okButton);
```

```
        // Pack the frame.
frame.pack();

        //Display the pane and OK button.
frame.show();
```

範例：使用 VJobList 顯示工作清單

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Job list example.This program presents a list of jobs in an
// explorer pane.
//
// Command syntax:
//VJobListExample system
//
// This source is an example of IBM Toolbox for Java "AS400ExplorerPane".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VJobListExample
{


 public static void main(String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：VJobListExample system");
return;
        }

 try
        {
// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create a VJobList object which represents the list
// of jobs named QZDASOINIT.
VJobList jobList = new VJobList (system);
jobList.setName ("QZDASOINIT");

// Create a frame.
JFrame f = new JFrame ("工作清單範例");

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);
```

```
// Create an explorer pane to present the job list.
// Use load to load the information from the system.
AS400ExplorerPane explorerPane = new AS400ExplorerPane (jobList);
explorerPane.addErrorListener (errorHandler);

explorerPane.load ();

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// Layout the frame with the explorer pane.
f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", explorerPane);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
 System.exit(0);
        }
    }


}
```

範例：使用按鈕呼叫伺服器上的程式

註： 請閱讀 程式碼不保事項聲明 中的重要法律資訊。

```
///////////////////////////////////////////////////////////////////////
//
// Program call button example.This program demonstrates how to
// use a button that calls a program on the server.It will exchange data
// with the server program via an input and output parameter.
//
// Command syntax:
//ProgramCallButtonExample system
//
// This source is an example of IBM Toolbox for Java "ProgramCallButton".
//
///////////////////////////////////////////////////////////////////////

import com.ibm.as400.access.*;
import com.ibm.as400.vaccess.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ProgramCallButtonExample
{

private ProgramParameter parm1, parm2, parm3, parm4, parm5;
private JTextField cpuField;
private JTextField dasdField;
private JTextField jobsField;


// Create a ProgramCallButtonExample object, then call the
// non-static version of main().If we don't to this then
// the class variables (parm1, parm2, ...) must be declared
// static.If they are static they cannot be used by the
// action completed listener in Java 1.1.7 or 1.1.8.
 public static void main(String[] args)
    {
 ProgramCallButtonExample me = new ProgramCallButtonExample();
 me.Main(args);
    }

public void Main (String[] args)
    {
// If a system was not specified, then display help text and
// exit.
if (args.length != 1)
        {
System.out.println("用法：ProgramCallButtonExample system");
return;
        }

 try
```

```
        {
// Create a frame.
JFrame f = new JFrame ("程式呼叫按鈕範例");

// Create an error dialog adapter.This will display
// any errors to the user.
ErrorDialogAdapter errorHandler = new ErrorDialogAdapter (f);


// Create an AS400 object.The system name was passed
// as the first command line argument.
AS400 system = new AS400 (args[0]);

// Create the program path name.
QSYSObjectPathName programName = new QSYSObjectPathName ("QSYS",
"QWCRSSTS", "PGM");

// Create a ProgramCallButton object.The button
// will have the text "Refresh" and no icon.
ProgramCallButton button = new ProgramCallButton ("重新整理", null);
button.setSystem (system);
button.setProgram (programName.getPath ());
button.addErrorListener (errorHandler);

// The first parameter is an 64 byte output parameter.
parm1 = new ProgramParameter (64);
button.addParameter (parm1);

// We use the second parameter to set the buffer size
// of the first parameter.We will always set this to
// 64.Remember that we need to convert the Java int
// value 64 to the format used on the server.
 AS400Bin4 parm2Converter = new AS400Bin4 ();
byte[] parm2Bytes = parm2Converter.toBytes (64);
parm2 = new ProgramParameter (parm2Bytes);
button.addParameter (parm2);

// The third parameter is the status format.We will
// always use "SSTS0200".This is a String value, and
// again we need to convert it to the format used on the server.
AS400Text parm3Converter = new AS400Text (8, system);
byte[] parm3Bytes = parm3Converter.toBytes ("SSTS0200");
parm3 = new ProgramParameter (parm3Bytes);
button.addParameter (parm3);

// The fourth parameter is the reset statistics parameter.
// We will always pass "*NO" as a 10 character String.
AS400Text parm4Converter = new AS400Text (10, system);
byte[] parm4Bytes = parm4Converter.toBytes ("*NO ");
parm4 = new ProgramParameter (parm4Bytes);
button.addParameter (parm4);

// The fifth parameter is for error information.It
// is an input/output parameter.We will not use it
// for this example, but we need to set it to something,
// or else the number of parameters will not match
```

```
// what the server is expecting.
byte[] parm5Bytes = new byte[32];
parm5 = new ProgramParameter (parm5Bytes, 0);
button.addParameter (parm5);

// When the program runs, we will get a bunch of data.
// We need a way to display that data to the user.
// In this case, we will just use simple labels and text
// fields.
JLabel cpuLabel = new JLabel ("CPU 使用率： ");
cpuField = new JTextField (10);
cpuField.setEditable (false);

JLabel dasdLabel = new JLabel ("DASD 使用率： ");
dasdField = new JTextField (10);
dasdField.setEditable (false);

JLabel jobsLabel = new JLabel ("作用中工作數：");
jobsField = new JTextField (10);
jobsField.setEditable (false);

// When the frame closes, exit.
f.addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent event)
                {
 System.exit(0);
                }
            });

// When the program is called, we need to process the
// information that comes back in the first parameter.
// The format of the data in this parameter was documented
// by the program we are calling.

button.addActionCompletedListener (new ActionCompletedListener ()
            {
public void actionCompleted (ActionCompletedEvent event)
                {
 try
                    {
 // Get the data from the first parameter.
 // It is in the server format.
 byte[] parm1Bytes = parm1.getOutputData ();

 // Each of the pieces of data that we need
 // is an int.We can create one converter
 // to do all of our conversions.
 AS400Bin4 parm1Converter = new AS400Bin4 ();

 // Get the CPU utilitization starting at byte 32.
 // Set this value in the corresponding text field.
 int cpu = parm1Converter.toInt (parm1Bytes, 32);
 cpuField.setText (Integer.toString (cpu / 10) + "%");

 // Get the DASD utilitization starting at byte 52.
 // Set this value in the corresponding text field.
```

```
 int dasd = parm1Converter.toInt (parm1Bytes, 52);
 dasdField.setText (Integer.toString (dasd / 10000) + "%");

 // Get the number of active jobs starting at byte 36.
 // Set this value in the corresponding text field.
 int jobs = parm1Converter.toInt (parm1Bytes, 36);
 jobsField.setText (Integer.toString (jobs));
                    }
catch (Exception e) { e.printStackTrace(); }
                 }
            });

// Layout the frame.
JPanel outputPanel = new JPanel ();
outputPanel.setLayout (new GridLayout (3, 2, 5, 5));
outputPanel.add (cpuLabel);
outputPanel.add (cpuField);
outputPanel.add (dasdLabel);
outputPanel.add (dasdField);
outputPanel.add (jobsLabel);
outputPanel.add (jobsField);

Panel buttonPanel = new Panel ();
buttonPanel.add (button);

f.getContentPane ().setLayout (new BorderLayout ());
f.getContentPane ().add ("中央", outputPanel);
f.getContentPane ().add ("南", buttonPanel);
f.pack ();
f.show ();
        }
catch (Exception e)
        {
 System.out.println ("錯誤： " + e.getMessage ());
 System.exit(0);
        }
    }


}
```

程式設計秘訣

本節舉出各種秘訣，協助您使用 IBM Toolbox for Java：

關閉 Java 程式
瞭解如何適當地關閉您的 Java 程式。

使用整合檔案系統路徑名稱
閱讀如何在您的程式中使用整合檔案系統路徑名稱。本節內容涵蓋整合檔案系統路徑名稱、參數及特殊值。

管理連線
瞭解如何使用 AS400 類別來啟動與結束 socket 連接，包括如何符合 Enterprise JavaBean 規格。

使用 OS/400 Java 虛擬機器 (JVM)
瞭解如何在 AS/400 Java 虛擬機器中執行 IBM Toolbox for Java 的類別。
本節內容涵蓋如何以最佳方式存取伺服器資源、如何執行類別以及要考慮的登入因素。

»連接獨立的輔助儲存體儲存區 (IASP)
讀取有關連接 IASP 的相關資訊。IASP 是獨立於系統中的其餘儲存體，您可以連線也可以離線的硬碟機集合。«

處理使用 Toolbox for Java access 類別時的錯誤
瞭解如何使用 Toolbox for Java exception 類別來處理您在程式中使用 Toolbox for Java 的 access 類別時發生的錯誤。

處理使用 Toolbox for Java vaccess 類別時的錯誤
瞭解如何使用 Toolbox for Java 錯誤事件類別來處理您在程式中使用 Toolbox for Java 的 vaccess 類別時發生的錯誤。

使用 Trace 類別
瞭解如何使用 Trace 類別來記錄追蹤點和診斷訊息，協助您重新產生以及診斷程式中的問題。

最佳化您的程式
瞭解如何使您的程式最佳化，取得更佳的效能。

使用 OS/400 JVM 以得到更佳的效能
讀取從 OS/400 JVM 使得效能增進的相關資訊。

在您的從屬站管理 Toolbox for Java 類別
瞭解如何在您的從屬站使用 AS400ToolboxInstaller 類別來管理 IBM Toolbox for Java 類別。

增進 JAR 檔案的效能
瞭解如何使用 Toolbox for Java JarMaker 類別來建立更小、更快的載入 IBM Toolbox for Java JAR 檔案。

使用 Java 國家語言支援 (NLS)

閱讀關於使用 IBM Toolbox for Java 與 Java 國家語言支援的資訊。

取得服務與支援

請使用這些資源來尋找 Toolbox for Java 的支援服務。

關閉 Java 程式

若要確定您的程式是否已正確關閉，請在結束 Java 程式之前，發出 System.exit(0) 作為最後一個指示。

註：請避免在 Servlets 中使用 System.exit(0)， 因為這樣的話，會關閉整個 Java 虛擬機器 (Java VM)。

IBM Toolbox for Java 會透過使用者緒連接到伺服器。因為這個緣故，當無法發出 System.exit(0) 時，可能會使得
Java 程式無法正確關閉。

使用 System.exit(0) 並非必要，而是預防措施。您會有必須使用這個指令來結束 Java 程式的時候，而在不必要時
System.exit(0) 也不會有什麼問題。

伺服器物件的整合檔案系統路徑名稱

您的 Java 程式必須使用整合檔案系統名稱來參照伺服器物件，
如程式、檔案庫、指令或排存檔。整合檔案系統名稱即是在 iSeries 或 AS/400e
伺服器上的整合檔案系統的檔案庫 檔案系統中存取的伺服器物件的名稱。

路徑名稱可以由下列元件所組成：

路徑名稱元件	說明
檔案庫	檔案庫常駐的檔案庫。檔案庫是整合檔案系統路徑名稱的 必要部份。檔案庫名稱必須是 10 個或更少的字元，且後面必須接著.lib
物件	整合檔案系統路徑名稱代表的物件的名稱。物件是整合檔案系統路徑名稱的必要部份。 物件名稱必須是 10 個或更少的字元，且其後須接著.type，其中type 是物件的類型。您可透過控制語言 (CL) 指令 (如「使用物件 (WRKOBJ)」) 上 的 OBJTYPE 參數的提示，來找到想要的類型。
類型	物件的類型。當指定物件時，亦必須指定 物件的類型。(請參閱上面的 物件。) 類型名稱必須是 6 個或更少的字元。
成員	此整合檔案系統路徑名稱代表的成員的名稱。成員是整合檔案系統路徑名稱的選擇部份。只有物件 類型 是 FILE 時，才能指定它。成員名稱 必須是 10 個或更少的字元，且其後須接著.mbr。

當決定及指定整合檔案系統名稱時，請遵循這些條件：

- 正斜線 (/) 為路徑分隔字元。
- 名為 QSYS.LIB 的根目錄含有伺服器檔案庫結構。
- 常駐在伺服器檔案庫 QSYS 中的物件具有下列格式：

/QSYS.LIB/object.type

- 常駐在其它檔案庫中的物件具有下列格式：

/QSYS.LIB/library.LIB/object.type

- 物件類型副檔名即是針對該類型的物件所使用的伺服器縮寫。

若要察看這些類型的清單，請輸入一個具有物件類型作為參數的 CL 指令， 然後按 F4 (提示)
取得類型的提示。例如， 「使用物件 (WRKOBJ)」指令即具有物件類型參數。

下表是一些常用物件類型的清單及每一種類型的縮寫：

物件類型	縮寫
指令	.CMD
資料佇列	.DTAQ
檔案	.FILE
字型資源	.FNTRSC

格式定義	.FORMDF
檔案庫	.LIB
成員	.MBR
套印格式	.OVL
頁定義	.PAGDFN
頁區段	.PAGSET
程式	.PGM
輸出佇列	.OUTQ
排存檔	.SPLF

您可以使用下列說明，決定如何指定整合檔案系統路徑名稱：

整合檔案系統名稱	說明
/QSYS.LIB/MY_LIB.LIB/MY_PROG.PGM	程式 MY_PROG 是在伺服器上的 檔案庫 MY_LIB 中
/QSYS.LIB/MY_LIB.LIB/MY_QUEUE.DTAQ	資料佇列 MY_QUEUE 是在伺服器上的 檔案庫 MY_LIB 中
/QSYS.LIB/YEAR1998.LIB/MONTH.FILE/JULY.MBR	成員 JULY 是在伺服器上 檔案庫 YEAR1998 內 的檔案 MONTH 中

整合檔案系統特殊值

有多種 IBM Toolbox for Java 類別會識別整合檔案系統路徑名稱中的特殊值。 這些特殊值（例如使用在 iSeries 指令行上）的傳統格式會以星號(*ALL) 開頭。 但是，在使用 Toolbox for Java 類別的 Java 程式中，這些特殊值的格式是以百分比 符號(%ALL%) 開頭及結束。

註： 在整合檔案系統中，星號代表萬用字元。

下表顯示 IBM Toolbox for Java 類別可以識別特定路徑名稱元件的哪些特殊值。
表格中也告訴您這些特殊值的傳統格式與 Toolbox for Java 類別中使用的格式有何不同。

路徑名稱元件	傳統格式	Toolbox for Java 格式
檔案庫名稱	*ALL	%ALL%
	*ALLUSR	%ALLUSR%
	*CURLIB	%CURLIB%
	*LIBL	%LIBL%
	*USRLIBL	%USRLIBL%
物件名稱	*ALL	%ALL%

成員名稱	*ALL	%ALL%
	*FILE	%FILE%
	*FIRST	%FIRST%
	*LAST	%LAST%

請參閱 QSYSObjectPathName 類別，取得關於建置與剖析整合檔案系統名稱的資訊。

有關整合檔案系統概念的詳細資訊，請參閱整合檔案系統概念。

管理連線

能建立、啟動及結束與伺服器的連接是很重要的。以下討論說明管理與伺服器連接的主要概念，並提供一些程式碼範例。

若要與 iSeries 系統連接，您的 Java 程式必須建立 AS400 個物件。AS400 物件對每一種 iSeries 伺服器類型，最多含有一個 socket 連接。 一個服務程式會對應一個伺服器上的工作，且為伺服器中資料的介面。

註：建立 Enterprise JavaBeans (EJB) 時，請遵守連接期間 不允許使用執行緒的 EJB 規格。雖然關閉 IBM Toolbox for Jav 執行緒支援可能會使 應用程式的執行速度降低，但仍應遵守 EJB 規格。

在 iSeries 上，每一個與各伺服器的連接均有自己的工作。 分別使用不同的伺服器來支援下列每一項：

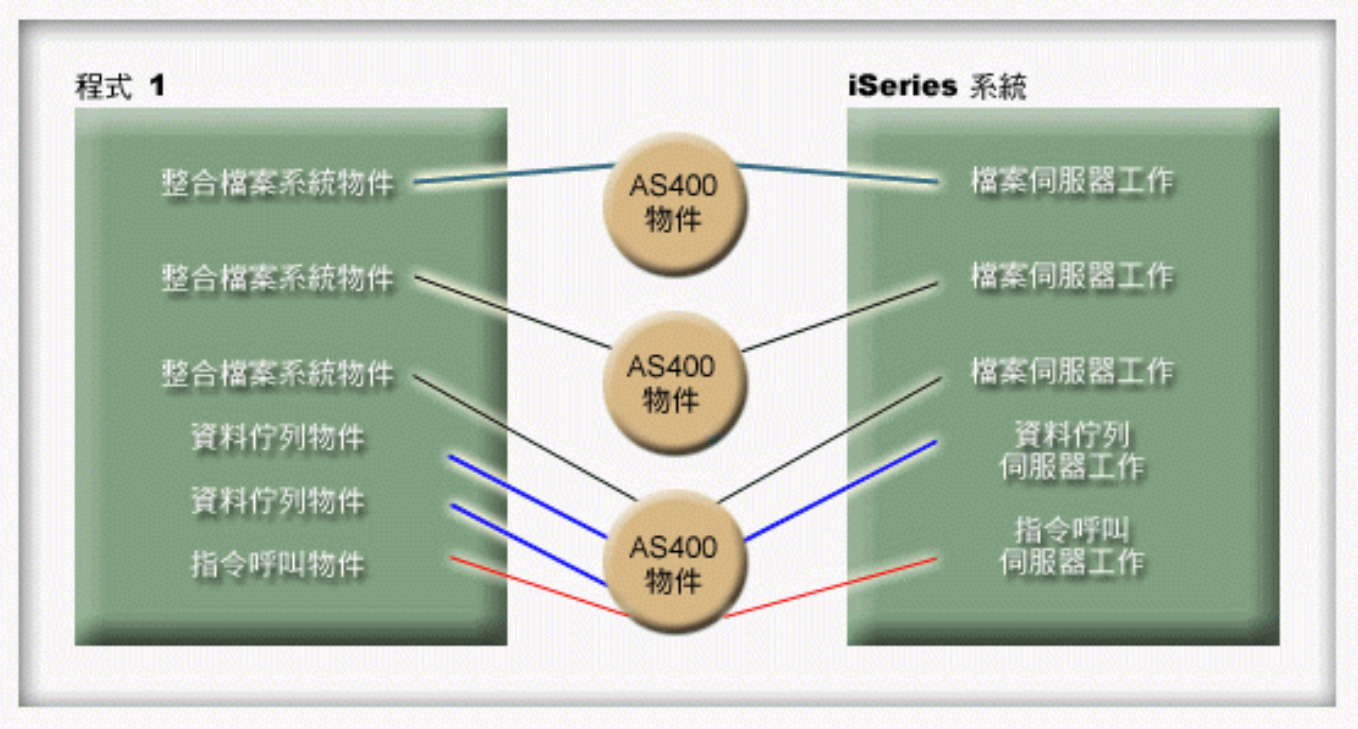
- JDBC
- 程式呼叫與指令呼叫
- 整合檔案系統
- 列印
- 資料佇列
- 記錄層次存取

備註：

- 如果應用程式不嘗試同時執行需要網路列印伺服器的兩個工作，則 PrintJob 類別會對每一個 AS400 物件使用 一個 socket 連接。
- 必要時，列印類別將會對網路列印伺服器 建立額外的 socket 連接。如果 5 分鐘內未使用額外交談，將切斷它們。

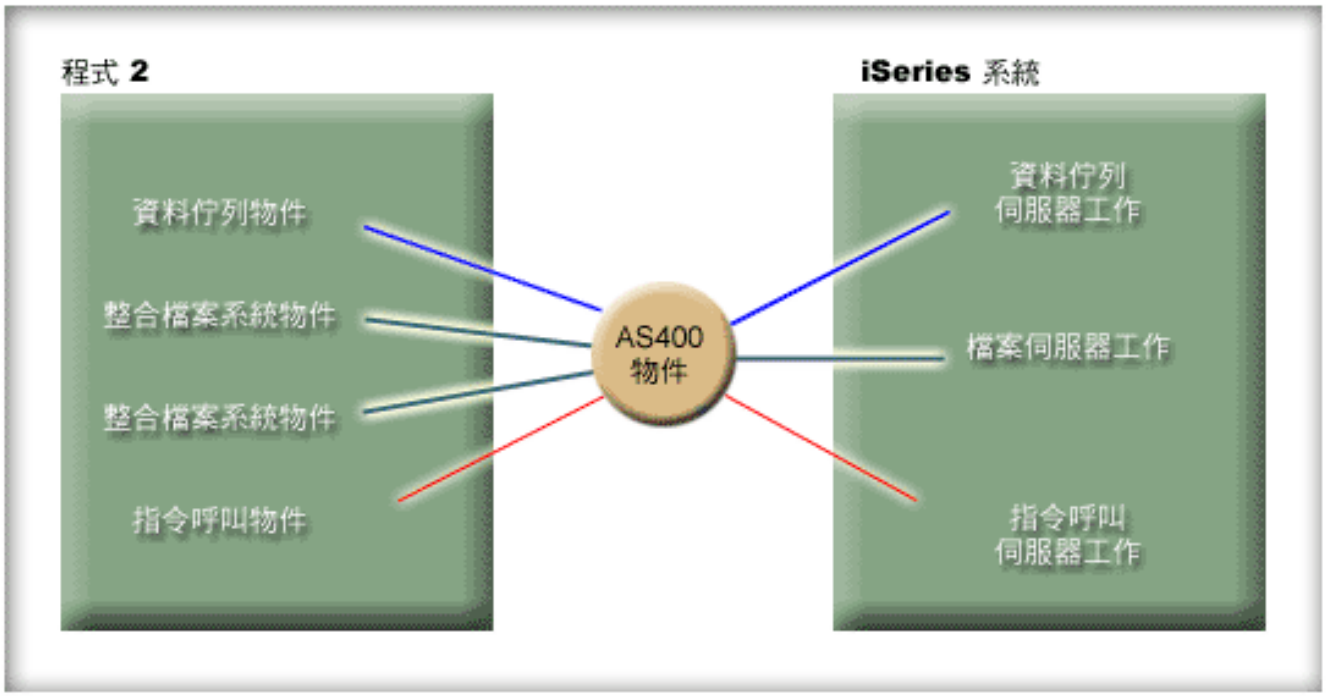
Java 程式可控制與 iSeries 的連接數。為了達到最佳的通信效能，一個 Java 程式可對 同一個系統建立多個 AS400 物件，如 中所示。這將建立 iSeries 的多個 socket 連接。

圖 1：對同一個 iSeries 系統建立多個 AS400 物件 與 socket 連接的 Java 程式



若要保留 iSeries 系統資源，請只建立一個 AS400 物件，如圖 2 所示。 此方法會減少連接數，因而減少系統中資源的使用量

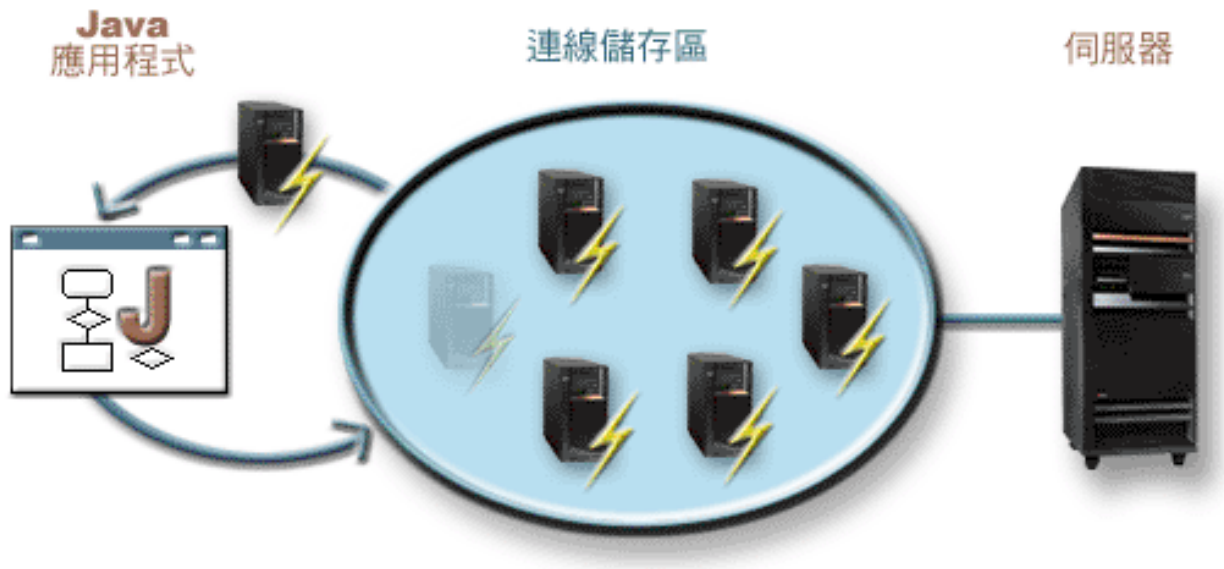
圖 2：對同一個 iSeries 系統建立單一 AS400 物件與 socket 連接的 Java 程式



註： 雖然建立多個連線會增加系統中資源的使用量，但建立多個連線是有好處的。 具有多個連線可讓您的 Java 程式以平行方式處理工作，因而使產能（每秒交易數）增加 並加速應用程式的執行。

您也可選擇使用連線儲存區來管理連線，如圖 3 所示。 此方法會重覆使用先前建立給使用者的連線，因而減少了與 iSeries 連接所要花費的時間。

圖 3：由 AS400ConnectionPool取得對 iSeries 伺服器連線的 Java 程式



下列範例說明如何建立及使用 AS400 物件：

範例 1： 在下列範例中， 將建立兩個 CommandCall 物件，傳送指令給同一個 iSeries 系統。 因為 CommandCall 物件使用同一個 AS400 物件，所以僅會建立一個與系統的連線。

```
                    // Create an AS400 object.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create two command call objects that use
                    // the same AS400 object.
    CommandCall cmd1 = new CommandCall(sys,"myCommand1");
    CommandCall cmd2 = new CommandCall(sys,"myCommand2");

                    // Run the commands.  A connection is made when the
                    // first command is run.  Since they use the same
                    // AS400 object the second command object will use
                    // the connection established by the first command.
    cmd1.run();
    cmd2.run();
```

範例 2： 在下列範例中， 將建立兩個 CommandCall 物件，傳送指令給同一個 iSeries 系統。 因為 CommandCall 物件使用不同的 AS400 物件，所以會建立兩個與系統的連線。

```
                    // Create two AS400 objects to the same server.
    AS400 sys1 = new AS400("mySystem.myCompany.com");
    AS400 sys2 = new AS400("mySystem.myCompany.com");

                    // Create two command call objects.  They use
                    // different AS400 objects.
    CommandCall cmd1 = new CommandCall(sys1,"myCommand1");
    CommandCall cmd2 = new CommandCall(sys2,"myCommand2");

                    // Run the commands.  A connection is made when the
                    // first command is run.  Since the second command
                    // object uses a different AS400 object, a second
                    // connection is made when the second command is run.
    cmd1.run();
    cmd2.run();
```

範例 3： 在下列範例中，將使用同一個 AS400 物件， 建立 CommandCall 物件與 IFSFileInputStream 物件。 因為在 iSeries 系統中，CommandCall 物件與 IFSFileInput Stream 物件使用不同的服務程式， 所以將建立兩個連線。

```
                    // Create an AS400 object.
    AS400 newConn1 = new AS400("mySystem.myCompany.com");

                    // Create a command call object.
    CommandCall cmd = new CommandCall(newConn1,"myCommand1");

                    // Create the file object.  Creating it causes the
                    // AS400 object to connect to the file service.
    IFSFileInputStream file = new IFSFileInputStream(newConn1,"/myfile");

                    // Run the command.  A connection is made to the
                    // command service when the command is run.
    cmd.run();
```

範例 4： 在下列範例中， 將使用一個 AS400ConnectionPool 來取得一個 iSeries 連線。 本範例 (如同以範例 3) 不會指定服務程式， 因此會於執行指令時進行與指令服務程式的連線。

```
                    // Create an AS400ConnectionPool.
     AS400ConnectionPool testPool1 = new AS400ConnectionPool();
                    // Create a connection.
     AS400 newConn1 = testPool1.getConnection("myAS400", "myUserID",
"myPassword");
                    // Create a command call object that uses the AS400
object.
     CommandCall cmd = new CommandCall(newConn1,"myCommand1");
                    // Run the command.  A connection is made to the
                    // command service when the command is run.
     cmd.run();
                    // Return connection to pool.
     testPool1.returnConnectionToPool(newConn1);
```

範例 5：下列範例於要求從儲存區進行 連線時，使用 AS400ConnectionPool 與特定服務程式連接。如此會在執行指令時，減少與服務程式連接所需的時間（請參閱以上的範例 4）。
如果連線傳回儲存區，則下一個取得連線的呼叫會傳回相同的連線物件。
這表示毋需額外的連接時間，不論是在建立時或使用時都一樣。

```
                    // Create an AS400ConnectionPool.
     AS400ConnectionPool testPool1 = new AS400ConnectionPool();
                    // Create a connection to the AS400.COMMAND service.
(Use the service number constants
                    // defined in the AS400 class (FILE, PRINT, COMMAND,
DATAQUEUE, etc.))
     AS400 newConn1 = testPool1.getConnection("myAS400", "myUserID",
"myPassword", AS400.COMMAND);
                    // Create a command call object that uses the AS400
object.
     CommandCall cmd = new CommandCall(newConn1,"myCommand1");
                    // Run the command.  A connection has already been
made
                    // to the command service.
     cmd.run();
                    // Return connection to pool.
     testPool1.returnConnectionToPool(newConn1);
                    // Get another connection to command service.  In
this case, it will return the same
                    // connection as above, meaning no extra connection
time will be needed either now or when the
                    // command service is used.
     AS400 newConn2 = testPool1.getConnection("myAS400", "myUserID",
"myPassword", AS400.COMMAND);
```

啟動與結束連接

Java 程式可控制連接的啟動與結束時間。根據預設值，當伺服器需要資訊時，即會啟動一個連線。 您可藉由在 AS400 物件中呼叫connectService()方法 ，預先連接至伺服器，來控制進行連線的正確時機。

您可使用一個AS400ConnectionPool， 在不呼叫 connectService()
方法的情況下，建立與服務程式先前連接的連線，如以範例 5 所述。

下列範例說明與 iSeries 連接及中斷與 iSeries 連接的 Java 程式。

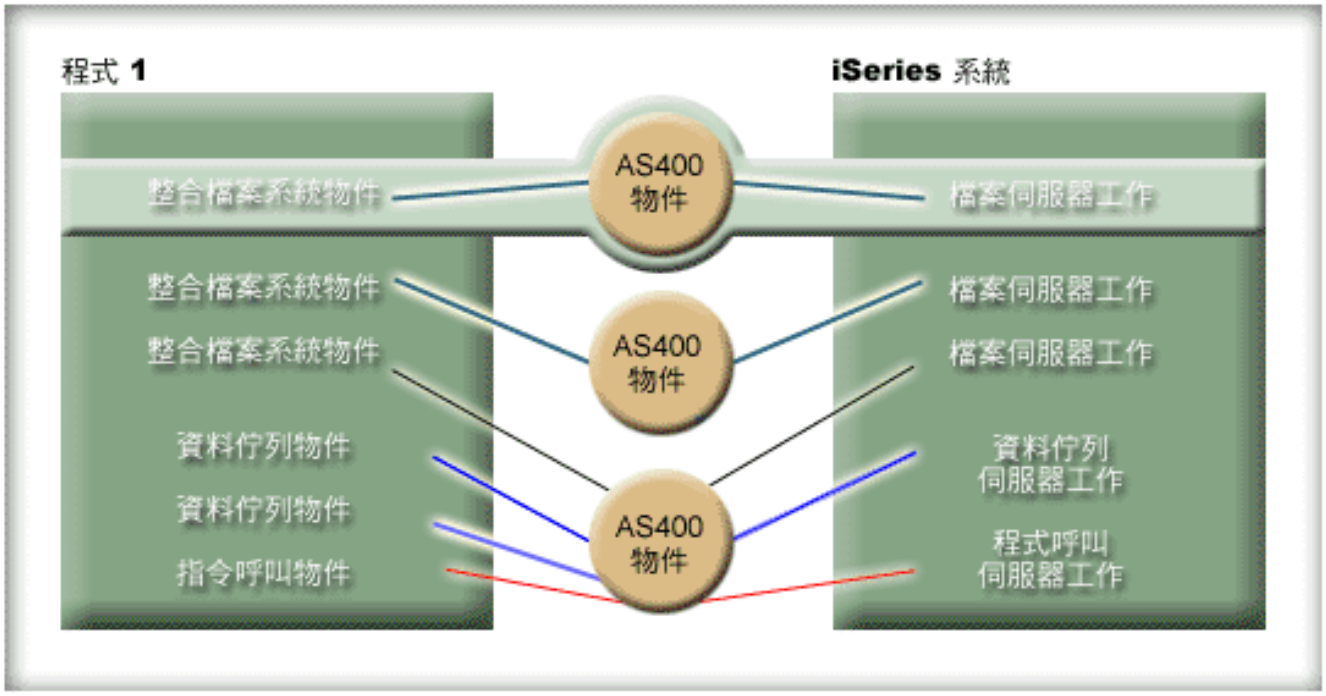
範例 1：本範例說明如何預先連接至 iSeries：

```
                    // Create an AS400 object.
AS400 system1 = new AS400("mySystem.myCompany.com");

                    // Connect to the command service.  Do it now
                    // instead of when data is first sent to the
                    // command service.  This is optional since the
                    // AS400 object will connect when necessary.
system1.connectService(AS400.COMMAND);
```

範例 2：一旦啟動連接，不管是隱含地透過 AS400 物件或明確地透過 Java 程式，Java 程式將負責切斷連接。Java 程式會藉著在 AS400 物件中呼叫 disconnectService() 方法來中斷連接。若要增進效能，僅在透過服務來完成程式執行時，Java 程式才應切斷連接。 如果在 Java 程式使用服務完成執行之前已斷線，則當需要從服務取得資料時，AS400 物件將重新連線 (如果可能重新連線的話)。

圖 4 說明切斷第一個整合檔案系統物件連接會如何僅結束 AS400 物件連線的單一案例，而非所有的整合檔案系統物件連線。

圖 4：對 AS400 物件的案例使用自己的服務程式之單一物件將被斷線

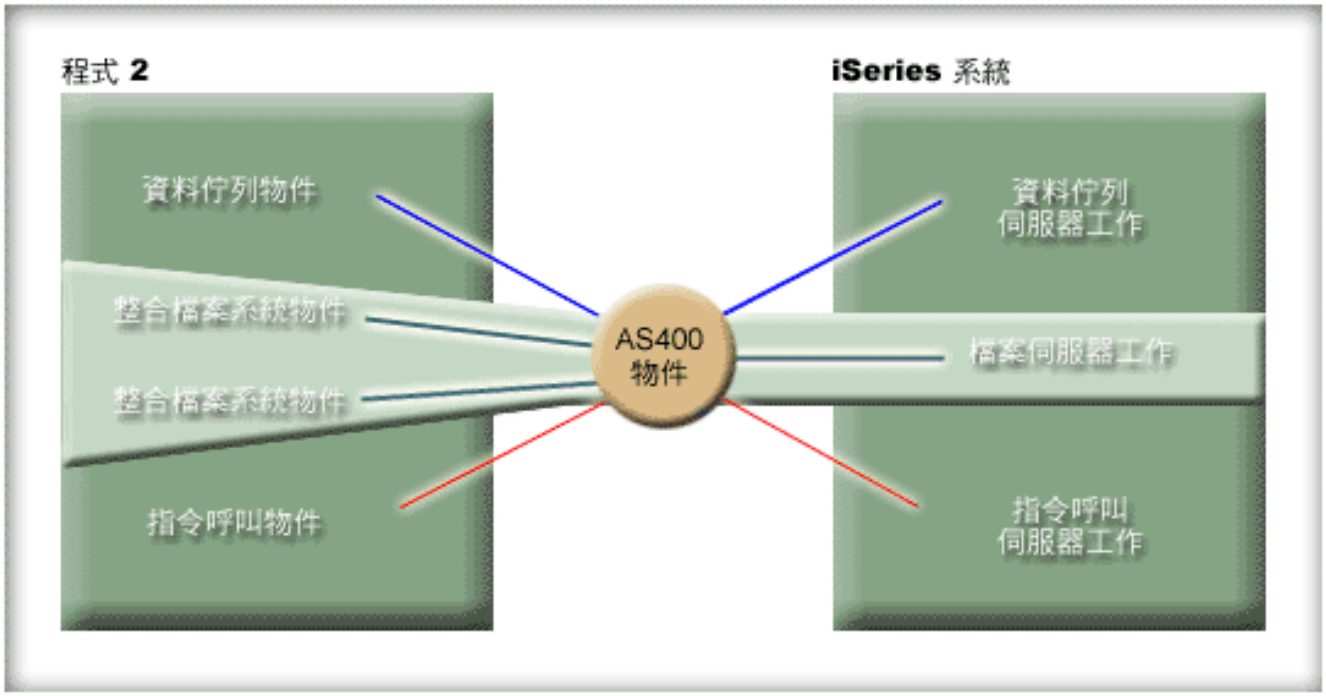


本例說明 Java 程式如何切斷連接：

```
                    // Create an AS400 object.
AS400 system1 = new AS400("mySystem.myCompany.com");

                    // ... use command call to send several commands
                    // to the server.  Since connectService() was not
                    // called, the AS400 object automatically
                    // connects when the first command is run.

                    // All done sending commands so disconnect the
                    // connection.
system1.disconnectService(AS400.COMMAND);
```

範例 3： 使用同一個服務與共用同一個 AS400 物件的多個物件將共用一個連接。 切斷連線將結束對 AS400 物件之各案例使用同一個服務程式之所有物件的連線，如圖 5 中所示。

圖 5：對 AS400 物件的案例使用同一個服務程式之所有物件將被切斷連線



例如，有兩個 CommandCall 物件使用同一個 AS400 物件。 當呼叫 disconnectService() 時， 將同時結束這兩個 CommandCal 物件的連接。當呼叫第二個 CommandCall 物件的 run() 方法時，AS400 物件必須重新連接至服務程式：

```
                    // Create an AS400 object.
AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create two command call objects.
CommandCall cmd1 = new CommandCall(sys,"myCommand1");
CommandCall cmd2 = new CommandCall(sys,"myCommand2");

                    // Run the first command
cmd1.run();

                    // Disconnect from the command service.
sys.disconnectService(AS400.COMMAND);

                    // Run the second command.  The AS400 object
                    // must reconnect to the server.
cmd2.run();

                    // Disconnect from the command service.  This
                    // is the correct place to disconnect.
sys.disconnectService(AS400.COMMAND);
```

範例 4： 並非所有的 IBM Toolbox for Java 類別均會自動重新連接整合檔案系統類別中，有些方法呼叫不會重新連線，因為檔案可能已有所變更。 當檔案被切斷連接時，某些其它處理可能已刪除了檔案，或已變更了它的內容。在下列範例中，有兩個檔案物件使用同一個 AS400 物件。 當呼叫 disconnectService() 時，將同時結束這兩個檔案物件的連接。 第二個 IFSFileInputStream 物件的 read()

將失敗，因為它不再具有與伺服器的連線。

```
                    // Create an AS400 object.
    AS400 sys = new AS400("mySystem.myCompany.com");

                    // Create two file objects.  A connection to the
                    // server is created when the first object is
                    // created.  The second object uses the connection
                    // created by the first object.
    IFSFileInputStream file1 = new IFSFileInputStream(sys,"/file1");
    IFSFileInputStream file2 = new IFSFileInputStream(sys,"/file2");

                    // Read from the first file, then close it.
    int i1 = file1.read();
    file1.close();

                    // Disconnect from the file service.
    sys.disconnectService(AS400.FILE);

                    // Attempt to read from the second file.  This
                    // fails because the connection to the file service
                    // no longer exists.  The program must either
                    // disconnect later or have the second file use a
                    // different AS400 object (which causes it to
                    // have its own connection).
    int i2 = file2.read();

                    // Close the second file.
    file2.close();

                    // Disconnect from the file service.  This
                    // is the correct place to disconnect.
    sys.disconnectService(AS400.FILE);
```

OS/400 Java 虛擬機器 (Java VM)

IBM Toolbox for Java 類別於 IBM Developer Kit for Java (OS/400) Java 虛擬機器 (Java VM) 上執行。事實上，此類別可於所有支援 Java Development Kit (JDK) 1.1.x 及 Java 2 Software Development Kit (J2SDK) 規格 的平台上執行。

當您在 OS/400 JVM 上執行 IBM Toolbox for Java 類別時，請執行下列動作：

- 選擇在 OS/400 JVM 中執行時，是否要 使用OS/400 JVM or the IBM Toolbox for Java 類別來存取 iSeries 伺服器資源。
- 移出 OS/400 JVM 上的執行 IBM Toolbox for Java 類別
- 讀取 OS/400 JVM 中有關設定系統名稱、使用者 ID 及密碼的資訊。

iSeries 伺服器支援不同 Java 平台的其它相關資訊， 請參閱Developer Kit for Java中的Support for multiple JDKs。

比較 OS/400 Java 虛擬機器 (Java VM) 與 IBM Toolbox for Java 類別

當 Java 程式在 IBM Developer Kit for Java (OS/400) Java 虛擬機器 (Java VM)中執行時, 您至少可使用兩種方法來存取 iSeries 伺服器資源。您可以使用下列任一個介面：

- 內建在 Java 中的機能
- 一個 IBM Toolbox for Java 類別

當決定將使用哪一個介面時, 請考慮下列因素：

- 位置 - 在決定要使用哪一個介面時, 程式執行之處是最重要的因素。程式將：
  - 僅在從屬站上執行嗎？
  - 僅在伺服器上執行嗎？
  - 同時執行於從屬站與伺服器上, 但在這兩個情形下, 資源是 iSeries 伺服器資源嗎？
  - 在某一台 OS/400 JVM 中執行, 並存取另一台 iSeries 伺服器的資源嗎？
  - 在不同種類的伺服器上執行嗎？

  如果程式同時執行於從屬站與伺服器 (包括作為第二台 iSeries 伺服器的從屬站之 iSeries 伺服器) 上, 並僅存取 iSeries 伺服器資源, 則使用 IBM Toolbox for Java 介面可能是最好的方式。

  如果程式必須存取多種類型伺服器中的資料, 使用 Java 原始介面 (JNI) 可能是最好的方式。

- 一致性 / 可移轉性 - 在 iSeries 伺服器上執行 IBM Toolbox for Java 類別的 能力, 意謂著可同時對從屬站程式與伺服器程式使用相同的介面。 對從屬站程式與伺服器程式, 您僅要學習一個介面時, 您將更有生產力。

  不過, 將資料寫入 IBM Toolbox for Java 介面將使您的程式較不具伺服器可攜性。

  如果程式必須執行於 iSeries 伺服器及其它伺服器上, 您可能會發現使用已在 Java 中建立的機能可能更

- 複雜性 - IBM Toolbox for Java 介面的建立是為了方便 存取 iSeries 伺服器資源。通常, 使用 IBM Toolbox for Java 介面的唯一另一個選擇方案為： 撰寫一個程式, 透過「Java 原始介面 (JNI)」存取資源以與該程式通信。

  您必須決定下列何者更重要：使 Java 更中立, 並撰寫一個程式來存取資源, 或是使用較不具可攜性的 IBM Toolbox for Java 介面。

- 功能 - IBM Toolbox for Java 介面通常比 Java 介面提供更多的功能。 例如, IBM Toolbox for Java 授權程式的 IFSFileOutputStream 類別的函數 比 java.io 的 FileOutputStream 類別為多。不過, 使用 IFSFileOutputStream 會使您的程式更適用 於 iSeries 伺服器。因為使用 IBM Toolbox for Java 類別, 會使您失去伺服器可移轉性。

  您必須決定可攜性較重要, 還是您想要利用其它的函數。

- 資源 - 執行於 OS/400 JVM 中時, 許多 IBM Toolbox for Java 類別 仍會透過主電腦伺服器提出要求。因此, 第二個工作 (伺服器工作) 將實施存取資源的要求。

此要求所使用的資源可能會比在 Java 程式的工作下執行之 Java 原始介面 (JNI) 所使用的資源要多。

- iSeries 伺服器作為從屬站- 如果程式執行於某一台 iSeries 伺服器上, 並存取第二台 iSeries 伺服器中的資料, 您最好的選擇可能就是使用 IBM Toolbox for Java 類別。這些類別會使您輕易地存取第二台 iSeries 伺服器中的資源。

  這種存取的範例為「資料佇列」存取。IBM Toolbox for Java 授權程式的「資料佇列」介面能使您輕易地存取資料佇列資源。

  使用 IBM Toolbox for Java 也意謂著您的程式會同時在從屬站與伺服器中運作, 以存取 iSeries 伺服器上的資料佇列。當執行於某一台 iSeries 伺服器上時, 它也會運作, 以存取另一台 iSeries 伺服器上的資料佇列。

  另一個選擇方案就是撰寫個別程式 (例如, 以 C 語言撰寫), 來存取資料佇列。 當 Java 程式需要存取資料佇列時, 它會呼叫此程式。

  此方法較具伺服器可攜性；您可具有一個處理資料佇列存取的 Java 程式, 及對每一個支援的伺服器具有不同版本的程式。

在 OS/400 Java 虛擬機器 (Java VM) 中執行 IBM Toolbox for Java 類別

以下為在 IBM Developer Kit for Java (OS/400) Java 虛擬機器 (Java VM) 中 執行 IBM Toolbox for Java 類別的特殊注意事項：

JDBC

有兩個 IBM 提供的 JDBC 驅動程式可供於 OS/400 JVM 中執行的程式使用：

- IBM Toolbox for Java JDBC 驅動程式
- IBM Developer Kit for Java JDBC 驅動程式

當程式在主從架構環境中執行時，最好使用 IBM Toolbox for Java JDBC 驅動程式。

當程式在 iSeries 伺服器中執行時，最好使用 IBM Developer Kit for Java JDBC 驅動程式。

如果同一個程式同時在工作站與伺服器上執行，則您應透過系統內容載入正確的驅動程式，
而不是在程式中編寫驅動程式的名稱。

程式呼叫

底下是呼叫程式的兩種常用方式：

- IBM Toolbox for Java 的 ProgramCall 類別
- 透過「Java 本機介面 (JNI)」呼叫

IBM Toolbox for Java 授權程式的 ProgramCall 類別 具有可呼叫任何 iSeries 伺服器程式的優點。

您可能無法透過 JNI 來呼叫 iSeries 伺服器程式。JNI 的優點就是具有可攜性而可在多個伺服器平台上執行。

指令呼叫

底下是呼叫指令的兩種常用方式：

- IBM Toolbox for Java 的 CommandCall 類別
- java.lang.runtime.exec()

CommandCall 類別會產生訊息清單， 以便一旦指令完成時，可供 Java 程式使用。這種訊息清單無法透過 java.lang.runtime.exec() 來使用。

java.lang.runtime.exec()
可跨平台使用，所以如果您的程式必須在不同類型的伺服器上存取檔案，java.lang.runtime.exec()
是較佳的解決方案。

整合檔案系統

以下是 iSeries 伺服器的整合檔案系統中存取檔案的常用方法：

- IBM Toolbox for Java 授權程式的 IFSFile 類別
- 本身為 java.io 一部份的檔案類別

IBM Toolbox for Java整合檔案系統類別的優點 在於它所提供的函數比 java.io 類別提供的函數還要多。IBM Toolbox for Java 類別 也可在 Applet 中運作，而且它們不需要重新導向的方法 (如 iSeries Access for Windows 即可從工作站到達伺服器。

java.io 類別具有可攜性而可在多個平台上執行，這就是它的優點。
如果您的程式必須在不同類型的伺服器上存取檔案，java.io 是較佳的解決方案。

如果在從屬站上使用 java.io 類別，您需要重新導向的方法 (如 iSeries Access for Windows)，
方可達到伺服器檔案系統。

在 OS/400 Java 虛擬機器 (Java VM) 中使用 AS400 物件設定系統名稱、使用者 ID 及密碼

當 Java 程式在 IBM Developer Kit for Java (OS/400) Java 虛擬機器 (Java VM) 中執行 時，物件允許以特殊值代表系統名稱、使用者 ID 及密碼。

在 OS/400 JVM 中執行程式時，請留意某些特殊值及其它注意事項：

- 程式於伺服器上執行時，系統會停用使用者 ID 及密碼的提示。 伺服器環境中使用者 ID 和密碼的其它相關資訊，請參閱 AS400 物件中的使用者 ID 和密碼摘要
- 如果未在 AS400 物件上設定系統名稱、使用者 ID 或密 碼，AS400 物件會使用啟動 Java 程式的工作之使用者 ID 和密碼， 與目前的伺服器連接在連接到 v4r3 或更早版本的機器時，必須在使用記錄層次存取時提 供密碼。連接到 v4r4 或更新版本的機器時，它會像其它的 IBM Toolbox for Java 元件一樣， 傳送登入使用者的密碼。
- 特殊值 localhost 可作為系統名稱。在此情形下，AS400 物件會連接至目前的伺服器。
- 特殊值 *current 可用來當作 AS400 物件中的使用者 ID 或密碼。 在此情形下，將使用啟動 Java 程式的工作的使用者 ID 或密碼 (或同時使用兩者)。*current 的其它相關資訊，請參閱以下的附註。
- 當 Java 程式在某一 iSeries 伺服器的 OS/400 JVM 上執行， 且程式正存取另一台 iSeries 伺服器的資源時，特殊值 current 可以當作 AS400 物件中的使用者 ID 或密碼使用。 在此情形下，當連接至目標系統時，將使用已在原始系統上， 啟動 Java 程式的工作的使用者 ID 與密碼。*current 的其它相關資訊，請參閱以下的附註。

  附註：

  - 如果您 使用的是記錄層次存取及 V4R3 或更早的版本，則 Java 程式不能將密碼設定為 "*current"。 當您使用記錄層次存取時，"localhost" 可用於系統名稱，而 "*current" 可 用於使 ID；不過，Java 程式必須提供密碼。
  - *current 僅能在執行版本 4 版次 3 (V4R3) 及更新的版本的系統中運作。 在執行 V4R2 的系統中，必須指定密碼及使用者 ID。

下列範例告訴您 AS400 物件如何與 OS/400 JVM 一起使用。

範例 1：Java 程式在 OS/400 JVM 中執行時， 程式毋需提供系統名稱、使用者 ID 或密碼。

當使用記錄層次存取時，必須提供一個密碼。

如果未提供這些值，AS400 物件將使用已啟動 Java 程式的工作的 使用者 ID 與密碼，連接至本端系統。

程式執行於 OS/400 JVM 上時，將系統名稱設定為 localhost 和 未設定系統名稱是一樣的。下列範例說明如何連接至目前的伺服器：

```
                    // Create two AS400 objects.  If the Java program is
// OS/400 JVM, the behavior of the two objects is the same.
// They will connect to the current server using the user ID and
// password of the job that started the Java program.
AS400 sys  = new AS400()
AS400 sys2 = new AS400("localhost")
```

範例 2：即使當程式正在 OS/400 JVM 中執 行，Java 程式仍可設定使用者 ID 和密碼。這些值將置換已啟動 Java 程式的工作的使用者 ID 與密碼。

在下列範例中，Java 程式連接至目前的伺服器，但程式使用的使用者 ID 和密碼 不同於啟動 Java 程式的工作之使用者 ID 和密碼。

```
                    // Create an AS400 object.  Connect to the current
// not use the user ID and password of the job that started the
// program. The supplied values are used.
AS400 sys = new AS400("localhost", "USR2", "PSWRD2")
```

範例 3：正在某一伺服器上執行的 Java 程式可連接至 其它 iSeries 系統，並使用其它 iSeries 系統的資源。

如果以*current 代表使用者 ID 和密碼，則當 Java 程式連接至 目標伺服器時，會使用啟動 Java 程式的工作之使用者 ID 和密碼。

在下列範例中，Java 程式在某一伺服器中執行，但使用另一台伺服器的資源。當程式連接至第二個伺服器時，會使用啟動 Java 程式的工作之使用者 ID 和密碼。

```
                    // Create an AS400 object.  This program will run on
// but will connect to a second server (called "target").
// Because    *current   is used for user ID and password, the user
// ID and password of the job that started the program will be
// used when connecting to the second server.
AS400 target = new AS400("target", "*current", "*current")
```

獨立輔助儲存體儲存區（IASP）

獨立輔助儲存體儲存區（IASP）是一個硬碟機的集合，可讓您設為上線或離線，
不受系統上其它儲存體的影響。IASP 包含下列其中一個項目：

- 一或多個使用者定義的檔案系統
- 一或多個外部檔案庫

每個 IASP 都包含所有與它包含的資料相關的必要系統資訊。 因此，當系統在作用中時，您可以將 IASP 設為離線、線上、或在系統之間切換。

詳細資訊請參閱 獨立 ASP 及使用者 ASP。

您可以使用"資料庫名稱" JDBC 內容 或 AS400JDBCDataSource 類別的 setDatabaseName() 方法
來指定您要連線的 ASP。

所有其它的 Toolbox for Java 類別（IFSFile、Print、DataQueues 等）
會使用連線到伺服器的使用者設定檔工作說明所指定的 IASP。

OS/400 最佳化

IBM Toolbox for Java 授權程式是以 Java 撰寫，因此它可以在任何具有已認證 的 Java 虛擬機器 (Java VM) 之平台上執行。不管在何處，IBM Toolbox for Java 類別 也依相同的方式執行。

在 iSeries JVM 執行 IBM Toolbox for Java 時，隨附 OS/400 的附加類別會 增強 IBM Toolbox for Java 的行為 iSeries JVM 中執行並連接至同一 台 iSeries 時，會增進登入行為及效能。從版本 4 版次 3 開始，OS/400 將包含附加的類別。

啟用最佳化

IBM Toolbox for Java 分成兩種套件：單獨的授權程式及 OS/400。

- Licensed Program 5722-JC1。授權程式版的 IBM Toolbox for Java 在下列目錄傳送檔案：

  ```
  /QIBM/ProdData/http/public/jt400/lib
  ```

  這些檔案不包含最佳化的 OS/400。如果要持續在從屬站執行 IBM Toolbox for Java，請使用這些檔案。
- OS/400。IBM Toolbox for Java 也在下列目錄中以 OS/400 的形式傳送

  ```
  /QIBM/ProdData/OS400/jt400/lib
  ```

  這些檔案包含在 iSeries JVM 中執行時，可最佳化 IBM Toolbox for Java 的類別。

其它相關資訊，請參閱在 Jar 檔案相關資訊中 之附註 1。

登入注意事項

有了 OS/400 提供的附加類別，Java 程式便可具有額外選項， 向 IBM Toolbox for Java 提供系統名稱、使用者 I 和密碼資訊。

存取 iSeries 資源時，IBM Toolbox for Java 類別必須有系統名稱、使用者 ID 和密碼。

- 在從屬站中執行時，是由 Java 程式提供系統名稱、 使用者 ID 和密碼，或是由 IBM Toolbox for Java 透過登入對話框， 從使用者擷取這些值。
- 在 iSeries Java 虛擬機器中執行時 IBM Toolbox for Java 有另一個選擇。 其可使用啟動 Java 程式的工作之使用者 ID 和密碼，傳送要求到目前 (本端) 伺服器。

有了附加的類別，則在一個 iSeries 中執行的 Java 程式存取另一個 iSeries 的資源時， 也可使用目前工作的使用 和密碼。在此情況下，Java 程式會設定系統名稱， 然後使用特殊值 "*current" 代替使用者 ID 及密碼。

如果您使用的是記錄層次存取 V4R4 或更新的版本，則 Java 程式僅能將 密碼設定為 "*current"。否則，您您使用記錄層次存取時，"localhost" 可用於 系統名稱，而 "*current" 可用於使用者 ID；不過，Java 程式必須提供密碼。

Java 程式會在 AS400 物件中設定系統名稱、使用者 ID 和密碼值。

如欲使用工作的使用者 ID 和密碼，Java 程式可以使用 "*current" 作為使用者 ID 及密碼，或它可以使用沒有使用
ID 及密碼參數的建構元。

如欲使用目前的 iSeries，Java 程式可使用 localhost 作為系統名稱，或使用預設建構元。亦即，

```
AS400 system = new AS400();
```

同於

```
AS400 system = new AS400("localhost", "*current", "*current");
```

會在下面範例中建立兩個 AS400 物件。 兩個物件具有相同的行為：它們同時使用工作的使用者 ID 和密碼，
對目前的 iSeries 執行指令。其中一個物件會使用特殊值代表使用者 ID 和密碼，另一個則會使用預設建構元，
且不會設定使用者 ID 或密碼。

```
                    // Create an AS400 object. Since the default
                    // constructor is used and system, user ID and
                    // password are never set, the AS400 object sends
                    // requests to the local iSeries using the job's
                    // user ID and password. If this program were run
                    // on a client, the user would be prompted for
                    // system, user ID and password.
AS400 sys1 = new AS400();

                    // Create an AS400 object. This object sends
                    // requests to the local iSeries using the job's
                    // user ID and password. This object will not work
                    // on a client.
AS400 sys2 = new AS400("localhost", "*current", "*current");

                    // Create two command call objects that use the
                    // AS400 objects.
CommandCall cmd1 = new CommandCall(sys1,"myCommand1");
CommandCall cmd2 = new CommandCall(sys2,"myCommand2");

                    // Run the commands.
cmd1.run();
cmd2.run();
```

在下列範例中，將建立一個 AS400 物件，來代表第二個 iSeries 系統。 因為使用了 *current，所以會在第二個
(目標) iSeries 上使用來自執行 Java 程式 的 iSeries 之工作的使用者 ID 和密碼。

```
                    // Create an AS400 object. This object sends
                    // requests to a second iSeries using the user ID
                    // and password from the job on the current iSeries.
AS400 sys = new AS400("mySystem.myCompany.com", "*current",
"*current");
```

```
                    // Create a command call object to run a command
                    // on the target iSeries.
CommandCall cmd = new CommandCall(sys,"myCommand1");


                    // Run the command.
cmd.run();
```

效能的增進

利用 OS/400 所提供的附加類別，在 iSeries Java 虛擬機器中執行的 Java 程式可提高效能。
在某些情況下會因為使用的通信功能減少而改善效能，但在某些情況下，會使用 iSeries API，而非呼叫伺服器程式

較短的下載時間

為了能下載最少數量的 IBM Toolbox for Java 類別檔案， 請使用附有AS400ToolboxJarMaker 工具 的Proxy 伺服器。

較快的通信

對所有 IBM Toolbox for Java 功能 (JDBC 與整合檔案系統存取除外) 而言， 在 iSeries Java 虛擬機器中執行的
程式，其執行速度會更快。 程式的執行速度之所以會更快，是因為在 Java 程式與伺服器中執行要求的
伺服器程式之間進行通信時，使用更少的通信碼。

JDBC 與整合檔案系統存取並未最佳化，因為已存在使這些功能的執行速度更快的機能。 執行於 iSeries
之上時，您可以使用 iSeries 的 JDBC 驅動程式， 代替 IBM Toolbox for Java 所隨附的 JDBC 驅動程式。
若要存取伺服器中的檔案，您可以使用 Java.io，代替 IBM Toolbox for Java 所附的 整合檔案系統存取類別。

直接呼叫 iSeries API

IBM Toolbox for Java 之下列類別的效能獲得改善， 因為這些類別會直接呼叫 iSeries
API，而非呼叫伺服器程式來執行要求。

- AS400Certificate 類別
- CommandCall
- DataQueue
- ProgramCall
- 記錄層次的資料庫存取類別
- ServiceProgramCall
- UserSpace

僅在使用者 ID 和密碼符合執行 Java 程式的工作的使用者 ID 和密碼時，才能直接呼叫 API。
若要改善效能，使用者 ID 和密碼必須符合啟動 Java 程式的工作的使用者 ID 和密碼。
若要取得最佳結果，請使用 localhost 代表系統名稱、*current 代表使用者 ID 及 *current 代表密碼。

埠對映變更

埠對映系統已變更過，可使得埠的存取更快速。在此變更之前，埠的要求會傳送到埠對映程式。 從那裡，iSeries
伺服器會判斷哪個埠可用，並傳回該埠傳給接受的使用者。
現在，您可以告知伺服器要使用哪一個埠，或指定使用預設埠。
此選項可免去伺服器替您決定埠所浪費的時間。您可使用 WRKSRVTBLE
指令來檢視或變更伺服器的埠清單。

為改善埠對映，已新增若干方法到AS400 類別 ：

- getServicePort
- setServicePort
- setServicePortsToDefault

MRI 變更

MRI 檔現在已隨附在 IBM Toolbox for Java 程式內提供，作為類別檔， 而非內容檔。iSeries 伺服器在類別檔中比在內容檔中可更快速地找到訊息。ResourceBundle.getString() 現在的執行速度更快，因為 MRI 檔儲存在 電腦會進行搜尋的第一個位置。變更為類別檔的另外一個優點為伺服器可以更快找到字串的轉換版本。

轉換器

容許 Java 與 iSeries 之間進行更快速、有效的轉換的兩種類別：

- 二進位轉換器: 在 Java 位元組陣列與 Java 簡單類型之間轉換。
- 字元轉換器: 在 Java 字串物件與 iSeries 程式碼套件之間進行轉換。

此外，IBM Toolbox for Java 目前也納入了本身的轉換表， 內有超過 100 個常用 CCSID。 先前的 IBM Toolbox for Java 延遲了幾乎所有的 Java 文字轉換。 要是 Java 沒有正確的轉換表，IBM Toolbox for Java 從伺服器中下載轉換表。

IBM Toolbox for Java 會對所能辨識的任何 CCSID 執行一切文字轉換。 遇到不明的 CCSID 時，會嘗試由 Java 來處理轉換作業。IBM Toolbox for Java 絕對不會嘗試從伺服器下載轉換表。這套技術使得 IBM Toolbox for Java 應用程式執行文字轉換所花的時間大幅縮短。使用者無需採取任何行動 即可享受這套文字轉換新制所帶來的優點；效能增益全部都由底層轉換表產生出來。

有關「建立 Java 程式」(CRTJVAPGM) 指令的效能要訣

如果您的 Java 程式是在 iSeries Java 虛擬機器 (JVM) 中執行，則當您 從 IBM Toolbox for Java zip 檔或 jar 檔建立 Java 程式時， 您可以大幅改善效能 。在 iSeries 指令行中輸入 CRTJVAPGM 指令，建立程式。(如需詳細資訊，請參閱 CRTJVAPGM 指令的線上說明資訊。) 利用CRTJVAPGM 指令， 您可儲存在 Java 程式啟動時所建立的 Java 程式 (且其中包含 IBM Toolbox for Java 類別)。 儲存已建立的 Java 程式可讓您節省啟動處理時間。 您可節省啟動處理時間，因為每次啟動您的 Java 程式時，伺服器中的 Java 程式不必再重建。

如果您使用的是 V4R2 或 V4R3 版本的 IBM Toolbox for Java， 則您不能對 jt400.zip 或 jt400.jar 檔執行 CRTJVAPGM 指令， 因為它太大了；不過，您可以對 jt400Access.zip 檔執行該指令。在 V4R3 中，IBM Toolbox for Java 授權程式包含一個額外的檔案：jt400Access.zip。jt400Access.zip 僅有存取類別，而沒有視覺化類別。

在 V4R5 (或更早的版本) 系統中執行 Java 應用程式時，請使用 jt400Access.zip。 在 V5R1 系統中執行 Java 應用程式時，請使用 jt400Native.jar。CRTJVAPGM 指令已對 jt400Native.jar 執行過。

Java 國家語言支援

Java 支援一組國家語言，但它是伺服器支援的語言之子集。

當語言之間有不符的情形時，例如，如果您正在使用 Java 不支援的語言之 本端工作站上執行作業，則 IBM Toolbo
for Java 授權程式能會以英文發出一些錯誤訊息 。

IBM Toolbox for Java 的服務及支援

使用下列資源取得服務及支援：

Toolbox for Java 疑難排解資訊
當您使用 Toolbox for Java 時，請使用此資訊來幫您解決問題。

JTOpen/Toolbox for Java 論壇
加入使用 Toolbox for Java 的 Java 程式設計師社群。這個論壇是取得其它 Java 程式設計師及偶而來自 Toolbox for Java 程式開發者本身的協助與建議的有效方法。

伺服器支援
使用 IBM Server 支援網站找出工具及資源，幫您將 iSeries 伺服器的技術規劃與支援流線化。

軟體支援中心
使用「IBM 軟體支援中心服務」網站可找出 IBM 所提供的大規模軟體支援服務。

IBM Toolbox for Java，5722-JC1 的支援服務，是依據 iSeries 軟體產品的一般規定與條款提供。支援服務包括程式服務、語音支援及咨詢服務。有關其它資訊，請連絡為您提供服務的 IBM 業務代表。

關於 IBM Toolbox for Java 程式的問題可透過程式服務及語音支援取得支援，關於應用程式設計及除錯的事項則是以諮詢服務方式支援。

除非下列有一項為 True，不然 IBM Toolbox for Java 應用程式介面 (API) 呼叫 方面是以諮詢服務方式支援：

- 它的確是 Java API 錯誤，且可經由在相當簡單的程式中重新建立，來示範該錯誤。
- 它是一個要求文件闡明的問題。
- 它是一個關於範例或文件的位置的問題。

所有程式設計輔助均是由諮詢服務加以支援，包括 IBM Toolbox for Java 授權程式中所提供的程式碼範例在內。其它範例可在網際網路上的 iSeries 首頁 取得，這些範例是無支援方式提供。

「IBM Toolbox for Java 授權程式產品」附有問題解決資訊。如果您認為 IBM Toolbox for Java API 中可能有錯，您需要提供一個可示範該錯誤的簡單程式。

IBM Toolbox for Java 的相關資訊

下列清單中包含與 IBM Toolbox for Java 資訊相關的網站和資訊中心主題。

IBM Toolbox for Java 的資源

請利用下列網站來更加瞭解 IBM Toolbox for Java：

- IBM Toolbox for Java and JTOpen：提供有關服務修正程式包、效能秘訣、範例等等豐富的資訊。您也可以下載這份資訊的壓縮套裝，包括 javadocs 在內。
- IBM Toolbox for Java Frequently Asked Questions (FAQ)：提供效能、疑難排解、JDBC 等等相關問題的答覆。
- IBM Toolbox for Java 及 JTOpen forum：提供有效的管道，能與使用 Toolbox for Java 的 Java 程式設計師以 Toolbox for Java 程式開發者本身的社群交流溝通。

»IBM Toolbox for Java 2 Micro Edition 資源

請利用下列網站來更加瞭解 ToolboxME for iSeries 和 Java 施行的無線通訊技術：

- IBM Toolbox for Java and JTOpen：提供 ToolboxME for iSeries 更多的相關資訊。
- IBM alphaWorks Wireless：提供新無線通訊技術的相關資訊，包括下載和開發資源的鏈結。
- Sun Java 2 Platform, Micro Edition：提供 Java 無線通訊技術更多的相關資訊，包括如下：
  - K Virtual Machine (KVM)
  - Connected Limited Device Configuration (CLDC)
  - Mobile Information Device Profile (MIDP)
- Java Wireless Developer：為 Java 無線通訊應用程式開發者提供廣泛的技術資訊。
- 無線通訊應用程式開發工具：
  - IBM WebSphere Studio Device Developer
  - Java 2 Platform Micro Edition, Wireless Toolkit

Java

Java 是可讓您開發可攜性物件導向應用程式和 applet 的程式設計語言。 請利用下列網站來更加瞭解 Java：

- IBM developer Works Javatechnology zone：提供資訊、教育和工具，協助您使用 Java、IBM 產品和其它技術，建立商務解決方案。
- IBM alphaWorks Java：提供新 Java 技術的相關資訊，包括下載和開發資源的鏈結。
- Sun Microsystems 提供的 "The Source for Java Technology"：提供 Java 各種用法的相關資訊，其中也包括新技術在內。
- Java for iSeries, PartnerWorld for Developers：提供 Java 相關資訊，以及 IBM 事業夥伴可以使用且正在使用的方式。

Java Naming and Directory Interface

- Java Naming and Directory Interface (JNDI)：提供 JNDI 的概況、技術資訊、範例，以及可利用的服務供應商名單。
- iSeries 400 Directory Services (LDAP)：提供 OS/400 中使用 LDAP (輕裝備目錄存取通信協定) 的相關資訊。

Java Secure Socket Extension

- Java Secure Socket Extension (JSSE)：提供 JSSE 的簡要概述和更多資訊的鏈結。

Servlet

Servlet 是在伺服器中執行的小型 Java 程式，能夠中介 一個或多數從屬站 (各在瀏覽器中執行) 對一部或更多資料庫的要求。 因為 servlet 是用 Java 程式設計而成，所以能夠以單一處理中多重執行緒來執行要求，於是可節省系統資源。請利用下列網站來更加瞭解 servlet：

- IBM Websphere, PartnerWorld for Developers：提供 servlet 型 Web 應用程式伺服器的相關資訊。
- Java Servlet technology：提供技術資訊、指令和工具，協助您瞭解和運用 servlet。

XHTML

XHTML 因為是 HTML 4.0 的繼承者而受到重視。它根據 HTML 4.0， 但納入了 XML 的延伸性。請利用下列網站來更加瞭解 XHTML：

- The Web Developer's Virtual Library：提供 XHTML 的介紹，包括範例和更多資訊的鏈結。
- W3C：提供 XHTML 標準建議的相關技術資訊。

XML

Extensible Markup Language (XML) 是可讓您用人類和電腦都很容易瞭解的方式來說明和組織資訊的中繼語言。中繼語言可讓您定義文件標記語言和這種語言的結構。請利用下列網站來更加瞭解 XML：

- IBM developerWorks XML zone：提供一個專屬於 IBM 研發 XML 工作的網站，以及它如何增加電子商務的便利性
- IBM alphaWorks XML：提供 XML 技術的新興標準和工具的相關資訊，包括下載和開發資源的鏈結。
- XML Support on iSeries, PartnerWorld for Developers：提供 XML 相關資訊，以及 IBM 事業夥伴可以使用且正在使用的方式。
- W3C XML：提供 XML 程式開發者的技術資源。
- XML.com：提供 XML 於電腦業界的最新資訊
- XML.org：提供 XML 社群的新聞和資訊，包括產業新聞、行事曆等等。
- XMLephant：提供學習 XML 的資源，包括許多其它 XML 網站的鏈結。

- XML Cover Pages：提供 XML、SGML 和相關的 XML 標準如 XSL、XSLT 的綜合性線上參考文章。

其它參考資料

- IBM HTTP Server for iSeries：提供 IBM HTTP Server for iSeries 的相關資訊、資源和祕訣。
- iSeries Access for Windows：提供 iSeries Access for Windows 的相關資訊，包括下載、常見問題集 (FAQ)，與其它網站的鏈結。
- IBM WebSphere Host On-Demand：提供支援 S/390、iSeries 和 DEC/Unix 模擬的瀏覽器型模擬器的相關資訊。
- IBM Support and downloads：提供 IBM 硬體、軟體支援的入口網站。

程式碼不保事項聲明

此文件包含程式設計範例。

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，
您可以依據這些範例，產生類似的函數，來符合您的需要。

IBM 提供的所有範例程式碼僅做為說明用途。這些範例尚未徹底經過所有情況的測試。 因此 IBM
不擔保或默示保證這些程式的可靠性、可用性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM
明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。