Net.Data

OS/2、Windows NT 和 UNIX 版管理及程式設計指南

版本 6 版次 1

Net.Data

OS/2、Windows NT 和 UNIX 版管理及程式設計指南

版本 6 版次 1

注意事項

請務必先詳讀第201頁的『附錄G. 注意事項』中的資訊，再使用此資訊及其所支援的產品。

第六版 (1999 年 5 月)

目錄

序文

感謝您選擇 Net.Data® 版本 6.1，這是用來建立動態 Web 網頁的 IBM™ 開發工具。使用 Net.Data，您可以藉由併入來自各種資料來源的資料，同時使用您已熟知之程式設計功能，快速地開發出具有動態內容的網頁。

關於 Net.Data

透過 IBM 的 Net.Data 產品，您便可以使用來自關聯式與非關聯式資料庫管理系統 (DBMS) 的資料 (這些系統包括DB2、IMS 與具有 ODBC 能力的資料庫) 以及使用以程式設計語言 (如 Java, JavaScript, Perl, C, C++ 與 REXX) 撰寫的應用程式，來建立動態網頁。

Net.Data 是一個巨集處理器，它以 Web 伺服器機器上的中間軟體之身份來執行。您可撰寫 Net.Data 應用程式 (稱爲 *macro*)， Net.Data 解譯該應用程式以自行設定的內容建立動態網頁，這些自行設定的內容是根據使用者的輸入、資料庫的現行狀態、其它資料來源、現有的企業邏輯以及您指定給巨集的其它因數。

使用格式 URL (一致資源定址器) 的要求從瀏覽器 (如 Netscape Navigator 或 Internet Explorer) 傳送到 Web 伺服器，該伺服器將要求轉送至 Net.Data 以供執行。 Net.Data 找到該巨集並加以執行，然後依您所寫的函數，建置一個自行設定的網頁。這些函數可以：

- 將企業邏輯封裝在 Perl script、C 與 C++ 應用程式，或 REXX 程式內。
- 存取資料庫，如 DB2
- 存取其它資料來源，如純本文檔。

Net.Data 會將這個網頁傳送給 Web 伺服器，然後，這個伺服器會透過網路轉送此網頁，以顯示在瀏覽器上。

Net.Data 可在伺服器環境中使用，因爲這些環境已被架構成可使用如「超本文轉送通信協定 (HTTP)」及「通用閘道介面 (CGI)」等介面。 HTTP 是一種用於瀏覽器及 Web 伺服器之間交談的工業標準介面，而 CGI 則是 Web 伺服器呼叫如 Net.Data 的閘道應用程式時所用的工業標準介面。這些介面可讓您選取喜愛的瀏覽器或 Web 伺服器，與 Net.Data 搭配使用。 Net.Data 也支援不同的 Web 伺服器「應用程式設計介面 (API)」，來提高執行效能。 Net.Data 系列產品會在 OS/400、OS/390、Windows NT、AIX、OS/2、HP-UX、 Sun Solaris、Linux 及 Santa Cruz Operating System (SCO) 作業系統上提供類似的功能。 Net.Data 也支援 FastCGI 與多重作業系統上的主要 Web 伺服器應用程式設計介面 (API)。

圖形管理工具可協助您管理 AIX、Windows NT 與 OS/2 作業系統的 Net.Data 架構設定。管理工具也可協助您指定連接使用「現場連線」之資料庫時的安全。

爲了協助您輕易地從資料庫中存取資料，Net.Data 提供有各種不同種類的工具，包括 NetObjects Fusion plug-in 及 Java 式開發的 SmartGuide。這些工具在 Java 環境中都可與 Net.Data Java servlet搭配運作，可讓您建立不同作業系統均適用的應用程式。 NetObjects Fusion plug-in 可讓您使用 NetObjects Fusion Web 開發工具，以利用關聯資料來源中的動態資料來建置複雜的應用程式。 Net.Data SmartGuide 會提供圖形式工具，引導您建立基本 Net.Data 巨集。

新增功能？

下列段落將描述 Net.Data 的新增強功能。

版本 2.0.7 修正檔案的新增功能？

Net.Data 版本 2.0.7 提供下列的增強功能：

- 可使用「 DB2 檔案管理程式」及 DATALINK 資料類型
- 支援新的 Net.Data 表格處理函數
- 安全的架構變數：DTW_DIRECT_REQUEST 及 DTW_SHOWSQL，以及提高您的應用程式安全性的指南

版本 6 版次 1 的新增功能？

Net.Data for OS/2, Windows NT 及 UNIX 在版本 6 版次 1 中提供下列特性：

- 執行效能增強包括：
  - 可在 Solaris 作業系統上支援 FastCGI
- 語言環境增強功能包括：
  - 可在 Windows NT 上的 SQL 語言環境中支援 Datalink 資料類型
  - 可在 Solaris 作業系統上支援 IMS Web 語言環境
  - 可在 ODBC 語言環境中支援大型物件 (LOB)
  - 支援新的 LOB 類型：樂器數位介面音效檔 (.mid)、音效交換檔案格式、音效檔 (.aif)、基本音效檔 (.au)、眞正音效檔 (.ra)、可攜性文件格式檔 (.pdf)，以及 Windows 音效影像檔 (.wav)
  - 使用 `cleanup` 常駐程式簡化暫時大型物件目錄 (`tmplobs`) 的維護工作
- 巨集語言增強功能包括：
  - 可在 DTW_TIME 函數中支援毫秒
  - 可在 MACRO_FUNCTION 區塊中支援 REPORT 區塊
  - 可在 FUNCTION 及 MACRO_FUNCTION 區塊中支援多個 REPORT 區塊
  - 可動態建置變數參照
  - 能夠在 DTW_SELECT() 的 OPTION 元素中設定 VALUE 值
  - 可在變數名稱中使用 # 字元。
- 可記錄「現場連線」錯誤及活動
- 可在 Linux 作業系統上支援 Net.Data
- 可透過 DTW_UNICODE 架構變數在巨集檔及 DB2 資料庫中支援 Unicode 字元

關於本書

本書討論的是 Net.Data 的管理及程式設計概念，以及如何架構 Net.Data 及其構成要素的方式、規劃安全，與提高執行效能。

依據程式設計語言及資料庫的知識，您可學習如何使用 Net.Data 巨集語言或 Java servlet，以開發巨集。您可以學到如何使用 Net.Data 提供的語言環境，使用 IMS Web 存取 DB2 資料庫 與 IMS 異動，，以及使用 Java, REXX, Perl, 與其他程式設計語言來存取您的資料。

本書可能會提及一些已發表，但尚未上市的產品或特性。

關於 Net.Data 巨集樣本、示範程式及本書最新版本的資訊，請參訪下列的「全球資訊網 (WWW)」網站：

http://www.software.ibm.com/data/net.data

誰應閱讀本書

本書適用於想規劃與撰寫 Net.Data 應用程式的人員。爲了瞭解本書中所提及的各種概念，您必須熟悉 Web 伺服器的工作，並了解簡單的 SQL 陳述式與 HTML 標籤 (包括 HTML 套表標籤)。

Net.Data 巨集語言、變數及內建函數，以及作業系統差異性會在 *Net.Data 參考手冊*中加以描述。

關於本書中的範例

本書中會舉簡單的範例，爲您說明某些特定的概念，這些範例並未涵蓋 Net.Data 結構的每一種使用方式。有些範例僅舉片段，若要能運作則還需其他的程式碼。

第1章 簡介

網際網路上的大多數網頁均是靜態的網頁；換言之，除非您加以編輯，否則這些網頁不會變更。若要將「現場」資料及應用程式放到 Web (如現行銷售統計)，網站軟體開發者通常會撰寫在 Web 伺服器上當成 middleware 使用的程式，以動態建置網頁。 然而撰寫各類型的程式並不容易。

透過*巨集*， Net.Data 可簡化交談式 Web 應用程式的撰寫。

本章將描述 Net.Data 及爲什麼要爲您的 Web 應用程式使用它的理由。

- 『何謂 Net.Data？』
- 第2頁的『爲什麼使用 Net.Data？』

何謂 Net.Data？

使用 Net.Data 巨集，您可以執行程式設計邏輯、存取及操作變數、呼叫函數，以及使用報表產生工具。巨集是一種含有 Net.Data 巨集語言結構、 HTML 標籤、Javascript 與語言環境陳述式 (如 SQL 與 Perl) 的文字檔。 Net.Data 會處理巨集，來產生可由 Web 瀏覽器顯示的輸出。巨集會將 HTML 的簡易性與 Web 伺服器程式的動態功能結合，使您能輕鬆地將動態資料新增到靜態網頁中。可從本端或遠端資料庫及純文字檔中擷取動態資料，或由應用程式及系統服務所產生。

圖1 會描述 Net.Data、Web 伺服器及支援的資料與程式設計語言環境之間的關係。

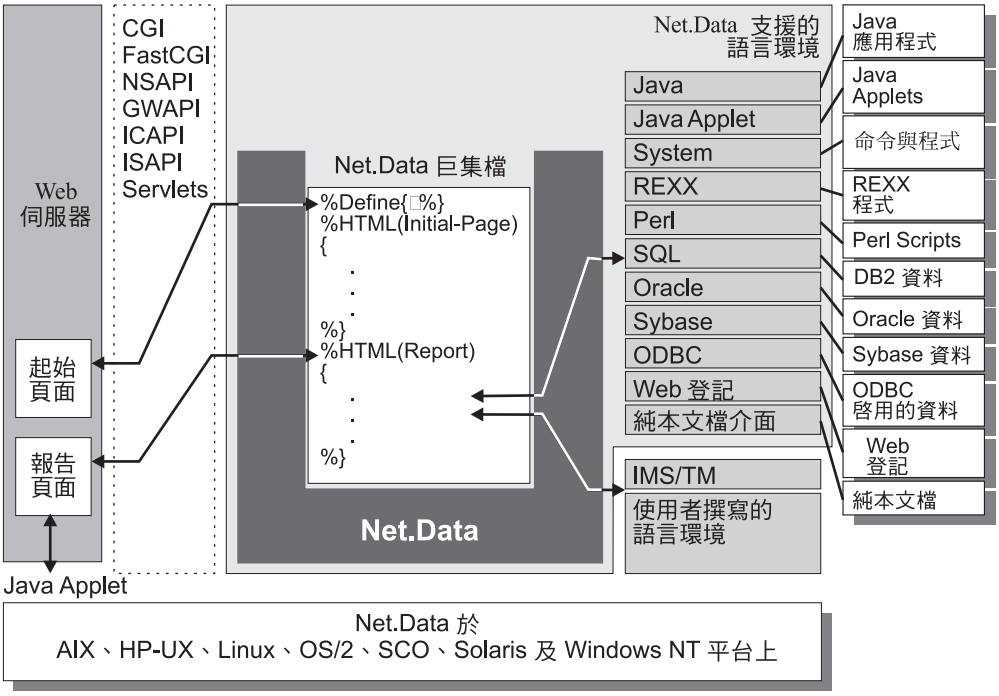


圖 1. Net.Data、Web 伺服器及支援之資料與程式來源之間的關係

當 Web 伺服器收到請求 Net.Data 服務的要求時，Web 伺服器會經由呼叫 Net.Data 作爲 DLL 或共用程式庫來呼叫 Net.Data 作爲 CGI、 FastCGI 或 Web 伺服器應用程式設計介面 (API)。 URL 包括 Net.Data 的特定資訊，包括將處理的巨集或直接呼叫的 SQL 陳述式或程式。 Net.Data 完成要求的處理時，會將結果網頁傳送到 Web 伺服器。伺服器會將它傳送到 Web 從屬站，這樣就可使用瀏覽器來顯示。

為什麼使用 Net.Data？

Net.Data 極適合用來建立動態的網頁，因爲使用巨集語言比撰寫自己的 Web 伺服器應用程式簡單，且 Net.Data 可讓您使用已知的語言 (如 HTML、SQL、Perl、REXX 及 JavaScript)。 Net.Data 也會提供存取 DB2 資料庫的語言環境、使用 IMS Web 執行 IMS 異動，或使用 REXX、Perl 及您的應用程式的其他語言。 此外，可以立即在瀏覽器上看到您對巨集所做的變更。

Net.Data 會經由啓用 Web 的資料及相關企業邏輯，來 已存在於您的作業系統上的資料管理能力。尤其，Net.Data：

- 提供簡單但功能強大的巨集語言，容許快速開發網際網路及 Intranet 應用程式。 Net.Data Web 應用程式環境提供下列特性：
- 允許在您的 Web 應用程式內將資料建立邏輯與呈現邏輯分開。 Net.Data 不會對資料的呈現方式 (例如，以 HTML 或 Javascript 呈現) 強加任何限制。這個分開容許使用者輕易地使用最新呈現技術來變更資料的呈現方式。
- 經由提供與 C, C++, REXX, Java 或其他語言互通的能力，讓您可使用舊有的技巧及企業邏輯來建立 Web 網頁。
- 提供使用簡單巨集語言，來迅速開發複雜的網際網路應用程式的能力。
- 提供對於儲存在 DB2 及任何遠端 DRDA 啓用的資料庫中之資料的存取有高效能的表現。
- 提供 Net.Data 系列產品支援的所有作業系統之間的簡易巨集移轉。

解譯的巨集語言

Net.Data 巨集語言是一種經過解譯的語言。當呼叫 Net.Data 來處理巨集時，Net.Data 會從檔案頂端起按順序直接解譯檔案中的每一條語言陳述式。使用這個方法，在您下次指定執行此巨集的 URL 時，您就可立即看到對巨集所作的變更。並不需要重新編譯。

直接要求

執行單一 SQL 陳述式、DB2 儲存程序、REXX 程式、 C 或 C++ 程式，或 Perl script 的簡式要求，並不需要建立巨集。這些要求可直接指定在從瀏覽器流向 Web 伺服器的 URL 中。

自由格式

Net.Data 巨集語言在程式設計格式上的規定不多。這個簡易性提供給程式設計師更多的自由及彈性。單一指令可跨多行，而多個指令也可全寫在同一行上。指令可開始於任何直欄。而空格或整行也可以被跳過。也可在任意位置使用說明。

無任何類型的變數

Net.Data 是將所有資料全視爲字串。Net.Data 使用內建函數以對代表有效數字 (包括採用指數格式的數字) 的字串執行算術運算。巨集語言變數的詳細討論，請參閱第84頁的『Net.Data 巨集變數』。

內建函數

Net.Data 會提供一些內建的函數，讓您用來對文字或數字執行各種處理、搜尋、與比較作業。 另外，其他的內建函數，還提供格式化功能與算術運算能力。

錯誤處理

Net.Data 偵測到錯誤時，會將含有說明的訊息傳回給從屬站。在錯誤訊息傳回給使用者或瀏覽器之前，您可自行設定它們。請參閱 *Net.Data 參考手冊*，以取得其他相關資訊。

第2章 架構 Net.Data

您可以使用產品隨附的 README 檔案中之指示，安裝符合您作業系統平台的 Net.Data。在安裝期間會完成大多部架構步驟；不過，這會隨著作業系統而有所不同。

在 安裝適合您作業系統的 Net.Data 之後，您必須架構 Net.Data 並修改 Web 伺服器的架構。架構作業包括：

- 自行設定 Net.Data 起始設定 (INI) 檔
- 架構 Net.Data 來使用於 FastCGI，或其中一個支援的 Web 伺服器 API (可選用的)
- 自行設定 Web 伺服器與環境變數檔案
- 架構「快取管理程式」(可選用的)
- 架構「現場連線」(可選用的)
- 設置 Net.Data 語言環境
- 設定存取權

您可以使用下列工具來架構 Net.Data：

- 文字編輯器

  在所有作業系統上，可使用文字編輯器來編輯起始設定檔及「現場連線」架構檔。您也可以使用文字編輯器來更新任何 Web 伺服器的架構檔。 在進行變更之前，建議您製作檔案備份。

- Net.Data 管理工具

  管理工具提供圖形介面，供您自行設定起始設定檔和「現場連線」架構檔。在 OS/2、Windows NT 及 AIX 作業系統上，您可以使用管理工具來架構 Net.Data。

使用方法視所要架構的元件及執行 Net.Data 的作業系統而定，其說明於表1中。如果您開始使用某一特定方法來進行架構作業，則爲了取得最佳效能，您應該繼續使用該方法。

表 1. 作業與作業系統架構方法的比較。 **A** *- 可使用管理工具或自行架構。* **M** *- 僅能自行架構。*

作業	作業系統			
	AIX	**NT**	**OS/2**	**HP|SUN|SCO**
架構 Net.Data INI 檔	A	A	A	M
定義 cliette 埠	A	A	A	M
定義 cliette	A	A	A	M
開啓 cliette 密碼加密	A	N/A	N/A	N/A
啓動錯誤日誌	A	A	A	M
架構用於 FastCGI、CGI 及 APIs* 的 Web 伺服器	M	M	M	M
定義「快取管理程式」埠	M	M	N/A	N/A
架構「快取管理程式」	M	M	N/A	N/A

***要訣：** 許多 Web 伺服器都具有可用來架構 Web 伺服器的管理工具。

本章說明如何架構 Net.Data，以及如何修改與 Net.Data 一起使用的 Web 伺服器架構。此外，本章亦說明如何架構選用性元件。

- 第11頁的『自行設定 Net.Data 起始設定檔』
- 第22頁的『設置語言環境』
- 第27頁的『架構「現場連線」』
- 第32頁的『架構 Web 伺服器與 CGI 一起使用』
- 第33頁的『架構 Net.Data for FastCGI』
- 第36頁的『架構 Net.Data 來使用 Java Servlet 及 Java Bean』
- 第36頁的『架構 Net.Data 來使用 Web 伺服器 API』
- 第38頁的『使用 Net.Data 管理工具來架構 Net.Data』
- 第48頁的『授與 Net.Data 存取的檔案的存取權』

關於 Net.Data 起始設定檔

Net.Data 使用其起始設定檔來建立各種架構變數的設定值，及架構語言環境和搜尋路徑。架構變數的設定會控制 Net.Data 作業的如下不同層面：

- 將字元資料以 Unicode 編碼
- 字串及字組函數是否可使用 DBCS
- 存取資料庫資料時所用的 DB2 案例的名稱
- 如何與語言環境、資料庫、連接管理與快取來進行連接與通信
- 是否啓動錯誤記錄

語言環境陳述式可定義一些可用的 Net.Data 語言環境，以及識別語言環境使用的特殊輸入和輸出參數值。語言環境可讓 Net.Data 存取不同資料來源，如 DB2 資料庫與系統服務程式。 路徑陳述式設定 Net.Data 所使用之檔案的目錄路徑，例如，巨集、REXX 程式、 Perl script。

Net.Data 起始設定檔 db2www.ini，其位於 Web 伺服器的文件目錄中。請參閱適合您作業系統的 README 檔，以取得有關的詳細資訊。

授權要訣：確定 Web 伺服器有權存取此檔。 有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

關於選用性元件之 Net.Data 架構檔

下列各節討論 Net.Data 選用性元件之架構檔。

『「現場連線」架構檔』

第7頁的『「快取管理程式」架構檔』

第7頁的『Net.Data 起始設定、控制和巨集檔的一般區段』

「現場連線」架構檔

「現場連線」在 Windows NT、OS/2、AIX、及 Sun Solaris 作業系統上提供連線管理，藉由取消啓動時的額外執行時間來改進執行效能。Net.Data「現場連線」架構檔包含關於一個或多個指名 cliette 之資訊。 cliette 是長期執行處理，它維護與資料庫或 Java 應

用程式的連線，能接受來自多個使用者的 Net.Data 巨集呼叫。 cliette 啓動之後，它會一直存在，直到 Net.Data「現場連線」終止爲止。 多個 cliette 可以連接到單一資料庫。

您設定 cliette 名稱、獨一無二的埠、處理數目的最小值和最大值，這些是架構檔中的 cliette 資訊的一部份。以資料庫 cliette 而言，亦可設定每一個 cliette 登錄的資料庫名稱、登入和密碼。在 AIX 上，您可加密密碼。

授權要訣：確定啓動「連線管理程式」的使用者 ID 具有此檔案的存取權力。有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

「快取管理程式」架構檔

「快取管理程式」架構檔包含「快取管理程式」和每一個快取的定義。關於 Net.Data 快取，說明於第153頁的『Net.Data 快取』 中。關於架構「快取管理程式」，說明於第158頁的『架構「快取管理程式」和 Net.Data 快取』。 檔案結構是一系列區段或段落：

「快取管理程式」段落

這個段落定義「快取管理程式」本身的參數，並包括網路資訊、記錄狀態、及追蹤狀態。此段落是必要的，且必須標示爲 `cache-manager`。

快取定義段落

這些段落定義每一個快取的參數；在架構檔中，「快取管理程式」所管理的每一個快取，都有一個快取定義段落；本節包含網路資訊、記憶體和空間需求、記錄狀態、和統計值狀態。對於「快取管理程式」所管理的每一個快取而言，快取定義段落是必要的。

「快取管理程式」架構檔不是由管理工具管理，而且可用任何的文字編輯器來更新。請參閱第153頁的『Net.Data 快取』以了解如何定義這個檔案。

授權要訣：確定啓動「快取管理程式」的使用者 ID 有權存取此檔。有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

Net.Data 起始設定、控制和巨集檔的一般區段

Net.Data 起始設定、架構與巨集檔案的某些部份須是一致的，如此 Net.Data 的所有元件方可以整體方式運作。下列表格彙總必須相配的這些檔案的區域。

表 2. Net.Data 架構檔和巨集的一致性需求

檔案	一般區段	附註
Net.Data INI 檔案	環境陳述式	使用「現場連線」的語言環境必須在其環境陳述式中設定資料庫 cliette 名稱
	「現場連線」架構變數	使用「Net.Data 現場連線」時，請設定「現場連線」埠 DTW_CM_PORT。此變數值須符合「現場連線」架構檔中的 MAIN_PORT 值。
	快取架構變數	使用 Net.Data 快取時，可以選用性地包括埠號和機器名稱變數。這些值必須符合「快取管理程式」架構檔已使用的值。

表 2. Net.Data 架構檔和巨集的一致性需求 (繼續)

檔案	一般區段	附註
「現場連線」架構檔	Cliette 定義	每一個 cliette 定義必須符合 INI 檔案中對應的定義。 此外，MAIN_PORT 值須符合 INI 檔中的 DTW_CM_PORT 變數值。
「快取管理程式」架構檔	「快取管理程式」架構變數	使用 Net.Data 快取時，您可以選用性地包括埠號和機器名稱變數。這些值必須符合 INI 檔案中已使用的值。

下列片斷說明巨集、Net.Data 起始設定檔和「現場連線」架構檔之間的關係。兩個 cliette 是由巨集 (DTW_SQL:SAMPLE, DTW_SQL:CELDIAL) 使用，並存取兩個 DB2 資料庫，稱爲 SAMPLE 與 CELDIAL。 「現場連線」架構檔含有 cliette 名稱與定義。 Net.Data 起始設定檔案中的 ENVIRONMENT 陳述式代表 cliette 名稱。 LOGIN 和 PASSWORD 的值，是在「現場連線」架構檔中設定。

圖2 顯示巨集的片斷，它包含 @DTW_ASSIGN 陳述式，定義要用來存取資料庫的 cliette。

```
<3**********************************************>
<3** 這是 HTML 備註                              **>
<3** 使用 cliette DTW_SQL:SAMPLE 來存取        **>
<3** SAMPLE 資料庫                              **>
<3**********************************************>
@DTW_ASSIGN (DATABASE, " SAMPLE ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)


<3**********************************************>
<3** 這是 HTML 備註                              **>
<3** 使用 cliette DTW_SQL:CELDIAL               **>
<3** 來處理 CELDIAL 資料庫                       **>
<3**********************************************>
@DTW_ASSIGN (DATABASE, " CELDIAL ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)
```

圖 2. Net.Data 巨集片斷

請注意，DATABASE 架構變數將替代起始設定檔案的 ENVIRONMENT，來產生 cliette 名稱。這容許您從同一巨集中存取多個資料庫。

第9頁的圖3 顯示 Net.Data 起始設定檔案中含有 ENVIRONMENT 陳述式與相關聯的 cliette 類型的片斷。在起始設定檔中，每一個 cliette 類型都有一個 ENVIRONMENT 陳述式。對於每一個資料庫 cliette 類型而言， ENVIRONMENT 陳述式設定一個 cliette 名稱。名稱是由 cliette 類型及在執行時將被解析的變數參照 $(DATABASE) 所構成。

使用「現場連線」的每一個語言環境，在 ENVIRONMENT 陳述式中必須包括 cliette 定義。

```
ENVIRONMENT (DTW_SQL)
    (IN DATABASE, LOGIN, PASSWORD, TRANSACTION_SCOPE, SHOWSQL,
    ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
    CLIETTE "DTW_SQL:$(DATABASE)"
```

圖 3. Net.Data 起始設定檔片斷

第10頁的圖4 顯示「現場連線」架構檔中含有 DTW_SQL:SAMPLE 與 DTW_SQL:CELDIAL 的 cliette 定義的片斷。

```
CONNECTION_MANAGER{
      MAIN_PORT=7128
      ADMIN_PORT1=7131
      ADMIN_PORT2=7133
      ENCRYPTION=OFF
      }

      ##################################################################
      # 這是現場連線的備註。
      # 備註以 # 字元開始，
      # 在行尾結束，除非下一行
      # 另設定 # 字元，否則不換行。
      # 您可以在行尾加備註，包含
      # 現場連線關鍵字，但不可在密碼行尾加備註。
      # 您無法在任何含有密碼關鍵字
      # 的行內加備註。
      # 您不能在任何名稱，如 cliette 名稱或資料庫 cliette
      # 密碼內加空格與 # 字元。
     ##################################################################
      CLIETTE  DTW_SQL:SAMPLE{
      MIN_PROCESS=1
      MAX_PROCESS=3
      START_PRIVATE_PORT=7100
      START_PUBLIC_PORT=7300
      EXEC_NAME=dtwcdb2.exe
      DATABASE=SAMPLE
            LOGIN=USER1
      PASSWORD=HAMLET
      }

      CLIETTE  DTW_SQL:CELDIAL{
      MIN_PROCESS=1
      MAX_PROCESS=5
      START_PRIVATE_PORT=7500
      START_PUBLIC_PORT=7700
      EXEC_NAME=dtwcdb2.exe
      DATABASE=CELDIAL
      LOGIN=USER2
      PASSWORD=OPHELIA
      }
```

圖 4. 「現場連線」架構檔片斷

自行設定 Net.Data 起始設定檔

起始設定檔所包含的資訊是使用三種架構陳述式設定，請參閱下列區段中的說明：

- 『架構變數陳述式』
- 第16頁的『路徑架構陳述式』
- 第20頁的『環境架構陳述式』

圖5中顯示的起始設定檔案範例，含有這些陳述式的例子，且對 OS/2 與 Windows NT 有效。

每一個別架構陳述式的文字均須在同一行上。 (爲了便於閱讀，ENVIRONMENT 陳述式會顯示在多行上。) 確定起始設定檔案對您可以從巨集呼叫的每一語言環境均含有 ENVIRONMENT 陳述式。如果您已完整限定巨集內的所有檔案參照，您就不需要設定任何路徑架構陳述式。

```
1  DTW_CM_PORT 7128
2  DTW_INST_DIR c:\db2www
3  DTW_LOG_DIR c:\db2www\logs
4  DB2INSTANCE DB2
5  DTW_DIRECT_REQUEST NO
6  DTW_SHOWSQL NO
7  MACRO_PATH c:\DB2WWW\Macro
8  HTML_PATH  c:\www\html
9  INCLUDE_PATH c:\db2www\Macro
10 EXEC_PATH c:\db2www\Macro
11 FFI_PATH c:\pub\ffi;pub\ffi\data
12 ENVIRONMENT (DTW_SQL)          [DLL 路徑]   [參數列示]
13 ENVIRONMENT (DTW_SYB)          [DLL 路徑]   [參數列示]
14 ENVIRONMENT (DTW_ORA)          [DLL 路徑]   [參數列示]
15 ENVIRONMENT (DTW_ODBC)         [DLL 路徑]   [參數列示]
16 ENVIRONMENT (DTW_DEFAULT)      [DLL 路徑]   [參數列示]
17 ENVIRONMENT (DTW_APPLET)       [DLL 路徑]   [參數列示]
18 ENVIRONMENT (DTW_REXX)         [DLL 路徑]   [參數列示]
19 ENVIRONMENT (DTW_PERL)         [DLL 路徑]   [參數列示]
20 ENVIRONMENT (DTW_SYSTEM)       [DLL 路徑]   [參數列示]
21 ENVIRONMENT (DTW_FILE)         [DLL 路徑]   [參數列示]
22 ENVIRONMENT (DTW_WEBREG)       [DLL 路徑]   [參數列示]
23 ENVIRONMENT (DTW_JAVAPPS)      [DLL 路徑]   [參數列示]
24 ENVIRONMENT (HWS_LE)           [DLL 路徑]   [參數列示]
```

- 第 1 - 6 行定義架構變數
- 第 7 - 11 行定義處理巨集時所需檔案的路徑
- 第 12 - 24 行定義可用的環境陳述式。

圖 5. Net.Data 起始設定檔

下列段落描述如何，以及如何自行設定自行設定檔中的架構陳述式。

架構變數陳述式

Net.Data 架構變數陳述式設定了架構變數的值。架構變數有各種不同的用途。 有些變數是語言環境適當運作或在替代模態中操作所必要的。有些變數控制字元編碼或建構中 Web 首頁的內容。此外，您可以使用架構變數陳述式來定義應用程式特定變數。

您使用的架構變數視您使用的語言環境、資料庫而定，同時也視僅跟應用程式有關的其他因素而定。

更新架構變數陳述式：

以您應用程式所需的架構變數來自行設定起始設定檔。架構變數具有下列語法：

NAME[=]*value-string*

等號是可選用的，以方括弧（[）表示。

下列子段落描述您可以在起始設定檔案中設定的架構變數陳述式：

- 『「快取管理程式」架構變數』
- 第13頁的『DB2INSTANCE: DB2 案例變數』
- 第13頁的『DTW_CM_PORT: 「現場連線」埠號變數』
- 第13頁的『DTW_DIRECT_REQUEST：啓用直接要求變數』
- 第14頁的『DTW_INST_DIR: Net.Data 安裝目錄變數』
- 第14頁的『DTW_LOG_DIR: 錯誤日誌位置變數』
- 第14頁的『DTW_MBMODE: 原始語言支援變數』
- 第15頁的『DTW_SHOWSQL：啓用或停用 SHOWSQL 架構變數』
- 第15頁的『DTW_SMTP_SERVER: 電子郵件 SMTP 伺服器變數』
- 第15頁的『DTW_UNICODE：Unicode 變數』
- 第16頁的『DTW_VARIABLE_SCOPE：變數範圍變數』

「快取管理程式」架構變數

如果「快取管理程式」不是在 Net.Data 巨集執行的機器上執行，則使用兩個可選用的架構變數：

- DTW_CACHE_PORT 可指定 Net.Data 使用哪一個埠號來連接到「快取管理程式」。
- DTW_CACHE_HOST 可指定本端或遠端機器的 TCP/IP 主電腦名稱。

如果「快取管理程式」在本端機器上執行，則使用 UNIX 領域 socket 或具名管線來通信，不需要架構。

「快取管理程式」僅會在 AIX 與 Windows NT 機器上執行。請參閱 第153頁的『Net.Data 快取』 以了解關於 Net.Data 快取。

DTW_CACHE_PORT：快取管理程式埠變數

設定「快取管理程式」監聽的 TCP/IP 埠。這個埠號必須符合「快取管理程式」架構檔中所設定的埠號， Net.Data 才能夠和「快取管理程式」通信。若未設定，「快取管理程式」將使用預設埠 7175。

語法：

```
DTW_CACHE_PORT=port_number
```

參數：

port_number

設定給「快取管理程式」來服務快取要求的獨一無二的埠號。預設值是 7175。

表3 說明選項，對這些變數設定機器 ID 和埠號。

表 3. 「快取管理程式」架構變數：架構選項

預設「連線管理程式」值	如果有設定快取機器...	如果未設定快取機器...

表 3. 「快取管理程式」架構變數：架構選項 (繼續)

如果有設定快取埠...	Net.Data 使用設定的埠連接到設定機器上的「快取管理程式」。	Net.Data 使用設定的埠連接到本端機器上的「快取管理程式」。
如果未設定快取埠...	Net.Data 使用預設埠 7175 連接到設定機器上的「快取管理程式」。	Net.Data 使用預設埠 7175 連接到本端機器上的「快取管理程式」。

DTW_CACHE_HOST：快取管理程式機器 ID 變數

設定「快取管理程式」常駐的機器。若未設定， Net.Data 將假定正確機器爲區域機器。

語法：

```
DTW_CACHE_HOST=host_name
```

參數：

host_name

「快取管理程式」執行所在的區域或遠端機器的完整 TCP/IP 主電腦名稱。預設值爲區域機器的主電腦名稱。

DB2INSTANCE: DB2 案例變數

設定 SQL 語言環境使用的 DB2 案例。當 Net.Data 連接到在 Windows NT、OS/2、和 UNIX 作業系統上執行的 DB2 時，這個變數是必要的。

OS/2, Windows NT 與 UNIX 作業系統上的 DB2 需要 DB2INSTANCE 定義爲一個環境變數。如果 Net.Data 偵測到 DB2INSTANCE 未定義成環境變數，在嘗試連接到 DB2 之前，它會將 DB2INSTANCE 環境變數設定爲起始設定檔案中的 DB2INSTANCE 值。

語法：

```
DB2INSTANCE instance_name
```

DTW_CM_PORT: 「現場連線」埠號變數

設定 Net.Data 用於「現場連線」的獨一無二埠號。

語法：

```
DTW_CM_PORT port_number
```

其中 *port_number* 設定用於「現場連線」的獨一無二埠號。

DTW_DIRECT_REQUEST：啓用直接要求變數

啓用或停用 Net.Data 直接要求呼叫。依據預設值，將停用直接要求。

呼叫 Net.Data 的直接要求方法容許使用者指定直接在 URL 內執行 SQL 陳述式或 Perl、 REXX 或 C 程式。當停用直接要求時，使用者必須使用巨集要求方法來 Net.Data，如此容許使用者僅執行在巨集中定義或呼叫的那些 SQL 陳述式及函數。請參閱第54頁的『使用 Net.Data 機制』，取得何時使用 DTW_DIRECT_REQUEST 的安全相關建議。

語法：

```
DTW_DIRECT_REQUEST YES|NO
```

其中：

YES 啓用 Net.Data 直接要求。

NO 停用 Net.Data 直接要求。NO 是預設值。

DTW_INST_DIR: Net.Data 安裝目錄變數

在 Net.Data 執行期間，找出某些檔案。安裝時，您會設定此變數來設定 Net.Data 起始目錄 (<*inst_dir*>)， Net.Data 將安裝在這個目錄中。 安裝之後請勿變更這個值。

DTW_LOG_DIR: 錯誤日誌位置變數

指定將儲存錯誤日誌的目錄。在巨集中以 DTW_LOG_LEVEL 變數啓用記錄之後，日誌檔會儲存在 DTW_LOG_DIR 變數的路徑陳述式所設定的目錄中。預設值爲 *inst_dir*\logs\netdata.logs。請參閱第175頁的『記錄 Net.Data 錯誤訊息』 以了解關於以 Net.Data 記錄錯誤訊息以及關於 DTW_LOG_LEVEL 變數。

需求： DTW_LOG_DIR 變數必須定義以使 Net.Data 記錄檔案。若未定義，則即使在巨集中 DTW_LOG_LEVEL 已設定爲 ERROR 或 WARNING，也不會執行記錄。

語法：

```
DTW_LOG_DIR \inst_dir\路徑
```

範例： 起始設定檔案架構

```
DTW_LOG_DIR \inst_dir\mylogfiles\
```

DTW_MBMODE: 原始語言支援變數

對於字組和字串函數啓動國家語言支援。當這個變數的值是 YES，所有字串和字組函數可正確處理字串內的 DBCS 字元，因爲它把字串視爲混合資料 (也就是說，視爲可能包含單位元組字集和雙位元組字集的字元的字串)。預設值是 NO。您可以置換掉設定在起始設定檔中的值，方法是在 Net.Data 巨集中設定 DTW_MBMODE 變數。

這個架構變數會使用 DTW_UNICODE 架構變數。如果 DTW_UNICODE 使用預設值 NO，將使用 DTW_MBMODE 的值。 如果 DTW_UNICODE 設定爲非 NO 的值，將使用它的值。 表4描述這兩個變數的設定如何決定內建函數處理字串的方式：

表 4. DTW_UNICODE 與 DTW_MBMODE 的設定之間的關係

如果 DTW_UNICODE 設定為	如果 DTW_MBMODE=YES	如果 DTW_MBMODE=NO
NO	支援混合 SBCS 的 DBCS	僅支援 SBCS
UTF8	支援 UTF-8	支援 UTF-8

語法：

```
DTW_MBMODE [=] NO|YES
```

DTW_SHOWSQL：啓用或停用 SHOWSQL 架構變數

置換您的 Net.Data 巨集內的設定 SHOWSQL 的效果。

語法：

```
DTW_SHOWSQL YES|NO
```

其中：

YES 啓用任何巨集中將 SHOWSQL 的值設定爲 YES 的 SHOWSQL。

NO 停用您的巨集中的 SHOWSQL，即使變數 SHOWSQL 設定爲 YES，也是如此。NO 是預設值。

表5描述 Net.Data 起始設定檔及巨集中的設定如何決定是否要對特殊巨集啓用或停用 SHOWSQL 變數。

表 5. SHOWSQL 的 Net.Data 起始設定檔及巨集中的設定之間的關係

DTW_SHOWSQL 的設定	設定 SHOWSQL	顯示 SQL 陳述式
NO	NO	NO
NO	YES	NO
YES	NO	NO
YES	YES	YES

DTW_SMTP_SERVER: 電子郵件 SMTP 伺服器變數

指定使用 DTW_SENDMAIL 內建函數送出電子郵件訊息時使用 SMTP 伺服器。這個變數的值可以是主電腦名稱或是 IP 位址。如果這個變數沒有被定義，Net.Data 將使用區域主電腦作爲 SMTP 伺服器。

語法：

```
DTW_SMTP_SERVER server_name
```

其中 server_name 指傳送電子郵件訊息時將使用的 SMTP 伺服器的主電腦名稱或 IP 位址。

效能要訣：指定這個值的 IP 位址，將防止當取回指定 SMTP 伺服器的 IP 位址時，Net.Data 連接到領域名稱伺服器。

範例：

```
DTW_SMTP_SERVER us.ibm.com
```

DTW_UNICODE：Unicode 變數

指定 Net.Data 是否支援下列中的 Unicode：

- 巨集
- 套表資料
- 從 DB2 資料庫取回的資料
- Net.Data 內建函數處理的字串

Net.Data 支援巨集、套表資料及內建函數中的 UTF-8 Unicode 格式，且輸出格式恆是 UTF-8。Net.Data 可以存取含有 UTF-16 資料的資料庫，且會將它轉換爲 UTF-8。

當設定爲 UTF8，DTW_UNICODE 會告訴 Net.Data 要在 Unicode 環境中執行，這表示它預期巨集檔資料、套表資料、來自瀏覽器的輸入及來自 DB2 資料庫的資料，具有 UTF-8 格式。設定這個變數將告訴 Net.Data，巨集輸入及輸出具有 UTF-8 格式；因此，Net.Data 將建立 UTF-8 格式的 Web 網頁。

這個架構變數會使用 DTW_MBMODE 架構變數。當處理字組及字串內建函數時，DTW_UNICODE 架構變數的值會置換 DTW_MBMODE 變數的設定。如果 DTW_UNICODE 使用預設值 NO，將使用 DTW_MBMODE 的值。如果 DTW_UNICODE 設定爲非 NO 的值，將使用它的值。 表6 描述這兩個變數的設定如何決定內建函數處理字串的方式：

表 6. DTW_UNICODE 與 DTW_MBMODE 的設定之間的關係

如果 DTW_UNICODE 設定為	如果 DTW_MBMODE=YES	如果 DTW_MBMODE=NO
NO	支援混合 SBCS 的 DBCS	僅支援 SBCS
UTF8	支援 UTF-8	支援 UTF-8

語法：

```
DTW_UNICODE NO|UTF8|UTF16
```

其中：

NO 指定將服從 DTW_MBMODE 變數的值。表6 會依據 DTW_MBMODE 的值來描述 Net.Data 支援。

UTF-8 指定要支援 UTF-8 字碼頁，並忽略 DTW_MBMODE 架構變數的值。 UTF-8 以可變的位元組數代表字元，且在 ASCII 模式顯示時也不會有問題。

DTW_VARIABLE_SCOPE：變數範圍變數

指定 Net.Data 如何對待區域變數範圍：區域變數是否仍是區域，或區域變數是否可在建立它們的函數區塊外使用。提供這個變數的目的在於能夠與先前版本的 Net.Data 相容。

語法：

```
DTW_VARIABLE_SCOPE = LOCAL|GLOBAL
```

其中：

LOCAL
指定區域變數仍是區域。這個行爲是在 Net.Data 版本 2.0 引進的，且是預設值。

GLOBAL
指定區域變數可在建立它們的函數區塊外使用。這個行爲在 Net.Data 版本 2 之前即是預設值，而且指定這個值就不需要變更巨集檔。

路徑架構陳述式

Net.Data 根據路徑架構陳述式的設定，來決定 Net.Data 巨集所使用之檔案和可執行程式的位置。路徑陳述式如下：

- 『MACRO_PATH』
- 第18頁的『EXEC_PATH』
- 第19頁的『INCLUDE_PATH』
- 第20頁的『FFI_PATH』
- 第20頁的『HTML_PATH』

這些路徑陳述式識別 Net.Data 在嘗試尋找下列檔案時所搜尋的一個或多個目錄：巨集、可執行檔、文字檔、LOB 檔以及併入檔。 您需要的路徑陳述式視巨集使用的 Net.Data 功能而定。

更新準則：

許多一般準則適用於路徑陳述式。例外狀況將會在每一路徑陳述式的說明中提到。

- 每一個指定的目錄都是以分號 (;) 作爲結尾。
- 斜線 (/) 與反斜線 (\) 則視爲一樣。
- 每一個路徑陳述式可以設定多重路徑，但 HTML_PATH 除外，它只能有一個路徑陳述式。路徑是從左到右依設定次序搜尋。此種多路徑功能，可讓您在多重目錄中組織檔案。例如，您可以將每一個 Web 應用程式於在自己的目錄中。
- 建議使用絕對路徑陳述式。

下列幾節說明每一個路徑陳述式的目的和語法，並提供有效路徑陳述式的範例。範例可能與您的應用程式不同，需根據您的作業系統和架構而定。

MACRO_PATH

MACRO_PATH 架構陳述式識別 Net.Data 搜尋 Net.Data 巨集的目錄。例如，設定下列 URL 是將要求具有路徑和檔案名稱爲 `/macro/sqlm.d2w` 的 Net.Data 巨集：

```
http://server/cgi-bin/db2www/macro/sqlm.d2w/report
```

語法：

```
MACRO_PATH [=] path1;path2;...;pathn
```

等號 (=) 是可選用性的，由方括弧（[）表示。

Net.Data 會將路徑 `/macro/sqlm.d2w` 附加到 MACRO_PATH 架構陳述式的路徑中 (從左到右)，直到 Net.Data 找到巨集或搜尋過所有路徑爲止。 關於呼叫 Net.Data 巨集的資訊，請參閱第59頁的『第4章 呼叫 Net.Data』。

範例： 下列範例顯示在起始設定檔中的 MACRO PATH 陳述式以及呼叫 Net.Data 的相關鏈結。

Net.Data 起始設定檔：

```
MACRO_PATH /u/user1/macros;/usr/lpp/netdata/macros;
```

HTML 鏈結：

```
<A HREF="http://server/cgi-bin/db2www/query.d2w/input">Submit another query.</A>
```

如果在目錄 `/u/user1/macros` 中找到檔案 *query.d2w*，則完整的路徑爲 `/u/user1/macros/query.d2w`。

如果在 MACRO_PATH 陳述式中指定的目錄中找不到檔案：

- 如果指定的路徑是絕對的，Net.Data 將在指定的路徑中搜尋檔案。 例如，如果提出下列的 URL：

  ```
  http://server/cgi-bin/db2www/u/user1/macros/myfile.txt/report
  ```

  Net.Data 將在 /u/user1/macros/myfile.txt 目錄路徑中搜尋檔案。
- 如果指定的路徑是相對的，Net.Data 將在所有目錄中搜尋檔案，從根 (/) 目錄開始。例如，如果提出下列的 URL：

  ```
  http://server/cgi-bin/db2www/myfile.txt/report
  ```

  且在 MACRO_PATH 中設定的任何目錄中找不到檔案 myfile.txt， Net.Data 將嘗試在根 (/) 目錄中尋找檔案：/myfile.txt

EXEC_PATH

EXEC_PATH 架構陳述式會識別一個或多個目錄，Net.Data 將搜尋它 (它們) 來取得 EXEC 陳述式或可執行變數呼叫的外部程式。路徑陳述式中的目錄次序決定 Net.Data 搜尋目錄的次序。 一旦找到該程式，則會將外部程式的名稱附加到路徑設定中，形成完整的檔名以便傳遞給語言環境準備執行。

語法：

```
EXEC_PATH [=] path1;path2;...;pathn
```

範例： 下列範例顯示在起始設定檔中的 EXEC PATH 陳述式，以及巨集中呼叫外部程式的 EXEC 陳述式。

Net.Data 起始設定檔：

```
EXEC_PATH /u/user1/prgms;/usr/lpp/netdata/prgms;
```

Net.Data 巨集：

```
%FUNCTION(DTW_REXX) myFunction() {
  %EXEC{ myFunction.cmd %}
%}
```

如果在 /usr/lpp/netdata/prgms 目錄中找到檔案 myFunction.cmd，則程式的完整名稱爲 /usr/lpp/netdata/prgms/myFunction.cmd。

如果在 EXEC_PATH 陳述式中設定的目錄中找不到檔案：

- 如果指定的路徑是絕對的，Net.Data 將在指定的路徑中搜尋檔案。 例如，如果提出下列的 URL：

  ```
  http://myserver/cgi-bin/db2www/usr/user1/prgms/myFunction.cmd
  ```

  Net.Data 將在 /u/user1/prgms/myFunction.cmd 目錄路徑中搜尋檔案。
- 如果設定的路徑是相對的，Net.Data 將搜尋現行工作目錄。 例如，如果提出下列的 URL：

  ```
  http://myserver/cgi-bin/db2www/myFunction.cmd/report
  ```

  且在 EXEC_PATH 中設定的任何目錄中找不到檔案 myFunction.cmd， Net.Data 將嘗試在現行工作目錄中尋找檔案。

INCLUDE_PATH

INCLUDE_PATH 架構陳述式定義 Net.Data 搜尋的一或多個目錄， (按您所設定的次序)，以便找出 Net.Data 巨集中 INCLUDE 陳述式所設定的檔案。一旦找到該檔，Net.Data 會將此併入檔名稱附加在路徑規格中，以產生完整的併入檔名稱。

語法：

```
INCLUDE_PATH [=] path1;path2;...;pathn
```

要訣： 如果要從區域 Web 伺服器中併入 HTML 檔，則使用 *Net.Data 參考手冊*中關於 INCLUDE_URL 的區域 Web 伺服器範例中所顯示的 INCLUDE_URL 結構。使用所示範的語法，您不需更新 INCLUDE_PATH 即可設定 Web 伺服器已知的目錄。

範例 1： 下列範例顯示起始設定檔中的 INCLUDE_PATH 陳述式，以及設定併入檔的 INCLUDE 陳述式。

Net.Data 起始設定檔：

```
INCLUDE_PATH /u/user1/includes;/usr/lpp/netdata/includes;
```

Net.Data 巨集：

```
%INCLUDE "myInclude.txt"
```

如果在 /u/user1/includes 目錄中找到了檔案 *myInclude.txt*，則併入檔的完整名稱爲 /u/user1/includes/myInclude.txt。

範例 2：下列範例顯示 INCLUDE_PATH 陳述式以及具有次目錄名稱的 INCLUDE 檔。

Net.Data 起始設定檔：

```
INCLUDE_PATH /u/user1/includes;/usr/lpp/netdata/includes;
```

Net.Data 巨集：

```
%INCLUDE "OE/oeheader.inc"
```

將在目錄 /u/user1/includes/OE 與 /usr/lpp/netdata/includes/OE 中搜尋併入檔。如果在 /usr/lpp/netdata/includes/OE 找到此檔案，則併入檔的完整名稱是 /usr/lpp/netdata/includes/OE/oeheader.inc。

如果在 INCLUDE_PATH 陳述式中設定的目錄中找不到檔案：

- 如果指定的路徑是絕對的，Net.Data 將在指定的路徑中搜尋檔案。 例如，如果提出下列的 URL：

  ```
  http://myserver/cgi-bin/db2www/u/user1/includes/oeheader.inc
  ```

  Net.Data 將在 /u/user1/includes/oeheader.inc 目錄路徑中搜尋檔案。
- 如果設定的路徑是相對的，Net.Data 將搜尋現行工作目錄。 例如，如果提出下列的 URL：

  ```
  http://myserver/cgi-bin/db2www/my.cmd/report
  ```

  且在 INCLUDE_PATH 中設定的任何目錄中找不到檔案 myFunction.cmd， Net.Data 將嘗試在現行工作目錄中尋找檔案。

FFI_PATH

FFI_PATH 架構陳述式定義 Net.Data 搜尋的一或多個目錄， (按您所設定的次序)，以便找出被純本文檔介面 (FFI) 函數參考到的純本文檔。

語法：

```
FFI_PATH [=] path1;path2;...;pathn
```

範例： 下列範例顯示在起始設定檔中的 FFI_PATH 陳述式。

Net.Data 起始設定檔：

```
FFI_PATH /u/user1/ffi;/usr/lpp/netdata/ffi;
```

當呼叫 FFI 語言環境時，Net.Data 會尋找 FFI_PATH 陳述式設定的路徑。

因爲 FFI_PATH 陳述式是用來對不在路徑陳述式的目錄中的那些檔案提供安全保護，所以對找不到的 FFI 檔案將有特別的規定。請參閱 *Net.Data 參考手冊*中 FFI 內建函數段落。

HTML_PATH

HTML_PATH 架構陳述式會設定一個目錄，讓 Net.Data 將大型物件 (LOB) 寫入其中。這個路徑陳述式只接受一個目錄路徑。

在安裝期間，Net.Data 會建立一個名爲 tmplobs 的目錄，這個目錄位在 HTML_PATH 路徑架構變數中指定的目錄下。Net.Data 會在這個目錄中儲存所有 LOB 檔。如果您變更 HTML_PATH 的值，請在新目錄下建立新的次目錄。

語法：

```
HTML_PATH [=] path
```

範例： 下列範例顯示在起始設定檔中的 HTML PATH 陳述式。

Net.Data 起始設定檔：

```
HTML_PATH /db2/lobs
```

當查詢傳回一個 LOB 時， Net.Data 會將之儲存到 HTML_PATH 架構陳述式所設定的目錄中。

執行效能要訣： 由於 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮到系統限制問題。詳細資訊，請參閱第118頁的『使用大型物件』。

環境架構陳述式

ENVIRONMENT 陳述式旨在架構語言環境。語言環境是 Net.Data 的構成要素之一，Net.Data 用它來存取資料來源 (如：DB2 資料庫) 或執行以 REXX 之類語言撰寫的程式。 Net.Data 提供一組語言環境，以及可讓您建立自己語言環境的介面。這些語言環境會在第113頁的『第6章 使用語言環境』中加以描述，而語言環境介面則會在 *Net.Data 語言環境介面參考手冊*中加以描述。

Net.Data 需要特殊語言環境的 ENVIRONMENT 陳述式先存在，方可呼叫該語言環境。

您可以經由將變數指定爲 ENVIRONMENT 陳述式中的參數，使變數與語言環境產生關聯。 Net.Data 會隱含地將 ENVIRONMENT 陳述式上指定的參數傳遞給語言環境，作爲巨集變數。若要變更巨集中 ENVIRONMENT 陳述式上指定的參數值，請使用 DTW_ASSIGN() 函數指定變數值，或在 DEFINE 區段中定義變數。**重要事項**： 如果有一個巨集變數定義在巨集中，但未在 ENVIRONMENT 陳述式上指定它，則巨集變數將不會傳遞給語言環境。

例如，巨集可定義 DATABASE 變數來設定資料庫名稱，該位置將執行 DTW_SQL 函數內的 SQL 陳述式。DATABASE 的值必須傳遞至 SQL 語言環境 (DTW_SQL)，這樣 SQL 語言環境可以連接至設定的 資料庫。要傳遞變數至語言環境，您必須新增 DATABASE 變數到 DTW_SQL 之環境陳述式的參數列示。

樣本 Net.Data 起始設定檔在自行設定 Net.Data 環境架構陳述式方面已有一些假設。這些假設不一定適合您的環境。修改陳述式使其適合您的環境。

新增或更新 ENVIRONMENT 陳述式：

ENVIRONMENT 陳述式具有下列語法：

`ENVIRONMENT(`*`type`*`) `*`library_name`*` (`*`parameter_list`*`, ...) [CLIETTE "`*`cliette_name`*`"]`

參數：

- *type*

  Net.Data 藉由此名稱使這個語言環境和 Net.Data 巨集中定義之 FUNCTION 區塊互相關聯。 您必須在 FUNCTION 區塊定義中設定語言環境的類型來識別 Net.Data 應使用來執行函數的語言環境。

- *library_name*

  含有 Net.Data 呼叫的語言環境介面的 DLL或共用程式庫的名稱。

  – 在 AIX 中，共用程式庫名稱是以 *.o* 副檔名來指定的。

  – 在 HP-UX 中，共用程式庫名稱是以 *.sl* 副檔名來指定的。

  – 在 SUN, SCO 及 LINUX 中，共用程式庫名稱是以 *.so* 副檔名來指定的。

  – 在 OS/2 及 Windows NT 中，於指定 DLL 名稱時，不用 *.dll* 副檔名。

- *parameter_list*

  除了在 FUNCTION 區塊定義中設定的參數之外，此參數列示還包含每一個函數呼叫時傳遞給語言環境的參數。

  若要設定及傳遞參數列示中的變數，請在巨集中定義變數。

  在執行由語言環境所處理的函數之前，您必須先將這些參數定義爲架構變數，或定義爲巨集中的變數。如果函數修改其中一個輸出參數，則函數完成之後參數會保留修改過的值。 下列列示會指定 ENVIRONMENT 陳述式可傳遞哪些變數：

  DTW_SQL: TRANSACTION_SCOPE, LOCATION, DB2SSID, DB2PLAN

  DTW_ODBC: TRANSACTION_SCOPE, LOCATION

- *cliette_name*

  cliette 的名稱。 *cliette_ name* 可以指「Java 應用程式」語言環境 cliette，或它可以是資料庫 cliette。*cliette_name* 參數是與 CLIETTE 關鍵字一起使用，且兩者僅能與「現場連線」一起使用。`CLIETTE` 與 *`cliette_name`* 是可選用的，且僅能對資料庫與 Java 應用程式語言環境設定它，以及對具有針對它們撰寫的 cliette 的使用者定義的語言環境設定它。

Java 應用程式 cliette

這個 cliette 名稱設定 Java 應用程式語言環境。

語法：

```
CLIETTE "DTW_JAVAPPS"
```

資料庫 cliette

這個 cliette 名稱設定與資料庫相關聯的 cliette。

語法：

```
CLIETTE "type:db_name"
```

參數：

type 與 cliette 相關聯的資料庫語言環境。請參閱第 46 頁，取得有效類型列示。

db_name

資料庫 cliette 名稱。這個名稱通常與 cliette 相關的資料庫同名，例如 MYDBASE，但也可以是其他名稱。 當使用 Oracle 語言環境時，*db_name* 是可選用的。

當 Net.Data 處理起始設定檔時，它不會載入語言環境 DLL 或共用程式庫。當 Net.Data 第一次執行指明語言環境的函數時，它即會載入該語言環境的 DLL 或共用程式庫。只要載入 Net.Data，即會持續載入 DLL 或共用程式庫。

範例： 供 Net.Data 提供的語言環境所用之 ENVIRONMENT 陳述式。

針對您的應用程式自行設定 ENVIRONMENT 陳述式時，請在 ENVIRONMENT 陳述式上新增變數，這些變數需要從起始設定檔中傳送到語言環境，或爲 Net.Data 巨集撰寫者在其巨集內所需設定或置換的。

```
ENVIRONMENT (DTW_SQL)      DTWSQL    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
  CLIETTE "DTW_SQL:MYDBASE"
ENVIRONMENT (DTW_SYB)      DTWSYB    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_ORA)      DTWORA    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_ODBC)     DTWODBC   ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_APPLET)   DTWJAVA   ( )
ENVIRONMENT (DTW_JAVAPPS)  DTWJAVAPPS ( OUT RETURN_CODE ) CLIETTE "DTW_JAVAPPS"
ENVIRONMENT (DTW_PERL)     DTWPERL   ( OUT RETURN_CODE )
ENVIRONMENT (DTW_REXX)     DTWREXX   ( OUT RETURN_CODE )
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ( OUT RETURN_CODE )
ENVIRONMENT (HWS_LE)       DTWHWS    ( OUT RETURN_CODE )
```

要求每一 ENVIRONMENT 陳述式須在單一行上。

設置語言環境

在您爲 Net.Data 語言環境修改架構變數和 ENVIRONMENT 架構陳述式之後，需要一些額外的設定，才能使下列語言環境的功能正常。下列幾節將描述設置語言環境所需的步驟：

- 第23頁的『設置 IMS Web 語言環境』

- 『設置 Java 語言環境』
- 第24頁的『設置 Oracle 語言環境』
- 第26頁的『設置 Sybase 語言環境』

設置 IMS Web 語言環境

若要使用 IMS Web 語言環境，您必須完成下列步驟：

1. 在執行 Net.Data 的 Web 伺服器上安裝 IMS Web Runtime 元件。關於如何安裝及使用 IMS Web Runtime 元件的資訊，請參閱 *IMS Web User's Guide*：

   `http://www.software.ibm.com/data/ims/about/imsweb/document/`

2. 建立異動 DLL。
   a. 使用 IMS Web Studio 工具，依據您的交易的，建立 C++ 程式碼、makefile (DTW*proj*.mak) 及 Net.Data 巨集 (DTW*proj*.d2w) 檔。
   b. 如果執行 Net.Data 的作業系統不同於執行 IMS Web Studio 工具的作業系統，請將 DLL 原始檔案移到目標作業系統的 IMS Web 開發機器。例如，如果您想要在 Windows NT 上執行 IMS Web Studio 工具，但目標平台爲 AIX 或 OS/390，請將異動 DLL 的原始檔案分別移到執行 AIX 或 OS/390 的 IMS Web 開發機器。
   c. 使用建立的 make 檔，建置異動 DLL 的可執行格式。
3. 將異動 DLL 檔 (DTW*proj*.dll) 及 Net.Data 巨集檔 (DTW*proj*.d2w) 複製到您的 Web 伺服器。
   a. 將巨集放在 Net.Data 將從其中取回巨集的目錄中。(詳細資訊，請參閱第17頁的『MACRO_PATH』。)
   b. 將異動 DLL 或共用程式庫放在 Web 伺服器將從其中取回 DLL 或共用程式庫的目錄中。.
4. 使用 IMS Web Studio 工具所建立的樣本檔案 (DTW*proj*.htm) 中的鏈結，來修改您的 Web 伺服器的 HTML 樹狀結構中的 HTML 檔。然後，您可以使用鏈結來呼叫 Net.Data，並顯示輸入 HTML 套表，來呼叫 IMS Web 語言環境。然後，IMS Web 語言環境會呼叫 IMS 異動 DLL，它會使用 IMS Web Runtime DLL 提供的服務程式，來執行交易並傳回它的輸出給 Web 瀏覽器。

   IMS Web Runtime DLL 會形成要求訊息，並透過 IMS TOC 將它傳送給 OTMA，因而使得適當的交易置於等待佇列中。然後，OTMA 會透過 IMS TOC 將交易的輸出傳回給 IMS Web。然後，交易 os 會透過 IMS Web 語言環境傳回到 Net.Data，並顯示在 Web 瀏覽器上。

設置 Java 語言環境

Java 語言環境需要有一些其他設置後，您才可以從巨集呼叫函數：

1. 建立一個批次檔來啓動 Java 應用程式，因爲 Net.Data 無法直接啓動 Java 應用程式。 Net.Data 將使用這個檔案來啓動「Java 虛擬機器」，它會執行您的 Java 函數。批次檔必須包括 java-classpath 陳述式，方可確保可以找到需要的 Java 套裝軟體 (標準及應用程式專用的套裝軟體)。例如，批次檔 `launchjv.bat` 含有下列 java-classpath：

   ```
   java -classpath %CLASSPATH%;C:\DB2WWW\Javaclas dtw_samp %1 %2 %3 %4 %5 %6
   ```

2. 定義一個 cliette 來使用「現場連線」架構檔 dtwcm.cnf 中的 Java 語言環境。以 EXEC_NAME 架構變數指定 cliette 的唯一埠號及相關的批次檔名稱。在下列範例中，Java cliette 名稱定義爲 DTW_JAVAPPS，且 EXEC_NAME 架構變數設定爲批次檔 launchjv.bat 的名稱：

```
CLIETTE DTW_JAVAPPS{
MIN_PROCESS=1                <= 必要的：這個值必須是 1，因為
                                JAVAPPS cliette 是多緒的。
MAX_PROCESS=1                <= 必要的：這個值必須是 1，因為
                                JAVAPPS cliette 是多緒的。
START_PRIVATE_PORT=5100      <= 須是唯一的埠號
START_PUBLIC_PORT=5300       <= 須是唯一的埠號
EXEC_NAME=launchjv.bat       <= 包括 classpath 陳述式的批次檔的名稱
}
```

當您啓動「Net.Data 連線管理程式」時，Net.Data 便會啓動架構檔中指定的 Java cliette。cliette 會變成可用，以便能夠處理來自您的 Net.Data 巨集應用程式的 Java 語言環境要求。

3. 更新 Net.Data 起始設定檔 db2www.ini 中的 DTW_JAVAPPS ENVIRONMENT 路徑陳述式，方法是新增每一個
4. cliette 名稱到陳述式。例如：

```
ENVIRONMENT DTW_JAVAPPS  ( OUT RETURN_CODE ) CLIETTE "DTW_JAVAPPS"
```

設置 Oracle 語言環境

使用下列步驟，從 Net.Data 巨集存取 Oracle 資料庫：

1. 確定 Oracle 適當元件已安裝且運作如下：
   a. 將 SQL*Net 安裝在 Net.Data 安裝所在的機器上，若尚未安裝它的話。詳細資訊，請參閱下列 URL：

      ```
      http://www.oracle.com/products/networking/html/stnd_sqlnet.html
      ```

   b. 驗證 Oracle *tnsping* 函數是否可使用您的 Web 使用的同一安全授權。若要驗證，請以 Web 伺服器使用者 ID 登入，並輸入：

      ```
      tnsping oracle-instance-name
      ```

      其中 *oracle-instance-name* 是您的 Net.Data 巨集將存取的 Oracle 系統的名稱。

      如果是在系統權限下執行您的 Web 伺服器，則您可能無法在 Windows NT 上驗證 *tnsping* 函數。若是如此，請跳過這個步驟。

   c. 驗證可用您的 Web 伺服器使用的同一安全授權來存取 Oracle 表格。若要驗證，請使用 SQL*Plus 行指令工具輸入 SQL SELECT 陳述式，以具有您的 Web 伺服器的權限的 SQL SELECT 陳述式，來存取 Oracle 表格。例如：

      ```
      SELECT * FROM tablename
      ```

      如果是在系統權限下執行您的 Web 伺服器，則您可能無法在 Windows NT 上驗證表格存取。 若是如此，請跳過這個步驟。

   疑難排解：如果上述步驟失敗，請不要繼續執行。如果任一步驟失敗，請檢查您的 Oracle 架構。

2. 確定已在您的 Web 伺服器處理中設定了正確的 Oracle 環境變數。
   - 對於 AIX，將下列幾行放在 /etc/environment 檔中：

```
ORACLE_SID=oracle-instance-name
ORACLE_HOME=oracle-runtime-library-directory
```

- 對於 Windows NT，請使用「系統內容控制」畫面，新增下列環境變數：

```
ORACLE_SID=oracle-instance-name
ORACLE_HOME=oracle-runtime-library-directory
```

提示：您可能需要額外的行來放置其他 Oracle 環境變數，取決於您計劃使用的 Oracle 機能，如國家語言支援及兩段式確定。請參閱 Oracle 管理文件，取得這些環境變數的詳細資訊。

3. 測試從 Net.Data 連接到 Oracle 的情況。在您的 Net.Data 巨集中，請在 LOGIN 及 PASSWORD 變數中指定適當的值。當存取 Oracle 資料庫時，請*不要*定義 Net.Data DATABASE 變數。 下列是巨集中連接陳述式的範例：

```
%DEFINE LOGIN=user_ID@remote-oracle-instance-name
%DEFINE PASSWORD=password
```

區域 Oracle 範例：

如果您僅存取區域 Oracle 範例，請不要指定 remote-oracle-instance 名稱，作爲登入使用者 ID 的一部份，範例如下：

```
%DEFINE LOGIN=user_ID
%DEFINE PASSWORD=password
```

現場連線

如果您使用「現場連線」，則您可以在「現場連線」架構檔中指定 LOGIN 及 PASSWORD，但基於安全目的，不建議您使用它。例如：

```
LOGIN=user_ID
PASSWORD=password
```

提示：不要對 Oracle 指定 DATABASE 變數。

4. 執行 CGI shell script 來測試您的架構，以確保可從您的 Web 伺服器存取 Oracle 範例，範例如下：

```
#! /bin/sh
echo "content-type; text/html
echo
echo "< html>< pre>"
set
echo "</pre>< p>< pre>"
tnsping oracle-instance-name
echo
```

另一種方式，您可以直接從 Net.Data 巨集執行 *tnsping*，範例如下：

```
%DEFINE testora = %exec "tnsping oracle-instance-name"
%HTML(report){
< P>將以 tnsping 測試 Oracle 存取。
<hr>
$(testora)
<hr>
< P>Oracle 測試已完成。
%}
```

疑難排解：

如果驗證步驟失敗，請驗證下列項目，檢查所有先前步驟是否已順利完成：

- 檢查您的 Oracle 架構。

- 驗證 Oracle 環境變數語法是否正確，且沒有遺失任何變數。
- 檢查 Oracle 連線，確定您已輸入正確的使用者 ID 和通行碼。

如果驗證步驟仍然失敗，請聯絡「IBM 服務人員」。

範例：

在您完成了存取驗證步驟後，您可以用巨集中的函數來呼叫 Oracle 語言環境，範例如下：

```
%FUNCTION(DTW_ORA) STL1() {
insert into $(tablename) (int1,int2) values (111,NULL)
%}
```

設置 Sybase 語言環境

若要從 Net.Data 存取 Sybase：

1. 確定 Sybase 適當元件已安裝且運作如下：
   a. 將 Sybase 的 Open Client 安裝在 Net.Data 安裝所在的機器上，若尚未安裝它的話。詳細資訊，請參閱 Sybase Open Client 文件。
   b. 驗證 Sybase *ping* 函數是否可使用您的 Web 使用的同一安全授權。若要驗證，請以 Web 伺服器使用者 ID 登入，並輸入：

      ```
      ping sybase-instance-name
      ```

      其中 *sybase-instance-name* 是您的 Net.Data 巨集將存取的 Sybase 系統的名稱。

      如果是在系統權限下執行您的 Web 伺服器，則您可能無法在 Windows NT 上驗證 *ping* 函數。若是如此，請跳過這個步驟。
   c. 驗證可用您的 Web 伺服器使用的同一安全授權來存取 Sybase 表格。若要驗證，請使用 SQL 行指令工具輸入 SQL SELECT 陳述式，以您的 Web 伺服器的權限存取 Sybase 表格。例如：

      ```
      SELECT * FROM tablename
      ```

      如果是在系統權限下執行您的 Web 伺服器，則您可能無法在 Windows NT 上驗證表格存取。若是如此，請跳過這個步驟。

   疑難排解：如果上述步驟失敗，請不要繼續執行。如果任一步驟失敗，請檢查您的 Sybase 架構。
2. 確定已在您的 Web 伺服器處理中設定了正確的 Sybase 環境變數。
   - 對於 AIX，將下列幾行放在 `/etc/environment` 檔中：

     ```
     DSQUERY=sybase-instance-name
     SYBASE=sybase-runtime-library-directory
     ```
   - 對於 Windows NT，請使用「系統內容控制」畫面，新增下列環境變數：

     ```
     DSQUERY=sybase-instance-name
     SYBASE=sybase-runtime-library-directory
     ```

   提示：您可能需要額外的行來放置其他 Sybase 環境變數，取決於您計劃使用的 Sybase 機能，如國家語言支援及兩段式確定。請參閱 Sybase 管理文件，取得這些環境變數的詳細資訊。

3. 測試從 Net.Data 連接到 Sybase 的情況。在您的 Net.Data 巨集中，請在 LOGIN、PASSWORD 及 DATABASE 變數中指定適當的值。下列是巨集中連接陳述式的範例：

```
%DEFINE DATABASE=database-name
%DEFINE LOGIN=user_ID@remote-sybase-instance-name
%DEFINE PASSWORD=password
```

「現場連線」：如果您使用「現場連線」，則您可以在「現場連線」架構檔中指定 LOGIN 及 PASSWORD，但基於安全目的，不建議您使用它。例如：

```
DATABASE=database-name
LOGIN=user_ID
PASSWORD=password
```

4. 執行 CGI shell script 來測試您的架構，以確保可從您的 Web 伺服器存取 Sybase 案例，範例如下：

```
#! /bin/sh
echo "content-type; text/html
echo
echo "< html>< pre>"
set
echo "</pre>< p>< pre>"
isql -u user_ID -p password << EOFF
SELECT * FROM tablename
EOFF
echo
```

疑難排解：

如果驗證步驟失敗，請驗證下列項目，檢查所有先前步驟是否已順利完成：

- 檢查您的 Sybase 架構。
- 驗證 Sybase 環境變數語法是否正確，且沒有遺失任何變數。
- 檢查 Sybase 連線，確定您已輸入正確的使用者 ID 和通行碼。

如果驗證步驟仍然失敗，請聯絡「IBM 服務人員」。

範例：

一旦您完成了存取驗證步驟後，您可以用巨集中的函數來呼叫 Sybase 語言環境，範例如下：

```
%function(DTW_SYB) STL1() {
insert into $(tablename) (int1,int2) values (111,NULL)
%}
```

架構「現場連線」

「現場連線」管理資料庫和 Java 應用程式連線，以改進 Net.Data 在 Windows NT、OS/2、 AIX 及 Sun Solaris 作業系統上的執行效能。經由使用「連線管理程式」和 cliette (亦即維護開放連線的處理)，「現場連線」可消除連接到資料庫或啓動「Java 虛擬機器」的額外啓動時間。

「現場連線」使用架構檔 dtwcm.cnf，來決定需啓動哪些 cliette。它包含了「現場連線」所用之每一個 cliette 的管理資訊和定義。有關「現場連線」的詳細資訊，請參閱第150頁的『管理連線』。

圖6 顯示的樣本架構檔，包含下列類型的資訊：

- 「連線管理程式」埠資訊
- DB2 連線的 SQL cliette 資訊
- Java 應用程式 cliette 資訊

```
 1  CONNECTION_MANAGER{
 2  MAIN_PORT=7100
 3  ADMIN_PORT1=7101
 4  ADMIN_PORT2=7102
 5  }
 6
 7  CLIETTE DTW_SQL:CELDIAL{
 8  MIN_PROCESS=1
 9  MAX_PROCESS=5
10  START_PRIVATE_PORT=7200
11  START_PUBLIC_PORT=7210
12  EXEC_NAME=./dtwcdb2
13  DATABASE=CELDIAL
14  LOGIN=marshall
15  PASSWORD=stlpwd
16  }
17
18  CLIETTE DTW_JAVAPPS{
19  MIN_PROCESS=1
20  MAX_PROCESS=5
21  START_PRIVATE_PORT=7300
22  START_PUBLIC_PORT=7310
23  EXEC_NAME=./javaapp
24  }
```

- 1 至 5 行是架構檔所必備的，定義著「現場連線」所用的獨一無二埠號。
- 第 7 - 16 行定義所有資料庫 cliette，識別 cliette 名稱、將執行的處理數目、資料庫名稱、埠號碼及 cliette 執行檔。您可以加入其他資訊，例如，如果連接 DB2 資料庫，則可加入使用者 ID 和通行碼。 這些額外之值會出現在 13 - 15 行。
- 第 19 - 25 行定義 Java 應用程式的所有 cliette，識別 cliette 名稱、將執行的處理數目、唯一的埠號碼及 cliette 執行檔。

圖 6. 「現場連線」架構檔

開始之前：在自行設定「現場連線」架構檔之前，請閱讀這些步驟後面的提示和秘訣區段。

若要架構「現場連線」埠：

1. 使用編輯程式來開啓架構檔 `dtwcm.cnf`。
2. 架構三個「現場連線」埠號：
   - MAIN_PORT
   - ADMIN_PORT1
   - ADMIN_PORT2

   圖6 顯示預設埠號。若這些號碼不是獨一無二的，則必須變更爲獨一無二的埠號。
3. **重要事項**：確定 MAIN_PORT 的值符合 Net.Data 起始設定檔案中 DTW_CM_PORT 的值。

若要架構資料庫 cliette：

1. 鍵入 cliette 環境陳述式。

   `CLIETTE` *`type`*`:`*`db_name`*

   參數：

   type 連結語言環境和 cliette 所用的名稱。關於有效類型的列示，請參閱 46。

db_name

資料庫 cliette 名稱通常與 cliette 相關的資料庫同名，例如 MYDBASE，但也可以是其他名稱。 當使用 Oracle 語言環境時， *db_name* 是可選用的。

2. 決定 MIN _PROCESS 和 MAX_PROCESS 的值。 MIN_PROCESS 設定「連線管理程式」啓動時，所要啓動的處理數。之後，當同時存在的額外要求抵達時，「連線管理程式」會啓動更多的 cliette (若有需要的話)，直到到達 MAX_PROCESS 所指的上限值爲止。您所使用的值，可能會影響執行效能，但是您稍後可以再做變更。

   鍵入 MIN_PROCESS 和 MAX_PROCESS 陳述式：

   ```
   MIN_PROCESS=min_num
   MAX_PROCESS=max_num
   ```

   參數：

   min_num

   「連線管理程式」啓動時所要啓動的 cliette 數。您必須擁有足夠的獨一無二埠號以支援這麼多 cliette。

   max_num

   可同時執行的最大 cliette 數。您必須擁有足夠的獨一無二埠號以支援這麼多 cliette。

3. 決定在系統上資料庫 cliette 所要使用的埠號。這些號碼必須是獨一無二，以避免與「快取管理程式」或其他應用程式所用的埠號發生衝突。每一個 cliette 使用兩個埠。設定一組埠時，您必須設定要使用的埠號範圍。 前兩個值爲 START_PUBLIC_PORT 和 START_PRIVATE_PORT。另一個是 MAX_PROCESS，指出最大 cliette 數。下列範例顯示所要使用的埠號。

   ```
   START_PUBLIC_PORT=1000
   START_PRIVATE_PORT=1010
        MAX_PROCESS=5
   ```

   範例使用下列埠：

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

   常見的錯誤是兩組 cliette 所用的埠號重疊，或與「快取管理程式」埠號重疊。 請向您的系統管理者查詢，以確保您要使用的埠號爲可用的。在您作業系統上的 Readme 檔中，會有一般性的指引，您可以從中瞭解您作業系統可使用哪些埠號。

4. 設定 cliette 可執行檔的名稱。這個檔名將設定爲：

   ```
   EXEC_NAME=./dtwcxxx
   ```

其中，*xxx* 是資料庫類型識別字。 請參閱 表7，取得有效的可執行檔名稱：

表 7. cliette 可執行檔名稱

Cliette 說明	Cliette 類型	姓名		平台可用性					
		UNIX	Windows NT 或 OS/2	AIX	NT	OS/2	HP	SUN	SCO
DB2 處理 cliette	DTW_SQL	dtwcdb2	dtwcdb2.exe	Y	Y	Y	Y	Y	N
ODBC 處理 cliette	DTW_ODBC	dtwcodbc	dtwcodbc.exe	Y	Y	N	N	N	N
Sybase 處理 cliette	DTW_SYB	dtwcsyb	dtwcsyb.exe	Y	Y	N	N	N	N
Oracle 處理 cliette	DTW_ORA	dtwcora	dtwcora.exe	Y	Y	N	N	N	N

5. 設定 cliette 所連結的資料庫名稱：

   ```
   DATABASE=db_name
   ```

   其中 *db_name* 是 cliette 所連結的資料庫名稱，例如 MYDBASE。

6. 可選用的：變更 LOGIN 和 PASSWORD 變數的預設值，讓 Net.Data 使用與啓動「連線管理程式」者相同的使用者 ID，去連接 DB2 資料庫。設定這些預設值之後，可讓您避免將此資訊放進架構檔中。例如，在第28頁的圖6 的樣本架構檔中，將 14 和 15 行換成下列這兩行：

   ```
   LOGIN=*USE_DEFAULT
   PASSWORD=*USE_DEFAULT
   ```

   要訣：若您在架構檔中定義多重 cliette 登錄，則可以針對特定資料庫設定不同的資料庫登入和密碼。

若要架構 Java 應用程式 cliette：

1. 鍵入 cliette 環境陳述式：

   ```
   CLIETTE DTW_JAVAPPS
   ```

2. 決定 MIN _PROCESS 和 MAX_PROCESS 的值。 MIN_PROCESS 設定「連線管理程式」啓動時，所要啓動的處理數。 之後，當同時存在的處理抵達時，「連線管理程式」會啓動更多的 cliette (若有需要的話)，直到到達 MAX_PROCESS 所指的上限值爲止。 您所使用的值，可能會影響執行效能，但是您稍後可以再做變更。

   鍵入 MIN_PROCESS 和 MAX_PROCESS 陳述式。

   ```
   MIN_PROCESS=min_num
   MAX_PROCESS=max_num
   ```

   參數：

   min_num

   「連線管理程式」啓動時所要啓動的 cliette 數。 您必須擁有足夠的獨一無二埠號以支援這麼多 cliette。

   max_num

   可同時執行的最大額外 cliette 數。您必須擁有足夠的獨一無二埠號以支援這麼多 cliette。

3. 決定在系統上資料庫 cliette 所要使用的埠號。這些號碼必須是獨一無二，以避免與「快取管理程式」或其他應用程式所用的埠號發生衝突。每一個 cliette 使用兩個埠。設定一組埠時，您必須設定要使用的埠號範圍。 前兩個值爲 START_PUBLIC_PORT 和 START_PRIVATE_PORT。另一個是 MAX_PROCESS，指出最大 cliette 數。下列範例顯示所要使用的埠號。

```
START_PUBLIC_PORT=1000
START_PRIVATE_PORT=1010
     MAX_PROCESS=5
```

範例使用下列埠：

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

常見的錯誤是兩組 cliette 所用的埠號重疊，或與「快取管理程式」埠號重疊。請向您的系統管理者查詢，以確保您要使用的埠號爲可用的。在您作業系統上的 Readme 檔中，會有一般性的指引，您可以從中瞭解您作業系統可使用哪些埠號。

架構「現場連線」的提示和秘訣：

- 「連線管理程式」所用的 Cliette 名稱，必須是能夠獨一無二地認出一組 cliette。
- 如果是資料庫 cliette，則對於您要存取之每一個資料庫來說，都必須有一組已指名的 cliette。對於不常存取的資料庫，您可將 cliette 的 MIN 和 MAX 數設定爲 1。另外，您也可將 MIN 設爲 0，表示直到 Net.Data 提出 cliette 要求時，才啓動處理。
- 對於起始設定檔中的 cliette 類型， cliette 的 NAME 必須與 ENVIRONMENT 陳述式所參考到的 cliette 名稱一致。 cliette 名稱可以包含變數，以 DB2 cliette 爲例，應該包含變數參照 $(DATABASE)。在 ENVIRONMENT 陳述式中，cliette 名稱的預設值是 DTW_SQL:$(DATABASE)。您可以在起始設定檔中使用變數參照，但不能在「現場連線」架構檔中使用。

  DATABASE 變數定義在 Net.Data 巨集中。當巨集中發現 SQL 陳述式時， Net.Data 起始設定檔中的 $(DATABASE) 變數參照會換成 DATABASE 的現行值。

  您可以使用這個方法來存取多重資料庫。如果您想在 Net.Data 巨集中存取三個資料庫 (例如，D1、D2 和 D3)，且您的起始設定檔具有標準的 CLIETTE ″DTW_SQL:$(DATABASE)″ 行，則您的架構檔中必須有下列三個區段：

```
CLIETTE DTW_SQL:D1{ ...}
CLIETTE DTW_SQL:D2{....}
CLIETTE DTW_SQL:D3{....}
```

- 處理已啓動，但無法停止。如果您將最大處理數設定爲 M，不管任何時候都會同時進行 M 個處理，它們會維持在作用中狀態，直到您將「連線管理程式」關閉爲止。所以您大概不會想將 MAX_PROCESS 的值設得太高，以免將系統資源全用來啓動一些較少使用之處理。

  建議值： 嘗試不同的 MIN_PROCESS 和 MAX_PROCESS 值，看看何值最適用於您的系統。若「連線管理程式」收到的要求，比所設定的最大值還多，則會將最後的要求放在等待佇列中，直到 cliette 完成處理爲止。當 cliette 可用時，就會處理等待佇列中的要求。將要求放入等待佇列中的這種處理，應用程式的使用者會容易明瞭。

- 您可以在不同名稱的區段中，使用相同類型的 cliette。例如，架構檔中所有的 DB2 資料庫區段，全都使用了相同的 cliette 類型。您不可以讓兩個區段的名稱相同。

若您使用 CGI，而您僅想讓部份的資料庫使用「現場連線」，則您僅需在架構檔中列出您要的資料庫。當 Net.Data 在處理 Net.Data 巨集時，若遇到 SQL 區段，其會向「連線管理程式」要求取得某個特定的 cliette。若「連線管理程式」式沒有該類型的 cliette，其會以 NO_CLIETTE_AVAIL 訊息回應之。此時，Net.Data 會改用 DLL 版本來處理該要求。

若要自動啓動「連線管理程式」為 Windows NT 服務：

在 Windows NT 上，您可以設定令「連線管理程式」啓動爲 Windows NT 服務，而不需透過指令行。將「連線管理程式」當成 Windows NT 服務來執行，可在每次啓動機器時自動啓動「連線管理程式」。

要訣：在設定「連線管理程式」爲自動啓動之前，請先從指令行上啓動，來確保「現場連線」架構檔的正確性。

- 從 Windows NT 工作列上，選取 **啓動->設定->控制台->服務程式**。
- 選取 **Net.Data 連線管理程式**，然後按一下**啓動**按鈕。
- 選取**自動啓動類型**，再按一下**確定**。

架構 Web 伺服器與 CGI 一起使用

「通用閘道介面 (CGI)」是一種工業標準介面，可讓 Web 伺服器呼叫應用程式，例如 Net.Data。Net.Data 對 CGI 的支援，可讓您以最喜歡的 Web 伺服器來使用 Net.Data。

經由將 Map、Exec 及 Pass 指令新增到 HTTP 架構檔，呼叫 Net.Data，來架構 Web 伺服器，以便呼叫 Net.Data。

建議：以 HTTP 架構檔內的下列次序組織指令，來防止這些指令不被處理：Map、Exec、Pass。例如，如果下列 Pass 指令在 Map 或 Exec 指令之前，將不處理 Map 及 Exec 指令：

```
Pass /*
```

Map 指令

Map 指令會將使用格式 `/cgi-bin/db2www/*` 的登錄對映到您系統上 Net.Data 程式常駐的程式庫。 (字串尾端的星號 (*) 代表該字串後的任何字元。) 大寫與小寫 map 陳述式均包括在內，因爲指令會區分大小寫。在這個範例中，這兩個 Map 陳述式均指向同一位置。

Exec 指令

Exec 指令將啓用 Web 伺服器執行 CGI 程式庫中的任何 CGI 程式。 請在指令上設定程式常駐的程式庫 (非程式本身)。

Pass 指令

Net.Data 不會使用 Pass 指令。如果您想要簡化您的 URL，請在第17頁的『MACRO_PATH』中討論的 Net.Data 起始設定檔案中，使用 MACRO_PATH 陳述式。

架構 Net.Data for FastCGI

Net.Data 可在 Apache Web Server 及 Domino Go Webserver 上以 FastCGI 處理形式執行。 FastCGI 會以可信賴的 CGI 提供類似於其他 Web API 程式的執行效能。 AIX 及 Sun Solaris 作業系統支援 FastCGI。

開始之前：

在使用 FastCGI 之前，請確定已安裝了必備產品：

- **對於 Apache**： 下載並安裝 Apache Web Server 1.2.0 或更高版本。
- **對於 Domino Go Webserver**：從下列下載及安裝 Domino Go Webserver for AIX 或 Sun：

  `http://www.ics.raleigh.ibm.com/dominowebserver`

若要架構 Net.Data for FastCGI：

1. 爲您的作業系統架構 Web 伺服器和 FastCGI 架構檔：

   對於 Apache Web 伺服器：

   更新 `http.conf` 檔案。

   - 宣告新的應用程式：

     ```
     AppClass inst_dir
     -processes proc_num
     -initial-env LIBPATH=libpath
     -initial-env ORACLE_HOME=oracle_path
     -initial-env ORACLE_SID=oracle_instance
     -initial-env SYBASE=sybase_path
     -initial-env DSQUERY=sybase_instance
     -initial-env DB2INSTANCE=db2_instance
     -initial-env RXQUEUE_OWNER_PID=REXX_perf_var
     -initial-env LANG=locale
     ```

   - 宣告 FastCGI 模組：

     ```
     <location /fcgi-bin>
     SetHandler fastcgi-script
     </location>
     ```

   對於 Domino Go Webserver：

   更新 `httpd.conf` 和 `fcgi.conf` 檔案：

   - 在 `httpd.conf` 檔案中，宣告服務程式區段：

     ```
     ServerInit /u/mydir/http/fcgi-bin/fcgi.o:FCGIInit
     /u/mydir/http/fcgi.conf service/fcgi-bin/*
     /u/mydir/http/fcgi-bin/fcgi.o:FCGIDispatcher*ServerTerm
     /u/mydir/http/fcgi-bin/fcgi.o:FCGIStop
     ```

   - 在 `fcgi.conf` 檔案中，宣告應用程式：

     ```
     Local {
     Exec inst_dir
     Role Responder
     URL /fcgi-bin/db2www
     BindPath /tmp/db2www.ibm
     NumProcesses proc_num
     Environ LIBPATH=libpath
     Environ ORACLE_HOME=oracle_path
     Environ ORACLE_SID=oracle_instance
     Environ SYBASE=sybase_path
     Environ DSQUERY=sybase_instance
     ```

```
Environ DB2INSTANCE=db2_instance
Environ RXQUEUE_OWNER_PID=REXX_perf_var
Environ LANG=locale
}
```

參數：

inst_dir

Net.Data 之可執行檔的路徑和目錄名稱。

對於 Apache：

```
AppClass /u/mydir/apache/fcgi-bin/db2www
```

對於 Domino Go Webserver：

```
Exec /u/mydir/http/fcgi-bin/db2www
```

Role Responder

Domino Go Webserver 專用的必要關鍵字。

URL Domino Go Webserver 專用的必要關鍵字和 URD 位址。URL 指向 EXEC_PATH 陳述式所設定的路徑。

BindPath

AIX 上 Domino Go Webserver 專用的必要關鍵字和路徑。 Net.Data 和 FastCGI 所用之獨一無二 UNIX socket 的路徑。

proc_num

可同時處理的要求數。預設值是 1，但應該基於應用程式需求來增加，以改進執行效能。有關調整的資訊，請參閱第149頁的『使用 FastCGI』。

對於 Apache：

```
-processes 7
```

對於 ICS 或 Domino Go Webserver：

```
NumProcesses 7
```

libpath Net.Data 起始設定檔案中每一個 ENVIRONMENT 陳述式所宣告的 LIBPATH (共用程式庫或 DLL) 陳述式。 當存取 DB2 時，LIBPATH 陳述式應該含有 DB2 UDB 程式庫目錄的路徑。例如：

```
/usr/lpp/db2_05_00/lib
```

對於 Apache：

```
-initial-env LIBPATH=/u/mydir/apache/lib:/u/mydir/apache:/usr/lib
```

對於 Domino Go Webserver：

```
Environ LIBPATH=/u/mydir/http/lib:/u/mydir/http:/usr/lib
```

oracle_path

使用 Oracle 時必備的。Oracle 資料庫可執行檔的路徑和目錄。

對於 Apache：

```
-initial-env ORACLE_HOME=/home.native/oracle/product/7.2
```

對於 Domino Go Webserver：

```
Environ ORACLE_HOME=/home.native/oracle/product/7.2
```

oracle_instance

使用 Oracle 時必備的。Oracle 資料庫的案例。 您須對 Oracle 使用「現場連線」。

對於 Apache：

```
-initial-env ORACLE_SID=mvpdb2
```

對於 Domino Go Webserver：

```
Environ ORACLE_SID=mvpdb2
```

sybase_path

使用 Sybase 時必備的。Sybase 資料庫可執行檔的路徑和目錄。

對於 Apache：

```
-initial-env SYBASE=/home.native/sybase/product
```

對於 Domino Go Webserver：

```
Environ SYBASE=/home.native/sybase/product
```

sybase_instance

使用 Sybase 時必備的。Sybase 資料庫的案例。 您須對 Sybase 使用「現場連線」。

對於 Apache：

```
-initial-env DSQUERY=SybaseAIX
```

對於 Domino Go Webserver：

```
Environ DSQUERY=SybaseAIX
```

db2_instance

使用 DB2 時必備的。DB2 資料庫的案例。

對於 Apache：

```
-initial-env DB2INSTANCE=wwwinst
```

對於 Domino Go Webserver：

```
Environ DB2INSTANCE=wwwinst
```

REXX_perf_var

在 AIX 上使用 REXX 時必備的。該效能變數使用於 AIX 作業系統上的 FastCGI 和 REXX。預設值是 0。對於其他產品和作業系統，請在 Net.Data 巨集中宣告這個變數。這個變數的詳細資訊，請參閱第185頁的『附錄B. Net.Data for AIX』。

對於 Apache：

```
-initial-env RXQUEUE_OWNER_PID=0
```

對於 Domino Go Webserver：

```
Environ RXQUEUE_OWNER_PID=0
```

locale UNIX 語言環境變數。使用 En_US 表示美式英語。

對於 Apache：

```
-initial-env LANG=En_US
```

對於 Domino Go Webserver：

```
Environ LANG=En_US
```

2. **對於 Apache：** 在 srm.conf 檔案中，新增 fgi-bin 目錄做爲新的 script 別名：
   ```
   ScripAlias /fcgi-bin/ /u/mydir/apache/fci-bin
   ```
3. 從 CGI-BIN 至 FCGI-BIN，移轉靜態或動態產生之網頁內的任何超鏈結。例如：
   ```
   <A HREF="http://server/fcgi-bin/db2www/filename.ext/block/
   [?name=val&...]">any text</A>
   ```
4. 以 FCGI-BIN 代替 CGI-BIN，修正 Net.Data 有關 URL 呼叫的一般使用者文件。例如：
   ```
   http://server/fcgi-bin/db2www/filename.ext/block/[?name=val&...]
   ```

架構 Net.Data 來使用 Java Servlet 及 Java Bean

請參閱 Web 伺服器文件，以取得有關安裝及使用 servlet 的指示。 Net.Data servlet 位在 NetDataServlets.jar 檔中。您的 Web 伺服器將需要您新增 *inst_dir*/servlet-lib/NetDataServlets.jar 及 *inst_dir*/servlet-lib 到您的 CLASSPATH。

有關安裝 Web 伺服器和 Web 伺服器架構檔指令的詳細資訊，請參閱您的 Web 伺服器文件：

架構 Net.Data 來使用 Web 伺服器 API

使用 Web 伺服器應用程式設計介面 (API) 來代替 CGI，可明顯地改善 Net.Data 的執行效能。 Net.Data 支援下列伺服器 API：

- IBM Internet Connection Server API (ICAPI)
- Lotus Domino Go Webserver API (GWAPI)
- Microsoft Internet Server API (ISAPI)
- Netscape API (NSAPI)

有關每一個 API 的詳細資訊，請參閱第149頁的『使用 Web 伺服器 API』 及您的 Net.Data 版本之 README 檔。

需求：若要在 ICAPI, GWAPI, ISAPI 或 NSAPI 模式中執行 Net.Data，您須重新架構 Web server，來使用 Net.Data DLL 或共用程式庫作爲它的服務程式指令。重新架構後，須重新啓動 Web 伺服器，以便您對 Net.Data 起始設定檔案所做的變更可以生效。依據預設值，Net.Data 是在 CGI 模式中執行。

下列段落描述如何架構 Net.Data 與 Web 伺服器來執行 Web 伺服器 API 模式。已提供一般性步驟和範例，但可能不同於您的作業系統。請參閱您的作業系統的 Net.Data README 檔，取得特定指示。

架構 ICAPI 與 GWAPI：

Domino Go Webserver 是繼 IBM Internet Connection Secure Server 之後開發的產品。若您正在升級，您可能想使用新的 Domino Go Webserver。請注意 GWAPI 和 ICAPI 是指同一產品，名稱不同是爲了區別所用的 Web 伺服器。

1. 停止 Web 伺服器。
2. 請確定 ICAPI 或 GWAPI DLL 或共用程式庫，是位於伺服器的 CGI-BIN 或 ICAPI-LIB 目錄內。

   有關特定檔案和目錄名稱，請參閱符合您作業系統的 Net.Data README 檔案或程式目錄。
3. 請新增服務程式陳述式至您的 Web 伺服器架構檔中 (httpd.conf 或 httpd.cnf)，以呼叫該 API。

   例如：

   ```
   Service /cgi-bin/db2www*   /usr/lpp/internet/server_root/cgi-bin/dtwicapi.o:dtw_icapi*
   ```

   請參閱作業系統的 Net.Data README 檔案，取得特定檔案與目錄名稱。
4. 重新啓動 Web 伺服器。

ICAPI 和 GWAPI 具備完整的相容性，而可支援現有的應用程式。使用 ICAPI 或 GWAPI 來呼叫 URL、套表、或鏈結，方法與使用 CGI 相同。使用 CGI 而順利執行的任何巨集，亦能夠使用 ICAPI 或 GWAPI 來順利執行。這些巨集不需要任何變更。

若要架構 ISAPI：

1. 停止 Web 伺服器。
2. 將 Net.Data 提供之 ISAPI 的 DLL 複製到伺服器的次目錄中。例如：

   ```
   /inetsrv/scripts/dtwisapi.filetype
   ```

   其中 *filetype* 在 Window NT 和 OS/2 上是指 .dll，在 UNIX 作業系統上是指 .o。

   請參閱作業系統的 Net.Data README 檔案，取得特定檔案與目錄名稱。
3. 因爲 ISAPI 會略過 CGI 處理，所以您不需要在套表與鏈結中具有 URL 的 cgi-bin/db2www/ 部份。請改用 dtwisapi.*filetype*。例如，如果下列 URL 呼叫 Net.Data 作爲 CGI 程式：

   ```
   http://server1.stl.ibm.com/cgi-bin/db2www/test1.d2w/report
   ```

   您應該以下列 URL 呼叫 Net.Data 作爲 ISAPI 增益集：

   ```
   http://server1.stl.ibm.com/scripts/dtwisapi.dll/test1.d2w/report
   ```
4. 如果您已將巨集 test1.d2w 儲存在 MACRO_PATH 中指定的其中一個目錄或 Web 伺服器的現行目錄下的次目錄 /order/，請使用下列 URL 呼叫 CGI 模式中的 Net.Data：

   ```
   http://server1.stl.ibm.com/cgi-bin/db2www/orders/test1.d2w/report
   ```

   呼叫 ISAPI 模式中的 Net.Data 的對等 URL 爲：

   ```
   http://server1.stl.ibm.com/scripts/dtwisapi.dll/orders/test1.d2w/report
   ```
5. 重新啓動 Web 伺服器。

若要架構 NSAPI：

1. 停止 Web 伺服器。
2. 將 Net.Data 提供之 NSAPI 的 DLL 複製到伺服器目錄中。例如：

   ```
   /netscape/server/bin/httpd/dtwnsapi.filetype
   ```

   其中 *filetype* 在 Window NT 和 OS/2 上是指 .dll，在 UNIX 作業系統上是指 .o。

請參閱作業系統的 Net.Data README 檔案，取得特定檔案與目錄名稱。

3. 使用下列的變更，來修改您的伺服器架構檔。有關作業系統的差異，請參閱符合您作業系統的 Net.Data README 檔案或程式目錄。

obj.conf 在檔案頂端加進下式：

```
Init fn="load-modules" shlib="<path>dtwnsapi.dll" funcs=dtw_nsapi
```

obj.conf 在 Services 部份加進下式：

```
Service fn="dtw_nsapi" method=(GET|HEAD|POST) type="magnus-internal/d2w"
```

mime.types 新增這個類型，其中 *d2w* 是巨集的預設副檔名。您可以設定任何三個字元組合。

```
type=magnum-internal/d2w exts=d2w
```

4. 將 Net.Data 巨集檔案從 `netdata/macro` 目錄移到伺服器的根文件目錄中：

   ```
   /netscape/server/docs/
   ```

5. 將伺服器的根文件目錄新增到起始設定檔案中的 MACRO_PATH 陳述式。此項變更將告訴 Net.Data 何處可找到巨集檔案。
6. 因爲 NSAPI 會略過 CGI 處理，所以您不需要在套表與鏈結中具有 URL 的 `cgi-bin/db2www/` 部份。 伺服器知道具有 `d2w` 檔案類型的檔案爲 Net.Data 巨集，因爲當您變更 Netscape 架構檔時已定義了它。例如，下列 URL 把 Net.Data 視爲 CGI 程式來呼叫：

   ```
   http://server1.stl.ibm.com/cgi-bin/db2www/test1.d2w/report
   ```

   當下列 URL 呼叫 Net.Data 作爲 NSAPI 增益集時：

   ```
   http://server1.stl.ibm.com/test1.d2w/report
   ```

7. 重新啓動 Web 伺服器。

若您將您的 Net.Data 巨集放在好幾個目錄中，則最後三個步驟會有所變動：

1. 將目錄連同其中的 Net.Data 巨集，一併移到伺服器的根文件目錄中。
2. 更新起始設定檔中的 MACRO_PATH 變數，以併入您的巨集檔所在之全部目錄和次目錄。
3. 修改指向這些 Net.Data 巨集的鏈結與套表，並保留其目錄名稱。 例如，在 CGI 模式中執行時，下列 URL 將呼叫儲存在 `/orders/` 目錄中的 Net.Data 巨集：

   ```
   http://server1.stl.ibm.com/cgi-bin/db2www/orders/test1.d2w/report
   ```

   更新過的 URL (用來呼叫 NSAPI 模式中的 Net.Data) 較短，但仍保留目錄名稱：

   ```
   http://server1.stl.ibm.com/orders/test1.d2w/report
   ```

使用 Net.Data 管理工具來架構 Net.Data

Net.Data 管理工具協助您在 Windows NT、AIX 及 OS/2 作業系統上，架構和管理 Net.Data 起始設定檔 (DB2WWW.INI) 及「現場連線」的架構檔 (dtwcm.cnf)。透過這個工具，您可以完成下列作業：

- 第39頁的『啓動管理工具』
- 第39頁的『架構路徑陳述式』
- 第41頁的『配置埠』
- 第41頁的『架構 Cliette』

- 第44頁的『架構語言環境』
- 第47頁的『定義架構變數』

請參閱『開始之前』，瞭解如何設定管理工具，並確定具有正確的軟體必備條件。

開始之前

1. 計畫 Net.Data 語言環境、資料庫、cliette、埠、及架構變數的架構。
2. 從 CD-ROM 來安裝 Net.Data。
3. 安裝 Java 執行期程式庫 (對每一個作業系統安裝 JDK 1.1 及後續的版本)。請檢查您的作業系統的 Net.Data README 檔，取得詳細資訊。

   安裝 JDK 後，確定在 CLASSPATH 中具有 `classes.zip`。
4. 若您已安裝 DB2 Universal Database 所附的 IBM JDBC 驅動常式，請新增驅動常式目錄至您的 Java CLASSPATH 陳述式，以啓用 DB2 登入測試。
5. 切換至儲存 Net.Data 管理工具程式的目錄：

   對於 OS/2 和 Windows NT：

   是指 *inst_dir*\connect*admin_directory*，其中 *inst_dir* 是您在安裝期間指定給 Net.Data 的目錄，而 *admin_directory* 是管理工具檔案所在的目錄。

   對於 AIX：

   是指 `/usr/lpp/internet/db2www/db2.v2/`*admin directory*，其中 *admin_directory* 是管理工具檔案所在的目錄

啓動管理工具

您使用的作業系統會決定管理工具的啓動方式。

對於 OS/2 和 Windows NT：

從 IBM Net.Data 版本 2 資料夾中，選取 **Net.Data 管理工具**圖示。

對於 AIX：

變更爲 Net.Data 安裝目錄 (inst_dir)。從指令行中，輸入 `ndadmin` 來啓動工具。

即會啓動管理工具，並顯示「Net.Data 管理」筆記本。

架構路徑陳述式

使用**路徑**頁面來新增、修改或刪除路徑陳述式，以找出 Net.Data 處理 Net.Data 巨集時所需的檔案。這些陳述式說明於 第16頁的『路徑架構陳述式』。 第40頁的圖7 顯示**路徑**頁面。

圖 7. Net.Data 管理工具的「路徑」頁面. 使用此頁來新增、修改或刪除路徑陳述式。

架構要訣：HTML 檔案類型只能有一個路徑。

若要新增路徑陳述式：

1. 啓動管理工具。
2. 在**路徑**頁面中，從**檔案類型**選取一種檔案類型，例如選取 Exec。
3. 在**編輯目錄**欄位中，鍵入新的路徑，並按一下**新增**按鈕。

   若您設定的路徑不存在，則會出現路徑視窗。若未選取任何目錄，則新的目錄會成爲列示中的最後一個項目。
4. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要修改路徑陳述式：

1. 啓動管理工具。
2. 在**路徑**頁面上，從**檔案類型**列示中，選取您要變更的檔案類型。
3. 在**目錄選擇**列示中，選取您要修改的路徑。所選取的路徑在**編輯目錄**欄位中開啓。
4. 修改**編輯目錄**欄位中的路徑，並按一下**修改**按鈕。若您輸入的路徑不存在，則會出現路徑視窗。
5. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要刪除路徑陳述式：

1. 啓動管理工具。
2. 在**路徑**頁面上，從**檔案類型**列示中，選取您要刪除的檔案類型。
3. 在**目錄選擇**欄位中，選取您要刪除的路徑。所選取的路徑在**編輯目錄**欄位中開啓。
4. 按一下**刪除**按鈕。

5. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

配置埠

使用**埠**頁面，來設定 Net.Data 所使用的 TCP/IP 埠號。 圖8 顯示**埠**頁面。

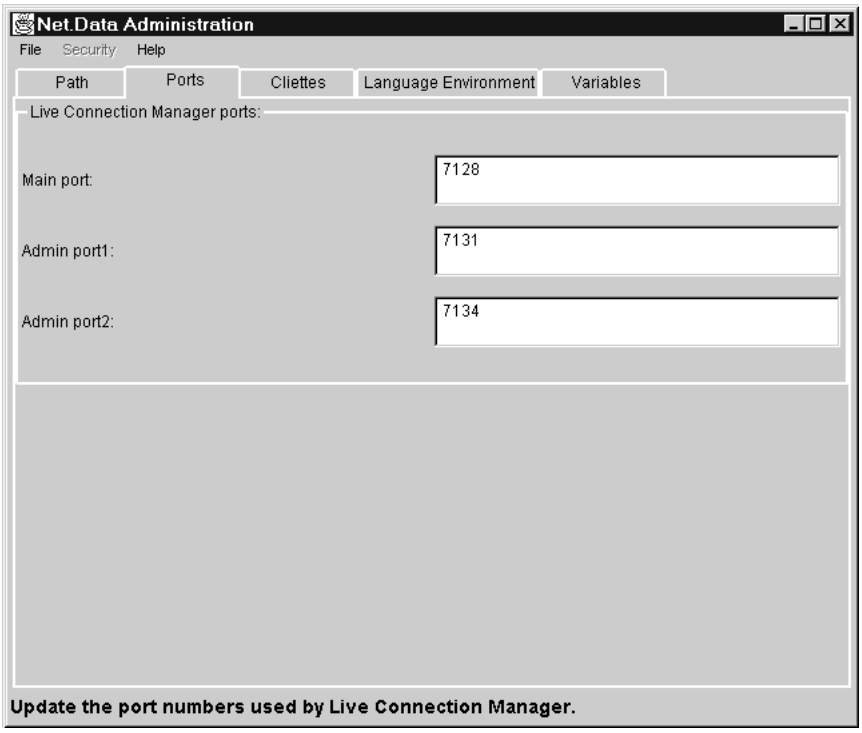


圖 8. Net.Data 管理工具的「埠」頁面. 使用此頁來設定埠。

若要設定 TCP/IP 埠號：

1. 啓動管理工具。
2. 在**埠**頁面上，於每一個埠欄位中鍵入一個獨一無二的埠號。當您跳至下一個欄位時，管理工具會驗證每一個欄位中所鍵入的埠號。
3. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

架構 Cliette

使用 **Cliette** 頁面來新增、修改或刪除「現場連線」資料庫 cliette，您也可以爲資料庫 cliette 管理資料庫和管理者之使用者 ID 和通行碼。有關 cliette 的詳細資訊，請參閱第 150頁的『管理連線』。 第42頁的圖9 顯示 **Cliette** 頁面。

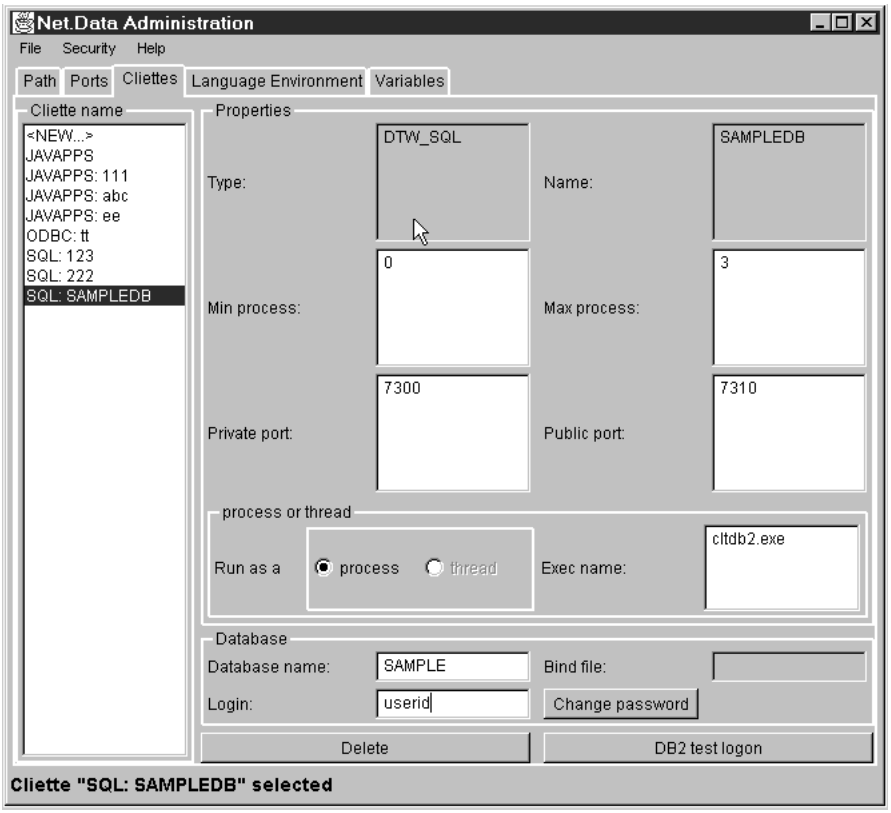


圖 9. Net.Data 管理工具的 Cliette 頁面. 使用此頁來新增、修改及刪除 cliette。

若要新增 cliette：

1. 啓動管理工具。
2. 在 **Cliette** 頁面上，從 **Cliette 名稱**列示中選取 **<新增...> 新增 cliette** 視窗會開啓。

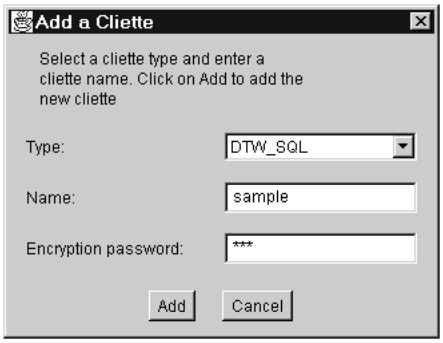


圖 10. Net.Data 管理工具的「新增 Cliette」視窗. 使用此頁來新增 cliette。

若您已啓用加密，則在首次建立或修改 cliette 時，會要求您輸入加密密碼。這個密碼會儲存起來，而您永不需再輸入一次。

3. 從**類型**列示中選取 cliette 類型。
4. 在**名稱**欄位中輸入新 cliette 的名稱。該名稱可以是資料庫的名稱，或另一個獨一無二的 cliette 名稱。 例如：MYCLIETTE。

5. 若**加密密碼**欄位已啓用，請輸入加密密碼。您將不需要再輸入一次密碼，因爲管理工具已爲您把密碼儲存起來。
6. 按一下**新增**按鈕。

   新的 cliette 會建立，且新增至 cliette 列示的底端。此外，新名稱會以高亮度顯示，且 cliette 的預設性質會顯示於**性質**群組框中。您可以變更這些值，來配合您的架構。
7. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要修改 cliette：

1. 啓動管理工具。
2. 在 **Cliette** 頁面上，從 **Cliette 名稱**列示中，選取您要變更的 cliette 之名稱。 cliette 的性質顯示於**性質**群組框中。
3. 依需要從**性質**群組框中修改性質。
   a. **類型**欄位顯示要定義的 cliette 類型，並對應一個語言環境類型名稱。 Net.Data 會於您新增新的 cliette 時移入這個欄位，且選項定義於「新增 Cliette」視窗的 **Cliette 類型**列示中。
   b. **名稱**欄位顯示 cliette 的名稱，通常是指資料庫的名稱。 Net.Data 會於您新增新的 cliette 時，填入這個欄位。
   c. 在**最小處理**欄位中，輸入「連線管理程式」啓動時能夠啓動的 cliette 處理數。每一個處理需要一個獨一無二的埠位址。 有關 MIN 處理值的詳細資訊，請參閱 第27頁的『架構「現場連線」』。
   d. 在**最大處理**欄位中，輸入可同時執行的最大 cliette 處理數 (除了「連線管理程式」啓動時所啓動的處理之外)。每一個處理需要一個獨一無二的埠位址。 有關 MAX 處理值的詳細資訊，請參閱第27頁的『架構「現場連線」』。
   e. 在**專用埠**欄位中輸入一個獨一無二埠號，來設定啓動埠號，給「連線管理程式」所啓動的 cliette 處理使用。額外的埠號給**最小處理**值所設定的每一個處理使用。例如，若您設定**專用埠**的埠號爲 7012，設定**最小處理**的值爲 5，則使用埠號 7012 至 7016，且不能與系統中設定的其他埠發生衝突。
   f. 在**公用埠**欄位中輸入一個獨一無二埠號，來設定啓動埠號，給額外處理啓動時所啓動的 cliette 處理使用，一直到**最大處理**欄位中設定的號碼爲止。額外的埠號使用於每一個處理。例如，若您設定**公用埠**的埠號爲 7020，設定**最大處理**的值爲 5，則使用埠號 7020 至 7024，且不能與系統中設定的其他埠發生衝突。
   g. **執行名稱**欄位會顯示 cliette 可執行檔的名稱。
4. 若 cliette 要使用於資料庫，請視情況修改**資料庫**群組框的值：
   a. 在**資料庫名稱**欄位中，設定 cliette 所連結的資料庫之名稱，例如 MYDBASE。
   b. **連結檔案**欄位包含連結檔案的名稱和路徑，指出您所使用的 cliette 類型。
   c. **登入**欄位設定連接資料庫時所用的登入使用者 ID。
   d. **變更密碼**按鈕可開啓「變更資料庫密碼」視窗。輸入加密密碼和新密碼兩次。您可以使用**安全**下拉功能表中設定的加密函數，來加密資料庫密碼。
5. 選取**檔案**及**儲存**，來儲存您的變更。
6. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要測試 DB2 資料庫登入和連線：

1. 在管理工具的 **Cliette** 頁面上，按一下 **DB2 測試登入**按鈕。測試完成時，確認視窗會出現，其中顯示連線測試的狀態。

2. 關閉該視窗，以繼續架構或關閉管理工具。

若要刪除 cliette：

1. 啓動管理工具。
2. 在 **Cliette** 頁面上，從 **Cliette 名稱**列示中，選取您要刪除的 cliette 之名稱。
3. 按一下**刪除**按鈕。
4. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要加密 cliette 使用者 ID 和通行碼：

加密對 cliette 的資料庫連線，提供一層安全保障。開啓加密時，「現場連線」架構檔中的所有資料庫密碼皆會加密，在存取和解密時就需要一個加密密碼。

要求：您必須使用 Net.Data 版本 2「現場連線」架構檔，才能使用加密。

1. **重要事項**：備製您的「現場連線」架構檔 <path>dtwcm.cnf。若您遺失加密密碼，或想要解密資料庫密碼而需復置密碼，就需要這個檔案。
2. 在管理工具的 **Cliette** 頁面上，選取 **安全 -> 開啓加密**下拉功能表選項。「開啓加密」確認視窗會出現。
3. 按一下**是**來繼續。「加密密碼」視窗會出現。
4. 輸入密碼兩次，以取得授權來使用具有加密密碼的 cliette。
5. 按一下 **OK**，來爲 cliette 定義新密碼並加密所有資料庫密碼。

若要關閉加密 cliette 使用者 ID 和通行碼：

1. 在管理工具的 **Cliette** 頁面上，選取**安全 -> 關閉加密**下拉功能表選項。 「關閉加密」確認視窗會出現。
2. 按一下**是**來繼續。因爲安全上理由，所有密碼皆設爲 *USE_DEFAULT。您可以使用「現場連線」的備份拷貝 <path>dtwcm.cnf，來復置您的密碼。

若要變更加密的密碼：

1. 在管理工具的 **Cliette** 頁面上，選取**安全 -> 變更加密密碼**下拉功能表選項。 「變更加密密碼」確認視窗會出現。
2. 按一下**是**來繼續。「變更加密密碼」視窗會出現。
3. 輸入一次舊的加密密碼，及新密碼兩次。
4. 按一下 **OK** 來變更加密密碼。

若要變更資料庫密碼：

1. 在管理工具的 **Cliette** 頁面上，按一下**變更密碼**按鈕。「變更資料庫密碼」視窗會出現。
2. 輸入一次加密密碼，新的資料庫密碼兩次。
3. 按一下 **OK** 來變更密碼並關閉視窗。 若您開啓加密，則已變更的資料庫密碼會被加密。

架構語言環境

使用**語言環境**頁面來新增、修改或刪除 Net.Data 語言環境。語言環境討論於第20頁的『環境架構陳述式』。第45頁的圖11 顯示**語言環境**頁面。

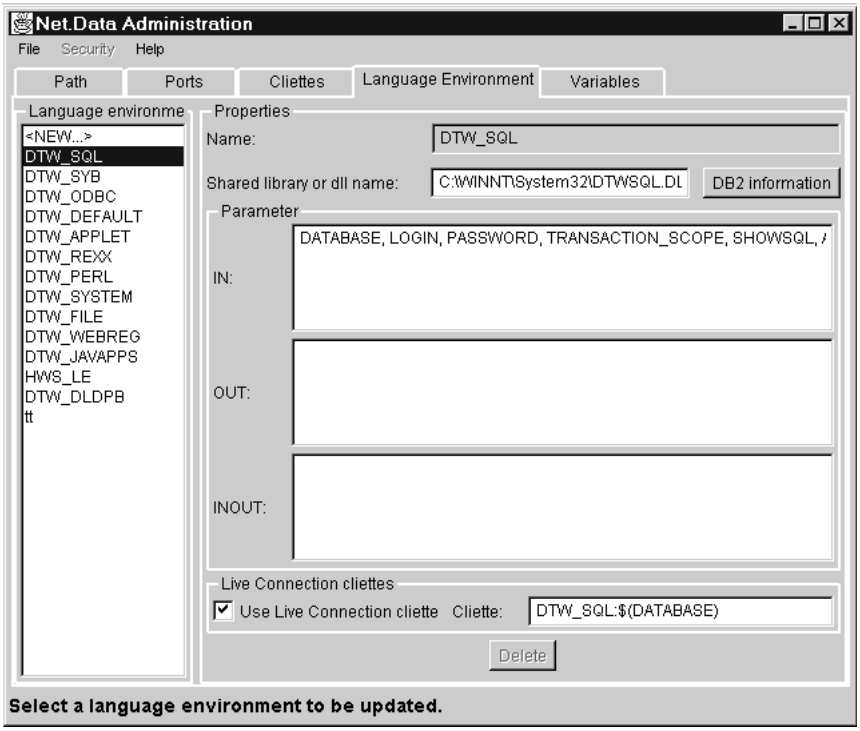


圖 11. Net.Data 管理工具的「語言環境」頁面. 使用此頁來設定語言環境。

若要新增語言環境：

1. 啓動管理工具。
2. 在**語言環境**頁面上，從**語言環境**列示中選取 **<新增...> 新增新的語言環境**視窗會開啓。
3. 在欄位中輸入語言環境的名稱，並按一下**新增**按鈕。 「新增語言環境」視窗會出現。

圖 12. Net.Data 管理工具的「新增語言環境」視窗. 使用此頁來設定新的語言環境。

新的語言環境會建立，且其名稱會加至語言環境列示的底端。此外，新名稱會以高亮度顯示，且語言環境的預設性質會顯示於**性質**群組框中。 您可以變更這些值，來配合您的架構。

4. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要修改語言環境：

1. 啓動管理工具。
2. 在**語言環境**頁面上，從**語言環境**列示中，選取您要變更之語言環境的名稱。 cliette 的性質顯示於**性質**群組框中。
3. 視情況修改**性質**群組框中的性質，如圖12 所示：

a. 在**名稱**欄位中，設定語言環境的名稱；這個名稱對應於定義 cliette 時所用的語言環境類型。若要變更這個值，請於**語言環境**列示中，在不同的名稱上按兩下滑鼠按鈕。有關語言環境類型的詳細資訊，請參閱第20頁的『環境架構陳述式』。

b. 在**共用程式庫或 dll 名稱**欄位中，設定語言環境的共用程式庫或 DLL 程式名稱和路徑。

c. 選取 **DB2 資訊**按鈕，來顯示「DB2 資訊」視窗，如圖13 所示。

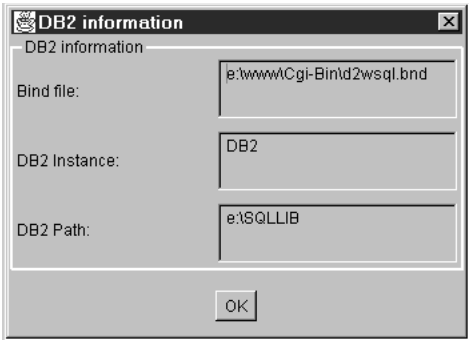


圖 13. Net.Data 管理工具的「DB2 資訊」視窗. 使用此頁來設定 DB2 資料庫的特定資訊。

設定 DB2 環境變數的值：

1) 在**連結檔案**欄位中，輸入連結檔案的路徑和檔案名稱。
2) 在 **DB2 案例**欄位中，設定使用 SQL 語言環境時所連結之資料庫的 DB2INSTANCE 值。
3) 在 **DB2 路徑**欄位中，設定 DB2 產品可執行檔的路徑目錄名稱，通常是 \SQLLIB。
4) 按一下 **OK** 來儲存您的變更並關閉視窗。

d. 在**參數**群組框中，設定每次呼叫語言環境時，對語言環境來回傳送的輸入和輸出參數。

要訣：請勿更新這些欄位，除非您要定義自己的語言環境。

e. 在**現場連線 cliette** 群組框中，設定是否使用 cliette 及那個 cliette 應該連結語言環境。

1) 勾選**使用「現場連線」cliette** 核對框，來設定語言環境的 cliette 是否爲作用中。若您想於呼叫語言環境時，使用 **Cliette** 欄位中設定的 cliette，則選取這個核對框。
2) 在 **Cliette** 欄位中設定 cliette 的名稱，以便於語言環境中執行。名稱的語法是根據您是架構資料庫，或 Java 應用程式語言環境而定。預設值是 `DTW_SQL:$(DATABASE)`。

   資料庫的語法：

   type`:`*name*

   其中：

   type cliette 的語言環境類型。可以是下列其中一個值：

對於 Windows NT：

DTW_ODBC, DTW_ORA, DTW_SYB, DTW_SQL, DTW_JAVAPPS

對於 OS/2：

DTW_SQL, DTW_JAVAPPS

對於 AIX：

DTW_ODBC, DTW_ORA, DTW_SYB, DTW_SQL, DTW_JAVAPPS

name **Cliette** 頁面上所定義之 cliette 的名稱。預設值是 $(DATABASE)。

Java 應用程式的語法：

```
DTW_JAVAPPS
```

4. 選取**檔案**及**儲存**，來儲存您的變更
5. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要刪除語言環境：

限制：您僅能刪除使用者所建的語言環境，而無法刪除 NetData 所附的預設語言環境

1. 啓動管理工具。
2. 在**語言環境**頁面上，從**語言環境**列示中，選取您要刪除之語言環境的名稱。
3. 按一下**刪除**按鈕。
4. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

定義架構變數

使用**變數**頁面來設定 Net.Data 的起始目錄，及選取錯誤訊息記錄的層次。 第48頁的圖 14 顯示**變數**頁面。

圖 14. Net.Data 管理工具的「變數」頁面. 使用此頁來設定起始設定變數。

若要設定 Net.Data 的起始目錄：

這個變數亦稱爲安裝目錄變數。

1. 啓動管理工具。
2. 在**變數**頁面上，於**安裝目錄**欄位中，輸入儲存日誌檔的目錄路徑。預設值爲 `\inst_dir\logs\`。例如 `e:\db2www`。
3. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

若要設定 Net.Data 的錯誤訊息記錄層次：

1. 啓動管理工具。
2. 在**變數**頁面上，從**錯誤日誌**群組框中選取一個錯誤日誌層次：
   - **關閉**
   - **只有錯誤**
   - **警告和錯誤**
3. 關閉管理工具，或按一下另一個 Tab 來進行額外的架構作業。

授與 Net.Data 存取的檔案的存取權

在使用 Net.Data 之前，您需要確定 Net.Data 執行時所依據的使用者 ID 具有 Net.Data 巨集中所參照的檔案的適當存取權，以及具有 URL 參照的巨集的適當存取權。 這表示這些檔案必須位於 Web 伺服器能夠連接，或這些使用者 ID 有明確存取權的 和目錄或程式庫中。

特別是要確定 Net.Data 執行時所依據的使用者 ID，是否具有下列授權：

- 讀取 Net.Data 起始設定檔 db2www.ini
- 執行 Net.Data 可執行檔與 DLL，以及搜尋可執行檔與 DLL的路徑中的目錄
- 讀取適當的 Net.Data 巨集檔，以及搜尋 MACRO_PATH 路徑架構陳述式所指明的適當目錄
- 執行適當的檔案，以及搜尋 EXEC_PATH 路徑架構陳述式所指明的適當目錄
- 讀取適當的檔案，以及搜尋 INCLUDE_PATH 路徑架構陳述式所指明的適當目錄
- 讀寫適當的檔案，以及搜尋 FFI_PATH 路徑架構陳述式所指明的適當目錄
- 讀取「現場連線」架構檔 dtwcm.cnf
- 讀取「快取管理程式」架構檔 CACHEMGR.CNF
- 讀取語言環境所參照的外部 Perl 與 REXX 可執行檔

授與這些檔案之存取權的方法，是根據執行 Net.Data 的作業系統而定。

第3章 維持資產的安全

Internet 安全的提供結合了防火牆技術、作業系統特性、 Web 伺服器特性、Net.Data 機制及存取控制機制 (爲資料原始檔的一部份)。

您必須決定出適用於資產的安全層次。這章說明可用來維持資產安全的方法，也可提供用來計畫網站之安全的額外資源參考。

下列各節含有保護資產的準則。所說明的安全機制包括：

- 『使用防火牆』
- 第53頁的『加密網路上的資料』
- 第53頁的『使用身份驗證』
- 第54頁的『使用授權』
- 第54頁的『使用 Net.Data 機制』

此外，Net.Data 提供資料庫 cliette 密碼加密；請參閱第41頁的『架構 Cliette』，以取得有關詳細資訊。

使用防火牆

*防火牆*是硬體、軟體及策略 (設計來限制網路環境中的資源存取) 的集合。

防火牆：

- 保護內部網路，免於潛入或侵入
- 保護內部網路，不讓內部使用者帶入資料及程式
- 限制內部使用者存取外部資料
- 如果防火牆遭破壞，則可限制所造成的損毀

Net.Data 可與防火牆產品搭配使用在您的環境下執行。

下列可能的架構將提供一些建議，告訴您如何管理 Net.Data 應用程式的安全。這些架構將提供高層次的資訊，並假定您已架構了防火牆，使您的安全 Intranet 與公用網際網路隔離。請根據您組織的安全策略，仔細考量這些架構問題。

- **高安全架構**

  本架構會建立一個次網路，使 Net.Data 及 Web 伺服器與安全 Intranet 及公用網際網路隔離。防火牆軟體係用來在 Web 伺服器與公用網際網路之間建立一道防火牆，而在 Web 伺服器與受保護的 Intranet (含有 DB2 伺服器) 之間建立另一道防火牆。第52頁的圖15 會顯示這個架構。

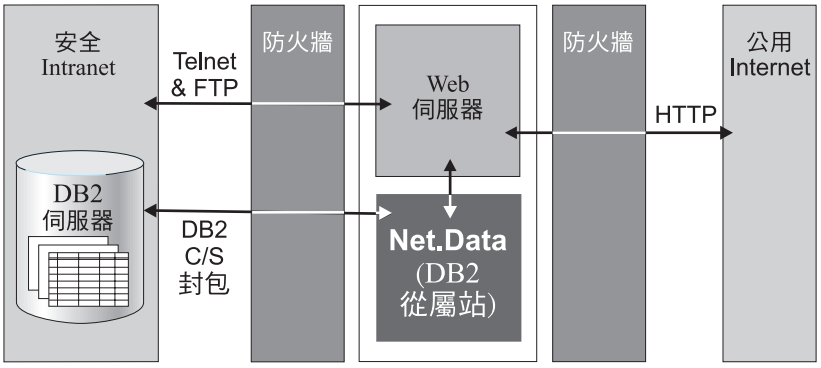


圖 15. 高安全架構

若要設定這個架構：

– 將 Net.Data 安裝在 Web 伺服器機器上，並確定 Net.Data 可經由:
  - 將從屬站應用程式啓用器 (CAE) 安裝在 Web 伺服器機器上。
  - 架構防火牆，容許 DB2 資料通過防火牆。有一種方法即是新增一個封包過濾規則，容許來自 Net.Data 的 DB2 從屬站要求，並確認從 DB2 伺服器到 Net.Data 的封包。

– 容許 Web 伺服器與安全 Intranet 之間的 FTP 與 Telnet 存取。有一種方法即是將 sock 伺服器安裝在 Web 伺服器機器上。

– 在防火牆軟體的封包過濾架構檔中，設定從標準 HTTP 埠進入的 TCP 封包可以存取 Web 伺服器。此外，設定離去的 TCP 確認封包可以從 Web 伺服器到達公用網際網路上的任何主電腦中。

- **中安全架構**

  在這個架構中，防火牆軟體會將具有 DB2 伺服器的受保護 Intranet 與公用網際網路隔離。 Net.Data 與 Web 伺服器位於工作站平台上的防火牆之外。這個架構比第一個架構簡單，但仍會提供資料庫保護。圖16 顯示這個架構。

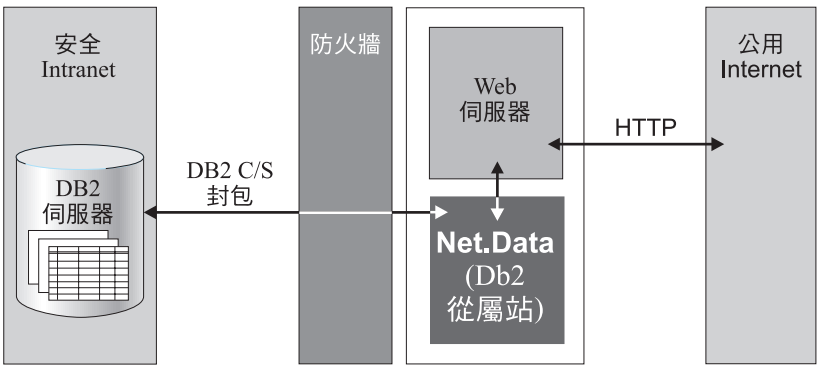


圖 16. 中安全架構：

您必須將 CAE 安裝在 Web 伺服器上，方可容許 Net.Data 與 DB2 伺服器通信。須架構防火牆，使得容許 DB2 從屬站要求可從 Net.Data 流到 DB2，並容許確認封包從 DB2 流到 Net.Data。

- **低安全架構**

在這個架構中， DB2 伺服器與 Net.Data 均安裝在防火牆與受保護的 Intranet 之外。在受到外來攻擊時，它們得不到保護。在此種架構方式下，防火牆不需用到任何封包過濾規定。圖17 顯示這個架構。

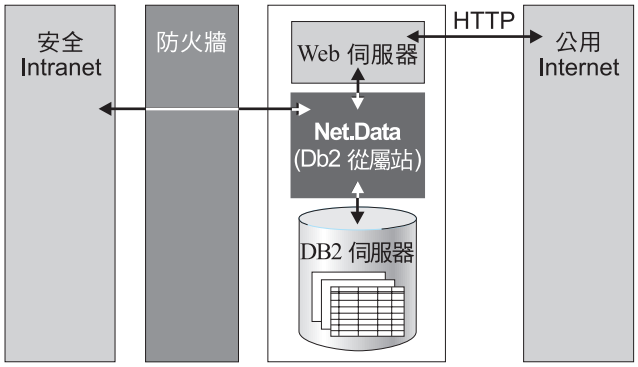


圖 17. 低安全架構：

加密網路上的資料

在使用支援 Secured Sockets Layer (SSL) 的 Web 伺服器時，您可加密從屬站系統及您 Web 伺服器之間傳送的所有資料。這個安全措施支援的加密項目包括：登入 ID、密碼，及透過 HTML 套表從從屬站系統傳輸到 Web 伺服器的所有資料，以及從 Web 伺服器傳送給從屬站系統的所有資料。大多數 Web 伺服器均支援 SSL，如 Internet Connection Secure Server, 版本 2 版次 2 或更高的版本，以及 Lotus Domino Go Webserver, 4.6.1 或更高的版本。

使用身份驗證

*身份驗證*係用來確定產生 Net.Data 要求的使用者 ID 有權存取和更新應用程式內的資料。身份驗證是一種使使用者 ID 與密碼相配的處理，以便驗證要求的確來自有效的使用者 ID。 Web 伺服器會將使用者 ID 與它所處理的每個 Net.Data 要求產生關聯。然後，處理此要求的處理或緒就可存取該使用者 ID 被授權的任何資源。

您可使用支援二種類型的身份驗證：一種是保護伺服器上的某些目錄，一種是保護您的資料庫。

- 大多數的 Web 伺服器可讓您設定要保護的伺服器目錄。您也可以讓系統要求使用者先輸入使用者 ID 及通行碼，才能存取到您所指定之目錄中的檔案。請參閱符合您 web 伺服器的*管理者手冊*，以決定您的系統功能。
- DB2 有一個有關資料庫存取的身份驗證系統，這個系統可限制只有某些使用者才可存取表格及直欄。您可以使用 Net.Data 的特殊變數 (如 LOGIN 及 PASSWORD)，來鏈結 DB2 身份驗證常式。

要訣: 若要保護 Net.Data 巨集，請執行下列：

1. 在 Net.Data 程式物件的 Web 伺服器架構檔中新增保護指令。
2. 確定 Net.Data 執行時所依據的使用者 ID 有權存取巨集檔案。 有關授與存取權的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

使用授權

*授權*提供使用者對於物件、資源或函數的完整或有限存取權。如 DB2 的資料來源會提供它們自己的授權機制，來保護它們管理的資訊。這些授權機制會假定與執行 Net.Data 要求的處理有關聯的使用者 ID 已經過適當的身份驗證，如第53頁的『使用身份驗證』中描述一般。然後，這些資料來源的舊有存取控制機制，會依據授權使用者 ID 所保留的授權來允許或拒絕存取。

使用 Net.Data 機制

除了上面描述的方法以外，您可以使用 Net.Data 架構變數或巨集開發技術，來限制一般使用者的活動、隱藏公司資產 (如資料庫的設計)，以及確認使用者在生產環境中提供的輸入值。

Net.Data 架構變數

Net.Data 會提供數個架構變數，您可以使用它們來限制一般使用者的活動，或隱藏資料庫的設計。

使用路徑陳述式控制檔案存取

Net.Data 會評估路徑架構陳述式的設定，以決定 Net.Data 巨集所用的檔案及可執行程式位置。這些路徑陳述式會識別當嘗試尋找巨集檔案、可執行檔、併入檔或其他純文字檔時，Net.Data 可搜尋的一個或多個目錄。藉由在這些路徑陳述式上自由選擇併入目錄，您可明確地控制使用者可從瀏覽器上存取的檔案。請參閱第5頁的『第2章 架構 Net.Data』，以取得路徑陳述式的其它詳細資料。

您也應該使用『使用授權』中所描述的授權檢查，以及驗證無法在第55頁的『巨集開發技術』中所描述的 INCLUDE 陳述式中變更檔名。

停用生產系統的 SHOWSQL

SHOWSQL 變數容許使用者指定 Net.Data 可在 Web 瀏覽器中顯示 Net.Data 函數內指定的 SQL 陳述式。這個變數主要是用於開發及測試應用程式內的 SQL，不是打算用於生產系統中 。

您可以使用下列方法之一，讓 SQL 陳述式無法顯示在生產系統中。

- 當使用 Net.Data 版本 2.0.7 或更新版本時，請在起始設定檔中使用 DTW_SHOWSQL 架構變數，來置換在您的 Net.Data 巨集中設定 SHOWSQL 的效果。請參閱第15頁的『DTW_SHOWSQL：啓用或停用 SHOWSQL 架構變數』，取得語法及其他資訊。
- Net.Data 版本 2.0.5 及早期版本的使用者可以使第55頁的『巨集開發技術』中描述的 DTW_ASSIGN() 函數。

請參閱 *Net.Data 參考手冊*的變數一章中的 SHOWSQL，取得 SHOWSQL Net.Data 變數的語法及範例。

請考量對生產系統啓用直接要求是否恰當

呼叫 Net.Data 的直接要求方法容許使用者指定直接從 URL 內執行 SQL 陳述式或 Perl、 REXX 或 C 程式。巨集要求方法容許使用者僅執行在巨集中定義或呼叫的那些 SQL 陳述式及函數。

您應該謹慎考慮是否容許直接要求的使用，因爲它能夠給與使用執行非常多的函數。當啓用這種呼叫方法時，請確定處理 Net.Data 要求時所用的使用者 ID 具有適當的授權層次。

您可以使用 DTW_DIRECT_REQUEST 架構變數，來停用直接要求。請參閱 第13頁的『DTW_DIRECT_REQUEST：啓用直接要求變數』，取得語法及其他資訊

巨集開發技術

Net.Data 會提供數種機制，容許使用指定值給輸入變數。若要確定巨集是以您想要的方式執行，巨集應該驗證這些輸入變數。在設計您的資料庫及應用程式時，您也應該限制使用者對有權看到的資料的存取權。

當撰寫 Net.Data 巨集時，請使用下列開發技術。這些技術將協助您確保您的應用程式是按照想要的方式執行，且僅有適當授權的使用者可存取資料。

確定無法在 URL 中置換 Net.Data 變數

使用者在 URL 中設定的 Net.Data 變數值將置換用來起始設定巨集中的變數的 DEFINE 陳述式。這可能會改變您的巨集的執行方式。若要避免這種可能性，請使用 DTW_ASSIGN() 函數起始設定 Net.Data 變數。

範例：不要使用 `%DEFINE SHOWSQL="NO"` 來設定 Net.Data SHOWSQL 變數，請使用 `@DTW_ASSIGN(SHOWSQL, "NO")`。然後，如 `SHOWSQL=YES` 查詢字串指定將不會置換巨集設定。

您可以使用下列方法之一，讓 SQL 陳述式無法顯示在生產系統中。

- 當使用支援 DTW_SHOWSQL 架構變數的 Net.Data 版本時，請在 Net.Data 起始設定檔中使用這個變數，來置換在您的 Net.Data 巨集中設定 SHOWSQL 的效果。請參閱第15頁的『DTW_SHOWSQL：啓用或停用 SHOWSQL 架構變數』，取得語法及其他資訊。
- 請使用上面範例中所描述的 DTW_ASSIGN() 函數，來指定 SHOWSQL 的值，防止它被置換。

請參閱 *Net.Data 參考手冊*的變數一章中的 SHOWSQL，取得 SHOWSQL Net.Data 變數的語法及範例。

您也可以使用 DTW_ASSIGN，來確保其他 Net.Data 變數 (如 RPT_MAX_ROWS 或 START_ROW_NUM) 不會被置換。請參閱 *Net.Data 參考手冊*中有關變數的那一章，取得這些變數的詳細資訊。

確認無法以更改應用程式行為的方式來修改您的 SQL 陳述式。

新增 Net.Data 變數到巨集內的 SQL 陳述式，將容許使用者在執行 SQL 陳述式之前，可動態更改它。驗證使用者提供的輸入值是巨集撰寫者的責任，且他要確保含有變數參照的 SQL 陳述式不會遭到不預期的修改。您的 Net.Data 應用程式應該確認使用者從 URL 提供的輸入值，以便 Net.Data 應用程式可以拒絕無效的輸入。您的驗證設計處理應該包括下列步驟：

1. 識別有效輸入的語法；例如，客戶 ID 須以字母開頭，且僅能含有英數字元。
2. 判斷容許不正確輸入、有意的破壞輸入或想取得 Net.Data 應用程式的內部資產的輸入，可能帶來的傷害。

3. 在巨集中包括必須符合應用程式需要的輸入驗證陳述式。如此的驗證取決於輸入的語法及使用方式。在較簡單的情況中，它足以檢查出輸入中的無效內容，或呼叫 Net.Data 來驗證輸入類型。如果輸入的語法相當複雜，則巨集開發者可能必須局部或完整剖析輸入，方可驗證它是否有效。

範例 1：使用 DTW_POS() 字串函數來驗證 SQL 陳述式

```
%FUNCTION(DTW_SQL) query1() {
   select * from shopper where shlogid = '$(shlogid)'
%}
```

shlogid 變數的值應該是購物者 ID。它的目的在於將 SELECT 陳述式傳回的橫列限制爲含有購物者 ID 識別的購物者的資訊的橫列。不過，如果傳遞字串『smith' or shlogid<>'smith』 作爲變數 shlogid 的值，則查詢將變成：

```
select * from shopper where shlogid = 'smith' or shlogid<>'smith'
```

這個使用者修改過的原始 SQL SELECT 陳述式將傳回整個購物者表格。

Net.Data 字串函數可用來驗證使用者不會以不適當方式修改 SQL 陳述式。例如，下列邏輯可用來確保與 shlogid 變數有關聯的輸入值是由單一購物者 ID 構成的：

```
@DTW_POS(" ", $(shlogid), result)
%IF (result == "0")
  @query1()
%ELSE
  %{ perform some sort of error processing %}
%ENDIF
```

範例 2：使用 DTW_TRANSLATE()

假設您的應用程式需要驗證輸入變數 number_of_orders 提供的值是一個整數。其中一種方法就是建立一個轉換表 input_translation_table，含有所有鍵盤字元，但數值字元 0-9 除外，並使用 DTW_TRANSLATE 及 DTW_POS 字串函數來驗證輸入：

```
@DTW_TRANSLATE(number_of_orders, "x", input_translation_table, "x", string_out)

 @DTW_POS("x", string_out, result)
     %IF (result = "0")
       %{ continue with normal processing %}
%ELSE
       %{ perform some sort of error processing %}
%ENDIF
```

請注意，使用者無法在 Web 瀏覽器中修改儲存程序內的 SQL 陳述式，且使用者提供的輸入參數會受到與輸入參數有關聯的 SQL 資料類型的限制。在使用 Net.Data 字串函數來驗證使用者輸入值不是很實際的情況中，您可以使用儲存程序。

確認無法以更改應用程式行為的方式來修改 INCLUDE 陳述式中的檔名。

如果您使用 Net.Data 變數，透過 INCLUDE 陳述式指定檔名的值，則將不會決定要併入的檔案，直到執行 INCLUDE 檔爲止。如果您打算在巨集內設定這個變數的值，但不容許瀏覽器的使用者置換巨集提供的值，則您應該使用 DTW_ASSIGN 而非使用 DEFINE 來設定變數值。如果您打算允許瀏覽器的使用者提供檔名，則您的巨集應該驗證提供的值。

範例：查詢字串指定 (如 `filename="../../x"`) 可能會造成併入的檔案是來自不是在 INCLUDE_PATH 架構陳述式中指定的目錄。假設您的 Net.Data 起始設定檔含有下列路徑架構陳述式：

```
INCLUDE_PATH /usr/lpp/netdata/include
```

且您的 Net.Data 巨集含有下列 INCLUDE 陳述式：

```
%INCLUDE "$(filename)"
```

`filename="../../x"` 的查詢字串指定將併入檔案 `/usr/lpp/x`，這不是 INCLUDE_PATH 架構陳述式規格想要的檔案。

Net.Data 字串函數可用來驗證提供的檔名是否適合應用程式。例如，下列邏輯可用來確保與檔名變數有關聯的輸入值沒有字串 "..."：

```
@DTW_POS("..", $(filename), result)
%IF (result > "0")
  %{ perform some sort of error processing %}
%ELSE
%{ continue with normal processing %}
%ENDIF
```

設計您的資料庫及查詢，以便使用者要求無權存取其他使用者的敏感資料。

有些資料庫設計會將敏感資料收集在單一表格中。除非 SQL SELECT 要求會以某種形式來加以限定，否則這種方法可能會使得 Web 瀏覽器的使用者取得所有敏感資料。

範例：下列 SQL 陳述式將傳回變數 `order_rn` 所識別的訂單的訂單資訊：

```
select setsstatcode, setsfailtype, mestname
from   merchant, setstatus
where  merfnbr   = setsmenbr
and    setsornbr = $(order_rn)
```

這種方法將容許瀏覽器中的使用者指定隨機訂單號碼，且可能取得其他客戶訂單的敏感資料。保護這種外洩的方法就是製作下列變更：

- 新增直欄到訂單資訊表格中，然後使用它來識別與特定橫列內的訂單資訊有關聯的客戶。
- 修改 SQL SELECT 陳述式，確定瀏覽器的使用者提供的鑑定客戶 ID 會限定 SELECT。

例如，如果 `shlogid` 是含有與訂單有關聯的客戶 ID 的直欄，且 SESSION_ID 是含有瀏覽器的使用者的鑑定 ID 的 Net.Data 變數，則您可以用下列陳述式置換先前的 SELECT 陳述式：

```
select setsstatcode, setsfailtype, mestname
  from   merchant, setstatus
  where  merfnbr   = setsmenbr
  and    setsornbr = $(order_rn)
  and    shlogid   = $(SESSION_ID)
```

使用 Net.Data 隱藏變數

您可以使用 Net.Data 隱藏變數來隱藏 Net.Data 巨集的不同性質，讓使用者無法用他們的 Web 瀏覽器來顯示您的 HTML 原始檔案。 例如，您可以隱藏您資料庫的內部結構。請參閱第90頁的『隱藏變數』，取得隱藏變數的詳細資訊。

自使用者要求驗證資訊

您可以依據使用者提供的輸入來建立自己的保護方案。例如，您可以透過

HTML 套表向使用者要求驗證資訊，並使用您的 Net.Data 巨集從資料庫取回的資料，或者從 Net.Data 巨集中定義的函數呼叫外部程式，來驗證它。

至於保護資產的詳細資訊，請參閱常見問題 (FAQ) 中有關 Internet 安全列示的部份，其 Web 網址如下：

```
http://www.w3.org/Security/Faq
```

第4章 呼叫 Net.Data

本章將描述您將如何使用不同 Web 伺服器介面來呼叫 Net.Data。在您可以使用其中一個呼叫方法之前，首先必須針對指定的介面來架構 Net.Data。您可以架構 Net.Data，來使用下列 Web 伺服器介面：

- 通用閘道介面 (CGI)
- FastCGI
- Lotus Domino Go Web 伺服器 (GWAPI)
- Internet Connection Server (ICAPI)
- Netscape Server (NSAPI)
- Microsoft Internet Server (ISAPI)
- Java servlet

請參閱第5頁的『第2章 架構 Net.Data』，瞭解如何替這些介面架構 Net.Data。依據預設值，Web 伺服器會將 Net.Data 呼叫成通用閘道介面程式，且每一 Net.Data 要求會在新的及個別處理中執行。當您架構 Web 伺服器時，您可以決定 Net.Data 的呼叫方式。

下列段落將描述 Net.Data 接受的要求類型，以及您可以用來透過不同 API 及 servlet 呼叫 Net.Data 的方法。

- 『呼叫使用的類型』
- 第68頁的『透過 Web 伺服器 API 呼叫 Net.Data』
- 第69頁的『使用 Java servlet 及 JavaBean 呼叫 Net.Data』

呼叫使用的類型

不管您是使用何種方法來呼叫 Net.Data，有兩個可指定的要求類型，視您想要執行巨集，或想要執行單一 SQL 陳述式、儲存程序或函數而定。

巨集要求

指定 Net.Data 執行指定的巨集。

直接要求

指定 Net.Data 執行 SQL 陳述式、儲存程序或函數。要求會指定：

- 語言環境的名稱。
- SQL 陳述式或函數名稱，以及呼叫函數所需的任意參數值。
- 呼叫 SQL 陳述式或函數所需的套表資料

想要撰寫單一 SQL 查詢或呼叫單一函數 (如 DB2 儲存程序、REXX 程式或 Perl 函數) 的 Web 軟體開發者，可對資料庫發出直接要求。直接要求並沒有任何需要 Net.Data 巨集的複雜 Net.Data 應用程式邏輯，因此會略過 Net.Data 巨集處理器。直接要求參數會被傳送給適當的語言環境來處理，以提供執行效能。

圖18 舉例說明了巨集要求與直接要求之間的差異。 巨集要求一定會在此要求的 URL 中指定巨集，且可使用套表資料。 直接要求從不會在 URL 中指定巨集，但仍可使用套表資料。

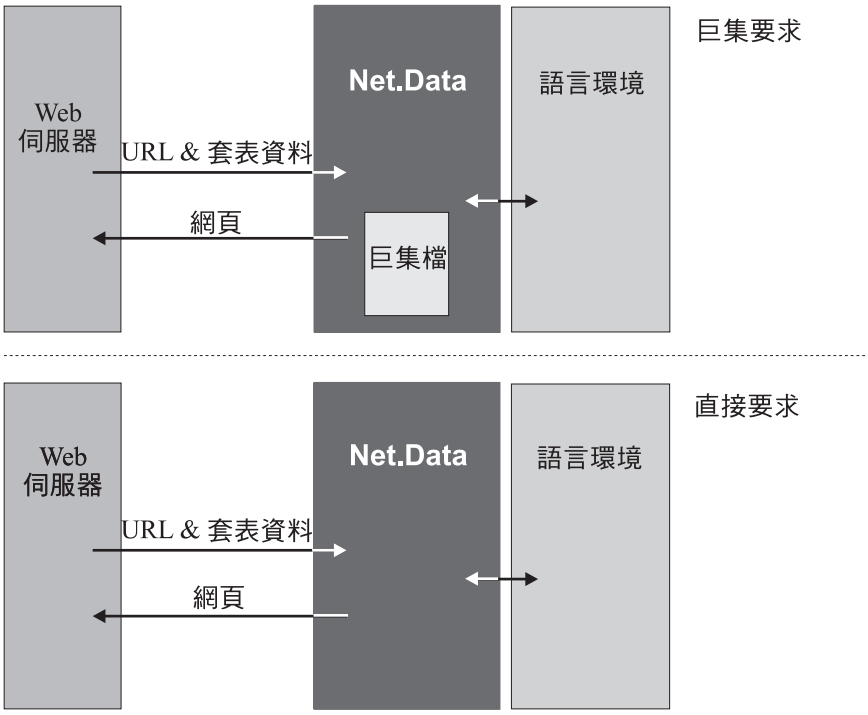


圖 18. 巨集要求與直接要求

呼叫 Net.Data 的語法會依據 Net.Data 的架構方式及您所產生的要求類型而異。 對於巨集及直接要求，均會使用 URL 來呼叫 Net.Data。使用者可以直接輸入 URL，或是可以在 HTML 頁面中編碼，作爲 HTML 鏈結或 HTML 套表。 Web 伺服器會使用下列來呼叫 Net.Data：CGI, FastCGI 或其中一個 Web 伺服器 API。此外，您可以使用 Net.Data servlet 來呼叫呼叫 Net.Data。

對於巨集要求，請在 URL 內指定 Net.Data 巨集的名稱及將在 Net.Data 巨集內執行的 HTML 區塊的名稱。對於直接要求，請在 URL 內指定 Net.Data 語言環境的名稱、 SQL 陳述式或函數名稱，以及任何其他必要的參數值。您可以使用 Net.Data 定義的語法，來指定這些值。

下列段落將詳細描述這些呼叫要求：

- 第61頁的『使用巨集呼叫 Net.Data (巨集要求)』
- 第63頁的『不使用巨集呼叫 Net.Data (直接要求)』

雖然範例會指定當使用 CGI 呼叫 Net.Data 時將使用的語法，但概念可引用到用來呼叫 Net.Data 的所有介面。若要取得每一介面類型的完整必要語法，請參閱每一介面專有的段落。

- 第68頁的『透過 Web 伺服器 API 呼叫 Net.Data』
- 第69頁的『使用 Java servlet 及 JavaBean 呼叫 Net.Data』

使用巨集呼叫 Net.Data (巨集要求)

這節說明如何透過指定巨集來呼叫 Net.Data。

下列語法陳述式顯示如何呼叫 Net.Data。

- URL：

```
http://server/Net.Data_invocation_path
 /filename/block[?name=val&...]
```

參數：

server 指定 Web 伺服器的名稱及路徑。若伺服器爲區域伺服器，您可以省略伺服器名稱，而使用相關的 URL。

Net.Data_invocation_path
Net.Data 可執行檔、servlet 類別、DLL 或共用程式庫的路徑與檔名。例如，/cgi-bin/db2www/。

filename
指定 Net.Data 的巨集檔名稱。Net.Data 會進行搜尋，並嘗試使這個檔名符合 MACRO_PATH 起始設定路徑變數中定義的路徑陳述式。請參閱第17頁的『MACRO_PATH』，以取得有關的詳細資訊。

block 指定參照的 Net.Data 巨集中的 HTML 區塊名稱。

method 指定與此套表搭配使用的 HTML 方法。

?name=val&...
指定要傳送給 Net.Data 的一或多個選用性參數。

然後，您可以在瀏覽器中直接指定 URL 目錄，或您可以在如下的 HTML 鏈結或格式中使用它：

- HTML 鏈結：

```
<A HREF="URL">任何文字</A>
```

- HTML 套表：

```
<FORM METHOD=方法 ACTION="URL">任何文字</FORM>
```

參數：

method 指定與此套表搭配使用的 HTML 方法。

URL 指定用來執行 Net.Data 巨集的 URL，這些是上面描述的參數。

範例

下列範例列出呼叫 Net.Data 的各種方法：

範例 1：使用 HTML 鏈結呼叫 Net.Data：

```
<A HREF="http://server/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</A>
```

範例 2：使用套表呼叫 Net.Data

```
<FORM METHOD=POST
 ACTION="http://server/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</FORM>
```

下列段落描述 HTML 鏈結與套表，以及如何使用它們呼叫 Net.Data 的詳細資訊：

- 『HTML 鏈結』
- 『HTML 套表』

HTML 鏈結

如果您正在製作 Web 網頁，則您可以建立一個 HTML 鏈結，造成 HTML 區塊的執行。使用者在瀏覽器上按一下定義爲 HTML 鏈結的文字或影像時， Net.Data 會執行巨集中的 HTML 區塊。

若要建立 HTML 鏈結，請使用 HTML <a> 標籤。決定哪些文字或圖形，您想要使用它們作爲 Net.Data 巨集的超鏈結，然後以 <a> 及 標籤來圍住它。在 <a> 標籤的 HREF 屬性中，請指定巨集及 HTML 區塊。

下列範例顯示當使用者在網頁上選取文字「列示所有監視器」時，可執行 SQL 查詢的鏈結。

```
<a href="http://server/cgi-bin/db2www/listA.d2w/report">
列示所有監視器</a>
```

按一下鏈結即可呼叫名爲 listA.d2w 的巨集，它具有名爲 ″report″ 的 HTML 區塊，如同下列範例所示一般：

```
%DEFINE DATABASE="MNS97"


 %FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='MONITOR'
%}

%HTML(report){
@myQuery()
%}
```

此查詢會傳回一個表格，內容包含由 EQPTABLE 表格內所描述之每種監視器的型號、成本及說明資訊。這個範例會藉由因產生預設報告來顯示查詢結果。請參閱第104頁的『報告區塊』，以取得如何使用 REPORT 區塊來自行設定報告的資訊。

HTML 套表

您可使用 HTML 套表動態地自行設定 Net.Data 巨集的執行。套表可讓使用者提供會影響巨集執行，及 Net.Data 所建置之網頁內容的輸入值。

下列範例藉由讓使用者在瀏覽器中使用簡式 HTML 套表，以選取所要顯示的產品類型資訊，來建置『HTML 鏈結』中的監視器列示範例。

```
<H1>硬體查詢套表</H1>
<HR>
<FORM METHOD=POST ACTION="/cgi-bin/db2www/equiplst.d2w/report">
```

```
<P>您要查閱哪類硬體？
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="MON" checked> 監視器
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PNT"> 指標裝置
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PRT"> 印表機
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="SCN"> 掃描器
</MENU>

<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
```

使用者在瀏覽器上選擇並按一下 Submit 按鈕後， Web 伺服器就會處理用來呼叫 Net.Data 之 FORM 標籤的 ACTION 參數。然後， Net.Data 會在 equiplst.d2w 巨集中執行 HTML 報告區塊：

```
%DEFINE DATABASE="MNS97"

 %FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='$(hdware)'
%REPORT{
<H3>這是您所要求的列示</H3>
%ROW{
<HR>
$(N1): $(V1), $(N2): $(V2)
<P>$(N3): $(V3)
%}
%}
%}

%HTML(report){
@myQuery()
%}
```

在上面範例中，SQL 陳述式中的 TYPE=$(hdware) 的值取自於 HTML套表輸入。

請參閱 *Net.Data 參考手冊*，以取得 ROW 區塊中所用之變數的詳細說明。

不使用巨集呼叫 Net.Data (直接要求)

本段落將告訴您如何使用*直接要求*來呼叫 Net.Data。當您使用直接要求時，您不必在 URL 中指定巨集的名稱。相反地，您將使用 Net.Data 定義的語法，指定 Net.Data 語言環境名稱、將執行的 SQL 陳述式或程式，以及 URL 中任何其他的必要參數值。請參閱第13頁的『DTW_DIRECT_REQUEST：啓用直接要求變數』，學習如何啓用或停用直接要求。

SQL 陳述式或程式以及其它任意指定的參數會直接傳送到所指定的語言環境，以供處理。因爲 Net.Data 不需要讀取與處理巨集，所以直接要求可提高執行效能。SQL、ODBC、Oracle、Sybase、Java、System、Perl 及 REXX Net.Data 提供的語言環境都支援直接要求，且您可使用 URL、HTML 套表或鏈結來呼叫 Net.Data。

直接要求會經由傳遞 URL 或套表資料的查詢字串中的參數來呼叫 Net.Data。下列範例舉例說明了您可指定直接要求的上下文。

```
<A HREF="http://server/cgi-bin/db2www/?direct_request">任何文字</A>
```

其中的 *direct_request* 表示直接要求的語法。例如，下列的 HTML 鏈結包含直接要求：

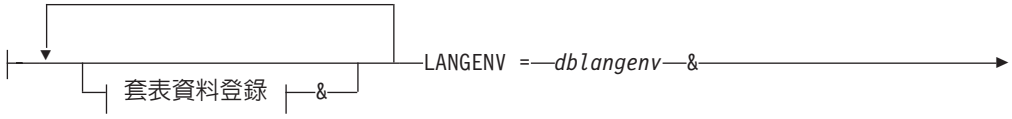
```
<A HREF="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</A>
```

直接要求語法

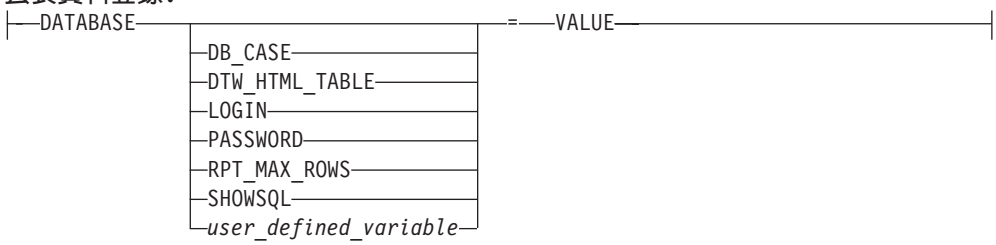
使用直接要求來呼叫 Net.Data 的語法，可含有資料庫或非資料庫語言環境的呼叫。

語法

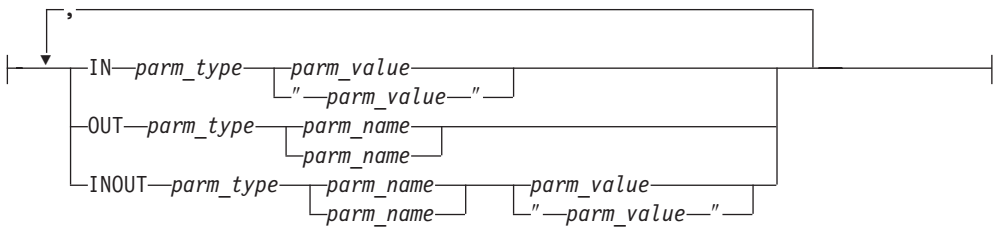
資料庫語言環境呼叫：



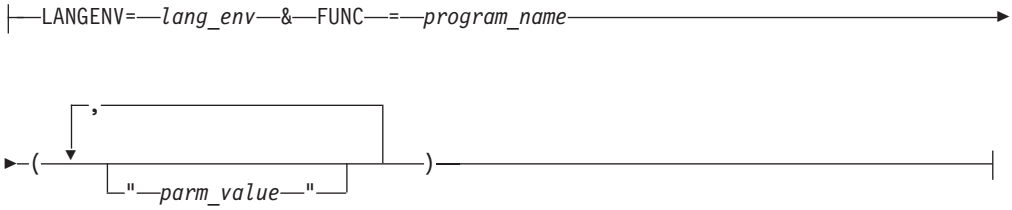
套表資料登錄：



參數列示：



非資料庫語言環境呼叫：



參數

資料庫語言環境呼叫

指定對 Net.Data 直接要求，由 Net.Data 呼叫資料庫語言環境。

套表資料登錄

可讓您指定 SQL 變數的設定，或要求簡易 HTML 格式的參數。請參閱 *Net.Data 參考手冊*中變數的章節，以學習更多關於這些變數的資訊。

DATABASE
指定 Net.Data 應傳送 SQL 要求的目的資料庫。這個參數是必要的。

DB_CASE
指定 SQL 陳述式的大小寫。

DTW_HTML_TABLE
指定 Net.Data 應該傳回 HTML 表格或預先格式化的文字表格。

LOGIN
指定資料庫使用者 ID。

PASSWORD
指定資料庫密碼。

RPT_MAX_ROWS
指定在報告中函數所傳回之表格的最大列數。

SHOWSQL
指定 Net.Data 是否應隱藏或顯示執行中的 SQL 陳述式。

START_ROW_NUM
指定函數的表格中哪一個列號將作爲報表的開頭。

user_defined_variable
傳送給 Net.Data 並提供必要資訊，或影響 Net.Data 行爲的變數。使用者定義的變數是爲您的應用程式而定義的變數。

VALUE
指定 Net.Data 變數的值。

LANGENV

指定 SQL 陳述式或儲存程序呼叫的目標語言環境。如果語言環境是一種資料庫語言環境，則也必須指定資料庫名稱。

dblangenv
資料庫語言環境的名稱：

- DTW_SQL
- DTW_ODBC

- DTW_ORA
- DTW_SYB

SQL
表示直接要求可指定列入 SQL 陳述式的執行。

sql_stmt
指定一字串，此字串含有任何可用動態 SQL 執行的有效 SQL 陳述式。

FUNC
表示直接要求可指定儲存程序的執行。

stored_proc_name
指定任何有效的 DB2 儲存程序名稱。

parm_type
指定 DB2 儲存程序的任意有效參數類型。

parm_name
指定任意有效的參數名稱。

parm_value
指定 DB2 儲存程序的任意有效參數值。

IN 指定 Net.Data 應使用此參數將輸入資料傳送給儲存程序。

INOUT
指定 Net.Data 應使用此參數將輸入資料傳送給儲存程序，並從語言環境傳回輸出資料。

OUT
指定語言環境應使用此參數，以從儲存程序傳回輸出資料。

非資料庫語言環境呼叫
指定直接要求給 Net.Data，由 Net.Data 呼叫非資料庫語言環境。

LANGENV
指定目標語言環境，以執行此函數。

lang_env
指定非資料庫語言環境的名稱：

- DTW_PERL
- DTW_REXX
- DTW_SYSTEM

FUNC
表示直接要求可指定程式的執行。

program_name
指定含有要執行之程式的檔案。

parm_value
指定函數的任意有效參數值。

直接要求範例

下列範例顯示在使用直接要求方法時，可用來呼叫 Net.Data 的不同方式。

HTML 鏈結: 下列範例使用直接要求，透過鏈結呼叫 Net.Data。

範例 1：一個鏈結，可呼叫 Perl 語言環境，並呼叫位在 Net.Data 起始設定檔案之 EXEC 路徑陳述式中的 Perl script

```
<A HREF="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</A>
```

範例 2：一個鏈結，可呼叫 Perl 語言環境 (如前例中所述)，但傳送的字串中，URL 內的雙引號與空格字元已經過編碼

```
<A HREF="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl
     (%22Hello+World%22)">any text</A>
```

要訣: 您必須將 URL 中的特定字元 (如空格及雙引號) 加以編碼。在這個範例中，參數值中的雙引號字元及空格必須分別編碼成 %22 或 + 字元。您可以使用內建函數 DTW_URLESCSEQ，來編碼必須在 URL 內編碼的任何文字。DTW_URLESCSEQ 函數的詳細資訊，請參閱它在 *Net.Data 參考手冊*中的說明。

HTML 套表: 下列範例使用直接要求，透過套表呼叫 Net.Data。

範例 1：一種 HTML 套表，可使用 SQL 語言環境來執行 SQL 查詢、連接 CELDIAL 資料庫，及查詢表格

```
<FORM METHOD="POST"
 ACTION="http://server/cgi-bin/db2www/">
<INPUT TYPE=hidden NAME="LANGENV" VALUE="DTW_SQL">
<INPUT TYPE=hidden NAME="DATABASE" VALUE="CELDIAL"
 <INPUT TYPE=hidden NAME="SQL" VALUE="select * from Table1 where col1=$(InputName)">
Enter Customer name:
<INPUT TYPE=text NAME="InputName" VALUE="John">
<INPUT TYPE=SUBMIT>
</FORM>
```

這個範例含有 SQL 陳述式中的替代變數，它可以使得 WHERE 子句更具變化。

URL: 下列範例使用直接要求，透過 URl 呼叫 Net.Data。

範例 1：可使用 SQL 語言環境來執行 SQL 查詢的 URL

```
http://server/cgi-bin/db2www/?LANGENV=DTW_SQL&DATABASE=CELDIAL
 &SQL=select+*+from+customer
```

範例 2：一個 URL，可呼叫 Perl 語言環境，並呼叫不在 Net.Data 起始設定檔案之 EXEC 路徑陳述式中的可執行檔。

```
http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=/u/MYDIR/macros/myexec.pl
```

範例 3：呼叫 System 語言環境，並呼叫外部 Perl script 的 URL

```
http://server/cgi-bin/db2www/?LANGENV=DTW_SYSTEM&FUNC=perl+/u/MYDIR/macros/myexec.pl
```

範例 4：呼叫 REXX 語言環境、呼叫 REXX 程式，並將參數傳送給此程式的 URL

```
http://server/cgi-bin/db2www/?LANGENV=DTW_REXX&FUNC=myexec.cmd(parm1,parm2)
```

範例 5：呼叫儲存程序並將參數傳送給 SQL 語言環境的 URL

```
http://server/cgi-bin/db2www/?LANGENV=DTW_SQL&FUNC=MY_STORED_PROC
   (IN+CHAR(30)+Salaries)&DATABASE=CELDIAL
```

透過 Web 伺服器 API 呼叫 Net.Data

Net.Data 會支援下列列示中的 Web API，取決於您的作業系統而定：

GWAPI plug-in 與 ICAPI plug-in
Lotus Domino Go Webserver API plug-in 是在 IBM Internet Connection Secure Sever plug-in 之後開發的 plug-in

ISAPI plug-in
Microsoft Internet Server API plug-in

NSAPI plug-in
Netscape Server API plug-in

請參閱 *Net.Data 參考手冊*中的作業系統參考附錄，來判斷您的作業系統支援哪些 Web 伺服器 API。請參閱 第36頁的『架構 Net.Data 來使用 Web 伺服器 API』，學習如何架構 Net.Data 及 Web 伺服器，與 API 搭配使用。

需求：

- Net.Data 若在 GWAPI, ICAPI, ISAPI 或 NSAPI 模式中執行，請重新啓動 Web 伺服器，以便 Web 伺服器可以重新載入 Net.Data，並將它當作一個處理來處理。
- 在 Web 伺服器呼叫 API 模式中的 Net.Data 後，若您對起始設定檔案做了變更，則須重新啓動 Web 伺服器。任何對 Net.Data 起始設定檔案 (db2www.ini) 的變更均沒有效果。在 API 模式中， Net.Data 僅會讀取起始設定檔案一次，以便減少效能的額外負擔。
- 在 API 模式中執行時，Oracle 與 Sybase 語言環境需要「現場連線」。

若要呼叫 Web 伺服器 API：

ICAPI 及 GWAPI：

語法：

```
http://server_name/CGI-BIN/db2www/macro_name/html_block
```

參數：

server_name
伺服器名稱。

macro_name
位於 MACRO_PATH 設定的目錄下的巨集的相對路徑名稱。

html_block
在巨集中要處理的 HTML 區塊名稱。

範例：

```
http://myserver/CGI-BIN/db2www/mymacro.d2w/report
```

ISAPI：

語法：

```
http://server_name/server_HTML_root_directory/dll_name/macro_name/
  html_block
```

參數：

server_name
伺服器名稱。

server_HTML_root_directory
Web 伺服器 HTML 根目錄名稱。

dll_name
Net.Data 的 ISAPI .dll 檔名 `dtwisapi.dll`。

macro_name
位於 MACRO_PATH 設定的目錄下的巨集的相對路徑名稱。

html_block
在巨集中要處理的 HTML 區塊名稱。

範例：

```
http://myserver/scripts/dtwisapi.dll/mymacro.d2w/report
```

NSAPI：

語法：

```
http://server_name/macro_name/html_block
```

參數：

server_name
伺服器名稱。

macro_name
位於 MACRO_PATH 設定的目錄下的巨集的相對路徑名稱。巨集檔的副檔名 (如 `.d2w`) 必須定義在 Web伺服器架構檔中。請參閱第36頁的『架構 Net.Data 來使用 Web 伺服器 API』，以取得有關詳細資訊。

html_block
在巨集中要處理的 HTML 區塊名稱。

範例：

```
http://myserver/mymacro.d2w/report
```

使用 Java servlet 及 JavaBean 呼叫 Net.Data

servlet 是 Java 類別，可執行類似於 CGI 程式或 Web 伺服器 API plug-in 的角色。Web 伺服器可使用 servlet，動態建置 HTML 頁面。Servlet 並沒有它們自己的圖形使用者介面，但可在區域或透過網路，動態載入它們的類別。您可以使用 URL 位址 (遠端)，或使用類別名稱 (區域)，來呼叫它們。

Net.Data 會提供 servlet，您可以使用它們來呼叫 Net.Data 巨集、執行 Net.Data 函數，或在 Net.Data 支援之可使用 Java 的作業系統內，透過 Net.Data 來執行 SQL 陳述式。

這章說明下列項目的概念及作業：

『Net.Data servlet』

第75頁的『Net.Data JavaBean』

Net.Data servlet

Net.Data 提供含有 Net.Data 函數的 servlet 及 NetObjects Fusion plug-in，可用於 Java 環境。使用 servlet 及 plug-in，您可以：

- 從 URL 及當成 Server-Side-Include (SSI) 來執行 Net.Data 巨集
- 從 URL 及當成 SSI 來執行 Net.Data 函數
- 使用 NetObjects Fusion (NOF) 來管理您的巨集；可對您的網站管理提供較好的整合及容易使用的圖形式使用者介面，以開發簡易巨集。 請參閱第193頁的『附錄E. 搭配使用 NetObjects Fusion NOF Plug-in 與 Net.Data servlet』，以學習如何使用 NOF plug-in。

這節說明下列 servlet 主題：

- 『關於 Net.Data servlet』
- 第71頁的『執行 Net.Data servlet』

關於 Net.Data servlet

Net.Data 提供 servlet，以協助您使用 Java 環境來開發與管理巨集。servlet 是 Java 類別，可執行類似於 CGI 程式或 Web 伺服器 API plug-in 的角色。servlet 係被具有 Java servlet 能力的 Web 伺服器用來建立 HTML 頁面。servlet 並沒有它們自己的圖形使用者介面，但可區域性 (或透過網路) 並動態地載入它們的類別，且可使用 URL 位址 (遠端) 或類別名稱 (區域) 予以呼叫。 servlet 可供 Windows NT 與 AIX 作業系統使用。

Net.Data servlet 是 Java 式的外層，執行的是使用原始 DLL 檔的 Net.Data 2 版巨集或直接要求。這些 servlet 可讓您執行舊有的巨集 (MacroServlet) 或單一函數 (FunctionServlet)。要使用這些 servlet 必須要有 Net.Data版本 2。 Net.Data servlet 可執行資料庫或非資料庫語言環境，且可與「現場連線」搭配執行。

servlet 是隨 API 而來，可與應用程式搭配使用。 servlet API 的資訊是與 Net.Data 書寫在一起。請參閱 <*inst_dir*>/servlet/NetDataservlet.jar 檔，以取得 API 說明文件。

Net.Data 提供兩種 servlet：

巨集 servlet (com.ibm.netdata.servlet.Macroservlet)

執行舊有 Net.Data 巨集。

使用巨集 servlet 與透過 CGI-BIN 介面執行 Net.Data 巨集類似，不同的是您是透過 Java servlet 來執行巨集。巨集 servlet 需要安裝 Net.Data 版本 2 或更新版本。

使用 巨集 servlet 的優點包括：

- 使用「現場連線管理程式」透過 ODBC 執行 SQL 查詢，來提高執行效能。
- 您可透過 Server-Side-Include (SSI) 執行巨集，將多個巨集內嵌到 HTML 檔案中。

巨集 servlet 也容許異質資料庫 (如 DB2、Oracle 及 Sybase) 的原始存取，以及不同的語言環境 (如 Perl、REXX 及 Java)。

函數 servlet (com.ibm.netdata.servlet.Functionservlet)

透過 servlet 介面 (例如，%FUNCTION DTW_SQL()) 來執行 Net.Data 函數或 SQL 陳述式。有關的詳細資訊，請參閱第63頁的『不使用巨集呼叫 Net.Data (直接要求)』。

函數 servlet 需要安裝 Net.Data 版本 2。

執行 Net.Data servlet

Net.Data servlet 可用兩種方式執行：一種是從 URL，另一種是當成 HTML 檔案的 SSI。您可使用 NetObjects Fusion plug-in 將 Net.Data servlet 納入 NOF 地點中。下列各節討論的是如何藉由鍵入 servlet 的語法，來修改與執行 servlet。請參閱 第193頁的『附錄E. 搭配使用 NetObjects Fusion NOF Plug-in 與 Net.Data servlet』，以學習如何使用 NetObjects Fusion 來修改與執行 servlet。

- 『執行巨集 servlet』
- 第73頁的『執行函數 servlet』

執行巨集 servlet: 在 HTML 檔案中，請使用下列其中一個語法選項來輸入 servlet 參數：

1. URL：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet
  ?MACRO=macro_value&BLOCK=block_value&parmnn=valuenn
```

例如：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet?MACRO=my_macro
  &BLOCK=my_block&field1=custno
```

2. SSI：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
     <param name="MACRO" value="macro_value">
      <param name="BLOCK" value="block_value">
     <param name="parmnn" value="valuenn">
</servlet>
```

例如：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
     <param name="MACRO" value="my_macro.d2w">
     <param name="BLOCK" value="report">
<param name="field1" value="custno">
</servlet>
```

參數：

macro_value

舊有 Net.Data 巨集的完整路徑

block_value

位在將執行的指定 Net.Data 巨集中的 HTML 區塊名稱；預設值爲 `report` (可選用的)

parmnn

您的巨集需要的其他參數，如

```
<param name="field1" ...
```

valuenn

您的巨集需要的其他值，如

```
... value="custnum"
```

HTMLPATH 參數：如果獲得的錯誤訊息指出 HTMLPATH 參數遺失，請將 HTMLPATH 參數新增到 servlet 呼叫指令中：

- URL：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet
?MACRO=macro_name&BLOCK=block_value&htmlpath=html_path&parmnn=valuenn
```

例如：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet?MACRO=my_macro
  &BLOCK=my_blockhtmlpath=e:\html&field1=custno
```

- SSI：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
     <param name="MACRO" value="macro_value">
      <param name="BLOCK" value="block_value">
      <param name="htmlpath" value="html_path">
     <param name="parmnn" value="valuenn">
</servlet>
```

例如：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
<param name="MACRO" value="my_macro">
<param name="BLOCK" value="my_block">
<param name="htmlpath" value="e:\html">
<param name="field1" value="custno">
</servlet>
```

OUTBUFLEN 參數：如果您的巨集結果大於 32 KB，請須設定 OUTBUFLEN 參數。必要時若無法設定這些參數可能會造成無法預期的結果。

- URL：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet
?MACRO=macro_name&BLOCK=block_value
&OUTBUFLEN=output_buffer_size&parmnn=valuenn
```

例如：

```
http://myserver/servlet/com.ibm.netdata.servlet.Macroservlet?MACRO=my_macro
  &BLOCK=my_block&OUTBUFLEN=48&field1=custno
```

- SSI：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
     <param name="MACRO" value="macro_value">
      <param name="BLOCK" value="block_value">
<param name="OUTBUFLEN" value="output_buffer_size">
     <param name="parmnn" value="valuenn">
</servlet>
```

例如：

```
<servlet code="com.ibm.netdata.servlet.Macroservlet">
<param name="MACRO" value="my_macro">
<param name="BLOCK" value="my_block">
<param name="OUTBUFLEN" value="48K">
<param name="field1" value="custno">
</servlet>
```

執行函數 servlet: 函數 servlet 可以使用直接要求，呼叫 Net.Data 來執行函數 (如 REXX 函數) 或 SQL 陳述式。您對 servlet 設定的參數取決於您要執行函數或 SQL 陳述式而定。在 HTML 檔案中，請使用下列其中一個語法選項來輸入 servlet 參數：

1. URL：
   - ***呼叫函數：***

     ```
     http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
       ?LANGENV=lang_env_name&FUNC=program_name&parmnn=valuenn
     ```

     例如：

     ```
     http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
       ?LANGENV=DTW_REXX&FUNC=my_rexx&field1=custno
     ```

   - ***呼叫 SQL 陳述式：***

     ```
     http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
        ?LANGENV=database_lang_env_name&SQL=SQL_statement
        &DATABASE=database_name&parmnn=valuenn
     ```

     例如：

     ```
     http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
     ?LANGENV=DTW_SQL&SQL=select+*+from+myTable&DATABASE=CELDIAL
     ```

2. SSI：
   - ***呼叫函數：***

     ```
     <servlet code="com.ibm.netdata.servlet.Functionservlet">
          <param name="LANGENV" value="lang_env_name">
           <param name="FUNC" value="program_name">
          <param name="parmnn" value="valuenn">
     </servlet>
     ```

     例如：

     ```
     <servlet code="com.ibm.netdata.servlet.Functionservlet">
     <param name="LANGENV" value="DTW_REXX">
     <param name="FUNC" value="myREXX">
     <param name="field1" value="custno">
     </servlet>
     ```

   - ***呼叫 SQL 陳述式：***

     ```
     <servlet code="com.ibm.netdata.servlet.Functionservlet">
          <param name="LANGENV" value="lang_env_name">
           <param name="SQL" value="SQL_stmt_name">
           <param name="DATABASE" value="database_name">
          <param name="parmnn" value="valuenn">
     </servlet>
     ```

     例如：

     ```
     <servlet code="com.ibm.netdata.servlet.Functionservlet">
     <param name="LANGENV" value="DTW_SQL">
     <param name="SQL" value="select * from employee">
     <param name="DATABASE" value="CELDIAL">
     </servlet>
     ```

參數：

lang_env_name

被呼叫的 Net.Data 語言環境 (如 DTW_SQL, DTW_REXX)，以處理函數、SQL 陳述式或儲存程序。這個參數需要使用 FUNC 或 SQL。

program_name

含有要執行的函數的程式的名稱。例如，my_rexx；其中 my_rexx 是 REXX 可執行檔的名稱。 請使用 *parmnn* 及 *valuenn* 參數，來指定此函數的輸入參數。

database_name

與此 DATABASE 參數有關的資料庫名稱。指定的

SQL_stmt_name

存取資料庫的 SQL 陳述式或儲存程序名稱。例如，"select * from employee"。請使用 *parmnn*及 *valuenn* 參數，來指定 SQL 陳述式或儲存程序的輸入參數。

parmnn

您的巨集需要的其他參數，如 <param name="field1" ...。

valuenn

您的巨集需要的其他值，如 ... value="custnum"。

HTMLPATH 參數：如果獲得的錯誤訊息指出 HTMLPATH 參數遺失，請將 HTMLPATH 參數新增到 servlet 呼叫指令中：

- URL：

```
http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
?LANGENV=lang_env_name&FUNC=program_name&htmlpath=html_path
&parmnn=valuenn
```

例如：

```
http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
?LANGENV=DTW_REXX&FUNC=my_rexx&htmlpath=e:\html&field1=custno
```

- SSI：

```
<servlet code="com.ibm.netdata.servlet.Functionservlet">
     <param name="LANGENV" value="lang_env_name">
      <param name="SQL" value="SQL_stmt_name">
      <param name="htmlpath" value="html_path">
     <param name="parmnn" value="valuenn">
</servlet>
```

例如：

```
<servlet code="com.ibm.netdata.servlet.Functionservlet">
<param name="LANGENV" value="DTW_SQL">
<param name="SQL" value="select * from employee">
<param name="htmlpath" value="e:\html">
<param name="field1" value="custno">
<param name="DATABASE" value="SAMPLE">
</servlet>
```

其中 *html_path* 指定的是 Web 伺服器根 HTML 目錄的路徑；例如：htmlpath=e:\html。

OUTBUFLEN 參數：如果您的巨集結果大於 32 KB，則您必須設定 OUTBUFLEN 參數。 必要時若無法設定這些參數可能會造成無法預期的結果。

- URL：

```
http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet
?LANGENV=lang_env_name&FUNC=program_name&OUTBUFLEN=output_buffer_size&parmnn=valuenn
```

例如：

```
http://myserver/servlet/com.ibm.netdata.servlet.Functionservlet?LANGENV=DTW_REXX
  &FUNC=my_rexx&OUTBUFLEN=48&field1=custno
```

- SSI：

```
<servlet code="com.ibm.netdata.servlet.Functionservlet">
     <param name="LANGENV" value="lang_env_name">
      <param name="FUNC" value="program_name">
<param name="OUTBUFLEN" value="output_buffer_size">
     <param name="parmnn" value="valuenn">
</servlet>
```

例如：

```
<servlet code="com.ibm.netdata.servlet.Functionservlet">
<param name="LANGENV" value="DTW_REXX">
<param name="FUNC" value="my_rexx">
<param name="OUTBUFLEN" value="48K">
<param name="field1" value="custno">
</servlet>
```

Net.Data JavaBean

Net.Data 提供的 JavaBean 可不需執行 Web 伺服器，就可用於 Java 環境。JavaBean 是物件導向程式設計介面，可讓您建置可重複使用的應用程式或程式建置區塊。這些物件可用在啓用 Java 之作業系統的網路上。

使用原生的 Net.Data DLL，JavaBean 會呼叫 Net.Data，這樣會移入回覆碼及含有 Net.Data 輸出 (結果) 的字串。因爲 JavaBeans 使用本機 DLL，所以您不必使 Web 伺服器執行，便可使用 Net.Data 函數。

設計要訣：Net.Data JavaBeans 傳回的結果即是您的巨集或函數傳回的任何結果；通常，這是 HTML。 請考慮將這些結果傳送到瞭解 HTML，並可顯示這些結果的 HTML 式 JavaBean。

透過 JavaBeans，您可以：

- 執行 Net.Data 巨集
- 透過 Net.Data 來執行 SQL 陳述式

本段落描述下列 JavaBean 主題：

- 『關於 Net.Data JavaBean』
- 第76頁的『設定與執行 Net.Data JavaBean』

關於 Net.Data JavaBean

Net.Data 提供 JavaBean，以協助您使用 Java 環境來開發與管理巨集。JavaBeans 是 Java 物件，其提供有下列介面：

- 在 JavaBean 開發環境中使用時 (如 Lotus BeanMachine)，您可用提供的自訂程式將所要的構成要素連接在一起，以處理與顯示巨集或 SQL 陳述式的結果，並產生 Java applet。
- 使用 API 時，您可使用 JavaBean 將 Net.Data 功能提供給您自己的 Java applet 或應用程式。 API 文件在 *<inst_dir>*/beans/NetDataBeans.jar 中。

Net.Data 提供兩種類型的 JavaBean：

Net.Data 巨集 JavaBean
提供 Java 式介面，可透過 Net.Data 來執行舊有 Net.Data 巨集。

Net.Data SQL JavaBean
提供 Java 式介面，可透過 Net.Data 來執行 SQL 陳述式。

Net.Data JavaBean 是 Java 式外層，是透過使用原生 DLL 檔案的 Net.Data 來執行。兩者都需要安裝 Net.Data 版本 2 或更新版本及 JDK 版本 1.1 或更新版本。

設定與執行 Net.Data JavaBean

本段落描述如何使用 JavaBean 開發工具 (如 Bean 機器) 來設定及執行 Net.Data JavaBeans。使用開發工具的步驟是同屬的，所以您可使用您所選擇的工具。

- 『巨集 Bean』
- 『SQL Bean』

巨集 Bean: Net.Data 巨集 bean (即 com.ibm.netdata.beans.NetDataMacro) 可讓您使用 Java，來執行舊有的巨集。若要使用這個 bean，您需要將 Net.Data 性質指定給此 bean，這樣它才能與巨集搭配使用。

用 JavaBean 開發工具設定 Net.Data 巨集 JavaBean：

1. 將 <*inst_dir*>/beans/NetDataBeans.jar 檔案新增到或匯入到您的 JavaBean 開發工具中。
2. 使用開發工具自訂程式介面，可設定下列輸入性質：

   巨集 指定要執行的舊有巨集名稱。例如：MyMacro.mac

   區塊 指定要執行的 HTML 區塊區段名稱；預設值是 report。

   HTML 路徑
   指定 Net.Data db2www.ini 檔案的路徑。

   參數 指定執行巨集時所用的參數名稱及值。

   語法：

   ```
   name1=value1&nameN=valueN
   ```

用 JavaBean 開發工具執行 Net.Data 巨集 JavaBean：

1. 選取 JavaBean 開發工具提供的執行動作，以執行此巨集。
2. 執行巨集後，您可參考下列輸出性質：

   RC 指定 Net.Data 傳回的回覆碼。

   結果 指定因執行 Net.Data 巨集而傳回的資料。

SQL Bean: Net.Data SQL bean (即 com.ibm.netdata.beans.NetDataSQL) 可讓您使用 Java，透過 Net.Data 來執行 SQL 陳述式。若要使用這個 bean，您需要將 Net.Data 性質指定給此 bean，這樣它才能與巨集搭配使用。

JavaBean 開發工具設置 Net.Data SQL JavaBean ：

1. 將 NetDataBeans.jar 檔案新增或匯入到 JavaBean 開發工具。
2. 使用開發工具自訂程式介面，可設定下列輸入性質：

語言環境

設定要使用的語言環境；預設值爲 DTW_SQL。

SQL 指定要執行的 SQL 陳述式；預設值爲 select * from employee。

DATABASE

設定要使用的資料庫；預設值爲 SAMPLE。

HTML 路徑

指定 Net.Data db2www.ini 檔案的路徑。

參數 指定執行 SQL 陳述式時所用的參數名稱及值。

語法：

```
name1=value1&nameN=valueN
```

JavaBean 開發工具執行 Net.Data SQL JavaBean：

1. 選取 JavaBean 開發工具提供的執行動作，以執行此巨集。
2. 執行 SQL 陳述式後，您可使用存取元方法來參考下列輸出性質：

RC 指定 Net.Data 傳回的回覆碼。

結果 指定 SQL 陳述式傳回的資料。

第5章 開發 Net.Data 巨集

Net.Data 巨集是一個文字檔，包含一連串具下列用途的 Net.Data 巨集語言結構：

- 設定網頁的佈置
- 定義變數和函數
- 呼叫 Net.Data 的內建函數或定義在巨集的函數
- 將處理輸出製成格式並傳回到 Web 瀏覽器來顯示

Net.Data 巨集含有兩個組織部份：宣告部份與呈現部份，如圖19 中所顯示一般。

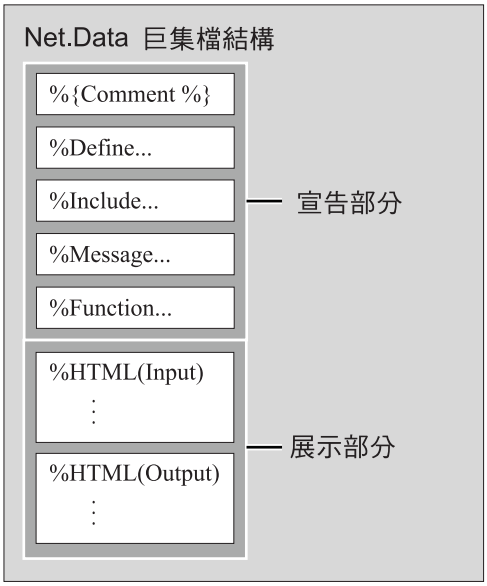


圖 19. 巨集結構

- *宣告部份*包含巨集中變數和函數的定義。
- *呈現部份*含有 HTML 區塊，它們會設定網頁的配置。 HTML 區塊是由 Web 瀏覽器支援的文字呈現陳述式所組成，如 HTML 與 JavaScript。

您可以任何順序來多次使用這些部分。關於巨集部份和結構的語法，請參閱 *Net.Data 參考手冊*。

授權要訣：確定 Web 伺服器有權存取此檔。有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

本章找出組成 Net.Data 巨集的不同區塊，並說明撰寫巨集所用的方法。

Net.Data 巨集的結構

巨集是由兩個部份所組成：

- 宣告部份，包含呈現部份中使用的定義。宣告部份使用兩個主要的可選用區塊：
  - DEFINE 區塊
  - FUNCTION 區塊

  宣告部份也可以含有其他語言結構與陳述式，如 EXEC 陳述式、IF 區塊、 INCLUDE 陳述式及 MESSAGE 區塊。有關語言結構的詳細資訊，請參閱 *Net.Data 參考手冊*中有關語言結構一章

  授權要訣：確定 Web 伺服器有權存取 EXEC 和 INCLUDE 陳述式所參照的檔案。有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。
- 呈現部份定義網頁的配置、參照變數，以及使用作爲巨集的進入及跳出點的 HTML 區塊來呼叫函數。當您呼叫 Net.Data 時，您會設定一個 HTML 區塊名稱作爲進入點，來處理巨集。HTML 區塊說明於第82頁的『HTML 區塊』。

在本節中，我們會舉一個簡單的 Net.Data 巨集爲例，來說明巨集語言的元素。這個範例巨集會呈現一個套表，提示您輸入資訊以傳給 REXX 程式。巨集會將此資訊傳給稱爲 ompsamp.cmd 的外部 REXX 程式，來回應使用者輸入的資料。 然後會將結果顯示在第二個 HTML 頁面上。

首先查看整個巨集，然後再查看每個區塊的明細：

```
%{ ********************       DEFINE 區塊        ************************%}
%DEFINE {
   page_title="Net.Data 巨集模版"
%}

%{ ********************** FUNCTION 定義區塊 ************************%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
{
  %EXEC{ompsamp.cmd %}
%}

%FUNCTION(DTW_REXX) today () RETURNS(result)
{
      result = date()
%}

%{ ********************      HTML 區塊：輸入     ************************%}
%HTML(INPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>輸入套表</h1>
今天是 @today()

<FORM METHOD="post" ACTION="OUTPUT">
請輸入一些要傳給 REXX 程式的資料：
<INPUT NAME="input_data" TYPE="text" SIZE="30">
<p>
<INPUT TYPE="submit" VALUE="Enter">

</form>

<hr>
<p>[<a href="/">首頁</a>]
</body></html>
```

```
%}

%{ **********************    HTML 區塊：輸出      ************************%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>輸出頁</h1>
<p>@rexx1(input_data)
<p><hr>
<p>[<a href="/">首頁</a> |
<a href="input">前一頁</a>]
</body></html>
%}
```

樣本巨集由四個主要區塊組成：DEFINE、FUNCTION、與兩個 HTML 區塊。 在同一 Net.Data 巨集中，您可以使用多個 DEFINE, FUNCTION 與 HTML 區塊。

這兩個 HTML 區塊含有如 HTML 文字呈現陳述式，使您便於撰寫 Web 巨集。如果您已經十分熟悉 HTML，則建置巨集的工作，只是新增一些要在伺服器上動態處理的巨集陳述式，以及要傳到資料庫的 SQL 陳述式。

雖然，巨集類似於 HTML 文件，但 Web 伺服器會使用 CGI、Web 伺服器 API 或 Java Servlet，透過 Net.Data 來存取它。若要呼叫巨集，Net.Data 需用到兩個參數：要處理的巨集名稱，以及在該巨集中要顯示的 HTML 區段。

當呼叫巨集時，Net.Data 會從頭開始處理。接下來的幾節中，我們會告訴您 Net.Data 在處理檔案時會發生什麼事。

DEFINE 區塊

DEFINE 區塊包含 DEFINE 語言結構，及 HTML 區塊後面所用的變數定義。下列範例顯示一個 DEFINE 區塊和一個變數定義：

```
%{ **********************       DEFINE 區塊        ************************%}
%DEFINE {
   page_title="Net.Data 巨集模版"
%}
```

第一行是一個備註。所謂備註，是指位於 %{ 和 %} 內的本文。備註可在巨集中的任意處。下一個陳述式會開始 DEFINE 區段。 您可以在一個定義區塊中定義多個變數。 在這個範例中，只可定義一個變數 page_title。 一旦定義之後，您可以使用語法 $(page_title)，在巨集中的任意處參照此變數。 透過變數的使用，可讓您以後在對您的巨集進行整體的變更時較爲簡單。 此區塊的最後一行 %}，定義 DEFINE 區塊的終止。

FUNCTION 區塊

FUNCTION 區塊含有 HTML 區塊所呼叫的函數的宣告。 函數是由語言環境來處理，可執行程式、SQL 查詢或儲存程序。

下列範例顯示兩個 FUNCTION 區塊。一個定義外部 REXX 程式的呼叫，另一個則含有列入 REXX 陳述式。

```
%{ ********************** FUNCTION 區塊 ***********************************%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- 此函數接受
                                                         一個參數，並傳回
                                                         變數 'result'，這是
                                                         外部 REXX 程式指定的
                                                         變數
     %EXEC{ompsamp.cmd %}  <-- 函數會執行名稱如下的外部 REXX 程式：
                               "ompsamp.cmd"
%}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
      result = date()  <-- 此函數的單一來源陳述式
                           是包含於列入程式中。
%}
```

第一個函數區塊 (rexx1) 爲一個 REXX 函數宣告，它會執行名爲 ompsamp.cmd 的外部 REXX 程式。一個輸入變數 input 是由此函數來接受，且自動地傳送至外部 REXX 指令。 REXX 指令也會傳回一個稱爲 result 的變數。 REXX 命令中 result 變數的內容，會置換 OUTPUT 區塊中呼叫的 @rexx1() 函數呼叫。變數 input 和 result 可由 REXX程式直接存取，請參閱 ompsamp.cmd 的原始程式：

```
/* REXX */
result = 'REXX 程式從巨集中收到 "'input'"。'
```

此函數中的程式碼會對傳給它的資料做出回應。 您可以使用一般的 mark-up 樣式標籤 (例如 <b> 或)，括住要求的 @rexx1() 函數呼叫，來按照自己的意思製作結果本文的格式。 不使用 result 變數，而是使用 REXX SAY 陳述式， REXX 程式即可將寫好的 HTML 陳述式以標準格式輸出。

第二個函數區塊，也參照 REXX 程式 today。不過，本例中的整個 REXX 程式包含在其本身的函數宣告中。不需要一個外部程式。 REXX 和 Perl 函數都可接受列入 (inline) 程式，這是因爲它們皆爲解譯過的語言，可動態解析及執行之。 列入程式不需用到另一個程式檔來管理，因此具有簡便上的好處。第一個 REXX 函數也已用列入方式處理。

HTML 區塊

HTML 區塊會定義網頁的配置、參照變數，以及呼叫函數。HTML 區塊會作爲巨集的進入與跳出點。 HTML 區塊恆會在 Net.Data 巨集要求中指定，且每一巨集至少須有一個 HTML 區塊。

範例巨集中的第一個 HTML 區塊名爲 INPUT。 HTML(INPUT) 含有具有一個輸入欄位的簡單套表的 HTML。

```
%{ **********************      HTML 區塊：輸入     ************************%}
%HTML (INPUT) {               <--- 識別出此 HTML 區塊的名稱。
<html>
<head>
<title>$(page_title)</title>  <--- 請注意變數替代。
</head><body>
<h1>輸入套表</h1>
今天是 @today()             <--- 此行包含一個函數呼叫。

<FORM METHOD="post" ACTION="OUTPUT">  <--- 當提出這個套表時，
                                           將呼叫 "OUTPUT" HTML 區塊。
請輸入一些要傳給 REXX 程式的資料：
<INPUT NAME="input_data"      <--- 當提出套表時，"input_data" 會被定義
TYPE="text" SIZE="30">             而且會在此巨集的其他位置被參考到的。
                                   其被起始設定成使用者在輸入欄位中
                                   所鍵入的任何資料。
<p>
```

```
<INPUT TYPE="submit" VALUE="Enter">

<hr>
<p>
[
<a href="/">首頁</a>]
 </body><html>
%}                                  <--- 關閉此區塊。
```

整個區塊都用 HTML 區塊識別字 %HTML (INPUT) {...%} 圍住。 INPUT 指出此區塊的名稱。名稱可以含有任何英數字元、底線或句點。 HTML <title> 標籤包含變數替代的範例。 變數 page_title 的值，會取代套表的標題。

此區塊中還含有一個函數呼叫。 表示式 @today() 是對函數 today 的呼叫。此函數是定義在上面描述的 FUNCTION 區塊中。 Net.Data 將 today 函數的結果，亦即本日，插入與 @today() 表示式同一位置 HTML 本文中。

FORM 陳述式中的 ACTION 參數，提供 HTML 區塊之間或巨集之間的導引範例。 若是參照 ACTION 參數中的另一個區塊名稱，則當提出套表時，即會存取該區塊。任何來自 HTML 套表的輸入資料，都會被當成隱含變數來傳給區塊。定義在此套表中的單一輸入欄位，亦可適用。當提出套表後，在此套表中輸入的資料將透過變數 *input_data*，傳遞到 HTML(OUTPUT) 區塊。

只要巨集是位在同一 Web 伺服器上，您便可以使用相關參照方式，來存取其他巨集中的 HTML 區塊。例如， ACTION 參數 ACTION="../othermacro.d2w/main" 可存取巨集 othermacro.d2w 中稱爲 main 的 HTML 區塊。 同樣地，此套表中輸入的任何資料，皆會透過 *input_data* 變數傳送到這個巨集。

當您呼叫 Net.Data 時，您是以當成 URL 的一部份來傳遞變數。例如：

```
<a href="/cgi-bin/db2www/othermacro.d2w/main?input_data=value">下一個巨集</a>
```

您可以經由參照套表中指定的變數名稱，來存取或操作套表資料。

範例中的下一個區塊爲 HTML(OUTPUT) 區塊。它包含 HTML 標籤和 Net.Data 巨集陳述式，這些陳述式定義 HTML(OUTPUT) 要求所處理的輸出。

```
%{ **********************    HTML 區塊：輸出     ************************%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>  <--- 其它替代。

</head><body>
<h1>輸出頁</h1>
<p>@rexx1(input_data)  <--- 此行包含呼叫函數 rexx1，
                           及傳送 "input_data" 引數給函數。
<p>
<hr>
<p>
[
<a href="/">首頁</a> |
<a href="input">前一頁</a>]
%}
```

類似 HTML(OUTPUT) 區塊，此區塊是一個標準的 HTML，再加上用來替代變數的巨集陳述式以及一個函數呼叫。同樣地， page_title 變數被取代成標題陳述式。一如前述，本區塊中含有一個函數呼叫。 在此例中，它會呼叫函數 rexx1，並將變數 input_data

的內容傳遞給函數，而內容是來自輸入區塊中定義的套表。 您可以和函數間互傳多個變數。 函數定義會指定要傳遞的變數之數目與用法。

Net.Data 巨集變數

Net.Data 可讓您在 Net 巨集內定義及參照變數。 另外，您可以將巨集中的這些變數傳遞給語言環境後再傳回。

Net.Data 變數可根據變數類型及是否有預設值來定義。這些變數可基於定義的分式，分爲下列類型：

- 在 DEFINE 區塊中，使用 DEFINE 陳述式明確定義的變數
- 事先定義好的變數，這些變數可讓 Net.Data 使用，並且設有一值。 此值通常無法變更。
- 隱含的定義變數，有下列四種類型：
  - 雖未明確地定義，不過在首次指定值時可被案例化的變數。
  - 參數變數，爲 FUNCTION 區塊定義的一部份，且僅能在 FUNCTION 區塊中。
  - 可由 Net.Data 案例化，且會對應到套表資料或查詢字串資料的變數。
  - 此類變數乃和 Net.Data 表格相連結，且僅能在 ROW 區塊或 REPORT 區塊中被參考到。

下列段落描述：

- 『識別字範圍』
- 第85頁的『定義變數』
- 第86頁的『參照變數』
- 第88頁的『變數類型』

識別字範圍

識別字 (變數或函數呼叫) 變成*可見的*，表示被宣告或被案例化時，即可被參考。可看見識別字的區域，即稱爲其*範圍*。五種範圍類型如下：

- 廣域

  若您可在巨集的任一處參照到此識別字，則該識別字即具有廣域範圍。具有廣域範圍的識別字如下：
  - Net.Data 的內建函數
  - 套表資料
  - 查詢字串資料
  - 自 HTML 區塊中被案例化的變數
- 巨集

  當識別字的宣告是出現在任一區塊的外頭時，則該 ID 的範圍即屬此種。一個區塊開始於一個左方括弧 ({)，終止於一個百分比符號和右方括弧 (%}). (請注意：本定義中不包括 DEFINE 區塊，且 DEFINE 區塊乃被視爲獨立的 DEFINE 陳述式。) 不同於具有廣域範圍的識別字，具有巨集範圍的識別字僅能被巨集中其後有識別字宣告的項目所參照。
- FUNCTION 區塊或 MACRO_FUNCTION 區塊

若爲下列情況，識別字將具有函數區塊範圍：

– 在函數定義的參數列示中宣告識別字。

  如果具有同名的識別字已存在於函數定義之外，則 Net.Data 將使用來自函數區塊內的函數參數的識別字。

– 在函數區塊中已將識別字案例化，並且在函數呼叫之前不宣告它，或將它案例化。

若識別字已於函數之外宣告或起始設定，並且未於參數列示內宣告，則沒有函數區塊範圍。函數區塊內的識別字值仍會保持不變，除非函數更新它，才會有所改變。

- REPORT 區塊

  若識別字只能從 REPORT 區塊中來參考 (例如，表格直欄名稱 N1、N2、...、N*n*)，則有報告區塊範圍。只有 Net.Data 隱含定義成其表格處理之一部份的變數，才能具有報告區塊範圍。 而其它任何立即可用的變數，則屬於函數區塊範圍。

- ROW 區塊

  若識別字只能從 ROW 區塊中來參考到 (例如，表格值名稱 V1、V2、...、Vn)，則有橫列區塊範圍。只有 Net.Data 隱含定義成其表格處理之部份變數，才能具有橫列區塊範圍。 而其它任何立即可用的變數，則屬於函數區塊範圍。

定義變數

定義 Net.Data 巨集中變數的方法有下列 3 種：

- 定義陳述式或區塊
- HTML 套表標籤
- 查詢字串資料

自套表或查詢字串資料中接收的變數值，會置換掉 Net.Data 巨集中 DEFINE 陳述式所設定的變數值。

- **DEFINE 陳述式或區塊**

  在 Net 巨集中定義所用變數之最簡單方式就是使用 DEFINE 陳述式。語法如下：

```
%DEFINE variable_name="variable value"

%DEFINE variable_name={ variable value on multiple
                        lines of text  %}
%DEFINE {
    variable_name1="variable value 1"
    variable_name2="variable value 2"
%}
```

  variable_name 是您提供給變數的名稱。 變數名稱必須以一個字母或底線符號開始，並且可以包括任何的英數字字元、底線、句點或 #。所有變數名稱皆區分大小寫，但 N_*columnName* 和 V_*columnName* 除外，它們是表格變數。

  例如：

```
%DEFINE reply="hello"
```

  變數 `reply` 具有值 `hello`。

  只鍵入兩個連續的引號等於空字串。例如：

```
%DEFINE empty=""
```

變數 empty 具有空字串。

如果您的變數含有特殊字元 (如行尾)，請使用區塊大括弧括住值：

```
%DEFINE introduction={
Hello,
My name is John.
%}
```

若要在字串中包括雙引號，您可以連續使用兩個雙引號。

```
%DEFINE HI="say ""hello"""
```

您也可以使用區塊大括弧來跳出雙引號：

```
%DEFINE HI={ say "hello" %}
```

要以一個 DEFINE 陳述式定義數個變數時，請使用一個 DEFINE 區塊：

```
%DEFINE {
         variable1="value1"
         variable2="value2"
         variable3="value3"
         variable4="value4"
%}
```

- **HTML 套表標籤: SELECT, INPUT 及 TEXTAREA**

  您可以使用 HTML FORM 標籤，指定值給變數，亦即 SELECT, INPUT 及 TEXTAREA 標籤。下列範例使用標準 HTML 套表標籤來定義 Net.Data 變數：

  ```
  <INPUT NAME="variable_name" TYPE=...>
  ```

  或

  ```
  <SELECT NAME="variable_name">
    <OPTION>value one
    <OPTION>value two
  </SELECT>
  ```

  若要指定一個跨越多行或含有特殊字元 (如雙引號) 的變數，則可以使用 TEXTAREA 標籤：

  ```
  <TEXTAREA NAME="variable_name" ROWS="4">
  請在此輸入您的變數的
  多行值。
  </TEXTAREA>
  ```

  variable_name 是您提供給變數的名稱，而變數的值是由套表中接收的輸入所決定。 關於此種類型的變數定義如何使用於 Net.Data 巨集中，請參閱第62頁的『HTML 套表』中的範例。

- **查詢字串資料**

  您可以透過查詢字串，傳遞變數給 Net.Data。例如：

  ```
  http://www.ibm.com/cgi-bin/db2www/stdqry1.d2w/input?field=custno
  ```

  在上述範例中，變數名稱 field 和值 custno 指定 Net.Data 從查詢字串中接收的其它資料。 Net.Data 會像套表資料一樣地接收和處理這些資料。

參照變數

您可以參照先前定義的變數，來傳回其值。 若要參照 Net.Data 巨集中的變數，請在 $(和) 內指定變數名稱。 例如：

```
$(variableName)
$(homeURL)
```

當 Net.Data 發現一項變數參照時，它會以變數值取代該變數參照。變數參照可以含有字串、變數參照及函數呼叫。

您可以動態建立變數名稱。透過這個技術，當無法事先決定列示中的數字時，您可以使用迴路，來處理可變大小的表格，或在執行時所建置的列示的輸入資料。例如，您可以建立 HTML 套表元素的列示，這些元素是依據 SQL 查詢傳回的記錄而建立的。

若要使用變數作爲文字呈現陳述式的一部份，請在您的巨集的 HTML 區塊中參照它們。

無效變數參照：無效變數參照將解析爲空字串。例如，如果變數參照含有如 ！的無效字元，則參照將解析爲空字串。

有效變數名稱必須以英數字元或底線開頭，且它們可以由英數字元所構成，包括句點、底線及 # 字號。

範例 1：鏈結中的變數參照

如果您已定義變數 *homeURL*：

```
%DEFINE homeURL="http://www.ibm.com/"
```

您可以用 $(*homeURL*) 來參照首頁，並建立一個鏈結：

```
<A href="$(homeURL)">首頁</A>
```

您可以參照 Net.Data 巨集的許多部份中的變數；請參閱本章中的語言結構，來決定巨集中有哪些部份的變數容許被參照。如果在參照變數時，尚未定義它們， Net.Data 將傳回空的字串。僅有變數參照不能定義變數。

範例 2：動態建立變數參照

假設您執行具有任意數目的 SQL SELECT 陳述式。您可以使用下列 ROW 區塊，建立具有輸入欄位的 HTML 套表：

```
...
  %ROW {
<input type=text name=@dtw_rconcat("I", ROW_NUM) size=10 maxlength=10>
%}
...
```

因爲您建立了 INPUT 欄位，所以您可能想要存取當套表提出到您的巨集進行處理時，使用者所輸入的值。您可以編寫迴路，來取回可變長度列示中的值：

```
<PRE>
...
@dtw_assign(rowIndex, "1")
%while (rowIndex <= rowCount) {
The value entered for row $(rowIndex) is: $(I$(rowIndex))
@dtw_add(rowIndex, "1", rowIndex) %}
...
</PRE>
```

Net.Data 首先會使用 I$(rowIndex) 參照來建立變數名稱。例如，第一個變數名稱將是 I1。然後，Net.Data 將使用該值，並解析爲變數的值。

範例 3：具有巢狀變數參照及函數呼叫的變數參照

```
%define my = "my"
%define u = "lower"
%define myLOWERvar = "hey"
$($(my)@dtw_ruppercase(u)var)
```

變數參照會傳回 hey 的值。

變數類型

您可以在巨集中使用下列類型的變數。

- 『條件式變數』
- 『環境變數』
- 第89頁的『執行變數』
- 第90頁的『隱藏變數』
- 第91頁的『列示變數』
- 第91頁的『表格變數』
- 第92頁的『雜項變數』
- 第93頁的『表格處理程序變數』
- 第93頁的『報告變數』
- 第94頁的『語言環境變數』

如果您將字串指定給 Net.Data 以某種方式 (如 ENVVAR、LIST、條件列示變數) 定義的變數，變數將不再以定義的方式運作。換言之，變數將變成含有字串的簡單變數。

請參閱 *Net.Data 參考手冊*，取得每一變數的語法及範例。

條件式變數

透過類似於 IF THEN 結構的方法，條件式變數可讓您爲變數定義一個條件值。定義條件式變數時，您可以設定兩個可能的變數值。若所參照的第一個變數存在，則條件式變數會取得第一個值，否則，條件式變數取得第二個值。條件變數的語法爲：

```
varA = varB ? "value_1" : "value_2"
```

若 varB 已定義，則 varA="*value_1*"，否則爲 varA="*value_2*"。這與使用 IF 區塊是相同的，如下列範例所示：

```
%IF ($(varB))
    varA = "value_1"
%ELSE
    varA = "value_2"
%ENDIF
```

關於使用條件式變數與列示變數，請參閱 第91頁的『列示變數』 中的範例。

環境變數

一旦 Web 伺服器使環境變數可供正在處理您的 Net.Data 要求的處理或緒使用後，您便可以參照這些環境變數。當參照 ENVVAR 變數時，Net.Data 會以相同名稱傳回環境變數的現行值。

定義環境變數的語法如下：

```
%DEFINE var=%ENVVAR
```

其中 *var* 是將定義的環境變數的名稱。

例如，變數 SERVER_NAME 可定義爲環境變數：

```
%DEFINE SERVER_NAME=%ENVVAR
```

然後被參考到：

```
伺服器是 $(SERVER_NAME)
```

其輸出結果如下：

```
伺服器是 www.software.ibm.com
```

請參閱 *Net.Data 參考手冊*，取得 ENVVAR 陳述式的詳細資訊。

執行變數

透過執行變數，可讓您從一個變數參照呼叫其他程式。

使用 DEFINE 區塊中的 EXEC 語言結構，在 Net.Data 巨集中定義執行變數。有關 EXEC 語言元素的詳細資訊，請參閱 *Net.Data 參考手冊*中有關語言結構這一章。在下列範例中，變數 runit 被定義來執行可執行的程式 testProg：

```
%DEFINE runit=%EXEC "testProg"
```

runit 變成執行變數。

Net.Data 在 Net.Data 巨集中發現有效變數參照時，會執行可執行程式。例如，若 Net.Data 巨集中對變數 runit 有一個有效變數參照，則會執行 testProg 程式。

最簡單的方法是從另一個變數定義中來參照執行變數。下列範例示範這個方法。變數 date 將定義爲可執行的變數，且 dateRpt 含有可執行變數的參照。

```
%DEFINE date=%EXEC "date"
%DEFINE dateRpt="Today is $(date)"
```

不論 $(dateRpt) 位於 Net.Data 巨集中的何處， Net.Data 會搜尋可執行程式 date，一旦發現，即傳回：

```
今天是 Tue 11-07-1999
```

若 Net.Data 在巨集中發現執行變數，則會使用下列方法來尋找被參考到的可執行程式：

1. 它會搜尋 Net.Data 起始設定檔案中的 EXEC_PATH 所設定的目錄。有關詳細資料，請參閱第18頁的『EXEC_PATH』。
2. 如果 Net.Data 找不到程式，則系統會搜尋系統 PATH 環境變數或檔案庫列示中所定義的目錄。若找到可執行的程式，則 Net.Data 會執行該程式。

限制：請勿將執行變數設定爲其呼叫之可執行程式輸出的值。在前一個範例中，變數 date 的值是 NULL (空值)。若您在 DTW_ASSIGN 函數呼叫中使用此變數，來指定其值給另一個變數，則指定後新變數的值也會是 NULL (空值)。執行變數的唯一目的，是呼叫其定義的程式。

您也可以藉由在變數定義中指定參數及程式名稱，然後將參數傳給要執行的程式。在本範例中，distance 和 time 的值被傳遞給程式 *calcMPH*。

```
%DEFINE mph=%EXEC "calcMPH $(distance) $(time)"
```

下一範例會傳回系統日期作爲報告的一部份：

```
%DEFINE database="celdial"
 %DEFINE tstamp=%EXEC "date"

%FUNCTION(DTW_SQL) myQuery() {
SELECT CUSTNO, CUSTNAME from dist1.customer
%REPORT{
%ROW{
<A HREF="/cgi-bin/db2www/exmp.d2w/report?value1=$(V1)&value2=$(V2)">
$(V1) $(V2) </A> <BR>
%}
%}
%}

%HTML(report){
<H1>報告製作於： $(tstamp) </H1>
@myQuery()
%}
```

每一個報告都有日期，以便於追蹤。本範例中，亦會替另一個 Net.Data 巨集，在鏈結中加入客戶號碼和名稱。在報告中按一下任何客戶，即呼叫 `exmp.d2w` Net.Data 巨集，並將編號和名稱傳給 Net.Data 巨集。

隱藏變數

您可以使用隱藏變數，來隱藏變數的實際名稱，讓使用 Web 瀏覽器來顯示網頁原始碼的應用程式使用者無法看見。若要定義隱藏變數：

1. 在 HTML 區塊中最後一次參照變數的後面，爲您想要隱藏的每一個字串定義一個變數。變數在 HTML 區塊中使用之後，永遠都是以 DEFINE 語言結構來定義，如下列範例所示。 $$(*variable*) 會被參考到，然後被定義。
2. 在變數被參照的 HTML 區塊中，使用兩個貨幣符號 (而不是一個) 來參照變數。 例如，採用 `$$(X)` 而不是 `$(X)`。

   ```
   %HTML(INPUT) {
   <FORM ...>
   <P>選取您要檢視的欄位：
   shanghai<SELECT NAME="欄位">
   <OPTION VALUE="$$(name)"> 姓名
   <OPTION VALUE="$$(addr)"> 地址
   ...
   </FORM>
   %}

   %DEFINE {
   name="customer.name"
   addr="customer.address"
   %}

   %FUNCTION(DTW_SQL) mySelect(){
     SELECT $(Field) FROM customer
   %}

   ...
   ```

當 Web 瀏覽器顯示 HTML 套表時， $$(name) 和 $$(addr) 會分別置換成 $(name) 和 $(addr)，因此，實際的表格和直欄名稱永遠不會出現在 HTML 套表上。應用程式使用者無從得知眞正變數名稱已隱藏。當使用者提出該套表時，會呼叫 HTML(REPORT) 區塊。當 @mySelect() 呼叫 FUNCTION 區塊時， SQL 陳述式中的 $(Field)，會置換成 SQL 查詢中的 customer.name 或 customer.addr。

列示變數

使用列示變數可讓您建置一串以分隔符號區隔的值。 這特別有助於您建立多項目的 SQL 查詢 (例如，常見於某些 WHERE 或 HAVING 子句中)。 列示變數語法如下：

```
%LIST " value_separator " variable_name
```

建議值：空白是有意義的。在大部份的情況下，請在值分隔字元的前後各插入一個空格。 大部份的查詢在值區隔符號上，會使用布林或算術運算子 (例如，AND、OR 或 >)。下列範例舉例說明如何使用條件、隱藏、以及列示變數：

```
%HTML(INPUT) {
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/example2.d2w/report">
<H2>選取一個過更多個都市：</H2>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond1)">Sao Paolo<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond2)">Seattle<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond3)">Shanghai<BR>
<INPUT TYPE="submit" VALUE="Submit Query">
</FORM>
%}

%DEFINE{
DATABASE="custcity"
 %LIST " OR " conditions
cond1="cond1='Sao Paolo'"
cond2="cond2='Seattle'"
cond3="cond3='Shanghai'"
whereClause= ? "WHERE $(conditions)" : ""
%}

%FUNCTION(DTW_SQL) mySelect(){
SELECT name, city FROM citylist
$(whereClause)
%}

%HTML(REPORT){
@mySelect()
%}
```

在 HTML 套表中，若未勾選出任何勾選框，則 conditions 爲 NULL (空值)，因此查詢中的 whereClause 亦爲 NULL (空值)。否則， whereClause 有 OR 所分隔的選取值。例如，若三個都市皆選取，則 SQL 查詢如下：

```
SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'
```

本範例顯示選取了 Seattle，其產生的 SQL 查詢如下：

```
SELECT name, city FROM citylist
WHERE cond1='Seattle'
```

表格變數

表格變數可以定義一套相關資料。它含有一組包括一列直欄表頭的橫列及直欄。在 Net.Data 巨集中定義表格的方式，如下列陳述式所示：

```
%DEFINE myTable=%TABLE(30)
```

位在 %TABLE 後面的數字會限制這個表格變數可含有的列數。如果您想指定一個沒有列數限制的表格，您可以如下列範例所示，使用預設值或指定 ALL：

```
%DEFINE myTable2=%TABLE
%DEFINE myTable3=%TABLE(ALL)
```

定義表格時，並無橫列與直欄。把值移入表格內的唯一方法，是將它當做 OUT 或 INOUT 參數來傳給函數，或使用 Net.Data 提供的內建表格函數。 DTW_SQL 語言環境自動地將 SELECT 陳述式的結果放入表格中。

對於非資料庫語言環境，例如 DTW_REXX 或 DTW_PERL，語言環境也會負責設定表格值。不過，語言環境 script 或 program 會按著一個資料格接著一個資料格，來定義表格值。請參閱第113頁的『第6章 使用語言環境』，取得語言環境如何使用表格變數的詳細資訊。

您可以參照表格變數名稱，在函數之間傳遞表格。 在函數的 REPORT 區塊中可以參照表格的個別元素， 或使用 Net.Data 表格函數來參照它們。. 請參閱第93頁的『表格處理程序變數』，瞭解如何在 REPORT 區塊內，存取表格中的個別元素，以及參閱第102頁的『表格函數』，瞭解如何使用表格函數來存取表格的個別元素。 表格變數通常是在 SQL 函數中移入值，然後當做參數傳遞給 SQL 函數或另一個函數，以用來當做報告的輸入。您可以將表格變數當做 IN、OUT 或 INOUT 參數，傳遞給任何非 SQL 函數。表格只能當作 OUT 參數傳給 SQL 函數。

如果您參照一個表格變數，將依據 DTW_HTML_TABLE 變數的設定，來顯示表格內容並製作它的格式。在下列範例中，將顯示 myTable 的內容：

```
%HTML (output) {
  $(myTable)
}
```

表格中的直欄名稱和欄位值，是以 1 起始的陣列元素來定址。

雜項變數

這些變數是 Net.Data 定義的變數，用途如下：

- 影響 Net.Data 處理程序
- 找出函數呼叫的狀態
- 取得資料庫查詢之結果集的相關資訊
- 決定有關檔案位置和日期的資訊

雜項變數可以有 Net.Data 決定的預定值，或您所設定的值。例如，Net.Data 基於目前所處理的檔案，來決定 DTW_CURRENT_FILENAME 變數值，而您可以設定 Net.Data 是否要除去定位字元和換行字元所造成的額外空白。

預設變數可用來當做巨集內的變數參照，並提供有關檔案現行狀態、日期、或函數呼叫狀態之資訊。例如，若要取回目前的檔案名稱，您可以使用：

```
%REPORT {
<p>此檔案為 <i>$(DTW_CURRENT_FILENAME)</i>。</P>
}
```

可更改的變數值通常是使用 DEFINE 陳述式或 @DTW_ASSIGN() 函數來設定，且可讓您影響 Net.Data 處理巨集的方式。例如，若要設定是否除去空白，您可以使用下列 DEFINE 陳述式：

```
%DEFINE DTW_REMOVE_WS="YES"
```

表格處理程序變數

Net.Data 定義表格處理變數乃是爲了用於 REPORT 及 ROW 區塊。請使用這些變數來參照來自 SQL 查詢和函數呼叫的值。

表格處理變數具有 Net.Data 決定的預設值。這些變數可讓您根據直欄、橫列或被處理欄位，參照來自 SQL 查詢的結果集合或函數呼叫的值。您也可以存取有關被處理的橫列數或所有直欄名稱的列表。

例如，當 Net.Data 處理 SQL 查詢的結果集時，它會針對每一個現行直欄名稱指定變數 Nn 的值，例如指定 N1 給第一直欄、 N2 給第二個直欄，依此類推。您可以參照網頁輸出的現行直欄名稱。

使用表格處理程序變數，做爲巨集內的變數參照。例如，若要取回被處理之現行直欄的名稱，您可以使用：

```
%REPORT {
<p>直欄 1 為 <i>$(N1)</i>。</P>
}
```

表格處理程序變數亦提供有關查詢結果的資訊。 您可以參照巨集中的變數 TOTAL_ROWS，來顯示 SQL 查詢傳回的列數，如下列範例所示：

```
所找到的名稱：$(TOTAL_ROWS)
```

有些表格處理程序變數，會被其他變數或內建式函數所影響。 例如，TOTAL_ROWS 需要啓用 DTW_SET_TOTAL_ROWS SQL 語言環境變數，以便 Net.Data 在處理 SQL 查詢的結果集或函數呼叫時，能夠指定 TOTAL_ROWS 的值，如下列範例所示：

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
...

所找到的名稱：$(TOTAL_ROWS)
```

報告變數

Net.Data 會以預設報表格式顯示巨集所產生的網頁輸出。預設報表格式會顯示在使用 <PRE> </PRE> 標籤的表格格式中。 您可以經由下列方式置換預設報表：以顯示輸出的指示來定義 REPORT 區塊，或使用其中一個報表變數來阻止預設報表的產生。

報表變數會協助您自行設定網頁輸出的顯示方式，以及如何搭配預設報表與 Net.Data 表格一起使用。您必須先使用 DEFINE 陳述式或 @DTW_ASSIGN() 函數來定義這些變數，才能使用它們。

報表變數會設定間隔、置換預設報表格式、設定 HTML 表格輸出對預設表格輸出，以及設定其他顯示特性。例如，您可以使用 ALIGN 變數，來控制表格處理程序變數的前端和尾端空間。 下列範例使用 ALIGN 變數，對查詢所傳回的列示，以空格來分隔其中每一個直欄名稱。

```
%DEFINE ALIGN="YES"
...
<p>您的查詢是針對這些直欄： $(NLIST)
```

START_ROW_NUM 報告變數可讓您決定從何列開始顯示查詢的結果。 例如，下列變數值指定 Net.Data 從第三列開始顯示查詢的結果。

```
%DEFINE START_ROW_NUM = "3"
```

您也可以決定 Net.Data 是否要對預設格式使用 HTML 標籤。若 DTW_HTML_TABLE 設爲 YES，則會產生 HTML 表格，而不是文字格式的表格。

```
%DEFINE DTW_HTML_TABLE="YES"

%FUNCTION(DTW_SQL){
SELECT NAME, ADDRESS FROM $(qTable)
%}
```

語言環境變數

這些變數與語言環境一同使用，並影響語言環境處理要求的方式。

透過這些變數，您可以執行如下的作業：建立資料庫連線、提供 Java applet 的替代文字、啓用 NLS 支援，以及決定是否已順利執行 SQL 陳述式。

例如，您可以使用 SQL_STATE 變數，來存取或顯示資料庫傳回的 SQL 狀態值。

```
%FUNCTION (DTW_SQL) val1() {
 select * from customer
%REPORT {
 ...
  %ROW {
 ...
%}
 SQLSTATE=$(SQL_STATE)
%}
```

下一範例顯示如何定義將存取的資料庫。

```
%DEFINE DATABASE="CELDIAL"
```

Net.Data 函數

Net.Data 會提供要在應用程式中使用的內建函數，如字組與字串操作函數，以及取回與設定表格變數函數的函數。 例如，您也可以定義要與應用程式一起使用，來呼叫外部程式或儲存程序的函數。

使用者定義的函數

例如，您定義要與應用程式一起使用，來呼叫外部程式或儲存程序的那些函數。

Net.Data 的內建函數

Net.Data 會提供將在應用程式中使用的那些函數，如操作字組與字串的函數，以及取得與設定表格變數的函數。

這些段落將描述下列主題：

- 第95頁的『定義函數』
- 第99頁的『呼叫函數』

- 第99頁的『呼叫 Net.Data 的內建函數』

定義函數

若要在巨集中定義自己的函數，請使用 FUNCTION 區塊或 MACRO_FUNCTION 區塊：

FUNCTION 區塊
定義可從 Net.Data 巨集呼叫，且由語言環境來處理的次常式。 FUNCTION 區塊必須含有語言陳述式或外部程式的呼叫。

MACRO_FUNCTION 區塊
定義可從 Net.Data 巨集呼叫，且透過 Net.Data 而非語言環境來處理的次常式。MACRO_FUNCTION 區塊可以含有 HTML 區塊中所容許的任何陳述式。

語法：使用下列語法來定義函數：

FUNCTION 區塊：

```
%FUNCTION(type) function-name([usage] [datatype] parameter, ...) [RETURNS(return-var)] {
  executable-statements
  [report-block]
  ...
  [report-block]
  [message-block]
%}
```

MACRO_ FUNCTION 區塊：

```
%MACRO_FUNCTION function-name([usage] parameter, ...)  {
  executable-statements
  report-block
  ...
  report-block
%}
```

其中：

type 指出架構在起始設定檔中之語言環境。語言環境會呼叫某個特定的語言處理器(處理可執行的陳述式)，並提供 Net.Data 和語言處理器之間的標準介面。

function-name
指定 FUNCTION 或 MACRO_FUNCTION 區塊的名稱。函數呼叫會指定 *function-name*，它的前面會有一個 @ 符號。有關詳細資料，請參閱第99頁的『呼叫函數』。

您可以使用相同名稱來定義多個 FUNCTION 或 MACRO_FUNCTION 區塊，以便同時處理它們。每一個區塊皆必須有相同的參數列示。 當 Net.Data 呼叫函數時，相同名稱的所有 FUNCTION 區塊或相同名稱的 MACRO_FUNCTION 區塊，會按它們在 Net.Data 巨集中定義時的次序來執行。

usage 指定參數是輸入 (IN) 參數、輸出 (OUT) 參數、或同時兼具這兩種類型 (INOUT)。此指定表示參數是否在 FUNCTION 區塊或 MACRO_FUNCTION 區塊或兩者上來回傳遞。用法類型適用於參數列示中所有後續的參數，直到被另一個用法類型改變爲止。 預設類型爲 IN。

datatype
參數的資料類型。有些語言環境預期傳遞的參數是資料類型。例如，當呼叫儲存程序時，SQL 語言環境會預期它們，請參閱第113頁的『第6章 使用語言環境』，瞭解您正在使用的語言環境的支援資料類型。

parameter

區域變數的名稱，可換成指定於函數呼叫中之相對應引數的值。 例如，可執行之陳述式或 REPORT 區塊中的參數參照 $(*parm1*)，可換成參數的實際值。 另外，使用該語言的自然語法或當作環境變數使用時，即可將這些參數傳送至語言環境，供可執行的陳述式存取。 參數變數參照無法在 FUNCTION 或 MACRO_FUNCTION 區塊外使用。

return-var

請在 RETURNS 關鍵字後指定此參數，來識別特殊的 OUT 參數。傳回變數的值是在函數區塊中指定的，且它的值會傳回到巨集中呼叫該函數之處。例如，在下列句子中，<p>My name is @my_name().，@my_name() 會被傳回變數的值所取代。如果未設定 RETURNS 子句，函數呼叫的值爲：

- NULL，若從呼叫到語言環境的回覆碼爲零的話
- 回覆碼的值，若回覆碼爲非零的話。

executable-statements

在取代變數和處理函數之後，傳送至指定語言環境來處理的一組語言陳述式。 *executable-statements* 可以含有 Net.Data 變數參照與 Net.Data 函數呼叫。 *executable-statements* 包括 HTML 區塊中所容許的那些可執行的陳述式。

對於 FUNCTION 區塊，Net.Data 會以變數值來置換所有的變數參照、執行所有的函數呼叫，並在可執行的陳述式傳送至語言環境之前，以結果值來置換函數呼叫。 每一個語言環境以不同的方式處理陳述式。 有關指定可執行的陳述式或呼叫可執行的程式，請參閱第89頁的『執行變數』。

對於 MACRO_FUNCTION 區塊，可執行的陳述式是文字和 Net.Data 巨集語言結構的組合。此情況下，不會包括任何語言環境，因爲 Net.Data 扮演語言處理器的角色，且會處理可執行的陳述式。

report-block

定義一個或多個 REPORT 區塊，來處理 FUNCTION 或 MACRO_FUNCTION 區塊的輸出。請參閱第104頁的『報告區塊』.

message-block

定義 MESSAGE 區塊，用以處理 FUCNTION 所傳回的任何訊息。請參閱第97頁的『訊息區塊』。

在 Net.Data 巨集中呼叫函數之前，請先在任何其他區塊外定義它們。

在函數中使用特殊字元

當符合 Net.Data 語言結構語法的字元在函數區塊的語言陳述式區段中使用，作爲語法有效的內含程式碼 (如 REXX 或 Perl) 的一部份時，它們可能會被誤譯爲 Net.Data 語言結構，導致在巨集中發生錯誤或無法預期的結果。

例如，Perl 函數可以使用 COMMENT 區塊定界字元 (%{)。當巨集執行時，%{ 字元會被解譯爲 COMMENT 區塊的開頭。然後， Net.Data 會尋找 COMMENT 區塊的結尾，當它讀到函數區塊的結尾時，會認爲找到它。Net.Data 會繼續尋找函數區塊的結尾，當它找不到時，即會發出錯誤。

使用下列其中一種方法，使用 COMMENT 區塊定界字元或任何其他 Net.Data 特殊字元，作爲內含程式碼的一部份，不使它們被 Net.Data 解譯爲特殊字元：

- 使用 EXEC 陳述式呼叫程式碼，而不是將程式碼置於行內。
- 使用變數參照來設定特殊字元。

例如，下列 Perl 函數含有代表 COMMENT 區塊定界字元 %{ 作爲 Perl 語言陳述式一部份的字元：

```
%FUNCTION(DTW_PERL) func() {
  ...
  for $num_words (sort bynumber keys %{ $Rtitles{$num} }) {
  &make_links($Rtitles{$num}{$num_words});
  }
  ...
%}
```

若要確定 Net.Data 將 %{ 字元解譯爲 Perl 原始碼而非 Net.Data COMMENT 區塊定界字元，請以下列任一方式重寫函數：

- 使用 %EXEC 陳述式：

```
%FUNCTION(DTW_PERL) func() {
  %EXEC{ func.prl %}
%}
```

- 使用變數參照來設定 %{ 字元：

```
%define percent_openbrace = "%{"

%FUNCTION(DTW_PERL) func() {
  ...
  for $num_words (sort by number keys $(percent_openbrace) $Rtitles{$num} } {
  &make_links($Rtitles{$num}{$num_words});
  }
  ...
%}
```

訊息區塊

MESSAGE 區塊可讓您決定在順利完成 (或未順利完成) 函數呼叫後，要如何繼續處理，並且可讓您顯示相關資訊給函數呼叫者。 當處理訊息時，Net.Data 會爲呼叫 FUNCTION 區塊的每一個函數，設定一個語言環境變數 RETURN_CODE。不會在 MACRO_FUNCTION 區塊的函數呼叫上設定 RETURN_CODE。

MESSAGE 區塊是由一系列的訊息陳述式所組成，每個陳述式都設有回覆碼值、訊息文字與要執行的動作。 有關 MESSAGE 區塊的語法，請參閱 *Net.Data 參考手冊*中語言結構一章

MESSAGE 區塊的範圍可以是廣域或區域。若 MESSAGE 區塊是定義在 FUNCTION 區塊中，則對該 FUNCTION 區塊來說其範圍是區域性的。若它是指定在最外層的巨集上，則 MESSAGE 區塊的範圍是廣域的，且可供在 Net.Data 巨集中執行的所有函數呼叫使用。如果您定義多個廣域 MESSAGE 區塊，則會採用最近一次定義的。

Net.Data 會採用下列的規則，來處理從某函數呼叫所傳回的 RETURN_CODE 變數之值：

1. 檢查區域 MESSAGE 區塊是否有完全相符的；然後根據所指定的，看是要跳出或繼續。
2. 如果 RETURN_CODE 不是 0，請檢查區域 MESSAGE 區塊是否爲 `+default` 或 `-default`；然後根據所指定的，看是要跳出或繼續，不過得視 RETURN_CODE 的符號而定。

3. 若 RETURN_CODE 不爲 0，則檢查區域 MESSAGE 區塊是否爲預設值，然後根據所指定的，看是要跳出或繼續。
4. 檢查廣域 MESSAGE 區塊是否有完全相符的；然後根據所指定的，看是要跳出或繼續。
5. 若 RETURN_CODE 不爲 0，則根據 RETURN_CODE 的符號，檢查廣域 MESSAGE 區塊是否爲 +default 或 -default，然後依指定跳出或繼續。
6. 若 RETURN_CODE 不爲 0，則檢查廣域 MESSAGE 區塊是否爲 default，然後依指示跳出或繼續。
7. 若 RETURN_CODE 不爲 0，則發出 Net.Data 內部預設的訊息並跳出。

下列範例顯示一部份 Net.Data 巨集，其中帶有整體 MESSAGE 區塊，與一個函數的 MESSAGE 區塊。

```
%{ 廣域 message 區塊 %}
  %MESSAGE {
     -100     : "回覆碼 -100 訊息"   : exit
      100     : "回覆碼 100 訊息"    : continue
   +default : {
此為超出一行的長訊息。
您可以在此訊息中使用 HTML 標籤，包括
鏈結與套表。%}   : continue
%}

%{ FUNCTION 區塊中的區域 message 區塊 %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE {
     -100     : "回覆碼 -100 訊息"   : exit
      100     : "回覆碼 100 訊息"    : continue
     -default : {
此為超出一行的長訊息。
您可以在此訊息中使用 HTML 標籤，包括
鏈結與套表。%}   : exit
  %}
```

my_function() 傳回一個 RETURN_CODE 值爲 50 時， Net.Data 會以下列順序來處理錯誤：

1. 檢查區域 MESSAGE 區塊中是否有完全相符的。
2. 檢查區域 MESSAGE 區塊中是否爲 +default。
3. 檢查區域 MESSAGE 區塊中是否爲 default。
4. 檢查廣域 MESSAGE 區塊中是否有完全相符的。
5. 檢查廣域 MESSAGE 區塊中是否爲 +default。

當 Net.Data 找到相配時，會將訊息文字送給 Web 瀏覽器，並檢查所要求的動作。

當您設定 continue 時， Net.Data 會在印出訊息文字後繼續處理 Net.Data 巨集。 例如，若巨集呼叫 my_functions() 5 次，且在處理範例中之 MESSAGE 區塊時發現了錯誤 100，則該程式的輸出如下：

```
.
.
.
11 May 1997                  $245.45
13 May 1997                  $623.23
19 May 1997                  $ 83.02
```

```
回覆碼 100 訊息
22 May 1997                 $ 42.67

總計：                      $994.37
```

呼叫函數

使用 Net.Data 函數呼叫陳述式，來呼叫使用者定義的函數與內建函數。請在函數名稱或巨集函數名稱之後使用 @ 字元：

```
@function_name([ argument,... ])
```

function_name

這是要呼叫的函數或巨集函數的名稱。 除非該函數爲內建函數，否則此函數必須已定義在 Net.Data 巨集中。

argument

這是變數、引號內的字串、變數參照或函數呼叫的名稱。函數呼叫上的引數會符合函數或巨集函數參數列示上的參數。此外，當處理函數或巨集函數時，每一個參數均會指定爲它的對應引數的值。引數的數目和類型，必須與相對應參數的相同。

作爲引數的引號內的字串可以含有變數參照及函數呼叫。

範例 1：具有字串引數的函數呼叫

```
@myFunction("abc")
```

範例 2：具有變數及函數呼叫引數的函數呼叫

```
@myFunction(myvar, @DTW_rADD("2","3"))
```

範例 3：具有含有變數參照及函數呼叫的字串的函數呼叫

```
@myFunction("abc$(myvar)def@DTW_rADD("2","3")ghi")
```

呼叫 Net.Data 的內建函數

Net.Data 提供一大組內建函數，可幫助您簡化 Web 頁面的開發。這些函數已經過 Net.Data 的定義，所以您不需要定義它們。您可以如同呼叫其他函數一般，來呼叫這些函數。

第100頁的圖20 顯示 Net.Data 內建函數與巨集的互動方式。

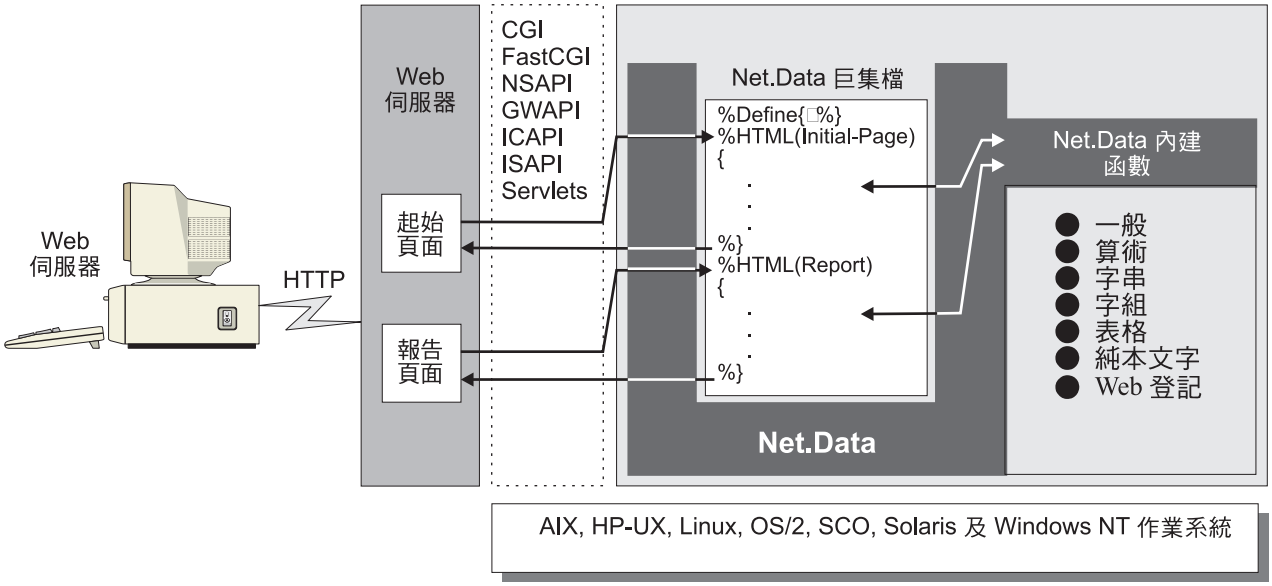


圖 20. Net.Data 內建函數

內建函數可採三種方式傳回其結果，取決於它的字首：

- **DTW_**、**DTWF_** 及 **DTWR_**： 以輸出參數傳回呼叫的結果，或未傳回任何結果。(**DTWF_** 是指純本文檔函數的字首。 **DTWR_** 是指 Web 登記函數的字首。)
- **DTW_r**、**DTWF_r** 及 **DTWR_r**： 函數呼叫的結果會置換巨集中的函數呼叫，同樣地， RETURNS 關鍵字的值也會置換使用者定義函數的函數呼叫，此函數已指定 RETURNS 關鍵字。
- **DTW_m**：透過傳給函數的每一個參數傳回多個結果。

有些內建函數沒有所屬類型。 若要判斷具有哪種類型的特殊內建函數，請參閱 *Net.Data 參考手冊*中有關 Net.Data 內建函數的那一章。

下列段落將提供 Net.Data 內建函數的高階概觀。請使用這些函數來執行一般目的、算術、字串、字組或表格操作函數。 請參閱 *Net.Data 參考手冊*，取得每一函數的說明及其語法和範例。這些函數中有一些要求變數在它們被使用前就被設定，方可使用它們，或須在特定上下文中使用它們。 並非所有作業系統均支援每一個內建函數。請參閱 *Net.Data 參考手冊*，您的作業系統支援哪些函數。

- 第101頁的『一般目的函數』
- 第101頁的『算術函數』
- 第101頁的『字串函數』
- 第102頁的『字組函數』
- 第102頁的『表格函數』
- 第102頁的『純本文檔函數』
- 第102頁的『Web 登記函數』

一般目的函數

此組函數可讓您改變資料或存取系統服務程式，藉此協助您開發網頁。 您可以使用它們來傳送郵件、處理 HTTP cookie、建立 HTML 跳出碼 (escape code)，以及從系統中取得其他有用的資訊。

例如，若要設定當發生特定狀況時，Net.Data 應跳出巨集，不處理巨集的其餘部份，請使用 DTW_EXIT 函數：

```
%HTML(cache_example) {

<html>
<head>
 <title>這是頁面標題</title>
</head>
<body>
  <center>
 <h3>這是主要標題</h3>
 <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
 < ! Joe Smith 看到一篇非常短的頁面                    ! >
 <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
 %IF (customer == "Joe Smith")
  </body>
</html>

@DTW_EXIT()

%ENDIF

...

  </body>
</html>
 %}
```

另一個有用函數就是 DTW_URLESCSEQ 函數，它會以它們的跳出值來置換 URL 中不容許的字元。例如，如果輸入變數 `string1` 等於 `"Guys & Dolls"`，則 DTW_URLESCSEQ 會將輸出變數指定爲值 `"Guys%20%26%20Dolls"`。

算術函數

這些函數會執行算術運算，協助您計算或改變數字性資料。除了標準的算數運算外，您也可以執行模數的除法，指定結果的精確度，以及使用科學記數法。

例如，函數 DTW_POWER 會產生第一個參數的第二個參數次方並傳回結果，如下列範例所示一般：

```
@DTW_POWER("2", "-3", result)
```

DTW_POWER 會在變數 `result` 中傳回 `".125"`

字串函數

這些函數可讓您處理字串中的字元。 您可以變更字串的大小寫，插入或刪除字元，指定一個字串值給另一個變數，以及其他有用的函數。

例如，您可以使用 DTW_ASSIGN 將輸入變數的值指定給輸出變數。您亦可以使用這個函數在巨集中變更變數。 在下列範例中，變數 `RC` 被指定爲 `0`。

```
@DTW_ASSIGN(RC, "0")
```

其他字串函數包括連接字串的 DTW_CONCAT 及在特定位置插入字串的 DTW_INSERT，以及許多其他字串操作函數。

字組函數

這些函數可讓您處理字串中的字組。這些函數中的絕大部份，乃與字串函數的作用類似，只不過其是以整個字組運作。 例如，可讓您計算字串中的字組數、除去字組、從字串中找出某個字組。

例如，使用 DTW_DELWORD 從字串中刪除設定的字數：

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

DTW_DELWORD 傳回字串 `"Now time"`。

其他字組函數包括傳回字組中的字元數的 DTW_WORDLENGTH 及傳回字串內字組位置的 DTW_WORDPOS。

表格函數

您可以使用這些函數來產生報告或套表，以便使用 Net.Data table 變數中的資料。 您也可以使用這些函數，來建立 Net.Data 表格，以及操作及取回那些表格中的值。表格變數含有一組值，以及相關的直欄名稱。 這些函數方便您將整群的值傳給函數。

例如，DTW_TB_APPENDROW 會附加一列到表格中。在下列範例中， Net.Data 會附加 10 列到表格 `myTable` 中：

```
@DTW_TB_APPENDROW(myTable, "10")
```

此外，DTW_TB_DUMPH 會傳回以 <PRE></PRE> 標籤括住的巨集表格變數的內容，且表格的每一列會顯示在不同行上。DTW_TB_CHECKBOX 則會從巨集表格變數中傳回一個或多個 HTML 核對框輸入標籤。

純本文檔函數

使用純本文檔介面 (FFI) 函數來開啓、讀取及操作純本文檔來源 (文字檔) 的資料，以及在純本文檔中儲存資料。

例如，DTWF_APPEND 會將表格變數的內容寫到檔案的尾端，而 DTWF_DELETE 則會刪除檔案中的記錄。

此外，FFI 函數容許以 DTWF_CLOSE 與 DTWF_OPEN 鎖定的檔案。DTWF_OPEN 會鎖定檔案，以便另一個要求無法讀取或更新檔案。當 Net.Data 完成該檔案時，DTWF_CLOSE 即會釋放它，以容許其他要求存取檔案。

Web 登記函數

使用 Web 登記函數來維護登記和其所包含之登錄。 Web 登記是一個由 Net.Data 維護密碼的檔案，可讓您輕易地新增、取回與刪除登錄。

例如，DTWR_ADDENTRY 可新增登錄，而 DTWR_DELENTRY 則會刪除登錄。DTWR_LISTSUB 會傳回關於 OUT 表格參數中的登記登錄的資訊，而 DTWR_UPDATEENTRY 則會以新值取代設定的登記登錄的舊有值。

在巨集中建立網頁

Net.Data 可讓您輕易地將標準網頁呈現在應用程式使用者的瀏覽器上。下列段落描述巨集的 HTML 和 REPORT 區塊，並顯示如何在 Net.Data 巨集中製作網頁的格式。有關這些區塊的語法資訊，請參閱 *Net.Data 參考手冊*中有關語言結構的那一章。

HTML 區塊

Net.Data 巨集包含 HTML 區塊，這些區塊可建立 Web 瀏覽器的文字呈現陳述式，如 HTML。 在巨集中，您必須至少設定一個 HTML 區塊，但若您想要的話，可以設定多個。每一個 HTML 區塊會在瀏覽器中建立單一網頁。 每一次呼叫 Net.Data 時，它僅會處理一個 HTML 區塊。若要建立一個由多個網頁組成的應用程式，您可以呼叫 Net.Data 多次，使用標準的導引技術 (如鏈結與套表) 來處理 HTML 區塊。

任何有效的文字呈現陳述式 (如 HTML 或 JavaScript) 可以出現在 HTML 區塊中。另外，您可在 HTML 區塊中使用 INCLUDE 陳述式、函數呼叫及變數參照。 下列範例顯示在 Net.Data 巨集中使用 HTML 區塊的一般方式：

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT) {
<H1>硬體查詢套表</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/equiplst.d2w/report">
<dl>
<dt>您要列出什麼硬體？
<dd><input type="radio" name="hdware" value="MON" checked>監視器
<dd><input type="radio" name="hdware" value="PNT">指標裝置
<dd><input type="radio" name="hdware" value="PRT">印表機
<dd><input type="radio" name="hdware" value="SCN">掃描器
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}

%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hdware)
%REPORT{
<B>這是您要求的列示：</B><BR>
%ROW{
<HR>
$(N1): $(V1)    $(N2): $(V2)
<P>
$(V3)
%}
%}
%}

%HTML(REPORT){
@myQuery()
%}
```

您可以從類似下面的 HTML 鏈結中來呼叫 Net.Data 巨集：

```
<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.d2w/input">List of hardware</a>
```

當應用程式使用者按一下此鏈結時，Web 瀏覽器會呼叫 Net.Data，且 Net.Data 會解析巨集。 當 Net.Data 開始處理呼叫時指定的 HTML 區塊後，在本例中爲 HTML(INPUT) 區塊，就會開始處理區塊中的文字。 若有任何事物無法被 Net.Data 辨識爲 Net.Data 巨集語言結構，它將傳送至瀏覽器來顯示。

當使用者做好選擇並按「提出」按鈕後，Net.Data 即開始執行 HTML FORM 元素的 ACTION 部份，它指定呼叫 Net.Data 巨集的 HTML(OUTPUT) 區塊。然後， Net.Data 會以 HTML(INPUT) 區塊的方式來處理 HTML(OUTPUT) 區塊。

然後，Net.Data 會處理 myQuery() 函數呼叫，該呼叫接著再呼叫 SQL FUNCTION 區塊。當 SQL 陳述式中的 $(hdware) 變數參照，被換成從輸入格式中傳回的值後，Net.Data 即開始查詢。此時，Net.Data 會再度處理報表，並根據 REPORT 區塊中設定的呈現文字陳述式來顯示查詢結果。

在 Net.Data 完成 REPORT 區塊處理程序之後，即會傳回 HTML(OUTPUT) 區塊，並完成處理程序。

報告區塊

REPORT 區塊可用來顯示 FUNCTION 區塊所輸出的資料並加以格式化。 雖然您所指定的可能是文字、巨集變數參照與函數呼叫等任何有效的組合，但一般來說，此種輸出類型通常是套表資料。 您可以選擇在 REPORT 區塊中指定表格名稱。 除了 SQL 及 ODBC 語言環境之外，您未指定表格名稱，則 Net.Data 將使用來自 FUNCTION 參數列示中第一個輸出表格的表格資料。

REPORT 區塊是由三個可選用的部份所組成：

- 表頭資訊，它含有在表格橫列資料之前顯示一次的文字。
- ROW 區塊，它含有對結果表格的每一列顯示一次的文字和表格變數。
- 標底資訊，它含有在表格橫列資料之後顯示一次的文字。

範例：

```
%REPORT{
<H2>查詢結果</H2>
<P>選取名稱以顯示明細。
<TABLE BORDER=1>
  <TR>
    <TD>Name</TD>
    <TD>Location</TD></TR>
%ROW{
  <TR>
  <TD>
    <a href="/cgi-bin/db2www/name.d2w/details?name=$(V1)&location;=$(V2)">$(V1)</a>
    </TD>
<TD>$(V2)</TD>
  </TR>
  %}
     </TABLE>
%}
```

REPORT 區塊指南

當建立 REPORT 區塊時，請使用下列指南：

- 若要避免顯示來自 ROW 區塊的任何表格輸出，請將 ROW 區塊空白，或乾脆整個省略掉。

- 請在 REPORT 區塊內使用 Net.Data 提供的變數，來存取 Net.Data 巨集結果表格中的資料。這些變數說明於 第93頁的『表格處理程序變數』。 詳細資訊，請參閱 *Net.Data 參考手冊*中有關「報告變數」的那一節。
- 若要提供標題和標底資訊，請在 ROW 區塊的前後提供文字。 Net.Data 將在 ROW 區塊之前找到的一切事物都當成標題資訊來處理。 Net.Data 將在 ROW 區塊之後找到的一切事物都當成註腳資訊來處理。如同 HTML 區塊一樣，Net.Data 會將標題、ROW 及註腳區塊中無法辨識爲巨集語言結構的一切，都視爲文字呈現陳述式，並將這些陳述式傳給瀏覽器。
- 您可以呼叫 REPORT 區塊中的函數及參照函數。
- 若要讓 Net.Data 使用預先格式化文字來列印預設報告，請勿在巨集中併入 REPORT 區塊。 下列範例顯示預設報告格式：

```
 SHIPDATE   | RECDATE    | SHIPNO   |
-------------------------------------
 25/05/1997 | 30/05/1997 | 1495194B |
-------------------------------------
 25/05/1997 | 28/05/1997 | 2942821G |
-------------------------------------
```

- 若要使用 HTML 標籤來代替預先格式化文字，請設定 DTW_HTML_TABLE 爲 YES。
- 若要停用列印預設報告，請設定 DTW_DEFAULT_REPORT 爲 NO 或設定空白的 REPORT 區塊。例如：

```
%REPORT{%}
```

範例：自行設定報表

下面的範例說明如何使用特殊變數及 HTML 標籤，來自行設定報告格式。 它會在表格 CustomerTbl 中，顯示姓名、電話號碼、以及傳眞號碼:

```
%DEFINE SET_TOTAL_ROWS="YES"
...
%FUNCTION(DTW_SQL) custlist() {
     SELECT Name, Phone, Fax FROM CustomerTbl
%REPORT{
<I>電話查詢結果：</I>
<BR>
=====================
<BR>
%ROW{
   姓名： <B>$(V1)</B>
<BR>
電話： $(V2)
<BR>
傳真： $(V3)
<BR>
------------------------------
<BR>
    %}
    取回的記錄總數：$(TOTAL_ROWS)
  %}
%}
```

查詢報告在瀏覽器中看來如下：

電話查詢結果：
```
====================
姓名： Doen, David
電話：422-245-1293
傳真：422-245-7383
```

```
------------------------------
姓名： Ramirez, Paolo
電話：955-768-3489
傳真：955-768-3974
------------------------------
姓名：Wu, Jianli
電話：525-472-1234
傳真：525-472-1234
------------------------------
取回的記錄總數：3
```

Net.Data 會以下列方式來產生報告：

1. 在報告開頭列印一次*電話查詢結果：*。 此本文以及區隔符號行均是 REPORT 區塊的標題部份。
2. 在取回的每一橫列上，對變數 V1、V2 及 V3，分別以「姓名」、「電話」及「傳眞」的值來置換它們。
3. 在報告尾端列印一次字串*取回的記錄總數：* 和 TOTAL_ROWS 的值。 (這個本文是 REPORT 區塊的註腳部份。)

多個 REPORT 區塊

您可以在單一 FUNCTION 或 MACRO FUNCTION 區塊內，設定多個 REPORT 區塊，建立多個具有一個函數呼叫的報表。

一般而言，您將搭配 DTW_SQL 語言環境與多個 REPORT 區塊一起使用，而這個語言環境具有一個函數，可呼叫會傳回多個結果集合的儲存程序 (請參閱 第120頁的『儲存程序』)。不過，多個 REPORT 區塊可與任一語言環境一起用來建立多個報表。

若要使用多個 REPORT 區塊，請將結果集合名稱置於每一結果集合的儲存程序 CALL 之上。如果從儲存程序傳回的結果集合的數目比您設定的 REPORT 報表區塊數目還要多，且如果 Net.Data 內建函數 DTW_DEFAULT_REPORT = ″MULTIPLE″，將對與 REPORT 區塊沒有關聯的每一結果集合建立預設報表。 如果未設定任何報表區塊，且如果 DTW_DEFAULT_REPORT = ″YES″，將僅建立一個預設報表。 請注意，僅適用於 SQL 語言環境，DTW_DEFAULT_REPORT 值 ″YES″ 等於 ″MULTIPLE″ 的值。

範例: 下列範例描述您可以用哪些方式使用多個報表區塊。

使用預設報表格式顯示多個報表：

範例 1：DTW_SQL 語言環境

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION (dtw_sql) myStoredProc () {
     CALL myproc (table1, table2)  %}
```

在這個範例中，儲存程序 myproc 傳回兩個結果集合，置於 table1 與 table2 中。 因爲未設定任何 REPORT 區塊，將顯示的這兩個表格爲預設報表，首先 table1，然後 table2。

範例 2： MACRO_FUNCTION 區塊。 在這個範例中，這兩個表格會傳遞到 MACRO_FUNCTION 區塊。因爲設定了 DTW_DEFAULT_REPORT=″MULTIPLE″，所以 Net.Data 僅會顯示第一個表格的預設報表。當指定 MULTIPLE 時，將會對這兩個表格建立預設報表。

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%MACRO_FUNCTION multReport (INOUT tablename1, tablename2) {
%}
```

在這個範例中，這兩個表格會傳遞到 MACRO_FUNCTION multReport。 再一次，Net.Data 會依據這兩個表格出現在 MACRO FUNCTION 區塊參數列示中的次序，來顯示它們的預設報表，首先 table1，然後 table2。

範例 3：DTW_REXX 語言環境

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (dtw_rexx) multReport (INOUT table1, table2) {
     SAY '正在建立多個預設報表...<BR>'
%}
```

在這個範例中，這兩個表格會傳遞到 REXX 函數 multReport。 因爲設定了 DTW_DEFAULT_REPORT=″YES″，所以 Net.Data 僅會顯示第一個表格的預設報表。

經由設定要進行顯示處理的 REPORT 區塊來顯示多個報表：

範例 1：已命名的 REPORT 區塊

```
%FUNCTION(dtw_sql) myStoredProc () {
   CALL myproc (table1, table2)

  %REPORT(table2) {
     ...
     %ROW {  ....   %}
     ...
   %}

   %REPORT(table1) {
     ...
    %row {  ....   %}
     ...
   %}
%}
```

在這個範例中，已對 FUNCTION 區塊參數列示中傳遞的這兩個表格設定了 REPORT 區塊。表格會依據它們在 REPORT 區塊上設定的次序來顯示，首先 table2，然後 table1。經由在 REPORT 區塊上設定表格名稱，您可以控制報表的顯示次序。

範例 2：沒有名稱的 REPORT 區塊

```
%FUNCTION(dtw_sql) myStoredProc  () {
     CALL  myproc

%REPORT {
     ...
     %ROW {  ....   %}
     ...
   %}
%REPORT {
     ...
     %ROW {  ....   %}
     ...
   %}
%}
```

在這個範例中，已對 FUNCTION 區塊參數列示中傳遞的這兩個表格設定了 REPORT 區塊。 因爲沒有在 REPORT 區塊上設定任何表格名稱，所以會依據這兩個表格從儲存程序傳回的次序，來顯示它們的報表。

使用預設報表與 REPORT 區塊的結合來顯示多個報表：

範例：預設報表與 REPORT 區塊的結合

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION(dtw_system) editTables (INOUT table1, table2, table3) {
  %REPORT(table2) {
    ...
    %ROW {  ....   %}
    ...
  %}
%}
```

在這個範例中，僅會指定一個 REPORT 區塊。因爲區塊指定 table2，且 table2 是 CALL 陳述式上列示的第二個結果集合，所以將使用第二個結果集合來顯示報表。因爲設定的 REPORT 區塊的數目少於從儲存程序傳回的結果集合數目，所以將顯示剩餘的結果集合的報表：首先顯示第一個結果集合的預設報表 table1，然後顯示第三個結果集合的預設報表 table3。指定一個輸出表格 table1，稍後可在巨集檔中處理它。

多個 REPORT 區塊的指南與限制: 在 FUNCTION 或 MACRO_FUNCTION 區塊時設定多個 REPORT 區塊時，請使用下列指南與限制。

準則：

- 您可以對每一結果集合或表格名稱指定一個或多個 REPORT 區塊。對 REPORT 區塊指定的名稱必須符合 CALL 陳述式或 FUNCTION 區塊參數列示中的對應結果集合名稱或表格名稱參數。
- 依多個表格的處理次序來設定它們的 REPORT 區塊。
- 當未設定表格的 REPORT 區塊時，若要設定預設處理，請定義 DTW_DEFAULT_REPORT = ″MULTIPLE″。當 Net.Data 建置網頁時，它會先顯示含有 REPORT 區塊的表格的報表，再顯示表格的預設報表。
- 若要阻止 Net.Data 顯示沒有 REPORT 區塊的表格，請設定 DTW_DEFAULT_REPORT = ″NO″。
- 當搭配一個會傳回多個表格的函數與 DTW_SAVE_TABLE_IN 變數一起使用時，從函數中傳回的第一個表格將指定爲 DTW_SAVE_TABLE_IN 表格。
- 多個報表區塊可以與任何語言環境一起使用。

限制：

- 函數中所有報表變數的值將套用到該函數中的所有 REPORT 區塊。您無法替個別的 REPORT 區塊來修改報告變數的值。
- MESSAGE 區塊必須位於 REPORT 區塊的前或後，不可位於 REPORT 區塊之間。
- 表格變數在傳遞到函數之前，必須在 TABLE 陳述式內定義它們。
- 如果第一個報表區塊指定表格名稱，則所有報表區塊均須指定表格名稱。
- 如果第一個報表區塊未指定表格名稱，則所有報表區塊不可指定表格名稱。
- 單一儲存程序的最大表格數目爲 32。

巨集中的條件式邏輯和迴路

Net.Data 可讓您使用 IF 和 WHILE 區塊，在您的 Net.Data 巨集中納入條件式邏輯和迴路。

IF 區塊及 WHILE 區塊會使用一個條件列示，協助您測試一個或數個條件，然後依據條件測試的結果，來執行陳述式區塊。條件列示含有邏輯運算子，如 = 及 <+，以及函數項，這些會構成引號內的字串、變數、變數參照及函數呼叫。引號內的字串可以含有變數參照及函數呼叫。您可以將條件列示巢狀化。

下列段落將描述條件性邏輯及迴路：

- 『條件性邏輯：IF 區塊』
- 第111頁的『迴路結構：WHILE 區塊』

條件性邏輯：IF 區塊

在 Net.Data 巨集中，請使用 IF 區塊來執行條件式處理。該 IF 區塊類似於大部份高階語言中的 IF 陳述式，因其提供測試一個或數個條件，然後基於條件測試的結果，來執行陳述式區塊的能力。

您幾乎可於巨集中的任意處設定 IF 區塊，且可以使用巢狀。有關 IF 區塊的語法，請參閱 *Net.Data 參考手冊*中有關語言結構的那一章。

IF 區塊規則：IF 區塊語法的規則，是由區塊在巨集中的位置所決定。IF 區塊的可執行區塊陳述式中，容許的元素是基於 IF 區塊本身的位置而定。

- 在含有 IF 區塊的區塊中有效之任何元素，在該 IF 區塊之內亦有效。 例如，若您在 HTML 區塊中設定 IF 區塊，則 HTML 區塊中容許的任何元素，在 IF 區塊中也容許，例如 INCLUDE 陳述式和 WHILE 區塊。

```
  %HTML 區塊
...
     %IF 區塊
...
    %INCLUDE
...
    %WHILE
...
%ENDIF
%}
```

- 同樣地，若於 Net.Data 巨集之宣告部份的其他區塊以外設定 IF 區塊，則只有其他區塊 (例如 DEFINE 區塊或 FUNCTION 區塊) 之外容許的那些元素，在 IF 區塊中才容許。

```
%IF
...
%DEFINE
...
  %FUNCTION
...
%ENDIF
```

- 當某個 IF 區塊巢狀於另一個 IF 區塊內，而後者位於宣告部份中其他區塊之外時，可以使用外部區塊所使用的任何元素。當某個 IF 區塊巢狀於另一個區塊內，而後者位於某個 IF 區塊中時，請使用所在區塊的語法規則。

  例如，巢狀 IF 區塊必須遵循當它在 HTML 區塊之中時的規則。

```
%IF
...
  %HTML {
...
%IF
...
```

```
  %ENDIF
    %}
  ...
  %ENDIF
```

例外狀況：請勿在 IF 區塊中設定 ROW 區塊。

IF 區塊字串比較

Net.Data 基於構成條件的函數項內容所產生的方式之一，來處理 IF 區塊條件列示。 預設動作是將所有函數項視爲字串，並依條件所示執行字串比較。不過，如果在兩個代表整數的字串之間作比較，這是一種數值比較。如果字串僅含有數字，且選用性在前面有 '+' 或 '-' 字元，則 Net.Data 將假設它爲數字。 字串不能包含 '+' 或 '-' 以外的任何非數字字元。 Net.Data 不支援非整數數字的數值比較。

有效整數字串的範例：

```
+1234567890
-47
000812
92000
```

無效整數字串的範例：

```
-  20     (包含空白字元)
234,000   (包含逗號)
57.987    (包含小數點)
```

Net.Data 於執行 IF 條件時會評估該區塊，這個時間可不同於 Net.Data 原來讀取它的時間。例如，若您在 REPORT 區塊中設定 IF 區塊，則 Net.Data 在讀取含有 REPORT 區塊的 FUNCTION 區塊定義時，不會評估與 IF 區塊相關的條件列示，而是在呼叫並執行函數時才評估。 這在 IF 區塊的條件列示部份和要執行的區塊陳述式兩者皆是如此。

IF 區塊範例：在其他區塊內含有 IF 區塊的巨集

```
%{ 此巨集是從另一個巨集被呼叫，在此套表資料中
   傳送作業系統及版本變數。
%}

%IF (platform == "AS400")
  %IF (version == "V3R2")
     %INCLUDE "as400v3r2_def.hti"
  %ELIF (version == "V3R7")
     %INCLUDE "as400v3r7_def.hti"
  %ELIF (version == "V4R1")
     %INCLUDE "as400v4r1_def.hti"
%ENDIF
%ELSE
     %INCLUDE "default_def.hti"
%ENDIF
%MACRO_FUNCTION numericCompare(IN term1, term2, OUT result) {
%IF (term1 < term2)
  @dtw_assign(result, "-1")
%ELIF (term1 > term2)
  @dtw_assign(result, "1")
%ELSE
  @dtw_assign(result, "0")
%ENDIF
%}

%HTML(report){
  %WHILE (a < "10") {
```

```
    外部 while 迴路 #$(a)<BR>
    %IF (@dtw_rdivrem(a,"2") == "0")
      這是奇數迴路<BR>
%ENDIF
    @DTW_ADD(a, "1", a)
  %}
%}
```

迴路結構：WHILE 區塊

使用 WHILE 區塊，於 Net.Data 巨集中執行迴路。如同 IF 區塊， WHILE 區塊提供測試數個條件，然後基於條件測試的結果，來執行陳述式區塊的能力。不像 IF 區塊，陳述式區塊可基於條件測試的結果，不限次數地執行。

您可以在 HTML 區塊、REPORT 區塊、ROW 區塊、 MACRO_FUNCTION 區塊及 IF 區塊內指定 WHILE 區塊，並且可以使用巢狀。有關 WHILE 區塊的語法，請參閱 *Net.Data 參考手冊*中有關語言結構的那一章。

Net.Data 處理 WHILE 區塊的方式，與處理 IF 區塊的方式完全相同，但會每次執行區塊後會重新評估條件。而且，如同其他條件式迴路結構一樣，若撰寫條件不正確，則處理程序也可能會陷入無限迴路內。

範例：具有 WHILE 區塊的巨集

```
%DEFINE loopCounter = "1"

%HTML(build_table) {
%WHILE (loopCounter <= "100") {
    %{ generate table tag and column headings %}
    %IF (loopCounter == "1")
      <TABLE BORDER>
      <TR>
      <TH>Item #
      <TH>說明
%ENDIF

  %{ 產生個別橫列 %}
    <TR>
    <TD>$(loopCounter)
    <TD>@getDescription(loopCounter)

  %{ 產生結束表格標籤 %}
  %IF (loopCounter == "100")
%ENDIF

  %{ 增加迴路計數 %}
    @DTW_ADD(loopCounter, "1", loopCounter)
  %}
%}
```

第6章 使用語言環境

Net.Data 會提供一些語言環境，您可以用它們來存取資料來源，以及執行包含企業邏輯的應用程式。例如： SQL 語言環境可讓您將 SQL 陳述式傳遞給 DB2 資料庫，而 REXX 語言環境可讓您呼叫 REXX 程式。 您也可以使用 SYSTEM 語言環境來執行程式或發出命令。

有了 Net.Data，您就可以在可插入的形態中，新增使用者撰寫的語言環境。每一個使用者撰寫的語言環境都必須支援一組由 Net.Data 定義的標準介面，而且必須當作動態鏈結程式庫或共用程式庫來實施。關於 Net.Data 提供的語言環境及如何建立使用者撰寫的語言環境的詳細資訊，請參閱*Net.Data 語言環境介面參考手冊*。

圖21顯示 Web 伺服器、Net.Data 及 Net.Data 語言環境之間的關係。

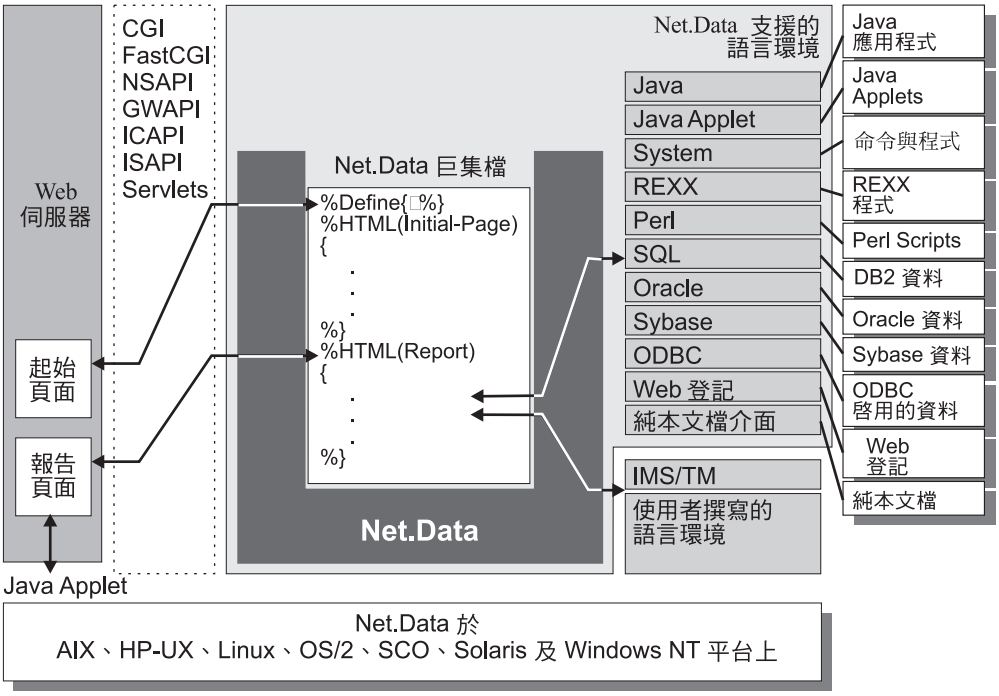


圖 21. Net.Data 語言環境

下列段落將描述 Net.Data 語言環境，以及如何在您的巨集中使用它們：

- 第114頁的『Net.Data 提供的語言環境的概觀』
- 第114頁的『呼叫語言環境』
- 第115頁的『資料語言環境』
- 第133頁的『程式設計語言環境』

關於 Net.Data 提供的語言環境的架構資訊，請參閱第22頁的『設置語言環境』。

關於使用語言環境時如何增進執行效能的資訊，請參閱第171頁的『將語言環境最佳化』。

Net.Data 提供的語言環境的概觀

Net.Data 會提供語言環境，讓您存取資料及程式設計資源，供您的應用程式使用。

Net.Data 提供兩種類型的語言環境：

- 第115頁的『資料語言環境』
- 第133頁的『程式設計語言環境』

表8提供每一語言環境的簡短說明。 請參閱 *Net.Data 參考手冊*的作業系統附錄，瞭解哪些作業系統支援哪些語言環境。

表 8. Net.Data 語言環境

語言環境	環境名稱	說明
純本文檔介面	DTW_FILE	純本文檔介面 (FFI) 提供可支援文字檔作爲資料來源的函數。
IMS Web	HWS_LE	IMS Web 語言環境可讓您使用 IMS Web 提出 IMS 異動，並在您的 Web 瀏覽器中接收異動的輸出。
Java Applet	DTW_APPLET	Java applet 語言環境可讓您在 Net.Data 應用程式中使用 Java applet。若要建立 applet 標籤，您必須提供 applet 標籤的限定元及 applet 的參數列示。
Java 應用程式	DTW_JAVAPPS	Net.Data 會支援具有 Java 語言環境的舊有 Java 應用程式。
ODBC	DTW_ODBC	ODBC 語言環境會透過 ODBC 介面，執行 SQL 陳述式，來存取多個資料庫管理系統。
Oracle	DTW_ORA	Oracle 語言環境可讓您直接存取 Oracle 資料。
Perl	DTW_PERL	Perl 語言環境可解譯 Net.Data 巨集的 FUNCTION 區塊中指定的內部 Perl script，或可以執行儲存於個別檔案中的外部 Perl script。
REXX	DTW_REXX	REXX 語言環境可解譯 Net.Data 巨集的 FUNCTION 區塊中指定的內部 REXX 程式，或可以執行儲存於個別檔案中的外部 REXX 程式。
SQL	DTW_SQL	SQL 語言環境會透過 DB2 來執行 SQL 陳述式。可在表格變數中傳回 SQL 陳述式的結果。
Sybase	DTW_SYB	Sybase 語言環境可讓您直接存取 Sybase 資料。
系統	DTW_SYSTEM	系統語言環境支援執行指令及呼叫外部程式。
Web 登記	DTW_WEBREG	「Web 登記」語言環境會提供函數，供應用程式相關的資料的持續儲存體使用。

呼叫語言環境

若要呼叫語言環境：

- 使用 FUNCTION 陳述式，來定義一個可呼叫語言環境的函數。
- 使用語言環境的函數呼叫。

例如：

```
%FUNCTION(DTW_SQL) custinfo() {
 select CUSTNAME, CUSTNO from ibmuser.customer
 %}
```

```
 ...
%HTML(REPORT){
 @custinfo()
 %}
```

處理錯誤狀況

在語言環境函數中偵測到錯誤時，語言環境將會用錯誤碼設定 Net.Data RETURN_CODE 變數。

您可以使用下列資源，來處理錯誤狀況：

- Net.Data 提供的語言環境會傳回錯誤碼，它的說明在 *Net.Data 訊息與訊息碼參考手冊*中。
- 資料庫語言環境 (如 SQL 語言環境) 會將 RETURN_CODE 設定爲資料庫管理系統 (DBMS) 傳回的錯誤碼，稱爲 SQLCODE。請參閱 DBMS 的訊息與訊息碼文件，來瞭解 DBMS 所使用的 SQLCODE。

安全

確定 Net.Data 執行時所用的使用者 ID 具有適當的權限，可存取語言環境陳述式的目標所參照的任何物件。例如，SQL 語言環境會執行 SQL 陳述式，而 SQL 陳述式會存取資料庫檔案，所以 Net.Data 執行時所用的使用者 ID 須具有資料庫檔案的權限。

資料語言環境

Net.Data 提供的資料語言環境能夠讓您從關聯式及階層式資料庫中存取資料，以及從 Net.Data 巨集存取其他來源。下列段落將描述 Net.Data 提供的語言環境，以及如何在您的 Net.Data 巨集中使用它們：

- 『關聯式資料庫語言環境』
- 第129頁的『純本文檔介面語言環境』
- 第130頁的『Web 登記語言環境』
- 第132頁的『IMS Web 語言環境』

關聯式資料庫語言環境

Net.Data 會提供關聯式資料庫語言環境，來協助您存取關聯式資料來源。Net.Data 會提供下列關聯式資料庫語言環境：

ODBC 語言環境

ODBC 語言環境會透過 ODBC 介面來執行 SQL 陳述式。 ODBC 是架構在 X/Open SQL CAE 規格上，它可以讓單一應用程式存取許多資料庫管理系統。

若要使用 ODBC 語言環境：

具有 ODBC 驅動常式及驅動常式管理程式。您的 ODBC 驅動常式文件會描述如何安裝及架構您的 ODBC 環境。

驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上。

```
ENVIRONMENT (DTW_ODBC)     DTWODBC   ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

限制：

- 當指定 DATABASE 變數時，您必須在起始設定檔案中指定與資料來源相同的資料庫。
- 列入陳述式區塊中的 SQL 陳述式最多可達到 64 KB。 DB2 Universal Database 具有下列限制：
  - 版本 6 或更新的版本：64 KB
  - 版本 5 版次 2 或更舊的版本：32 KB

  您的資料庫可能具有不同的限制：請參閱資料庫文件，來決定資料庫是否具有不同限制。

Oracle 語言環境

Oracle 語言環境提供 Oracle 資料的原始存取權。當使用 CGI, FastCGI, NSAPI, ISAPI, ICAPI 或 GWAPI 時，您可以從 Net.Data 存取 Oracle 資料庫。這個語言環境將支援 Oracle 7.2, 7.3 及 8.0。

若要使用 Oracle 語言環境：

驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上。

```
ENVIRONMENT (DTW_ORA)      DTWORA    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

重要事項：請參閱第24頁的『設置 Oracle 語言環境』，學習如何設置 Oracle 語言環境。

限制：

Oracle 語言環境具有下列限制：

- DATABASE 變數不會用來存取 Oracle 資料庫。
- LOGIN 變數必須含有 Oracle 資料庫案例名稱。例如， *ora73* 是在下列 LOGIN 變數中的已定義案例名稱：

  ```
  LOGON=admin@ora73
  ```

- 當使用非 CGI 的介面時，您必須使用「現場連線」。
- 不支援儲存程序。

SQL 語言環境

SQL 語言環境提供 DB2 資料庫的存取權。當存取 DB2 時，請使用這個語言環境，取得最佳效能。

若要使用 SQL 語言環境：

驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上。

```
ENVIRONMENT (DTW_SQL)      DTWSQL    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

限制：

列入陳述式區塊中的 SQL 陳述式最多可達到 64 KB。 DB2 Universal Database 具有下列限制：

- 版本 6 或更新的版本：64 KB
- 版本 5 版次 2 或更舊的版本：32 KB

您的資料庫可能具有不同的限制：請參閱資料庫文件，來決定 DBMS 是否具有不同限制。

Sybase 語言環境

Sybase 語言環境提供 Sybase 資料的原始存取權。當使用 CGI, FastCGI, NSAPI, ISAPI, ICAPI 或 GWAPI 模式時，您可以從 Net.Data 存取 Sybase 資料庫。

若要使用 Sybase 語言環境：

驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上。

```
ENVIRONMENT (DTW_SYB)      DTWSYB    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

重要事項：請參閱第26頁的『設置 Sybase 語言環境』，學習如何設置 Sybase 語言環境。

限制：

Sybase 語言環境具有下列限制：

- Sybase 語言環境不支援大型物件，如影像及音效。僅沒有 SELECT 陳述式的程序才支援儲存程序。
- 列入陳述式區塊中的 SQL 陳述式最多可達到 64 KB。您的資料庫可能具有不同的限制：請參閱資料庫文件，來決定資料庫是否具有較低限制。
- 當使用非 CGI 的介面時，您必須使用「現場連線」。
- 不支援儲存程序。

下列段落說明如何使用這些語言環境。

- 『管理 Net.Data 應用程式中的異動』
- 第118頁的『使用大型物件』
- 第120頁的『儲存程序』
- 第126頁的『編碼結果集合中的 DataLink URL』
- 第127頁的『關聯式資料庫語言環境範例s』

管理 Net.Data 應用程式中的異動

當您使用插入、刪除或更新陳述式來修改資料庫的內容時，修改的內容將不會變成永久的，除非資料庫從 Net.Data 收到一個確定陳述式。如果發生錯誤，則 Net.Data 將傳送一個回轉陳述式給資料庫，反轉自上次確定後所做的任何修改。

Net.Data 傳送確定及可能回轉的方式取決於您如何設定 TRANSACTION_SCOPE，以及您是否明確地在巨集中指定確定。TRANSACTION_SCOPE 的值是 MULTIPLE 及 SINGLE。

MULTIPLE

指定在發出確定及可能回轉陳述式之前，Net.Data 將執行所有 SQL 陳述式。在要求結束時，Net.Data 會傳送確定，而且如果順利發出了每一個 SQL 陳述式，則確定將使得資料庫中的所有修改內容變成永久的。如果任一陳述式傳回錯誤，則 Net.Data 將發出回轉陳述式，將資料庫設定回它的原始狀態。如果未設定 TRANSACTION_SCOPE，MULTIPLE 將是預設值。

若要啓動這個確定方法，請將 TRANSACTION_SCOPE 設定爲 MULTIPLE。

例如：

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"MULTIPLE")
```

SINGLE

指定在每一個 SQL 陳述式順利完成後，Net.Data 將發出一個確定陳述式。如果 SQL 陳述式傳回錯誤，將發出回轉陳述式。單一異動範圍可立即保障資料庫的修改；不過，使用這個範圍後，稍後可能無法使用回轉陳述式來還原修改的內容。

若要啓動這個確定方法，請將 TRANSACTION_SCOPE 設定爲 SINGLE。例如：

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"SINGLE")
```

您可以使用 COMMIT SQL 陳述式，在巨集中的任一 SQL 陳述式結束時，發出一個確定陳述式。經由使 TRANSACTION_SCOPE 設定爲 MULTIPLE，並在那些您覺得已合格作爲異動的陳述式群組結束時發出確定陳述式，您 (應用程式軟體開發者) 便可以完全控制應用程式中的確定及回轉行爲。 例如，在完成您的巨集中的每一次更新後發出確定陳述式，可協助您確保資料的完整性。

若要發出 SQL 確定陳述式，您可以定義一個可在您的 HTML 區塊中任一點呼叫的函數：

```
%FUNCTION(DTW_SQL) user_commit() {
  commit
%}
...
%HTML {
  ...
  @user_commit()
  ...
%}
```

使用大型物件

您可以將大型物件檔案 (LOB) 儲存在 DB2 資料庫中，然後使用 SQL 或 ODBC 語言環境來存取它們，供您的 Web 應用程式使用。

當 SQL 查詢在結果集合中傳回 LOB 時， SQL 或 ODBC 語言環境不會將大型物件儲存在處理變數 (如 V1 或 V2) 的 Net.Data 表格中, 或儲存在 Net.Data 表格欄位中。相反地，當 Net.Data 發現 LOB 時，它會將 LOB 儲存在 Net.Data 建立的檔案中。這個檔案位在 HTML_PATH 路徑架構變數所指定的目錄中。Net.Data 表格欄位及處理變數的表格將設定爲該檔案的路徑。僅執行 Net.Data 時所用的使用者 ID 方可存取這個檔案。

儲存 LOB 的檔案名稱是以動態方式建構的，具有下列格式：

名稱[.*副檔名*]

其中：

名稱 是識別大型物件的唯一字串

副檔名 是識別物件類型的字串。對於 CLOB 及 DBCLOB，副檔名爲 'txt'。對於 BLOB，SQL 語言環境會嘗試決定副檔名，其方法是尋找 LOB 檔的前幾個位元組中的標記，因爲它會指出 LOB 代表什麼。 SQL 語言環境可識別下列類型的資料 (括弧中的內容即是使用的副檔名)：

- 位元圖形影像 (.bmp)
- 圖形式影像格式 (.gif)
- JPEG 影像檔 (.jpg)
- Tagged 影像檔格式 (.tif)
- Postscript (.ps)
- 樂器數位介面音效檔 (.mid)
- AIFF 音效檔 (.aif)
- 音效視覺交錯音效檔 (.avi)
- 基本音效檔 (.au)
- 眞實音效檔 (.ra)
- Windows 音效視覺檔 (.wav)
- 可攜性文件格式 (.pdf)
- Midi 順序檔 (.rmi)

如果無法識別 BLOB 的物件類型，將不會對檔名新增副檔名。

在巨集檔中參照 LOB 時，SQL 語言環境將使用下列語法傳回檔案名稱，這個名稱會具有 LOB 檔案名稱前加的 /tmplobs/ 字串：

/tmplobs/*名稱*.[*副檔名*]

規劃要訣：當每一個查詢傳回 LOB 時，會造成檔案將在 HTML_PATH 路徑架構變數指定的目錄中建立。由於 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮到系統限制問題。您可能想要建立批次程式，定期清除目錄，或執行 dtwclean 常駐程式。請參閱第120頁的『管理暫時 LOBS』，取得詳細資訊。建議您最好使用 DataLinks，因爲它不需要透過 SQL 語言環境，將檔案儲存在目錄中，因而造成更佳的效能，且使用更少的系統資源。

範例：應用程式使用者必須按一下檔名，方可呼叫檢視器，因爲應用程式使用 MPEG 音效 (.MPA) 檔。 SQL 語言環境不認得此種檔案類型，因此會使用 EXEC 變數，將副檔名附加至檔案中。

```
%DEFINE{
docroot="/usr/lpp/internet/server_root/html"
myFile=%EXEC "更名 $(docroot)$(V3) $(docroot)$(V3).mpa"
%}
%{ 其中更名是您的作業系統中的更名指令 %}
%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
%REPORT{
<P>底下是您選取的資訊：<P>
%ROW{
$(myFile)
$(V1) Voice sample <IMG SRC="$(V2)">
<A HREF="$(V3).mpa")>Voice sample</A><P>
%}
%}
%}

%HTML(REPORT){
@queryData()
%}
```

queryData 函數會傳回下列 HTML 輸出：

```
<P>Here are the images you selected:<P>
Kinson Yamamoto
<IMG SRC="/tmplobs/p2345n1.gif">
<A HREF="/tmplobs/p2345n2.mpa">Voice sample</A><P>
Merilee Lau
<IMG SRC="/tmplobs/p2345n3.gif">
<A HREF="/tmplobs/p2345n4.mpa">Voice sample</A><P>
```

前一個範例中的 REPORT 區塊，使用隱含的表格變數 V1、V2、與 V3。

- V1 代表個人的名稱，爲純文字。
- V2 代表 .GIF 格式的個人照片檔。此圖片會以列入方式顯示。 SQL 語言環境會自動併入字首 /tmplobs/ 及 .GIF 副檔名。
- V3 是 .MPA 格式的個人聲音樣本檔。當 Net.Data 遇到不認得的檔案格式時，(如：.MPA 檔)，其會將檔案寫入 LOB 暫時目錄中，而不加上副檔名。 此範例是說明在處理此種檔案類型時，該如何使用 EXEC 變數來新增副檔名。 解析變數 *$(V3)* 時，會於檔名之前加上路徑 /tmplobs/。例如，/tmplobs/sound2a。 處理此類檔案的方法之一，是寫入 REXX 程式把斜線改成反斜線後，並且變更檔案的名稱。 當應用程式的使用者按一下「聲音樣本」時，就會播放聲音樣本。

LOB 的存取權： LOB 的預設 tmplobs 目錄是位在提供的 Net.Data 起始設定檔案中的 HTML_PATH 指定的目錄下。可以透過任何使用者 ID 來存取它。如果變更 HTML_PATH 值，請確定 Web 伺服器執行時所使用的使用者 ID 對 HTML_PATH 指定的目錄具有寫入權 (請參閱 第20頁的『HTML_PATH』，取得詳細資訊)。

管理暫時 LOB：

Net.Data 會將暫時 LOB 儲存在名爲 tmplobs 的次目錄下，這個次目錄位在 HTML_PATH 路徑架構變數中指定的目錄下。這些檔案可以是大型的，但應該定期清除，以保持可接受的效能。

Net.Data 提供一種名爲 dtwclean 的常駐程式，來協助您定期管理 tmplobs 目錄。dtwclean 會使用埠 7127。

若要執行 dtwclean常駐程式： 請從指令行視窗中輸入下列指令：

```
dtwclean [-t xx] [-d|-l]
```

其中：

-t 是指定 dtwclean 清除目錄的間隔的旗號

xx 爲一間隔 (秒)，表示在 dtwclean 消除檔案之前，檔案可留在目錄中的時間。這個值沒有限制。預設值爲 3600 秒。

-d 是指定除錯模式的旗號；追蹤資訊會顯示在指令視窗中。

-l 是指定記錄模式的旗號；追蹤資訊會列印到日誌檔。

儲存程序

儲存程序是一種已編譯的程式，儲存於 DB2 中，可執行 SQL 陳述式。在 Net.Data 中，儲存程序是從 Net.Data 函數上使用 CALL 陳述式來呼叫。儲存程序參數是從 Net.Data 函數參數列示中來傳入。您可以使用儲存程序，經由保存編譯過的 SQL 陳述式和資料庫伺服器，來改善執行效能和提升完整性。 Net.Data 支援透過 SQL 與 ODBC 語言環境，以 DB2 使用儲存程序。

本段落說明下列主題：

- 『儲存程序語法』
- 『呼叫儲存程序』
- 第122頁的『傳送參數』
- 第123頁的『處理結果集合』

儲存程序語法: 儲存程序的語法使用 FUNCTION 陳述式、CALL 陳述式，以及可選用的 REPORT 區塊。

```
%FUNCTION (DTW_SQL) function_name ([IN datatype arg1, INOUT datatype arg2,
    OUT tablename, ...]) {
  CALL stored_procedure [(resultsetname, ...)]
[%REPORT [(resultsetname)] { %}]
...
[%REPORT [(resultsetname)] { %}]
[%MESSAGE %}]
%}
```

其中：

function_name
指起始儲存程序呼叫的 Net.Data 函數的名稱

stored_procedure
儲存程序名稱

datatype
指 Net.Data 支援的其中一種資料庫資料類型，顯示在表9中。 參數列示中指定的資料類型必須與儲存程序中的資料類型相配。 有關這些資料類型的說明，請參閱您的資料庫文件。

tablename
指將儲存結果集合的 Net.Data 表格的名稱 (僅在結果集合將儲存在 Net.Data 表格時才會使用)。若設定的話，這個參數須符合 *resultsetname* 的關聯參數名稱

resultsetname
指與透過 REPORT 區塊及函數參數列示上的表格名稱 (或兩者) 從儲存程序傳回的結果有關聯的名稱。 REPORT 區塊上的 *resultsetname* 須符合 CALL 陳述式上的結果集合。

表 9. 儲存程序的資料類型

BIGINT	DOUBLEPRECISION	SMALLINT
CHAR	FLOAT	TIME
DATE	INTEGER	TIMESTAMP
DECIMAL	GRAPHIC	VARCHAR
DOUBLE	LONGVARCHAR	VARGRAPHIC
	LONGVARGRAPHIC	

呼叫儲存程序:

1. 定義起始儲存程序呼叫的函數。

   ```
   %FUNCTION (DTW_SQL) function_name()
   ```

2. 可選擇是否要對儲存程序設定任何 IN、INOUT 或 OUT 參數，包括將結果集合儲存在 Net.Data 表格時所用的表格變數名稱 (如果您想要將結果集合儲存在 Net.Data

表格，僅需設定一個 Net.Data 表格)。 您也可以從另一個儲存程序，指定表格名稱或結果集合同於 IN 或 INOUT 參數值。

```
%FUNCTION (DTW_SQL) function_name (IN datatype
arg1, INOUT datatype arg2, OUT tablename...)
```

3. 使用 CALL 陳述式來識別儲存程序名稱。

```
CALL stored_procedure
```

4. 如果儲存程序將建立一個結果集合，則可選擇是否要設定一個 REPORT 區塊，來定義 Net.Data 如何顯示結果集合。

```
%REPORT (resultsetname) {
...
%}
```

範例：

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1)  {
     CALL  myproc
 %REPORT (mytable){
  ...
  %ROW {  ...   %}
  ...
 %}
%}
```

5. 如果儲存程序將建立多個結果集合：
   - 在 CALL 陳述式上設定結果集合名稱。

     ```
     CALL stored_procedure (resultsetname1, resultsetname2, ...)
     ```

   - 可選擇是否要設定一個或多個 REPORT 區塊，來定義 Net.Data 如何顯示結果集合。

     ```
     %REPORT(resultsetname1) {
     ...
     %}
     ```

     範例：

     ```
     %FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1, OUT table1) {
        CALL myproc (table1, table2)
       %REPORT(table2) {
            ...
            %ROW {  ...   %}
            ...
      %}
     %}
     ```

傳送參數: 您可以將參數傳遞給儲存程序，並使儲存程序更新參數值，以便新值將傳回到 Net.Data 巨集。 函數參數列示上的參數數目及類型必須符合針對儲存程序定義的數目及類型。例如，如果針對儲存程序定義的參數列示上的參數爲 INOUT，則函數參數列示上對應的參數必須是 INOUT。例如，如果針對儲存程序定義的參數列示上的參數爲類型 CHAR(30)，則函數參數列示上對應的參數也必須是 CHAR(30)。

範例：傳送參數值至儲存程序

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) valuein)  {
     CALL  myproc
...
```

範例：從儲存程序傳回值

```
%FUNCTION (DTW_SQL) mystoredproc (OUT VARCHAR(9) retvalue)  {
     CALL  myproc
...
```

處理結果集合: 您可以使用 SQL 或 ODBC 語言環境，從儲存程序傳回一個或多個結果集合。結果集合可以儲存在 Net.Data 表格中，以便在您的巨集內進一步處理它們，或使用 REPORT 區塊處理它們。如果儲存程序建立多個結果集合，您必須使名稱與儲存程序所建立的每一個結果集合產生關聯。這是經由在 CALL 陳述式上設定參數來完成。然後，您針對結果集合設定的名稱可以與 REPORT 區塊或 Net.Data 表格產生關聯，使您能夠決定 Net.Data 將如何處理每一個結果集合。您可以：

- 不定義結果集合的報表區塊，使結果在 Net.Data 的預設報表樣式中處理。
- 使結果集合與 REPORT 區塊產生關聯，來引用自己的報告樣式。在 REPORT 區塊中，您可以使用 Net.Data 變數、文字處理陳述式 (如 HTML 或 JavaScript)，或其他函數，來設定報表資料如何顯示在瀏覽器上。
- 當您想要 Net.Data 稍後在巨集中使用資料，請將結果集合儲存在 Net.Data 表格中。例如，您可以將 Net.Data 表格傳遞給另一個函數，以便它可以使用資料來進行計算，並依據那些計算來顯示結果。

請參閱第108頁的『多個 REPORT 區塊的指南與限制』，取得當使用多個報表區塊時的指南與限制。

若要傳回單一結果集合及使用預設報告：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name () {
     CALL stored_procedure
%}
```

例如：

```
 %FUNCTION (DTW_SQL) mystoredproc() {
     CALL  myproc
%}
```

若要傳回單一結果集合及指定 REPORT 區塊：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name () {
     CALL stored_procedure  [(resultsetname)]
 %REPORT [(resultsetname)] {
  ...
 %}
%}
```

例如：

```
%FUNCTION (DTW_SQL) mystoredproc () {
     CALL  myproc
%REPORT {
      ...
      %ROW {  ...   %}
      ...
 %}
%}
```

另一種方式即是使用下列語法：

```
%FUNCTION (DTW_SQL) function_name () {
    CALL stored_procedure (resultsetname)

 %REPORT (resultsetname) {
  ...
 %}
%}
```

例如：

```
%FUNCTION (DTW_SQL) mystoredproc () {
    CALL myproc (mytable1)
 %REPORT (mytable1) {
  ...
  %ROW {  ...   %}
  ...
 %}
%}
```

若要將單一結果集合儲存在 Net.Data 表格中做進一步的處理：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name (OUT tablename) {
   CALL stored_procedure (resultsetname)
%}
```

例如：

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1) {
   CALL myproc (mytable1)
%}
```

請注意，DTW_DEFAULT_REPORT 將設定爲 NO，以便不會替結果集合建立一個預設報表。

若要傳回多個結果集合並使用預設報表格式顯示它們：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name () {
    CALL stored_procedure [(resultsetname1, resultsetname2, ...)]
%}
```

其中沒有報表區塊。

例如：

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (DTW_SQL) mystoredproc () {
     CALL  myproc
%}
```

若要傳回多個結果集合，並將結果集合儲存在 Net.Data 表格中做進一步的處理：

可使用下列語法：

```
 %FUNCTION (DTW_SQL) function_name (OUT tablename1, tablename2, ...) {
    CALL stored_procedure (resultsetname1, resultsetname2, ...)
%}
```

例如：

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1, mytable2) {
     CALL myproc (mytable1, mytable2)
%}
```

請注意，DTW_DEFAULT_REPORT 將設定爲 NO，以便不會替結果集合建立一個預設報表。

若要傳回多個結果集合並設定 REPORT 區塊來進行處理顯示：

每一個結果集合均會與它的一個或多個 REPORT 區塊 產生關聯。可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name (, ...) {
   CALL stored_procedure (resultsetname1, resultsetname2, ...)
 %REPORT (tablename1)
   ...
   %ROW {  ...   %}
   ...
 %}
 %REPORT (tablename2)
   ...
   %ROW {  ...   %}
   ...
 %}
...
%}
```

例如：

```
%FUNCTION (DTW_SQL) mystoredproc () {
   CALL myproc (mytable1, mytable2)

 %REPORT(mytable1) {
  ...
  %ROW {  ...   %}
  ...
 %}

 %REPORT(mytable2) {
  ...
  %ROW {  ...   %}
  ...
 %}
%}
```

若要傳回多個結果集合，並設定每一結果集合的不同顯示或處理選項：

您可以使用唯一的參數名稱，設定每一結果集合的不同處理選項。例如：

```
%FUNCTION (DTW_SQL) mystoredproc (OUT mytable2) {
    CALL myproc (mytable1, mytable2, mytable3)

 %REPORT(mytable1)
 ...
 %ROW {  ...   %}
 ...
 %}
%}
```

結果集合 mytable1 將被對應的 REPORT 區塊處理，並按照巨集撰寫者的設定方式顯示。結果集合 mytable2 儲存在 Net.Data 表格 mytable2 中，且現在可用來做進一步的處理，如傳遞給另一個函數。結果集合 mytable3 使用 Net.Data 的預設報表格式來顯示，因爲沒有替它設定任何 REPORT 區塊。

編碼結果集合中的 DataLink URL

DataLink 資料類型是擴充可儲存在資料庫檔案的資料類型的基本建置區塊之一。透過 DataLink，儲存在直欄中的眞正資料僅是檔案的指標。這個檔案可以是任一檔案類型；影像檔、聲音記錄檔或文字檔。 DataLinks 會儲存一個 URL，來解析檔案的位置。

DATALINK 資料類型需要使用「DataLink 檔案管理程式」。 「DataLink 檔案管理程式」的詳細資訊，請參閱您的作業系統的 DataLink 文件。在您使用 DATALINK 資料類型之前，您必須確定 Web 伺服器有權存取「DB2 檔案管理程式伺服器」管理的檔案系統。

當 SQL 查詢透過 DataLink 來傳回結果集合，且 DataLink 直欄是以具有 READ PERMISSION DB DataLink 選項的 FILE LINK CONTROL 來建立時，DataLink 直欄中的檔案路徑將含有存取符記。 DB2 會使用這個存取符記，來鑑定檔案的存取權。沒有這個存取符記，所有存取檔案的嘗試將因違反權限而失敗。不過，存取符記可能含有在將傳回給瀏覽器的 URL 中無法使用的字元，如分號 (;) 字元。例如：

```
/datalink/pics/UN1B;0YPVKGG346KEBE;baibien.jpg
```

URL 無效，因爲它含有分號 (;) 字元。若要使得 URL 有效，則必須使用 Net.Data 內建函數 DTW_URLESCSEQ 來編碼分號。不過，在引用這個函數之前，必須先執行某些字串操作，因爲這個函數也會編碼斜線 (/)。

您可以撰寫 Net.Data MACRO_FUNCTION，使字串操作自動化，並使用 DTW_URLESCSEQ 函數。請在每一個從 DATALINK 資料類型直欄取回資料的巨集中使用這個技術。

範例 1：自動編碼從 DB2 UDB 傳回的 URL 的 MACRO_FUNCTION

```
%{ 執行：將 DTW_URLESCSEQ 引用到 DATALINK URL，使它成為有效的 URL。
    輸入：來自 DB2 檔案管理程式直欄的 DATALINK URL。
    傳回：具有符記部份的 URL 是已編碼的 URL
%}
%MACRO_FUNCTION encodeDataLink(in DLURL) {
  @DTW_rCONCAT( @DTW_rDELSTR( DLURL,
  @DTW_rADD(@DTW_rLASTPOS("/", DLURL), "1" ) ),
  @DTW_rURLESCSEQ( @DTW_rSUBSTR(DLURL,
  @DTW_rADD( @DTW_rLASTPOS("/", DLURL), "1" ) ) ) )
%}
```

在使用這個 MACRO_FUNCTION 後，已適當編碼了 URL，且可以在 Web 瀏覽器上參照 DATALINK 直欄中指定的檔案。

範例 2：指定將傳回 DATALINK URL 的 SQL 查詢的 Net.Data 巨集

```
 %FUNCTION(DTW_SQL) myQuery(){
  select name, DLURLCOMPLETE(picture) from myTable where name like '%river%'
%REPORT{
%ROW{
      <p> $(V1) <br>
      Before Encoding: $(V2) <br>
      After Encoding: @encodeDataLink($(V2)) <br>
      Make HREF: <a href="@encodeDataLink($(V2))"> click here </a> <br> <p>
    %}
  %}
%}
```

請注意，將使用「DataLink 檔案管理程式」函數。這個函數 dlurlcomplete 會傳回完整的 URL。

關聯式資料庫語言環境範例s

下列範例顯示您可以如何從巨集中呼叫關聯式資料庫語言環境：

ODBC

下列範例定義及呼叫 ODBC 語言環境的多個函數：

```
%DEFINE {
   DATABASE="qesq1"
   SHOWSQL="YES"
   table="int_null"
   LOGIN="netdata1"
   PASSWORD="ibmdb2"%}
%function(dtw_odbc) sql1() {
create table int_null (int1 int, int2 int)
%}
%function(dtw_odbc) sql2() {
insert into $(table) (int1) values (111)
%}
%function(dtw_odbc) sql3() {
insert into $(table) (int2) values (222)
%}
%function(dtw_odbc) sql4() {
select * from $(table)
%}
%function(dtw_odbc) sql5() {
drop table $(table)
%}
%HTML(REPORT){
@sql1()
@sql2()
@sql3()
@sql4()
%}
```

Oracle

下列範例顯示一個具有 DTW_ORA 函數定義的巨集，它會查詢 Oracle 資料庫 udatabase，並使用變數參照來決定將查詢的資料庫表格。 FUNCTION 區塊也含有可處理錯誤狀況的 MESSAGE 區塊。當 Net.Data 處理巨集時，它會在瀏覽器中顯示一個預設報表。

```
%DEFINE {
   LOGIN="ulogin"
   PASSWORD="upassword"
   DATABASE="udatabase"
   table= "utable"
%}
%FUNCTION(DTW_ORA) myQuery(){
select ename,job,empno,hiredate,sal,deptno from $(table) order by ename
%}
%MESSAGE{
100 : "<b>WARNING</b>: No employee were found that met your search criteria.<p>"
    : continue
%}
%HTML(REPORT){
@myQuery()
%}
```

SQL

下列範例顯示一個具有可呼叫 SQL 儲存程序的 DTW_SQL 函數定義的巨集。它具有三種不同資料類型的參數。 DTW_SQL 語言環境會將每一個參數傳遞到符合參數的資料類型的儲存程序。當儲存程序完成處理後，將傳回輸出參數，且 Net.Data 會相應地更新變數。

```
%{************************************************************
                        定義區塊
************************************************************%}
%DEFINE {
 MACRO_NAME     = "TEST ALL TYPES"
  DTW_HTML_TABLE      = "YES"
 Procedure      = "TESTTYPE"
 parm1          = "1"              %{SMALLINT      %}
 parm2          = "11"             %{INT         %}
 parm3          = "1.1"            %{DECIMAL (2,1)  %}
 %}
 %FUNCTION(DTW_SQL)   myProc
  (INOUT SMALLINT     parm1,
   INOUT INT            parm2,
   INOUT DECIMAL(2,1)    parm3){
CALL $(Procedure)
%}
%HTML(REPORT){
<HEAD>
<TITLE>Net.Data : SQL Stored Procedure: Example '$(MACRO_NAME)'. </TITLE>
</HEAD>
<BODY BGCOLOR="#BBFFFF" TEXT="#000000" LINK="#000000">
<p><p>
呼叫將建立儲存程序的函數。
<p><p>
 @CRTPROC()
<hr>
<h2>
在呼叫儲存程序前 INOUT 參數的值：<p>
</h2>
<b>parm1  (SMALLINT)</b><br>
$(parm1)<p>
<b>parm2  (INT)</b><br>
$(parm2)<p>
<b>parm3  (DECIMAL)</b><br>
$(parm3)<p>
<p>
<hr>
<h2>
呼叫執行儲存程序的函數。
</h2>
<p><p>
 @myProc(parm1,parm2,parm3)
<hr>
<h2>
在呼叫儲存程序後 INOUT 參數的值：<p>
</h2>
<b>parm1  (SMALLINT)</b><br>
$(parm1)<p>
<b>parm2  (INT)</b><br>
$(parm2)<p>
<b>parm3  (DECIMAL)</b><br>
$(parm3)<p>
  </body>
%}
```

Sybase

下列範例顯示一個具有 DTW_SYB 函數定義的巨集，它會查詢 Sybase 資料庫

udatabase，並使用變數參照來決定將查詢的資料庫表格。 FUNCTION 區塊也含有可處理錯誤狀況的 MESSAGE 區塊。當 Net.Data 處理巨集時，它會在瀏覽器中顯示一個預設報表。

```
%DEFINE {
   LOGIN="ulogin"
   PASSWORD="upassword"
   DATABASE="udatabase"
   table= "utable"
%}
%FUNCTION(DTW_SYB) myQuery(){
select ename,job,empno,hiredate,sal,deptno from $(table) order by ename
%}
%MESSAGE{
100 : "<b>WARNING</b>: No employee were found that met your search criteria.<p>"
    : continue
%}
%HTML(REPORT){
@myQuery()
%}
```

純本文檔介面語言環境

如果您選擇要使用純本文檔作爲您的資料來源，請使用純本文檔介面 (FFI) 及其相關的函數，來開啓、關閉、讀取、寫入及刪除 Web 伺服器上的檔案。檔案語言支援會使用 FFI 函數，透過瀏覽器應 Web 從屬站的要求，從 Web 伺服器上的檔案讀取資料，或是將資料寫入其中。FFI 會將檔案當作記錄檔來檢視，每一記錄等於 Net.Data 巨集表格變數中的一列，而記錄中的每一個值等於 Net.Data 巨集表格變數中的欄位值。 FFI 會將檔案中的記錄讀入 Net.Data 巨集表格的橫列中，以及將表格中的橫列寫入記錄中。

請參閱 *Net.Data 參考手冊*，取得 FFI 內建函數的說明及語法。

架構 FFI 語言環境

請驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_FILE)   DTWFILE   ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

呼叫 FFI 內建函數

如同呼叫其他函數一樣的方式來呼叫 FFI 函數。請使用 DEFINE 陳述式，將任何您想要傳遞的參數定義爲變數；例如：

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1500"
    myRows = "2"
 %}
```

然後，使用函數呼叫陳述式來呼叫函數；例如：

```
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

範例

在這個範例中，Net.Data 會將 ffi001.dat 檔的內容讀入 Net.Data 表格中，並將這個表格的內容寫入 tmp.dat 檔中。最後， Net.Data 會刪除 tmp.dat 檔。

```
%DEFINE {
mytable = %TABLE(ALL)
myfile  = "/usr/lpp/netdata/ffi//ffi001.dat"
tmpfile = "/usr/lpp/netdata/ffi/tmp.dat"
%}
%HTML(report){
@DTWF_READ(myfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)
@DTWF_WRITE(tmpfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)
@DTWF_REMOVE(tmpfile)
%}
```

Web 登記語言環境

Net.Data Web 登記會提供持續儲存體，來存放應用程式相關的資料。Web 登記可以用來儲存架構資訊及其他在執行時可被 Web 型應用程式動態存取的資料。您僅能使用 Net.Data 及 Web 登記內建支援，透過 Net.Data 巨集來存取 Web 登記，以及從爲了這個目的而撰寫的 CGI 程式存取它。Web 登記可在作業系統的子集中使用。請參閱 *Net.Data 參考手冊*，取得 Web 登記內建函數的說明及語法，以及支援語言環境的作業系統列示。

標準 Web 網頁開發需要 URL 直接置於網頁的 HTML 來源中。這將使得不易變更鏈結。靜態自然也會限制可輕易放置在 Web 網頁上的鏈結類型。使用 Web 登記來儲存應用程式相關的資料 (例如， URL)，可協助建立具有動態鏈結的 HTML 網頁。

有權寫入存取登記的應用程式軟體開發者及 Web 管理者可在登記中儲存及維護資訊。在執行時，應用程式會從它們的相關的登記中取回資訊。這可設計有彈性的應用程式，且同時容許移動應用程式及伺服器。您可以使用 Net.Data 巨集，透過動態設定的鏈結，來建立 HTML 頁面。

資訊會登記項目的格式儲存在 Web 登記中。每一個登記項目是由一對字串所構成：RegistryVariable 字串及對應的 RegistryData 字串。可用一對字串來代表的資訊將能夠儲存爲登記項目。Net.Data 會使用變數字串作爲搜尋關鍵字，從登記中找出及取回特定項目。

表10顯示一個樣本 Web 登記：

表 10. 樣本 Web 登記

CompanyName	**WorldConnect**
Server	ftp.einet.net
JohnDoe/foreground	Green
CompanyURL/IBM Corp.	http://www.ibm.com
CompanyURL/Sun Microsystems Corp.	http://www.sun.com
CompanyURL/Digital Equipment Corp.	http://www.dec.com
JaneDoe/Home_page	http://jane.info.net

考慮使用 Web 登記的理由：

- 您可以使用 Web 登記，來儲存伺服器及 URL 的別名，因而可重新找出應用程式及伺服器。
- 應用程式軟體開發者可以運送具有在登記中預先定義的資料 (如 URL) 的 Web 型應用程式。一般使用者可以修改登記資料來變更應用程式的行爲。
- Web 登記可以用來依據產品名稱、國家語言、製造商等，來執行 URL 搜尋。

「Web 登記」中的索引登錄是其 RegistryVariable 字串具有其他索引字串的登錄，而這個字串是使用下列語法附加到這些登錄上：

```
RegistryVariable/Index
```

使用者會在個別參數中提供索引字串的值，這個值將給與設計來與索引登錄一起使用的內建函數。多個索引登記項目可以具有相同的 RegistryVariable 字串值，但它們可以經由具有不同的索引字串值，來維護它們的唯一性。

表 11. 樣本索引 Web 登記

Smith/Company_URL	http://www.ibmlink.ibm.com
Smith/Home_page	http://www.advantis.com

即使上面兩個索引登錄具有相同的 RegistryVariable 字串值 Smith，但每一種情況中，索引字串是不同的。 Web 登記函數會將它們視爲兩個不同的登錄。

架構 Web 登記語言環境

請驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_WEBREG)   DTWWEB    ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

呼叫 Web 登記內建函數

如同呼叫其他函數一樣的方式來呼叫「Web 登記」函數。請使用 DEFINE 陳述式，將任何您想要傳遞的參數定義爲變數。例如：

```
%DEFINE {
    name = "smith"
 %}
```

然後，使用函數呼叫陳述式來呼叫函數；例如：

```
@DTWR_ADDENTRY("URLLIST", name, "http://www.software.ibm.com/",
    "WORK_URL"
```

範例

下列範例會建立 Web 登記及新增登錄。然後，它會顯示含有登錄的報表。

```
%DEFINE {
RegTable = %TABLE(ALL)
%}
  %MESSAGE {
    default:"<p>Function Error: Return code: $(RETURN_CODE)." :continue
%}
%FUNCTION(DTW_WEBREG) ListTable(INOUT RegTable) {
%}
%HTML(report){
```

```
@DTWR_CREATEREG("MYREG")
@DTWR_ADDENTRY("MYREG", "Dept. 1", "Payroll")
@DTWR_ADDENTRY("MYREG", "Dept. 2", "Technical Support")
@DTWR_ADDENTRY("MYREG", "Dept. 3", "Research")
@DTWR_LISTREG("MYREG", RegTable)
<p>Report:<br>
@ListTable(RegTable)
%}
```

IMS Web 語言環境

IMS Web 語言環境是完整最終解決方案的一部份，它會使用 Net.Data，在全球資訊網 (WWW) 環境中執行 IMS 異動。IMS Web 語言環境提供：

- 具有下列的 Net.Data 巨集：
  - 用來輸入異動輸入資料的 HTML
  - 呼叫 IMS Web 語言環境的 Net.Data FUNCTION 區塊
  - 顯示異動輸出的 HTML
- IMS Web 語言環境呼叫的異動 DLL 的原始檔

IMS Web Studio 工具會依據異動的「訊息格式服務 (MFS)」原始檔及 IMS Web Net.Data 應用程式的樣本 HTML 頁面來建立 DLL 及巨集的程式碼，以及建立一個 make 檔案來建置 DLL 可執行檔或共用程式庫。在建置了 DLL 的可執行格式後，使用者會將 DLL 及巨集移到執行 Net.Data 的 Web 伺服器。異動已備妥，可在 Web 環境中執行。

IMS Web 會使用「IMS TCP/IP 開放式異動管理程式存取 (OTMA) 連線」，在 Web 伺服器及 IMS 環境之間進行通信。

請參閱 IMS Web 首頁，取得如何使用 IMS Web 的詳細資訊。

```
http://www.software.ibm.com/data/ims/about/imsweb/document/
```

架構 IMS Web 語言環境

若要使用 IMS Web 語言環境，您需要驗證 Net.Data 起始設定的設定值，並設置語言環境。

驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上。

```
ENVIRONMENT (HWS_LE)       DTWHWS    ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

重要事項：請參閱第23頁的『設置 IMS Web 語言環境』，學習如何設置 IMS 語言環境

限制

當 Net.Data 以 CGI 應用程式形式執行時，才支援 Net.Data 的 IMS Web 語言環境。

程式設計語言環境

Net.Data 會提供下列語言環境，讓您在呼叫外部程式時使用。

- 『Java Applet 語言環境』
- 第139頁的『Java 應用程式語言環境』
- 第141頁的『Perl 語言環境』
- 第143頁的『REXX 語言環境』
- 第146頁的『系統語言環境』

存取權：確定 Net.Data 執行時所用的使用者 ID 具有執行程式的存取權力，包括程式可以存取的任何物件。有關的詳細資訊，請參閱第48頁的『授與 Net.Data 存取的檔案的存取權』。

Java Applet 語言環境

Java applet 語言環境可讓您在 Net.Data 應用程式中輕易地建立 Java applet 的 HTML 標籤。當您呼叫 Java applet 語言環境時，您將指定 applet 的名稱，以及傳遞 applet 需要的任何參數。語言環境會處理巨集，並建立 HTML applet 標籤，然後 Web 瀏覽器會使用它來執行 applet。

此外，Net.Data 會提供一組介面，applet 可使用它們來存取表格參數。這些介面位在類別 DTW_Applet.class 中。

下列段落將描述如何使用 Java applet 語言環境，來執行 Java applet。

架構 Java Applet 語言環境

請驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_APPLET)   DTWJAVA    ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

建立 Java Applet

在使用 Net.Data Java applet 語言環境之前，您需要決定您計劃使用哪些 applet，或您需要撰寫哪些 applet。請參閱 Java 文件，取得如何建立 applet 的詳細資訊。

建立 Applet 標籤

您可以用 Net.Data 函數呼叫來建立 applet 語言環境的呼叫。不需要宣告函數呼叫。函數呼叫的語法如下：

```
@DTWA_AppletName(parm1, parm2, ..., parmN)
```

- DTWA_ 識別 applet 語言環境的函數呼叫。
- AppletName 是將對其建立標籤的 applet 的名稱。
- parm1 到 parmN 是用來建立 PARAM 標籤的參數。

若要撰寫一個建立 applet 標籤的巨集：

1. 在巨集的 DEFINE 區段串定義 applet 需要的任何參數。這些參數包括任何 applet 標籤屬性、Net.Data 變數及需要作爲 applet 的輸入的 Net.Data 表格變數。例如：

```
%define{
DATABASE = "celdial"                    <=Net.Data 變數：資料庫的名稱
MyGraph.codebase = "/netdata-java/"     <=必要的 applet 屬性
MyGraph.height = "200"                  <=必要的 applet 屬性
MyGraph.width = "400"                   <=必要的 applet 屬性
MyTitle = "This is my Title"            <=Net.Data 變數：Web 網頁的名稱
MyTable = %TABLE(all)                   <=儲存查詢結果的表格
%}
```

2. 選用性：指定資料庫的查詢，來建立一個結果集合，作爲 applet 的輸入。 當您使用 applet 來建立圖表或表格時，這是很有用的。例如：

```
%FUNCTION(DTW_SQL) mySQL(OUT table){
select name, ages from ibmuser.guests
%}
```

3. 在 Net.Data 巨集中指定函數呼叫，來呼叫 Java applet 語言環境，以及呼叫 applet。函數呼叫會指定 applet 的名稱，以及指定您要傳遞給語言環境的參數。這些參數包括任何 Net.Data 變數，以及需要作爲 applet 的輸入的 Net.Data 表格或直欄參數。例如：

```
%HTML(report){                                          <=HTML 區塊的開頭
@mySQL(MyTable)                                         <=SQL 函數 mySQL 的
                                                          呼叫
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable) <=Applet 函數呼叫
%}
```

Applet 標籤屬性: 您可以在 Net.Data 巨集中的任意處指定 applet 標籤的屬性。Net.Data 會將具有套表 *AppletName.attribute* 的所有變數替代成作爲屬性的 applet 標籤。在 applet 標籤上定義屬性的語法如下：

```
%define AppletName.attribute = "value"
```

下列屬性對所有 applet 是必要的：

- *codebase:* applet 的位置，以 URL 來識別它。
- *height:* applet 的高度 (圖點)。
- *width:* applet 的寬度 (圖點)。

下列屬性是選用性的：

- *align:* 對齊 applet
- *alt:* 如果瀏覽器瞭解 APPLET 標籤，但無法執行 Java applet 時，應該顯示的任何文字
- *archive:* 含有類別及其他資源的保存檔
- *hspace:* applet 每一邊的圖點數
- *name:* applet 案例的名稱
- *object:* 含有 applet 的序列化呈現的檔案的名稱
- *vspace:* applet 上下的圖點數

例如，如果您的 applet 稱爲 MyGraph，則您可以定義如下的這些必要屬性：

```
%DEFINE{
MyGraph.codebase = "/netdata-java/"
MyGraph.height   = "200"
MyGraph.width    = "400"
%}
```

在 DEFINE 區段中不需要眞正的指定。您可以用 DTW_ASSIGN 函數來設定值。如果您未定義一個變數代表 *AppletName.code* 變數，則 Net.Data 將新增預設 *code* 參數到 applet 標籤。 *code* 參數的值爲 *AppletName.class*，其中 *AppletName* 是您的 applet 的名稱。

Applet 標籤參數: 您可以在函數呼叫中定義要傳遞給 Java applet 語言環境的參數列示。您可以傳遞包括下列的參數：

- Net.Data 變數 (包括 LIST 變數)
- Net.Data 表格
- Net.Data 表格的直欄

當您傳遞參數時，Net.Data 會在 HTML 輸出中，以您指定給參數的名稱及值，來建立 Java applet PARAM 標籤。您無法傳遞字串文字或函數呼叫的結果。

Net.Data 變數參數:

您可以使用 Net.Data 變數作爲參數。如果您在巨集的 DEFINE 區塊中定義一個變數，並傳遞 DTWA_*AppletName* 函數呼叫中的變數值，則 Net.Data 將建立一個與變數具有相同名稱及值的 PARAM 標籤。例如，給與下列巨集陳述式：

```
%define{
...
MyTitle = "This is my Title"
%}
%HTML(report){
@DTWA_MyGraph( MyTitle, ...)
%}
```

Net.Data 將產生下列 applet PARAM 標籤：

```
<param name = 'MyTitle' value = "This is my Title" >
```

Net.Data 表格參數:

每次呼叫 Java applet 語言環境時，Net.Data 會自動以 DTW_NUMBER_OF_TABLES 的名稱，建立 PARAM 標籤，並指定函數呼叫是否已傳遞任何表格變數。值是 Net.Data 在函數中使用的表格變數的數目。如果未在函數呼叫中指定任何表格變數，將建立下列標籤：

```
<param name = "DTW_NUMBER_OF_TABLES" value = "0" >
```

您可以傳遞一個或多個 Net.Data 表格變數，作爲函數呼叫上的參數。如果您在 DTWA_*AppletName* 函數呼叫上指定 Net.Data 表格變數，則 Net.Data 將建立下列 PARAM 標籤：

表格名稱參數標籤：

這個標籤將指定要傳遞的表格的名稱。標籤具有下列語法：

```
<param name = 'DTW_TABLE_i_NAME' value = "tname" >
```

其中 i 是依據基於函數呼叫次序的表格號碼，而 *tname* 則是表格的名稱。

橫列及直欄規格參數標籤：

您可建立 PARAM 標籤，來指定特殊表格的列數及欄數。這個標籤具有下列語法：

```
<param name = 'DTW_tname_NUMBER_OF_ROWS' value = "rows" >
<param name = 'DTW_tname_NUMBER_OF_COLUMNS' value = "cols" >
```

其中表格名稱是 *tname*、*rows* 是表格中的列數，而 *cols* 則是表格中的欄數。這一對標籤是針對函數呼叫中指定的每一個唯一表格而建立的。

直欄值參數標籤：

這個 PARAM 標籤會指定特定直欄的直欄名稱。這個標籤具有下列語法：

```
<param name = 'DTW_tname_COLUMN_NAME_j' value = "cname" >
```

其中表格名稱是 *tname*，*j* 是欄號，而 *cname* 是表格中直欄的名稱。

橫列值參數標籤：

這個 PARAM 標籤會指定特殊橫列及直欄中的值。這個標籤具有下列語法：

```
<param name = 'DTW_tname_cname_VALUE_k' value = "val" >
```

其中表格名稱是 *tname*，*cname* 是直欄名稱，*k* 是列號，而 *val* 的值符合對應的橫列及直欄中的值。

表格直欄參數: 您可以傳遞一個表格直欄，當作函數呼叫上的參數，來建立特定直欄的標籤。 Net.Data 僅會對指定的直欄建立對應的 applet 標籤。表格直欄參數會使用下列語法：

```
@DTWA_AppletName(DTW_COLUMN( x )Table)
```

其中 *x* 是表格中的直欄名稱或欄號。

表格直欄參數會使用針對表格參數所定義的同一個 applet 標籤。

位於無法使用 Java 的瀏覽器上的 Applet 標籤的替代文字: 變數 DTW_APPLET_ALTTEXT 指定將顯示在不支援 Java 或已關閉 Java 支援的瀏覽器上的文字。例如，下列變數定義：

```
%define DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not Java-enabled."
```

將產生下列 HTML 標籤及文字：

```
<P>Sorry, your browser is not Java-enabled.<BR>
```

如果未定義這個變數，將不會顯示任何替代文字。

Java Applet 範例

下列範例將描述一個 Net.Data 巨集，它可以呼叫 Java applet 語言環境，以及呼叫語言環境所建立的結果 applet 標籤。

Net.Data 巨集含有 Java applet 語言環境的下列函數呼叫：

```
%define{
DATABASE = "celdial"
DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not Java-enabled."
DTW_DEFAULT_REPORT = "no"
MyGraph.codebase = "/netdata-java/"
```

```
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "This is my Title"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
select name, ages from ibmuser.guests
%}
%HTML(report){
@mySQL(MyTable)
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable)
%}
```

DEFINE 區段中的 Net.Data 巨集行會指定 applet 標籤的屬性：

```
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
```

語言環境會以下列限定元來建立 applet 標籤：

```
<applet code = 'MyGraph.class' codebase = '/netdata-java/' width = '400 'height = '200'>
```

Net.Data 會從輸出表格 MyTable 中的 Net.Data 巨集的 SQL 區段中，傳回 SQL 查詢結果。這個表格是在 DEFINE 區段中指定：

```
MyTable = %TABLE(all)
```

巨集中 applet 的呼叫是在 HTML 區段中指定：

```
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable)
```

依據函數呼叫中的參數，Net.Data 會建立一個完整的 applet 標籤，它含有結果的相關資訊，如欄數、傳回的列數，以及結果列。 Net.Data 會對結果表格中的每一資料格建立一個參數標籤，範例如下：

```
param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
```

參數名稱 *DTW_MyTable_ages_VALUE_1* 會指定表格 MyTable 中的表格資料格 (第一列, 直欄經歷時間)，它具有值 4。 applet 的函數呼叫中的關鍵字 DTW_COLUMN 指定您僅對在此顯示的結果表格 MyTable 的直欄經歷時間有興趣：

```
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
```

下列輸出會顯示 Net.Data 針對範例建立的完整 applet 標籤：

```
<applet code = 'MyGraph.class' codebase = '/netdata-java/' width = '400' height = '200' >
<param name = 'MyTitle' value = "This is my Title" >
<param name = 'DTW_NUMBER_OF_TABLES' value = "1" >
<param name = 'DTW_TABLE_1_NAME' value = "MyTable" >
<param name = 'DTW_MyTable_NUMBER_OF_ROWS' value = "5" >
<param name = 'DTW_MyTable_NUMBER_OF_COLUMNS' value = "1" >
<param name = 'DTW_MyTable_COLUMN_NAME_1' value = "ages" >
<param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
<param name = 'DTW_MyTable_ages_VALUE_2' value = "32">
<param name = 'DTW_MyTable_ages_VALUE_3' value = "31" >
<param name = 'DTW_MyTable_ages_VALUE_4' value = "28" >
<param name = 'DTW_MyTable_ages_VALUE_5' value = "40" >
<P>Sorry, your browser is not Java-enabled.<BR>
</applet>
```

使用 Net.Data Java Applet 介面

Net.Data 會在名爲 DTW_Applet.class 的類別中提供一組介面，您可以使用它，搭配您的 Java applet，來協助處理針對表格變數而建立的 PARAM 標籤。您可以建立一個 applet，擴充這個介面，以便可從您的 applet 呼叫常式。

Net.Data 提供這些介面：

- **int GetNumberOfTables()** 傳回 applet 標籤中找到的表格數。
- **String [] GetTableNames()** 傳回 applet 標籤中找到的表格名稱列示。
- **int GetNumberOfColumns(String table_name)** 傳回表格 table_name 中的欄數。
- **int GetNumberOfRows(String table_name)** 傳回表格 table_name 中的列數。
- **String[] GetColumnNames(String table_name)** 傳回表格 table_name 中的欄數。
- **String[][] GetTable(String table_name)** 傳回一個二維陣列的字串，它含有表格橫列及直欄的值。

若要存取介面，請在您的 applet 程式碼中使用 EXTENDS 關鍵字，從 DTW_APPLET 類別取出 applet 的次類別，範例如下：

```
import java.io.*;
import java.applet.Applet;
public class myDriver extends DTW_Applet
    {
    public void init()
        {
        super.init();
        if (GetNumberOfTables() > 0)
            {
            String [] tables = GetTableNames();
            printTables(tables);
            }
        }
    private void printTables(String[] tables)
        {
        String table_name;
        for (int i = 0; i < tables.length; i++)
            {
            table_name = tables[i];
            printTable(table_name);
            }
        }
    private void printTable(String table_name)
        {
        int nrows = GetNumberOfRows(table_name);
        int ncols = GetNumberOfColumns(table_name);
        System.out.println("Table: " + table_name + " has " + ncols + " columns and
                           " + nrows + " rows.");
        String [] col_names = GetColumnNames(table_name);
        System.out.println("-----------------------------------------------------");
        for (int i = 0; i < ncols; i++)
            System.out.print("   " + col_names[i] + "   ");
        System.out.println("\n-----------------------------------------------------");
        String [][] mytable = GetTable(table_name);
        for (int j = 0; j < nrows; j++)
            {
            for (int i = 0; i < ncols; i++)
                System.out.print("   " + mytable[i][j] + "   ");
            System.out.println("\n");
            }
        }
    }
```

Java 應用程式語言環境

Net.Data 會支援具有 Java 語言環境的舊有 Java 應用程式。 透過 Java applet 及 Java 方法 (或應用程式) 的支援，您可以透過 Java Database Connectivity (JDBC**) API 來存取 DB2。

關於 JDBC 的詳細資訊可從這些網站取得；

- IBM 軟體具有 JDK 1.1 或更新版本，需有它方可透過 Net.Data 來使用 JDBC：
  `http://www.software.ibm.com/data/db2/java/`
- JavaSoft 具有其他的 JDBC 驅動常式、JDBC API 文件及最新的 JDBC 更新版：
  `http://splash.javasoft.com/jdbc/`

架構 Java 語言環境

若要使用 Java 語言環境，您需要驗證 Net.Data 起始設定的設定值，並設置語言環境。

請驗證下列架構陳述式是否在起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_JAVAPPS)   ( OUT RETURN_CODE ) CLIETTE "DTW_JAVAPPS"
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

重要事項：請參閱第23頁的『設置 Java 語言環境』，學習如何設置 Java 語言環境。

呼叫 Java 函數

Java 語言環境提供類似於「遠端程序呼叫 (RPC)」的介面。您可以從 Net.Data 巨集，以 Net.Data 字串作爲參數，來發出 Java 函數呼叫，且您呼叫的 Java 函數可以傳回字串。當您使用 Java 語言環境時，您必須使用 Net.Data「現場連線」 (請參閱第150頁的『管理連線』，取得「現場連線」的詳細資訊)。

若要呼叫 Java 函數：

1. 撰寫 Java 函數。
2. 對所有 Java 函數建立一個 Net.Data cliette (Net.Data cliette 會啓動 Java 執行所在的「Java 虛擬機器」)。
3. 在「現場連線」架構檔中的 Java ENVIRONMENT 陳述式上定義一個 cliette。
   每次您引進新的 Java 函數時，您必須重新建立 Java cliette。
4. 啓動「連線管理程式」。
5. 執行可呼叫 Java 語言環境的 Net.Data 巨集。

建立 Java 函數: 修改 Java 函數樣本檔案 `UserFunctions.java`，或以下列名爲 `myfile.java` 的範例檔爲模型，來建立新的檔案：

```
====================myfile.java====================
import mypackage.*                                  <=含有您的函數
 public String myfctcall(...parameters from macro...)
{
 return ( mypackage.mymethod(...parameters...));    <=您的函數的高階呼叫
 }
public String lowlevelcall(...parameters...)
{
 string result;
```

```
  .......code using many functions of your package...
  return(result)
}
```

Java 語言環境檔案結構: 在安裝 Net.Data 期間，Net.Data 會建立數個目錄。這些目錄包括透過 Java 語言環境建立 Java 函數、定義 cliette 及執行巨集時所需的檔案：

- 名爲 `UserFunctions.java` 樣本 Java 函數。
- 名爲 `makeClas` 的樣本檔案。執行時，這個檔案會對 Java 函數建立一個 Net.Data cliette 類別。
- 名爲 `launchjv` 的樣本檔案，Net.Data cliette 會使用它，來啓動「Java 虛擬機器」並執行您的 Java 函數。

表12描述您的作業系統上的檔案的目錄及檔名。

表 12. 建立 Java 函數時使用的檔案

作業系統	檔名	目錄
OS/2	UserFunctions.java	javaapps
	launchjv.com	connect
Windows NT	UserFunctions.java	javaclas
	makeClas.bat	javaclas
	launchjv.bat	connect
UNIX	UserFunctions.java	javaapps
	launchjv	javaapps

定義 Java 語言環境 Cliette: 修改樣本檔 `makeClas.bat`，或建立新的 .bat 檔，對所有 Java 函數建立名爲 `dtw_samp.class` 的 Net.Data cliette 類別。下列範例顯示批次檔 `CreateServer` 如何處理三個 Java 函數：

```
rem Batch file to create dtw_samp for Net.Data
java CreateServer dtw_samp.java UserFunctions.java myfile.java
javac dtw_samp.java
```

批次檔會處理下列檔案，以及 Net.Data 提供、名爲 `Stub.java` 的 stub 檔，來建立 `dtw_samp.class`。

- `dtw_samp.java`
- `UserFunctions.java`
- `myfile.java`

撰寫 JDBC 應用程式或 applet 非常類似於使用 DB2 CLI 或 ODBC 撰寫 C 應用程式，來存取資料庫。應用程式與 applet 之間的主要差異性在於應用程式可能需要特殊軟體，方可與 DB2 通信，例如，「DB2 從屬站應用程式啓用器」。 applet 依賴可使用 Java 的 Web 瀏覽器，且不需要在從屬站上安裝任何 DB2 程式碼。

在使用 JDBC 之前，您的系統需要某些架構。這些注意事項會在 「DB2 JDBC 應用程式及 Applet 支援」網站中加以討論：

`http://www.software.ibm.com/data/db2/jdbc/db2java.html`

Java 語言環境範例

在您建立了 Java 函數、定義了 cliette 類別，以及架構了 Net.Data 後，您可以執行含有 Java 函數參照的巨集。**重要事項：** 在呼叫 Net.Data 巨集之前，請先啓動「連線管理程式」。

在下列範例中，函數呼叫 *myfctcall* 會使用 cliette DTW_JAVAPPS 呼叫 Net.Data 提供的樣本函數。

```
%FUNCTION (DTW_JAVAPPS) myfctcall( ....parameters from macro ....)
%{ to call the sample provided with Net.Data %}
%FUNCTION (DTW_JAVAPPS) reverse_line(str);
%HTML(report){
you should see the string "Hello World" in reverse.
@reverse_line("Hello World")
You should have the result of your function call.
@myfctcall( ... ....)
%}
```

Perl 語言環境

Perl 語言環境可解譯 Net.Data 巨集的 FUNCTION 區塊中指定的列入 Perl script，或可以處理伺服器上個別檔案中所儲存的 Perl script。

架構 Perl 語言環境

請驗證下列架構陳述式是否在 Net.Data 起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_PERL)     DTWPERL   ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

呼叫外部 Perl Script

EXEC 陳述式會使用下列語法，在 FUNCTION 區塊中識別外部 Perl script 的呼叫：

```
%EXEC{ perl_script_name [optional parameters] %}
```

必要的：確定 *perl_script_name* (Perl script 名稱) 列示在針對 Net.Data 起始設定檔案中的 EXEC_PATH 架構變數所指定的路徑中。

```
%FUNCTION(DTW_PERL) rexx1() {
%EXEC{MyPerl.pl %}
%}
```

傳送參數

有兩種方式可將資訊傳遞給 Perl (DTW_PERL) 語言環境所呼叫的程式：直接與間接。

直接 將呼叫上的參數直接傳遞給 Perl script。例如：

```
%DEFINE INPARM1 = "SWITCH1"
%FUNCTION(DTW_PERL) sys1() {
  %EXEC{
    MyPerl.pl $(INPARM1) "literal string"
 %}
%}
```

Net.Data 變數 INPARM1 將被參照並傳遞到 Perl script。參數傳遞到 Perl script 的方式將同於從指令行中呼叫 Perl script 時參數傳遞到 Perl script 的方式。使用這種方法傳遞給 Perl script 的參數被視爲輸入類型參數 (Perl script 可使用及操作已傳遞到 Perl script 的參數，但對參數所做的變更不會向 Net.Data 反映)。

間接

使用下列其中一種方法，直接將呼叫上的參數傳遞到 Perl script。

- 使 Net.Data 將輸入參數傳遞到 Perl script，作爲環境變數。然後 Perl script 可以透過環境變數來取回參數。
- 使 Perl script 透過寫入到 Net.Data 將在環境變數 DTWPIPE 中傳遞其名稱的具名管線，將輸出參數傳回給語言環境。請使用下列語法，將資料寫入到具名管線：

  name="value"

  對於多個資料項目，請以換行或空白字元分隔每一個項目。

  如果變數名稱同於輸出參數名稱，且使用上述語法，則新值將置換現行值。如果變數名稱沒有對應到輸出參數，則 Net.Data 將不會處理它。

下列範例將顯示 Net.Data 如何從巨集傳遞變數。

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
  $date = 'date';
  chop $date;
  open(DTW, "> $ENV{DTWPIPE}") || die "Could not open: $!";
  print DTW "result = \"$date\"\n";
%}
%HTML(INPUT) {
  @today()
%}
```

如果 Perl script 是在名爲 `today.pl` 的外部檔中，則可在下一範例中撰寫相同的函數：

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
  %EXEC { today.pl %}
%}
```

您可以將 Net.Data 表格傳遞到 Perl 語言環境所呼叫的 Perl script。 Perl script 會依據它們的 Net.Data 名稱來存取 Net.Data 巨集表格參數的值。 直欄標題與欄位值包含在以表格名稱與直欄號碼識別的變數中。 例如，在 `myTable` 中，直欄標題爲 `myTable_N_j`，而欄位值爲 `myTable_V_i_j`，其中 *i* 是橫列號碼，*j* 爲直欄號碼 表格的橫列與直欄數目爲 `myTable_ROWS` 與 `myTable_COLS`。

FUNCTION 區段中的 REPORT 及 MESSAGE 區塊

在任何 FUNCTION 區段中均容許 REPORT 及 MESSAGE 區塊。它們是透過 Net.Data 來處理，而不是透過語言環境來處理。不過， Perl script 可將文字寫入到將併入爲 Web 網頁一部份的標準輸出串流。

Perl 語言環境範例

下列範例將顯示 Net.Data 如何經由執行外部 Perl script 來建立表格：

```
%define {
 c = %TABLE(20)
 rows = "5"
 columns = "5" %}
 %function(DTW_PERL) genTable(in rows, in columns, out table) {
 %exec{ perl.pl
 %}
 %message{
 default: "genTable: Unexpected Error"
 %}
 %}
%HTML(REPORT){
 @genTable(rows, columns, c)
 return code is $(RETURN_CODE)
 %}
 The Perl script (perl.pl):
 open(D2W,"> $ENV{DTWPIPE}");
 print "genTable begins ...
 ";
 $r = $ENV{ROWS};
 $c = $ENV{COLUMNS};
 print D2W "table_ROWS=\"$r\" ";
 print D2W "table_COLS=\"$c\" ";
 print "rows: $r
 ";
 print "columns: $c";
 for ($j=1; $j<=$c; $j++)
 {
 print D2W "table_N_$j=\"COL$j\" ";
 }
 for ($i=1; $i<=$r; $i++)
 {
 for ($j=1; $j<=$c; $j++)
 {
 print D2W "table_V_$i","_","$j=\"¦ $i $j ¿\" ";
 }
 }
 close(D2W);
```

結果：genTable 產生：

```
rows: 5 columns: 5
  COL1  |  COL2   |  COL3   |  COL4   |  COL5   |
-------------------------------------------------
[ 1 1 ] | [ 1 2 ] | [ 1 3 ] | [ 1 4 ] | [ 1 5 ] |
-------------------------------------------------
[ 2 1 ] | [ 2 2 ] | [ 2 3 ] | [ 2 4 ] | [ 2 5 ] |
-------------------------------------------------
[ 3 1 ] | [ 3 2 ] | [ 3 3 ] | [ 3 4 ] | [ 3 5 ] |
-------------------------------------------------
[ 4 1 ] | [ 4 2 ] | [ 4 3 ] | [ 4 4 ] | [ 4 5 ] |
-------------------------------------------------
[ 5 1 ] | [ 5 2 ] | [ 5 3 ] | [ 5 4 ] | [ 5 5 ] |
-------------------------------------------------
return code is 0
```

REXX 語言環境

REXX 語言環境可讓您執行 REXX 程式。

架構 REXX 語言環境

若要使用 REXX 語言環境，您需要確認 Net.Data 起始設定的設定值，並設置語言環境。

請確認下列架構陳述式是否在起始設定檔案中，且位在同一行上：

```
ENVIRONMENT (DTW_REXX)     DTWREXX   ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

執行 REXX 程式

透過 REXX 語言環境，您可以執行列入 REXX 程式或外部 REXX 程式。列入 REXX 程式是一種在巨集中具有 REXX 程式來源的 REXX 程式。外部 REXX 程式則在外部檔中具有 REXX 程式來源。

若要執行列入 REXX 程式：

定義一個函數，它須使用 REXX (DTW_REXX) 語言環境，且在函數的語言環境可執行區段中須含有 REXX 程式碼。

範例：含有列入 REXX 程式的函數

```
%function(DTW_REXX) helloWorld() {
        SAY 'Hello World'
%}
```

若要執行外部 REXX 程式：

定義一個函數，它須使用 REXX (DTW_REXX) 語言環境，且須包括將在 EXEC 陳述式中執行 REXX 程式的路徑。

範例：含有指向外部程式的 EXEC 陳述式的函數

```
%function(DTW_REXX) externalHelloWorld() {
%EXEC{ helloworld.exe%}
%}
```

必要的：確定 REXX 檔案名稱列示在針對 Net.Data 起始設定檔案中的 EXEC_PATH 架構變數所指定的路徑中。請參閱第18頁的『EXEC_PATH』，以學習如何定義 EXEC_PATH 架構變數。

傳送參數到 REXX 程式

有兩種方式可將資訊傳遞給 REXX (DTW_REXX) 語言環境所呼叫的 REXX 程式：直接與間接。

直接　使用 %EXEC 陳述式直接將參數傳遞給外部 REXX 程式。例如：

```
 %FUNCTION(DTW_REXX) rexx1() {
  %EXEC{
    CALL1.CMD $(INPARM) "literal string"   %}
%}
```

Net.Data 變數 INPARM1 會被解除參照，並傳遞到外部 REXX 程式。REXX 程式可以經由使用 REXX PARSE ARG 指令，來參照變數。使用這種方法傳遞給程式的參數被視爲輸入類型參數 (程式可使用及操作已傳遞到程式的參數，但對參數所做的變更不會向 Net.Data 反映)。

間接

經由 REXX 程式*變數儲存池*方式間接傳遞參數。當啓動 REXX 程式時，REXX 直譯器即會建立並維護含有關於所有變數的資訊的空間。這個空間稱爲變數儲存池。

當呼叫 REXX 語言環境 (DTW_REXX) 函數時，在執行 REXX 程式之前，REXX 語言環境會先將任何輸入 (IN) 或輸入/輸出 (INOUT) 函數儲存在變數儲存池中。 當呼叫 REXX 程式時，它便可以直接存取這些變數。 一旦順利完成 REXX 程式，DTW_REXX 語言環境便會判斷是否有任何 (OUT) 或 INOUT 函數參數。 若有，環境則會從變數儲存池中取回對應於函數參數的值，並以新值更新函數參數值。當 Net.Data 收到控制時，它會以從 REXX 語言環境中取得的新值，更新所有 OUT 或 INOUT 參數。 例如：

```
%DEFINE a = "3"
%DEFINE b = "0"
%FUNCTION(DTW_REXX) double_func(IN inp1, OUT outp1){
   outp1 = 2*inp1
%}

%HTML(REPORT){
Value of b is $(b), @double_func(a, b) Value of b is $(b)
%}
```

在上面範例中，呼叫 *@double_func* 傳遞兩個參數 *a* 與 *b*。 REXX 函數 *double_func* 會將第一個參數加倍，並將結果儲存在第二個參數中。當 Net.Data 呼叫巨集時，*b* 具有值 6。

您可以將 Net.Data 表格傳遞到 REXX 程式。REXX 程式會存取 Net.Data 巨集表格參數的值，作爲 REXX stem 變數。對 REXX 程式而言，直欄標題與欄位值包含在以表格名稱與直欄號碼識別的變數中。例如，在 myTable 中，直欄標題爲 myTable_N.*j*，而欄位值爲 myTable_N.*i*.*j*，其中 *i* 是橫列號碼，*j* 爲直欄號碼。表格中的橫列數目爲 myTable_*ROWS*，表格中的直欄數目則爲 myTable_*COLS*。

增進 AIX 作業系統的效能：

若您在 AIX 系統多次呼叫 REXX 語言環境，您可考慮將 RXQUEUE_OWNER_PID 環境變數設定爲 0。可多次呼叫 REXX 語言環境的巨集可以很容易地產生許多處理程序及大量的系統資源。

您可以將環境變數設定成下列三種方式的其中之一：

- 在巨集中，使用 DTW_SETENV 內建函數：

  ```
  @DTW_rSETENV("RXQUEUE_OWNER_PID", "0")
  ```

- 在 AIX 系統環境檔中，插入下列陳述式：

  ```
  /etc/environment: RXQUEUE_OWNER_PID = 0
  ```

  這個方法會影響 REXX 對整個機器的行爲。

- 在 HTTP Web 伺服器環境檔中；例如，對於 Domino Go Webserver，插入下列陳述式：

  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  這個方法會影響 REXX 對 Web 伺服器的行爲。

REXX 語言環境範例

下列範例顯示一個巨集，它會呼叫一個 REXX 函數，來建立一個含有兩欄三列的 Net.Data 表格。在 REXX 函數的呼叫之後是一個內建函數 DTW_TB_TABLE()，被呼叫來建立將傳回給瀏覽器的 HTML 套表。

```
%DEFINE myTable = %TABLE
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION(DTW_REXX) genTable(out out_table) {
  out_table_ROWS = 3
  out_table_COLS = 2

  /* 設定直欄標題 */
  do j=1 to out_table_COLS
    out_table_N.j = 'COL'j
  end

  /* 設定橫列中的欄位 */
  do i = 1 to out_table_ROWS
    do j = 1 to out_table_COLS
      out_table_V.i.j = '[' i j ']'
  end
  end
%}

%HTML(REPORT){
 @genTable(myTable)
 @DTW_TB_TABLE(myTable)
%}
```

結果：

```
  COL1      COL2
[ 1 1 ]   [ 1 2 ]
[ 2 1 ]   [ 2 2 ]
[ 3 1 ]   [ 3 2 ]
```

系統語言環境

系統語言環境支援執行指令及呼叫外部程式。

架構系統語言環境

將下列架構陳述式新增到起始設定檔案，須在同一行上：

```
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ( OUT RETURN_CODE )
```

請參閱 第20頁的『環境架構陳述式』，瞭解 Net.Data 起始設定檔案及語言環境 ENVIRONMENT 陳述式。

發出指令及呼叫程式

若要發出一個指令，請定義一個使用系統 (DTW_SYSTEM) 語言環境的函數，而這個語言環境須包括將在 EXEC 陳述式中發出的指令的路徑。例如：

```
%FUNCTION(DTW_SYSTEM) sys1() {
      %EXEC { ADDLIBLE.CMD %}
 %}
```

如果您使用 EXEC_PATH 架構變數來定義含有物件 (如指令與程式) 的目錄的路徑，則您可以縮短可執行物件的路徑。請參閱第18頁的『EXEC_PATH』，以學習如何定義 EXEC_PATH 架構變數。

範例 1：發出指令

```
%FUNCTION(DTW_SYSTEM) sys2() {
      %EXEC { MYPGM %}
 %}
```

範例 2：呼叫程式

```
%FUNCTION(DTW_SYSTEM) sys3() {
      %EXEC {MYPGM.EXE %}
 %
```

傳遞參數到程式

有兩種方式可將資訊傳遞給「系統」(DTW_REXX) 語言環境所呼叫的程式：直接與間接。

直接 在程式的呼叫上直接傳遞參數。例如：

```
%DEFINE INPARM1 = "SWITCH1"

%FUNCTION(DTW_SYSTEM) sys1() {
  %EXEC{
  CALL1.CMD  $(INPARM1) "literal string"
 %}
%}
```

Net.Data 變數 INPARM1 將被參照並傳遞到程式。參數傳遞到程式的方式將同於從指令行中呼叫程式時參數傳遞到程式的方式。使用這種方法傳遞給程式的參數被視爲輸入類型參數 (程式可使用及操作已傳遞到程式的參數，但對參數所做的變更不會向 Net.Data 反映)。

間接

「系統」語言環境無法直接傳遞或取回 Net.Data 變數，所以將以下列方式使它們可供程式使用：

- Net.Data 會傳遞輸入參數到程式，作爲環境變數。然後程式可以透過環境變數來取回參數。
- 程式將透過寫入到 Net.Data 將在環境變數 DTWPIPE 中傳遞其名稱的具名管線，將輸出參數傳回給語言環境。請使用下列語法，將資料寫入到具名管線：

  name="value"

  對於多個資料項目，請以換行或空白字元分隔每一個項目。

  如果變數名稱同於輸出參數名稱，且使用上述語法，則新值將置換現行值。如果變數名稱沒有對應到輸出參數，則 Net.Data 將不會處理它。

下列範例將顯示 Net.Data 如何從巨集傳遞變數。

```
%FUNCTION(DTW_SYSTEM) sys1 (IN P1, OUT P2, P3) {
  %EXEC {
     UPDPGM
  %}
%}
```

您可以將 Net.Data 表格傳遞到「系統」語言環境所呼叫的程式。 程式會依據它們的 Net.Data 名稱來存取 Net.Data 巨集表格參數的值。 直欄標題與欄位值包含在以表格名稱與直欄號碼識別的變數中。 例如，在 myTable 中，直欄標題爲 myTable_N_*j*，而欄位值爲 myTable_V_i_*j*，其中 *i* 是橫列號碼，*j* 爲直欄號碼 表格的橫列與直欄數目爲 myTable_ROWS 與 myTable_COLS。

不建議您傳遞具有多列的表格，因爲處理的環境變數的數目受到限制。

系統語言環境範例

下列範例顯示一個巨集，它有一個具有三個參數 (P1, P2 與 P3) 的函數定義。P1 爲輸入 (IN) 參數，而 P2 與 P3 則爲輸出 (OUT) 參數。函數呼叫程式 UPDPGM，以 P1 的值更新參數 P2，並將 P3 設定爲字串。在處理 %EXEC 區塊中的陳述式之前，DTW_SYSTEM 語言環境會將 P1 與對應值儲存在環境空間中。

```
%DEFINE {
   MYPARM2  = "ValueOfParm2"
   MYPARM3  = "ValueOfParm3"
 %}
 %FUNCTION(DTW_SYSTEM) sys1 (IN P1, OUT P2, P3) {
   %EXEC {
      UPDPGM
   %}
 %}

 %HTML(upd1) {
<P>
 正將資料傳遞到程式。MYPARM2 的現行值為 "$(MYPARM2)"，
 MYPARM3 的現行值為 "$(MYPARM3)"。現在，我們將呼叫 Web 巨集函數。

 @sys1("ValueOfParm1", MYPARM2, MYPARM3)

<P>
 在函數呼叫後，MYPARM2 的值為 "$(MYPARM2)"，
 MYPARM3 的值為 "$(MYPARM3)"。
 %}
```

第7章 提高執行效能

提高執行效能是調整系統的重要部份。本章將討論提高 Net.Data 執行效能的策略。討論的主題如下：

- 『使用 Web 伺服器 API』
- 『使用 FastCGI』
- 第150頁的『管理連線』
- 第153頁的『Net.Data 快取』
- 第171頁的『設定錯誤日誌層次』
- 第171頁的『將語言環境最佳化』

此外，請確定已正確地調整了您的 Web 伺服器。 Web 伺服器的執行效能對回應時間有直接的影響，與 Net.Data 處理巨集或直接要求的快慢無關。

使用 Web 伺服器 API

您可呼叫 Net.Data 與 Web 伺服器 API (而非 CGI) 來提高執行效能。當 Net.Data 使用 Web 伺服器 API 來執行時，Net.Data 會在 Web 伺服器的處理內以緒的方式執行。因爲 Web 伺服器的處理是多緒的，所有可在同一位址空間內同時處理多個 Net.Data 要求，因此可消除呼叫 Net.Data 作爲 CGI 處理的額外執行時間。

注意事項：Web 伺服器 API 的使用提供了提高的執行效能，而不需隔離應用程式。因爲 Net.Data 是在多緒環境中執行，所以使用者撰寫之語言環境內引進錯誤、不正確的呼叫，甚至資料庫停電都可讓 Web 伺服器發生問題，且可能讓它當機。決定是否要使用其中一個 Web 伺服器時，請決定您應用程式的最高優先順序是否爲執行效能或應用程式隔離。

使用 FastCGI

FastCGI 利用 CGI-BIN 的可靠性而提供了提高的執行效能。 FastCGI 的使用可讓您利用 API 伺服器的速度來執行巨集，且伺服器中含有使用個別記憶體空間的較可靠方法。Net.Data 呼叫預設是與 CGI 搭配執行。

您可在支援 FastCGI 的所有伺服器上搭配使用 Net.Data 與 FastCGI。

請參閱第33頁的『架構 Net.Data for FastCGI』，以學習如何架構 FastCGI。

您可將 FastCGI 調整成執行適當數目的處理，以處理含處理架構參數之進來要求的數目。例如，每個客戶平均每秒鐘有 100 個要求，而每個要求需要半秒鐘的時間來處理。則他們設會將此處理參數設定成 50。

下列語言環境支援 FastCGI：

- SQL
- Oracle 7.2, 7.3, 8.0
- ODBC
- Sybase

- REXX
- Perl

需求：在 FastCGI 模式中執行時， Oracle 與 Sybase 語言環境需要「現場連線」。

調整同時處理的數目：

1. 開啓定義處理之配置參數的架構檔。
   - 若是 Apache，則爲 `httpd.conf` 檔。
   - 若是 ICS 或 Lotus Domino Go Webserver，則爲 `lgw_fcgi.conf` 檔。
2. 變更用來指定處理數目的配置參數值：
   - Apache：`Process=`*num*。
   - ISC：`NumProcess=`*num*。

   其中 *num* 是處理數目。

管理連線

Net.Data 提供一個名爲「現場連線」的元件，來管理資料庫與 Java 虛擬機器連線。「現場連線」會維持持續的連線，來提高執行效能。有些 Net.Data 動作所需要的啓動時間很長。例如，處理必須先讓 DBMS 能認出自己，並連上資料庫後，才能發出資料庫查詢。這對要存取資料庫的 Net.Data 巨集來說，會花上很長的處理時間。另外，像 Java 虛擬機器的啓動費用也是很貴的，執行 Java 應用程式 (但非 Java applet) 時需要使用它。由於 CGI 程式的作業方式所致，每次您提出要求至 Web 伺服器時，光是這些程式的啓動，就得花掉您一些成本。Net.Data 在 OS/2、Windows NT 及 UNIX 作業系統上提供有「現場連線」，以維護持續性的連線。

下列各節說明的是「現場連線」。

- 『關於「現場連線」』
- 第151頁的『「現場連線」的優點』
- 第151頁的『我應該使用「現場連線」嗎？』
- 第152頁的『啓動連線管理程式』
- 第153頁的『Net.Data 與現場連線處理串流』

關於「現場連線」

「現場連線」幫您省下額外的啓動時間，以大大提高執行效能。省錢的原因是因爲會有一項或多項專門執行啓動函數的處理持續進行著。這些處理然後會等著來服務的要求。 如果您將 Net.Data 當成 CGI 或 FastCGI 程式使用，或當成 Web 伺服器 API plug-in 使用，您便可以執行「現場連線」。

「現場連線」是由「連線管理程式」與 cliette 組成的。 *Cliette* 是「連線管理程式」啓動的處理，當伺服器在執行時，會一直保持在作用中。Cliette 會處理資料，並利用關鍵字 CLIETTE，與起始設定檔案中所指定的 Net.Data 語言環境通信。 每種 cliette 類型都處理一種特定的語言環境函數，以 DB2 cliette 爲例，其會連接 DB2 資料庫，並且會設定一些作業，以便在 Net.Data 處理任何 Net.Data 巨集前，先執行 SQL 呼叫。 可

執行檔是命名在「現場連線」架構檔 dtwcm.cnf 中。 圖22會顯示「現場連線」、巨集與語言環境之間的交談。

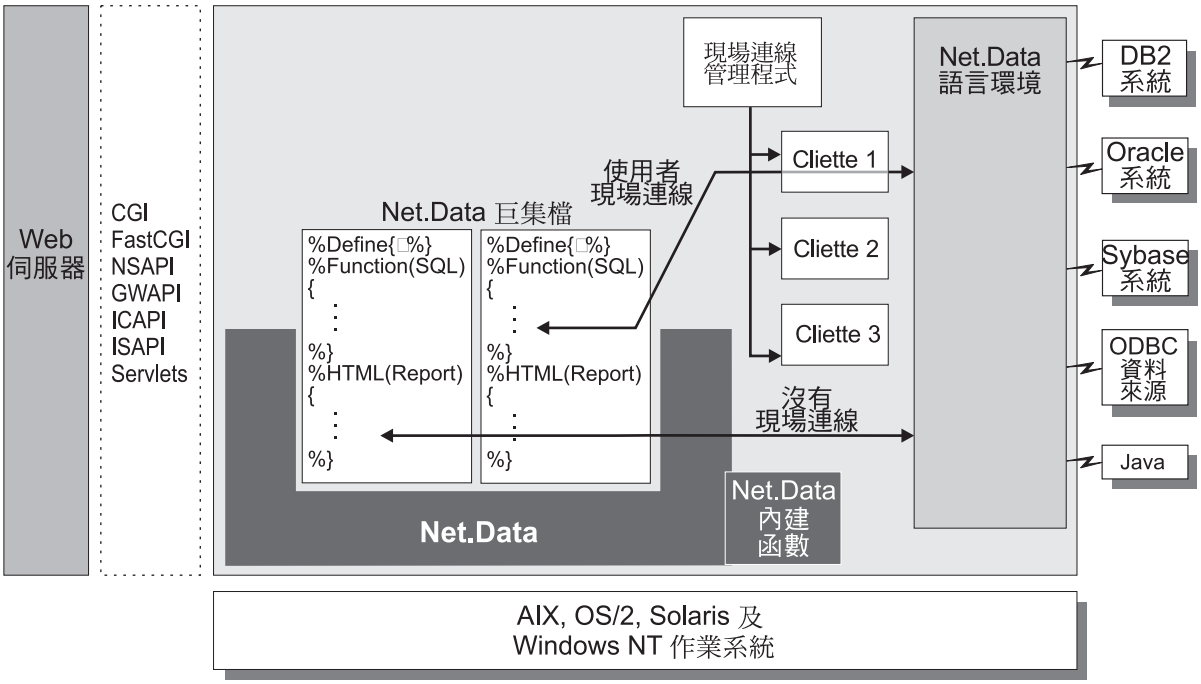


圖 22. 「現場連線」與 Cliette

下列各節會更詳細地說明「現場連線」。 若要學習如何架構「現場連線」，請參閱第27頁的『架構「現場連線」』。

「現場連線」的優點

使用「現場連線」的主要優點如下：

- **提高執行效能**

  重複使用連線乃比建立新連線來得有效率。一般而言，如果您要求小型的 SQL 陳述式 (例如，在小於 100000 列的資料庫上執行簡式查詢)，或如果資料庫連線十分困難 (例如，遠端伺服器)，則連接時間就很可觀。

- **多重資料庫存取**

  「現場連線」可在同一時間內，讓一個 Net.Data 巨集連接至多個資料庫。因爲每個資料庫都有一個獨一無二的 cliette，因此， Net.Data 就可與多個 cliette 進行通信。

我應該使用「現場連線」嗎？

您可以在 CGI、FastCGI 或 API 模式中使用「現場連線」，與您的資料庫進行通信。此外，如果您的應用程式需用到多個資料庫中的資料，則使用「現場連線」會較爲有利。

與 API plug-in 搭配使用的「現場連線」，可提高不少系統的執行效能，視這些系統的負荷與架構而定。您必須多次嘗試，才能決定出最適合您系統的架構。

有許多應用程式可在不使用「現場連線」的情況下，增進執行效能，亦即，它們可透過 ACTIVATE DATABASE 或 START DATABASE 指令，來節省建立資料庫連線的時間。有關您資料庫所用指令的詳細說明，請參閱您的資料庫文件。同時，請翻閱您的作業系統文件，以瞭解是否有其他步驟可幫助執行效能的提升。

需求：在 FastCGI 與 API 模式中執行時， Oracle 與 Sybase 語言環境需要「現場連線」。

啓動連線管理程式

「連線管理程式」是一個隨附於 Net.Data 的個別可執行檔，名爲 dtwcm。啓動 Web 伺服器時，就會啓動「連線管理程式」。

當您啓動「連線管理程式」時，它會讀取架構檔，並啓動一組處理。 在每個處理中，「連線管理程式」會開始執行特定的 cliette。若要學習如何架構「現場連線」，請參閱第27頁的『架構「現場連線」』。

若要透過 Windows NT 及 OS/2 來啓動「連線管理程式」：

1. 從指令行中，將目錄變更爲 <*inst_dir*>\connect\。
2. 輸入 dtwcm。

其中 <*inst_dir*> 是 Net.Data 的安裝目錄。

若要透過 AIX 啓動「連線管理程式」：

1. 從指令行中，將目錄變更爲 /usr/lpp/internet/db2www/db2/。
2. 輸入 dtwcm。

若要透過訊息選項啓動「連線管理程式」：

依據預設值，將抑制「連線管理程式」。如果想要顯示「連線管理程式」訊息，請在啓動「連線管理程式」時使用 -d 選項。

從指令行中，輸入：dtwcm -d

在使用了 -d 選項後，須重新啓動「連線管理程式」，方可重新抑制訊息。

若要自動啓動「連線管理程式」為 Windows NT 服務：

在 Windows NT 上，您可以設定令「連線管理程式」啓動爲 Windows NT 服務，而不需透過指令行。將「連線管理程式」當成 Windows NT 服務來執行，可在每次啓動機器時自動啓動「連線管理程式」。

要訣：在設定「連線管理程式」爲自動啓動之前，請先從指令行上啓動，來確保「現場連線」架構檔的正確性。

1. 從 Windows NT 工作列上，選取 **啓動->設定->控制台->服務程式**。
2. 選取 **Net.Data 連線管理程式**，然後按一下**啓動**按鈕。
3. 選取**自動啓動類型**，再按一下**確定**。

Net.Data 與現場連線處理串流

架構並啓動資料庫、Web 伺服器及「連線管理程式」後，則在啓用「現場連線」時，Net.Data 處理程序一般會包含下列步驟：

1. Web 伺服器會收到一個要求，並啓動 FastCGI、CGI 或 API 處理來執行 Net.Data。
2. Net.Data 會開始處理 Net.Data 巨集。
3. 當 Net.Data 遇到使用「現場連線」的函數呼叫時，其會從起始設定檔中判斷所需的 cliette 類型。如果是 DB2， cliette 類型通常是根據 DB2 資料庫名稱來命名，如 DTW_SQL:CELDIAL。
4. Net.Data 會向「連線管理程式」要求此類型的 cliette。
5. 「連線管理程式」會尋找那類型的可用 cliette。如果找不到，「連線管理程式」會將此項要求放進等待佇列中，等到有適當而可用的 cliette 類型時再行處理。
6. 有可用的 cliette 時，「連線管理程式」會告知 Net.Data 如何與 cliette 通信。
7. Net.Data 要求 cliette 去處理函數。
8. 從步驟 3 開始重複執行此項處理，直到 Net.Data 巨集處理完成。
9. 釋出所有的 cliette。

若起始設定檔中指定有 cliette，而「連線管理程式」不在執行狀態，則 Net.Data 會載入 DLL 並處理巨集。若您使用的是 API，則您可能會接收到錯誤，且應啓動「連線管理程式」。

Net.Data 快取

快取可協助您改進對應用程式使用者的回應時間。 Net.Data 將要求的結果儲存至區域環境的 Web 伺服器，以便快速擷取，直到復新資訊的時間到爲止。本章說明 Net.Data 快取概念、作業及限制。

- 『關於 Web 快取』
- 第154頁的『關於 Net.Data 快取』
- 第154頁的『Net.Data 快取術語』
- 第155頁的『Net.Data 快取概念』
- 第156頁的『Net.Data 快取限制』
- 第156頁的『Net.Data 快取介面』
- 第157頁的『規劃「快取管理程式」』
- 第157頁的『快取識別字』
- 第158頁的『架構「快取管理程式」和 Net.Data 快取』
- 第164頁的『啓動和停止「快取管理程式」』
- 第165頁的『快取網頁』
- 第167頁的『CACHEADM 指令』
- 第169頁的『快取日誌』

關於 Web 快取

許多軟體構成要素執行 Web 應用程式的快取。此處是快取應用程式的一些範例：

- Web 瀏覽器將網頁和相關的物件，例如影像、音效檔及 Java 小程式，儲存在區域環境的記憶體或磁碟上，以便在其使用者重複存取相同頁面時，能夠節省網路時間。
- Web 虛擬伺服器快取將網頁和相關的物件，儲存在接近使用者群組的區域伺服器上，以減少對遠端 Web 伺服器的網路存取時間；例如，減少 Web 伺服器取回所要求項目之次數。Web 虛擬伺服器快取也可有效共用多個使用者經常存取的網頁。
- Web 伺服器快取經常取回記憶體中的頁面和相關的物件，以便使用者重複取回相同頁面時，能夠節省磁碟存取時間。
- 資料庫管理系統可快取中的資料項目 (通常保留在磁碟上)，以便於重複取回相同資料項目時，能夠節省磁碟存取時間。

所有這些構成要素，皆獨立執行其快取功能，但整體結果是改進對使用者的回應時間。爲了決定復新快取項目的時間，Web 構成要素 (瀏覽器、虛擬伺服器及 Web 伺服器) 通常會考慮各種不同的選項，包括：

- 瀏覽器和伺服器架構選項
- 與網頁一起傳回的 HTTP 標題之內容，和來自 Web 伺服器的相關項目，特別是有效日期資訊。

關於 Net.Data 快取

對於經常存取的頁面及 Net.Data 巨集所產生的相關資料項目， Net.Data 本身提供自己的快取功能。 從 Net.Data 快取傳遞網頁，可讓您爲了建立網頁而執行 Net.Data 巨集與存取資料庫時，能夠節省所需要的時間。

您可以每一個伺服器使用一個「快取管理程式」。**建議值**：對 Net.Data 的多個案例使用一個「快取管理程式」，對每一個「快取管理程式」使用多重快取。

圖23 顯示 Net.Data 使用「快取管理程式」，來管理巨集檔的 HTML 輸出快取。此輸出可包含資料庫的資料。

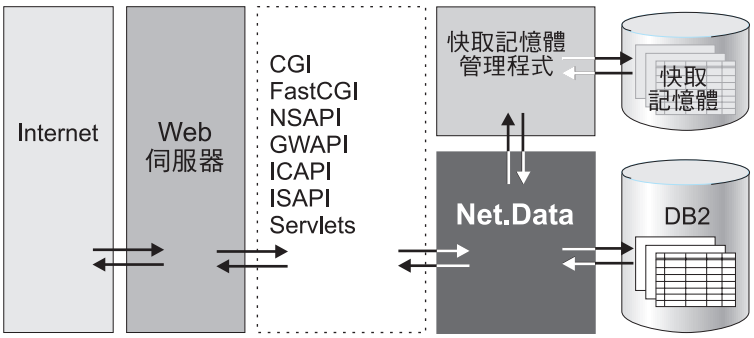


圖 23. Net.Data 快取

Net.Data 快取術語

Net.Data 文件使用下列詞彙來說明 Net.Data 快取。

快取 一種記憶體類型，包含最近存取的資料，是爲了加速後續存取相同資料而設計。快取通常是用來保留可透過網路存取之經常使用之資料的副本。在 Net.Data

中，是指含有 Net.Data 產生的 HTML 網頁 (供 Net.Data 巨集重覆使用) 之區域記憶體。若在快取中儲存網頁，則 Net.Data 就不必在快取中重新產生該資訊。每一個快取是由「快取管理程式」來管理，該程式負責多重快取，且可服務 Net.Data 的多重案例。

快取 ID

識別特定快取的字串。

快取管理程式

爲一台機器管理快取的程式。它可管理多個快取。

「快取管理程式」架構檔

含有設定值的檔案，Net.Data 使用這些設定值來決定記錄、追蹤、快取檔大小及其他選項的設定。它包含「快取管理程式」的設定，及特定「快取管理程式」所管理的所有快取檔。與 Net.Data 一起封裝時的檔名爲 `cachemgr.cnf`。

Net.Data 快取概念

根據您的系統上有多少個 HTTP 伺服器，及每一個 HTTP 伺服器是否執行其自己的 Net.Data 備份 (使用不同的 Net.Data 架構檔)，您可以把所有的 Net.Data 備份連結一個或多個「快取管理程式」。一個「快取管理程式」可以支援許多個快取，每一個快取有各自的識別字，稱爲*快取 ID*。 圖24 顯示一個「快取管理程式」使用多重巨集檔和管理兩個快取

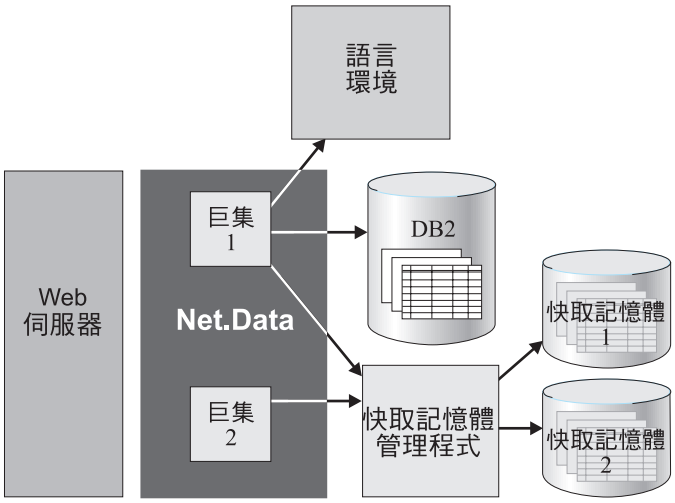


圖 24. 「快取管理程式」使用多重巨集檔和快取

快取中可放置項目 (稱爲*快取網頁*) 不限數量。每一個快取網頁有一個獨一無二的識別字，例如「一致資源定址器 (URL)」。一個網頁是一個區段或一個完整的 HTML 頁面。

當 Net.Data 收到快取資料的要求 (例如，來自內建函數 DTW_CACHE_PAGE 的要求) 時，將採取下列步驟：

1. Net.Data 連接到「快取管理程式」。
2. Net.Data 檢查是否已快取資料。
   - 如果資料已被快取且未到期，Net.Data 將向「快取管理程式」要求頁面，將它傳送到瀏覽器，並且停止執行巨集。

- 如果資料未被快取，則 Net.Data 會繼續處理巨集，然後將建立的 HTML 頁面傳送到 Web 瀏覽器，以及傳送到「快取管理程式」，將在此快取資料。

3. Net.Data 切斷與「快取管理程式」的連線。

當巨集檔順利完成處理程序時，「快取管理程式」會快取 HTML 輸出，來確定只快取順利產生的網頁。除非資料已傳送至瀏覽器，否則資料不會被快取且使用者看見的資料就是被快取的資料。

當 Net.Data 遭遇錯誤或預先從巨集跳出，「快取管理程式」的作法如下：

- 不接受不完整或損毀的頁面
- 在快取中保留舊有頁面

Net.Data 快取限制

Net.Data 快取有下列限制：

安全 「快取管理程式」不提供安全。例如，若某位資料庫使用者執行巨集，並快取一頁資料庫結果。另一位資料庫使用者可以取回該快取頁面。

直接要求

Net.Data 直接要求呼叫無法使用 Net.Data 快取。

Net.Data 快取介面

Net.Data 提供一組有彈性的介面，讓您用來架構和設置應用程式的快取。 表13 說明使用 Net.Data 快取特性的不同選項，以及在何處說明這些特性。

表 13. Net.Data 快取介面

介面	說明	跳至 ...
「快取管理程式」架構選項	您可以在「快取管理程式」架構檔的「快取管理程式」段落中，指定「快取管理程式」的許多選項，例如記錄和追蹤。	第158頁的『定義「快取管理程式」』
快取架構選項	在 Net.Data 「快取管理程式」的單一案例中，您可以定義許多快取，來保留被快取的項目。每一個快取有它自己的性質，例如大小、位置及快取 ID。這些性質定義於「快取管理程式」架構檔的快取段落中。每一個段落由快取 ID 來識別。	第160頁的『定義快取』
Net.Data 起始設定選項	若 Net.Data 和對應的「快取管理程式」分別在不同系統上執行，則您要在 Net.Data 起始設定檔中指定「快取管理程式」系統和埠號。	第12頁的『「快取管理程式」架構變數』

表 13. Net.Data 快取介面 (繼續)

介面	說明	跳至 ...
Net.Data 快取內建式函數	您可以使用 Net.Data 快取內建式函數，來操作 Net.Data 快取的內容。在適當的巨集函數中指定快取 ID，以選取具有最適當性質的快取。	請參閱 *Net.Data 參考手冊*的內建式函數這一章

規劃「快取管理程式」

規劃使用 Net.Data 快取函數時，您必須考慮：

- 哪些頁面會獲得快取的好處，以及所獲得的效能增進
- 快取項目的時間
- 復新快取項目的時間，及所用的復新方法

爲了使用 Net.Data 快取，您需要完成下列步驟，這需要先知道使用快取的方式。

建議值：在使用具有快取功能的主要應用程式之前，我們建議您先計畫和製作應用程式的原型，再付諸生產。

- 安裝 Net.Data，包括快取函數。
- 架構「快取管理程式」。請參閱 第158頁的『架構「快取管理程式」和 Net.Data 快取』。
- 決定如何將 Net.Data 應用程式付諸生產。
- 檢查各種 Net.Data 快取日誌，來決定是否改進快取的使用方式及其架構方式。

快取錯誤

在完成處理之前，若 Net.Data 遇到一個內部錯誤，使得它跳出巨集檔案，則「快取管理程式」不會快取網頁。「快取管理程式」不會快取不完整或含有 Net.Data 錯誤的頁面。這些類型的錯誤還包括巨集語法錯誤及 SQL 錯誤。

含有錯誤的頁面，共有在下列情形才會被快取：

- Net.Data 遇到一個錯誤，但因爲訊息區塊中的 CONTINUE 架構變數，所以 Net.Data 會繼續執行巨集檔案，且會正常地終止。
- 錯誤發生在 Net.Data 的錯誤判斷範圍之外，如資料庫回捲。

快取識別字

對您的應用程式設計快取時，您需要計畫兩種類型的識別字。

- **快取的識別字**： 此識別字是快取 ID，在定義快取的架構檔段落中指定名稱。您可以使用許多方式來分類和命名您的快取。例如，您可以根據應用程式來快取命名。您的每一個 Net.Data 應用程式可以有一個快取，並以其服務的 Net.Data 巨集所衍生的名稱，來爲每一個快取命名。
- **快取頁面的識別字**： 此識別字是快取頁面的 ID，指定所要快取之頁面的名稱。快取的頁面 ID 可以是任何字串，例如 URL 位址。您可使用 DTW_CACHE_PAGE() 內建函數來指定識別字。關於語法和範例，請參閱 *Net.Data 參考手冊*的內建式函數這一章。

架構「快取管理程式」和 Net.Data 快取

「快取管理程式」管理系統中的一個或多個快取。每一個快取包含動態產生的 HTML 頁面之內容。若要架構「快取管理程式」和每一個快取，您要更新「快取管理程式」架構檔 cachemgr.cnf 中的關鍵字值。

「快取管理程式」架構檔包含兩種段落：「快取管理程式」段落和「快取定義」段落。下列步驟說明如何爲您的應用程式自行設定這兩種段落。

定義「快取管理程式」

指定容許關鍵字的值，來定義「快取管理程式」段落。所有關鍵字是可選用的：除非不想接受預設值，否則不需要指定其值。

若要定義「快取管理程式」：

1. 設定「快取管理程式」日誌檔的名稱。該日誌顯示所有快取的全部異動活動，且供除錯和問題分析使用。

   預設值是將訊息寫至主控台。

   語法：

   ```
   log=path
   ```

   其中，*path* 是快取檔的路徑和檔案名稱。

   要訣：若要指定日誌檔給每一個快取，請在「快取定義」段落中使用 **tran-log** 關鍵字。

2. 指定「快取管理程式」爲了進入要求所用的 TCP/IP 通信協定埠號。這個埠號只用於聯絡「快取管理程式」和遠端機器。

   此值必須符合 Net.Data 起始設定檔中 DTW_CACHE_PORT 架構變數所指定之埠號。預設值的決定方式如下：

   a. 「快取管理程式」檢查路徑 */etc/services* 中，有關名稱 ibm-cachemgrd 的值。若找到此值，則「快取管理程式」使用該值。若找不到，則使用下一個方法。

   b. 「快取管理程式」用預設埠 7175。

   語法：

   ```
   port=port_number
   ```

   其中，*port_number* 是獨一無二的 TCP/IP 通信協定埠號。

3. 以秒爲單位，指定「快取管理程式」容許擱置中之讀取作業保持作用中的最大時間。若超過這個時間，則「快取管理程式」會中斷連線。

   預設值是 30 秒鐘。

   語法：

   ```
   connection-timeout=seconds
   ```

   其中，*seconds* 是擱置中之讀取作業保持作用中的時間長度，以秒數爲單位。

4. 設定是否記錄訊息。

   預設值是 **no** 或 **off**。

   語法：

   ```
   logging=yes|on|no|off
   ```

其中：

yes|on
表示記錄是必要的

no|off 表示不要執行記錄。

5. 設定是否折返日誌。

   預設值是 **no**。若指定爲 **yes**，則當檔案大小達到最大值時 (請參閱下列 **log-size**)、檔案類型爲 .old 時、及新日誌開啓時，現行日誌會關閉。只保留一份日誌檔 (舊有 .old 檔案會被改寫)。

   語法：

   ```
   wrap-log=yes|no
   ```

   其中：

   yes 指定要折返日誌。

   no 指定不折返日誌。

6. 以位元組爲單位，設定容許日誌擴大到最大之值 (若已指定折返日誌的話)。

   預設值是 64000。

   語法：

   ```
   log-size=bytes
   ```

   其中，*bytes* 是最大大小的位元組數。

7. 設定要寫入日誌中的訊息層次。若這些值出現在 *trace_flag_definitions* 列示中且無任何設定，則會設定爲 on。

   預設值是僅記錄「快取管理程式」啓動和關機訊息。

   語法：

   ```
   trace-flags=trace_flag_definitions
   ```

   其中：

   D_ALL
   啓用所有追蹤旗號。

   D_NONE
   停用所有追蹤旗號

範例：指定所有追蹤旗號皆啓用：

```
trace=flags=D_ALL
```

段落範例：有效的「架構管理程式」段落：

```
cache-manager {
log=/u/cached/logs/cached.log
port=7177
connection-timeout=0
logging=yes
wrap-log=yes
log-size=64000
trace-flags=D_ALL
}
```

定義快取

指定容許關鍵字的值，來定義「快取定義」段落。大多數關鍵字是可選用的，且不需指定，除非您不想使用預設值。

若要定義快取：

1. 指定用來保留快取頁面的路徑和目錄名稱。在啓動時，含有此目錄的檔案系統大小，至少必須爲 **fssize** 的值 (如下所示)；否則快取不會啓動。此值可指定爲絕對路徑名稱，或是對應於「快取管理程式」啓動路徑的相對路徑名稱。

   必要的。

   語法：

   ```
   root=path_name
   ```

   其中：

   path_name
   儲存快取頁面的絕對或相對之路徑和目錄名稱。

2. 設定啓動「快取管理程式」時，現行快取是否作用中。

   不必要的；預設值是 **yes**。 若設定爲 **no**，則快取定義於「快取管理程式」中，但不啓動。 您可以稍後使用 **cacheadm** 指令來啓動它。

   語法：

   ```
   caching=yes|no
   ```

   其中：

   yes 表示啓動「快取管理程式」時，亦啓用快取。

   no 表示啓動「快取管理程式」時，不要啓動快取。

3. 設定檔案系統中，由現行快取中的頁面所使用之最大空間。超過最大空間值時，「快取管理程式」會刪除足夠的頁數，從最舊的頁面開始刪除，以將快取所佔據的全部空間降至限制內。您可以將此值設爲最大，以有效停用自動除去登錄；但是，若超過實體檔案空間，則新增新的頁面至快取中就會失敗。

   不必要的；預設值是 0 (沒有對磁碟進行快取)。

   語法：

   ```
   fssize=nnB|nnKB|nnM
   ```

   其中：

   ***nn*B** 是位元組數；例如 5000B。

   ***nn*KB** 是千位元組數；例如 640KB。

   ***nn*MB** 是百萬位元組數；例如 30MB。

4. 設定此快取中之全部頁面所用的最大記憶體數量。超過最大記憶體數量時，「快取管理程式」會刪除足夠的頁數，從最舊的頁面開始刪除，使快取所佔用的全部記憶體在界限內。您可以設定頁數爲最大數，以便有效停用自動除去頁數；但是，若 **cachemgrd** 處理消耗太多記憶體，則作業系統可能會終止它。

   不必要的；預設值是 1MB。

   語法：

   ```
   mem-size=nnB|nnKB|nnMB
   ```

其中：

*nn***B** 是位元組數；例如 5000B。

*nn***KB** 是千位元組數；例如 640KB。

*nn***MB** 是百萬位元組數；例如 30MB。

5. 設定頁面在快取中保留的最大時間長度。超過此值時，「快取管理程式」會將頁面標示爲過期，但不會刪除該頁面，除非到達了 **fssize** (若是在磁碟上快取) 或 **memsize** (若是在記憶體中快取) 限制。若到達了 **memsize** 或 **fssize** 限制，則「快取管理程式」會在所有其他頁面之前，先刪除標示爲過期的頁面。您可以使用 **check_expiration** 關鍵字，來停用 **lifetime** 檢查。

   必要的： 否；預設值是 5 分鐘。

   語法：

   ```
   lifetime=time_length
   ```

   其中：

   *nn***S** 是指秒數；例如 600S。

   *nn***M** 是指分鐘數；例如 20M。

   *nn***H** 是指小時數；例如 30H。

6. 設定是否將快取頁面標示爲過期，及執行生命週期檢查。

   不必要的；預設值是 **yes**，預設生命週期長度是 60 秒鐘。此值也可設爲時間長度，表示 **yes** 值，並宣告項目保留在快取中的最大時間長度。設定爲 **no** 時，快取頁面就不會標示爲過期，且不會執行生命週期檢查。

   語法：

   ```
   check-expiration=yes|nnS|nnM|nnH|no
   ```

   其中：

   yes 表示「快取管理程式」執行生命週期檢查，且快取頁面標示爲過期。

   *nn***S** 是指秒數；例如 600S。

   *nn***M** 是指分鐘數；例如 20M。

   *nn***H** 是指小時數；例如 30H。

   no 表示「快取管理程式」不執行生命週期檢查，且快取頁面不標示爲過期。

7. 設定快取頁面在快取內所佔用的最大空間數量。若一頁面超過記憶體大小，則檢查檔案快取。若有足夠的空間，則「快取管理程式」會在檔案快取中儲存快取頁面。若頁面不適合於檔案快取，則快取嘗試會失敗。若頁面小於 datum_memory_limit 值 (cacheobj-memory-limit)，但如果快取無足夠的空間，則從記憶體快取中刪除最舊的快取頁面，以容納新建頁面。

   不必要的；預設值是 1KB。

   語法：

   ```
   datum-memory-limit (cacheobj-memory-limit)=nnB|nnKB|nnMB
   ```

   其中：

   *nn***B** 是位元組數；例如 5000B。

   *nn***KB** 是千位元組數；例如 640KB。

*nn***MB** 是百萬位元組數；例如 30MB。

8. 設定快取頁面在檔案快取內所佔用的最大空間數量。 若頁面小於 datum_memory_limit 值，但檔案快取中無足夠的空間，則從檔案快取中刪除最舊的快取頁面，以容納新建頁面。

   不必要的；預設值是 1KB。

   語法：

   ```
   datum-disk-limit (cacheobj-space-limit)=nnB|nnKB|nnMB
   ```

   其中：

   *nn***B** 是位元組數；例如 5000B。

   *nn***KB** 是千位元組數；例如 640KB。

   *nn***MB** 是百萬位元組數；例如 30MB。

9. 設定建立統計值記錄之間的時間。若設爲 0，則不會寫入統計值記錄。

   不必要的；預設值是 0 (沒有統計值)。

   語法：

   ```
   stat-interval = nnS|nnM|nnH
   ```

   其中：

   *nn***S** 是指秒數；例如 600S。

   *nn***M** 是指分鐘數；例如 1M。

   *nn***H** 是指小時數；例如 3H。

10. 設定記錄現行快取之統計值的路徑和檔案名稱。

    stat-interval 的值大於 0 時，則爲必要的。

    語法：

    ```
    stat-files=filename
    ```

    其中，*filename* 是記錄統計值檔案的路徑和名稱。

11. 設定統計值計數器每次被寫入日誌檔時，是否應該重設 0。

    不必要的；預設值是 **yes**。

    語法：

    ```
    reset-stat-counters=yes|no
    ```

    其中：

    yes 重設統計值計數器。

    no 不要重設統計值計數器。

12. 定義路徑和檔案名稱，來保留每一個快取的異動日誌。快取異動日誌檔不同於「快取管理程式」日誌檔，後者用來記錄「快取管理程式」的整體活動。

    必要的；若未指定，則不會建立快取的異動日誌。

    語法：

    ```
    tran-log=filename
    ```

    其中，*filename* 是每一個快取的異動之路徑和檔案名稱。

13. 設定首次啓動「快取管理程式」時，是否要開啓異動記錄，來記錄快取。除非透過 tran-log 參數來指定有效的異動日誌檔，否則系統不處理此參數。若「快取管理程式」架構檔中已指定有效的 tran-log 值，則可以使用 **cacheadm** 指令，以便於執行「快取管理程式」常駐程式的同時，亦啓動異動記錄。

    不必要的；預設值是 **no**。

    語法：

    ```
    tran-logging=yes|on|no|off
    ```

    其中：

    yes|on
    表示記錄是必要的。

    no|off 表示不要執行記錄。

14. 設定是否要折返異動日誌。

    不必要的；預設值是 **yes**。 若指定爲 **yes**，則當其大小達到最大值時 (請參閱 **tran-log-size**)、檔案類型爲 .old 時、及新日誌開啓時，現行日誌會關閉。 只保留一份日誌 (舊有 .old 檔案會被改寫)。

    語法：

    ```
    wrap-tran-log=yes|no
    ```

    其中：

    yes 表示要折返日誌。

    no 表示不要折返日誌。

15. 以位元組爲單位，設定容許異動日誌擴大到最大之值 (若已指定 **wrap-tran-log** 的話)。

    不必要的；預設值是 64000。

    語法：

    ```
    tran-log-size=bytes
    ```

    其中，*bytes* 是最大大小的位元組數。

段落範例：快取的有效「快取定義」段落：

```
test1
{
caching=on
fssize=5MB
mem-size=10MB
lifetime=6000000
check-expiration=150
datum-memory-limit=5KB
datum-disk-limit=500KB
stat-interval=60
reset-stat-counters=no
root=/u/cached/chaches/cache0
stat-files=/u/cached/logs/chache0.stats
tran-log=/u/cached/logs/cache0.log
tran-logging=yes
wrap-tran-log=yes
tran-log-size=100k
}
```

啓動和停止「快取管理程式」

下列段落說明如何啓動和停止「快取管理程式」。

- 『啓動「快取管理程式」』
- 『停止「快取管理程式」』

啓動「快取管理程式」

請使用 cachemgrd 指令，來啓動「快取管理程式」常駐程式。

語法：

參數：

cachemgrd
指令關鍵字。

config_file
指定檔案名稱，該檔案定義「快取管理程式」其及管理的每一個快取。隨著 Net.Data 產品一起發行的架構檔是 cachemgr.cnf。

範例：

```
cachemgrd -c myconfig.cfg
```

停止「快取管理程式」

請使用 cacheadm 指令來停止「快取管理程式」。

語法：

參數：

cacheadm
指令關鍵字。

hostname
指定執行快取的機器名稱，該機器與發出 cacheadm 指令的機器不同。

port_num
指定快取埠號，如果該號碼不同於預設值 (7175)。

terminate
指定要停止「快取管理程式」。

範例：

```
cacheadm hostname host1 port 7178 terminate
```

快取網頁

您可以使用 DTW_CACHE_PAGE 內建函數來快取網頁。當 Net.Data 在巨集檔中找到 DTW_CACHE_PAGE 函數時，它會聯絡「快取管理程式」，並開始將巨集檔的 HTML 輸出保存在記憶體中。在 Net.Data 順利處理巨集之後，HTML 輸出會傳送至瀏覽器，且「快取管理程式」會在一個異動中快取輸出，如 圖25 所示

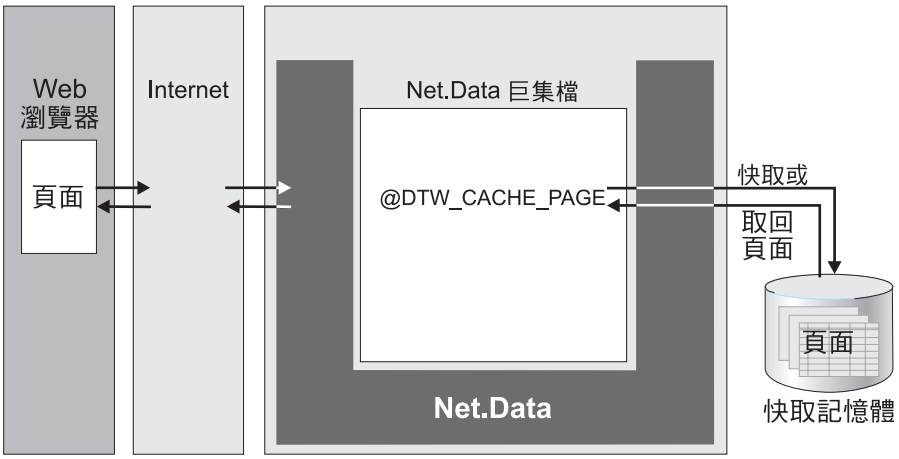


圖 25. DTW_CACHE_PAGE 函數起始快取

快取頁面

設定使用 Net.Data DTW_CACHE_PAGE() 內建函數，將 Net.Data 產生的頁面寫入快取中。

DTW_CACHE_PAGE() 一旦發覺頁面不存在快取中或已過期，則會快取函數陳述式後面之巨集檔的所有輸出。若頁面不存在快取中，或比指定的經歷時間還舊，則 Net.Data 會將輸出頁面傳回瀏覽器、從巨集執行產生新建輸出頁面、以及在快取中儲存頁面。

若「快取管理程式」找到快取頁面，且頁面仍爲現行，則會顯示快取內容，且 Net.Data 會跳出巨集。這個行爲確定從快取中取回網頁之後，不會再有任何不必要的處理程序。

執行效能要訣： 將 DTW_CACHE_PAGE() 置在首位，或置爲巨集檔中的前幾個陳述式之一，以將巨集檔的執行成本縮至最小。

若要快取頁面：

1. 在巨集檔的 HTML 區塊中，請於撰寫 HTML 程式碼之前，先插入下列函數陳述式：

   ```
   @DTW_CACHE_PAGE("cache_id", cached_page_id, "age", status)
   ```

   使用此函數來設定 Net.Data 要快取此陳述式之後的巨集的所有 HTML 輸出。若您想快取所有 HTML 輸出，請將此陳述式置於巨集檔較前面的位置。

   參數：

cache_id

用來識別將放置頁面之快取的字串。您可以連結快取 ID 和巨集 (或巨集群組)。

cached_page_id

含有識別字的變數，用來識別後續 @DTW_CACHE_PAGE 快取要求中快取內的頁面，例如頁面的 URL。

age 含有時間長度的字串變數 (以秒計算)，指定頁面過期的時間。若所要求的頁面存在快取中之時間，超過 *age* 的值，則 Net.Data 會執行巨集、重新產生頁面、快取所產生的頁面、以及置換過期的頁面。若所要求的頁面存在快取中的時間，小於或等於 age 的值，則 Net.Data 會從快取中取回該頁面，並傳送至瀏覽器。在這種情況中， Net.Data 會立即結束巨集的執行。

status Net.Data 傳回的字串變數，表示是否順利快取頁面。

範例：

```
%HTML(cache_example) {
 %IF (customer == "Joe Smith")
 @DTW_CACHE_PAGE("mymacro.d2w", "http://www.mypage.org", "-1", status)
%ENDIF
 ...
<html>

<head>
 <:title>This is the page title</title>
</head>

<body>
  <center>
 <h3>This is the Main Heading</h3>
 <p>It is $(time). Have a nice day!
  </body>

</html>
 %}
```

進階快取：動態決定是否快取

DTW_CACHE_PAGE() 函數從其位於巨集檔中的位置開始快取。一般而言，您可將該函數放在巨集檔的開頭，以得到較佳的執行效能，並確保所有 HTML 輸出皆被快取。

對於進階快取的應用，若您需要在處理期間於某一個點決定快取，而不是從巨集檔的開頭來快取，則可以將 DTW_CACHE_PAGE() 函數放在 HTML 輸出區段中。例如，您可能需要根據從查詢或函數呼叫所傳回的列數來做決定。

範例：因爲快取的決策是根據 HTML 輸出的預期大小而定，所以將函數放在 HTML 區塊中。

```
% DEFINE { ...%}

...

%FUNCTION(DTW_SQL) count_rows(){
  從客戶選取 count(*)
%REPORT{
%ROW{
  @DTW_ASSIGN(ALL_ROWS, V1)
%}
%}
```

```
%}

%FUNCTION(DTW_SQL) all_customers(){
  select * from customer
%}

%HTML (OUTPUT) {
<html>
<head>
 <title>這是客戶列示
</head>
<body>

@count_rows()

 %IF ($(ALL_ROWS) > "100")
 @DTW_CACHE_PAGE("mymacro.d2w", "http://www.mypage.org", "-1", status)
%ENDIF

@all_customers()

  </body>
</html>
%}
```

在此範例中，頁面是基於 HTML 輸出的預期大小來快取或取回的。只有在資料庫表格包含 100 列以上時，HTML 輸出頁面的快取才有價值。 Net.Data 在執行巨集之後都會將 OUTPUT 區塊中的本文 `This is the customer list`，傳送至瀏覽器；本文從不快取。在函數呼叫 `@count_rows()` 後面的幾行，會於 IF 區塊的條件滿足時快取或取回。兩個部份合起來形成一個完整的 Net.Data 輸出頁面。

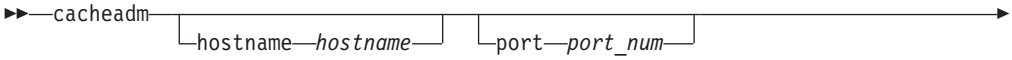
CACHEADM 指令

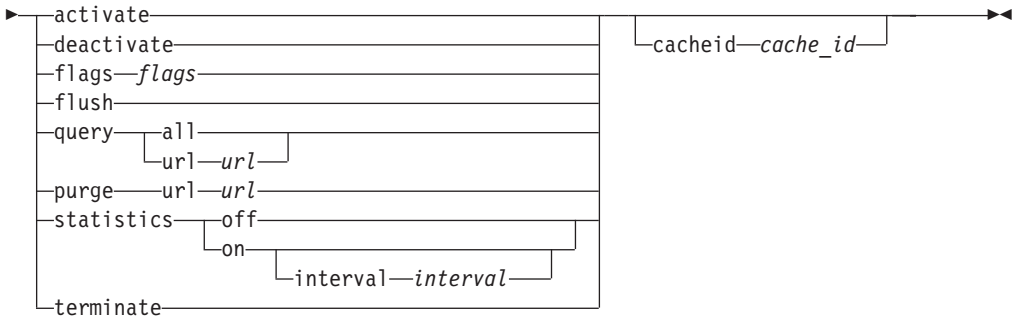
針對下列作業來使用 CACHEADM 指令：

- 停止「快取管理程式」
- 沖寫特定的快取
- 查詢特定的快取
- 啓用或停用記錄
- 記錄旗號
- 啓動和停止統計值聚集

所有參數皆可以縮寫成獨一無二的最小字元集。

語法：





參數：

activate
啓動指定的快取。若快取已在作用中，則「快取管理程式」不做任何處理。

cache_id
字串變數，識別要放置頁面的快取。例如：cache1。

deactivate
停用指定的快取。 若快取已非作用中，則「快取管理程式」不做任何處理。 所有擱置中作業皆會完成，且不接收任何新的作業。完成最後作業時，「快取管理程式」會將快取標示爲非作用中。

flags
指定列示的旗號是否應該開或關。

D_ALL
開啓所有追蹤旗號。

D_NONE
關閉所有追蹤旗號。

flush
沖寫由 *cache_id* 參數指定的快取，這是此參數所要求的。這個參數會無條件地從指定快取中刪除所有項目。

hostname
指定執行快取的機器名稱，該機器與發出 cacheadm 指令的機器不同。例如：myhost。

port_num
指定快取埠號，如果該號碼不同於預設值 (7175)。這個號碼必須是系統內獨一無二的。

purge
指定從快取中去除特定頁面。若已指定 *url*，則「快取管理程式」會除去關鍵字符合 *url* 的頁面。若已指定相依關係，則「快取管理程式」除去具相依關係的所有項目，並將其關鍵值寫入標準的輸出裝置 stdout。

query
根據指定的參數，傳回快取資料：

- 若僅指定快取 ID，則傳回關於快取的資訊。
- 若指定了 *url*，則傳回關於特定快取頁面的資訊。

- 若指定了 all，則傳回關於所有頁面的資訊。

  其它程式使用 all 選項來格式化或解譯結果。每一行包含下列資訊：

  – 頁面關鍵值
  – 頁面經歷時間
  – 頁面長度
  – 頁面建檔日期
  – 頁面到期日
  – 頁面上次被參考到的日期

  所有日期皆爲標準 UNIX 整數時間格式。

  執行效能要訣： 快取查詢全部選項會影響執行效能，應該謹慎使用。

statistics
啓用或停用記錄特定快取的統計值收集，需要 *cache_id* 參數。 若將 statistics 參數設定爲 on 來指定一個間隔，則 Net.Data 會設定或重設更新之間的間隔爲指定的秒數。

terminate
指定要停止「快取管理程式」。

tranlogging
啓用或停用特定快取的異動記錄，需要 *cache_id* 參數。只有在「快取管理程式」架構檔中以 tran-log 參數指定有效的異動日誌給快取時，此參數才會發生影響力。

url 「廣用相對位置」(URL) 位址，指定檔案在 Web 伺服器上的位置。例如：http:www.ibm.com/mydir/page1。

快取日誌

保存有關內部作業的數種類型之統計值，且會選用性地寫入快取日誌中。可爲每一個快取保存一個不同的日誌，或將所有統計值寫入相同的日誌中。本節討論下列快取日誌主題：

- 『架構日誌』
- 第170頁的『快取日誌格式』

架構日誌

若要記錄統計值，您必須架構「快取管理程式」架構檔。

若要架構日誌：

在「快取管理程式」架構檔的快取段落中，指定 stat-files 與 stat-interval 關鍵字。

您不需要停止、重新架構再重新啓動「快取管理程式」，即可修改統計值設定。

若要修改統計值聚集設定：

設定 cacheadm statistics 指令。 但請注意，以 cacheadm statistics 指令所做的變更，在「快取管理程式」重新啓動時並不會儲存。

快取日誌格式

統計值日誌是一般 ASCII 檔案，可由試算表或資料庫程式來處理或匯入。 寫成的記錄有三種類型：

- 起始設定記錄可針對特定快取，證明其統計值聚集的啓動。這些記錄的格式如下：

  mm/*dd*/*yy hh:mm:ss id* Initialization: interval *n* seconds

  其中：

mm/dd/yy	開始收集統計值的月、日、年
hh:mm:ss	開始收集統計值的時、分、秒
id	與記錄相關之快取的名稱
n	是收集間隔

- 終止記錄記錄了針對特定快取收集統計值的終止。 這些記錄的格式如下：

  mm/*dd*/*yy hh:mm:ss id* Termination

  其中：

mm/dd/yy	停止收集統計值的月、日、年
hh:mm:ss	停止收集統計值的時、分、秒
id	與記錄相關之快取的名稱

- 統計值記錄是一組以空格分隔的數字，用來顯示快取內的活動。這些記錄的格式如下：

  mm/*dd*/*yy hh:mm:ss id statistics*

  其中：

mm/dd/yy	建立統計值收集的月、日、年
hh:mm:ss	建立統計值收集的時、分、秒
id	與記錄相關之快取的名稱
<*statistics***>**	以空格區隔的統計值列示，它會列出針對 表14 中指定的快取收集的統計值：

表 14. 統計值列示

欄位號碼	目次	說明	計數器重新起始為 0
1	讀取	對快取執行讀取作業的次數	是
2	寫入	對快取執行寫入作業的次數	是
3	關閉	在快取的物件上執行關閉作業的次數	是
4	開啓讀取	在快取的物件上執行開啓讀取作業的次數	是
5	開啓寫入	在快取的物件上執行開啓寫入作業的次數	是
6	開啓寫入查詢	在快取的物件上執行開啓寫入查詢作業的次數	是
7	讀取選中	快取中，物件上的讀取選中次數	是
8	寫入選中	快取中，物件上的寫入選中次數	是
9	寫入查詢選中	快取中，物件上的寫入選中次數	是
10	起始設定	此快取所建立的新階段作業數量	是

表 14. 統計值列示 (繼續)

欄位號碼	目次	說明	計數器重新起始為 0
11	終止	此快取所終止的階段作業數量	是
12	除去	此快取中所刪除的物件數	否
13	已使用的記憶體	快取的記憶體部份中，物件已使用的記憶體數量	否
14	已使用的磁碟	快取的磁碟部份中，物件已使用的磁碟空間數量	否
15	可用的記憶體	快取的記憶體部份中，物件仍可使用的記憶體數量	否
16	可用的磁碟	快取的磁碟部份中，物件仍可使用的磁碟空間數量	否
17	記憶體物件計數	快取的記憶體部份中，物件的數量	否
18	檔案物件計數	快取的磁碟部份中，物件的數量	否
19	階段作業計數	快取中目前處於作用中的階段作業數	否

設定錯誤日誌層次

Net.Data 提供有錯誤日誌，這樣您就可監視 Net.Data 系統的錯誤或執行效能問題。

使用 Net.Data 錯誤日誌時，如果大部份的訊息都寫入到錯誤日誌中，您可能就會注意到對系統執行效能的影響。例如，每個使用者存取 Net.Data 找不到的巨集時， Net.Data 都會將訊息輸出到錯誤日誌中。

若要減少對執行效能的影響，請利用 DTW_LOG_LEVEL 關鍵字，來檢查 Net.Data 巨集中所設定之錯誤日誌的記錄層次。如果層次是設成 WARNING，請考慮將層次變更爲 ERROR，以增加些許的執行效能，或變更爲 OFF，以增加較大的執行效能。

將語言環境最佳化

下列段落將描述一些技術，當使用 Net.Data 提供的語言環境時，您可以使用它們來提高執行效能。

- 『REXX 語言環境』
- 第172頁的『SQL 語言環境』
- 第173頁的『系統及 Perl 語言環境』

REXX 語言環境

請使用下列要訣，改善 Net.Data 應用程式的執行效能：

- 儘可能結合您的 REXX 程式。具有更少、更大的程式會比較小的程式提供更好的執行效能，因爲每次在巨集中呼叫 REXX 語言環境函數時，將起始設定 REXX 直譯器。
- 將 REXX 程式儲存在外部檔案中，而不是將 REXX 程式包括在 Net.Data 巨集的行內。
- 對於外部 REXX 程式，請參照 %EXEC 陳述式中的指令行上的廣域變數。
- 經由定義廣域 Net.Data 變數並參照變數，將僅輸入參數直接傳遞給 REXX 程式。 對於列入 REXX 程式，請直接參照 REXX 來源中的廣域變數。

SQL 語言環境

在隨後的段落中，將描述某些關於資料庫及 SQL 語言環境的執行效能技術。若要瞭解 DB2 執行效能的注意事項，請探訪這個網站：*http://review.software.ibm.com/data/db2/performance*

資料庫技術

下列彙總將概述某些可改善資料庫存取的最簡單資料庫技術：

- 啓動資料庫。發出指令 `db2 activate database`，將以相當少的時間產生資料庫連線。請參閱 *DB2 Administration Guide*。 當使用 CGI 執行 Net.Data 時，這個執行效能技術可以完全代替「現場連線」(FastCGI 或 Web API 需要它) 的使用。
- 避免數字轉換。當比較直欄值與文字值時，請嘗試設定相同資料類型與屬性。如果文字值的精準度較直欄的精準度高，則 DB2 不會使用已命名的直欄的索引。如果要比較的兩個項目具有不同的資料類型，DB2 將須轉換這兩個值的其中一個或另一個，如此將會造成不精確的情況 (因爲有限的機器精準度所致)。

  例如，EDUCLVL 是半字整數值 (SMALLINT)。 請設定：

  ```
  ... WHERE EDUCLVL < 11 AND EDUCLVL >= 2
  ```

  而不要設定：

  ```
  ... WHERE EDUCLVL < 1.1E1 AND EDUCLVL > 1.3
  ```
- 避免字串填補。當固定長度的字串直欄值與文字值做比較時，請嘗試使用相同的資料長度。如果文字值的長度較直欄長度長，DB2 將不會使用索引。

  例如，EMPNO 爲 CHAR(6)，DEPTNO 爲 CHAR(3)。 請設定：

  ```
  ... WHERE EMPNO > '000300' AND  DEPTNO < 'E20'
  ```

  而不要設定：

  ```
  ... WHERE EMPNO > '000300 ' AND DEPTNO < 'E20 '
  ```
- 避免使用以 % 或 _ 開頭的 LIKE 型樣。百分比符號 (%) 與底線 (_) 當在 LIKE 述詞的型樣中使用時，將設定一個類似於您想要選取的橫列的直欄值的字串。當用來表示字串中間或結尾的字元時， LIKE 型樣可以利用索引。例如：

  ```
  ... WHERE LASTNAME LIKE 'J%SON%'
  ```

  不過，當在字串開頭使用時，LIKE 型樣可以阻止 DB2 使用任何可能已定義在 LASTNAME 直欄上的索引，來限制掃描的橫列數目。例如：

  ```
  ... WHERE LASTNAME LIKE '%SON'
  ```

  避免在字串開頭使用這些符號，尤其是在您存取特別大的表格時。

SQL 語言環境技術

您可以使用下列 SQL 語言環境技術，來改善執行效能。

- 使用 START_ROW_NUM 與 RPT_MAX_ROWS Net.Data 變數，減少傳回的表格的大小。 如果結果集合含有相當多的橫列，則您可以經由使用 START_ROW_NUM 及 RPT_MAX_ROWS，指定結果集合的次集，然後將它傳回給瀏覽器。START_ROW_NUM 指定要傳回的第一列的列號，而 RPT_MAX_ROWS 則是指定要傳回的列數。

重要事項: Net.Data 會重新發出每一要求的查詢，因爲在要求之間不會保持游標位置。

- 考慮呼叫使用靜態 SQL 的儲存程序。動態 SQL 是在執行時準備的，而靜態 SQL 則是在前置編譯階段時準備的。 SQL 語言環境會使用動態 SQL，以容許在程式執行時，它可以執行 SQL 陳述式。 因爲準備陳述式將需要額外的處理時間，所以靜態 SQL 可能更有效率。

系統及 Perl 語言環境

直接將僅輸入參數傳遞到「系統」或 Perl 語言環境將呼叫的程式。您可以經由定義廣域 Net.Data 變數並參照它們，來做到這一點。對於外部程式及 Perl script，請參照 %EXEC 陳述式中的指令行上的變數。對於列入 Perl script，請直接參照 Perl 原始檔中的變數。

第8章 Net.Data 記錄

Net.Data 會提供數個日誌，讓您在監視 Net.Data 執行效能及對錯誤進行疑難排解時使用。 Net.Data 日誌包括：

Net.Data 錯誤訊息日誌
含有所有 Net.Data 錯誤訊息的日誌。

「現場連線」日誌
含有「現場連線」錯誤訊息及 Net.Data、「現場連線」及 DB2 資料庫之間的通信的日誌。可對 DB2 資料庫的 DTW_SQL 及 DTW_ODBC 語言環境進行記錄。

下列段落將描述 Net.Data 記錄：

- 『記錄 Net.Data 錯誤訊息』
- 第177頁的『記錄現場連線 Cliette 及錯誤訊息』

記錄 Net.Data 錯誤訊息

Net.Data 會將錯誤訊息寫入到 Net.Data 錯誤日誌檔 `netdata.log`中。 錯誤日誌的最大大小是 Net.Data 所決定，固定爲 500 KB，大約爲 3000 個日誌項目。

您可定期瀏覽錯誤日誌檔案或保存的備份，以決定您的 Net.Data 系統是否有問題。

若要啓動 Net.Data 錯誤日誌：

- 設定 Net.Data 記錄架構變數 DTW_LOG_DIR：

  `DTW_LOG_DIR` *路徑*

  其中*路徑*是將儲存錯誤日誌檔案的目錄。
- 在巨集檔中設定 Net.Data 記錄變數 DTW_LOG_LEVEL：

  `@DTW_ASSIGN(DTW_LOG_LEVEL, "`*層次*`")`

  其中*層次*是記錄的層次。它可以具有下列值：

  off Net.Data 不會記錄錯誤。這是預設值。

  error Net.Data 記錄錯誤訊息。

  warning
  Net.Data 記錄警告以及錯誤訊息。

本段落將討論下列記錄主題：

- 第176頁的『Net.Data 錯誤日誌的規劃』
- 第176頁的『控制 Net.Data 記錄層次』
- 第176頁的『未記錄的 Net.Data 錯誤訊息的類型』
- 第176頁的『Net.Data 錯誤日誌檔案大小及輪替』
- 第177頁的『Net.Data 錯誤日誌格式』

Net.Data 錯誤日誌的規劃

在記錄錯誤時，您需要規劃下列議題：

- 決定磁碟空間：

  如果計畫使用錯誤日誌，您必須給予錯誤日誌額外的磁碟空間。

- 架構 Net.Data：

  如果計畫控制整個 Net.Data 系統的錯誤日誌，請在 Net.Data 起始設定檔案 DTW_LOG_DIR 中設定一個架構變數。

  即使您已在巨集中將 DTW_LOG_LEVEL 變數設定成錯誤或警告，錯誤日誌仍需要有這個變數。請參閱第14頁的『DTW_LOG_DIR: 錯誤日誌位置變數』，以學習如何更新起始設定檔案。

- 撰寫 Net.Data 巨集：

  以巨集中的 DTW_LOG_LEVEL 關鍵字設定記錄的層次。

- 執行 Net.Data：

  如果使用錯誤日誌，則您可檢查這些錯誤日誌及保存檔，以尋找 Net.Data 系統中的錯誤。

- 調整：

  請注意記錄會影響執行效能。請參閱第171頁的『設定錯誤日誌層次』，以取得執行效能議題的相關資訊。

控制 Net.Data 記錄層次

您可使用 DTW_LOG_LEVEL 變數字來指定記錄層次。請在 Net.Data 巨集中定義這個關鍵字。變數有三個設定：

off Net.Data 不會記錄錯誤。這是預設值。

error Net.Data 記錄錯誤訊息。

warning
Net.Data 記錄警告以及錯誤訊息。

未記錄的 Net.Data 錯誤訊息的類型

Net.Data 不會記錄巨集中由 MESSAGE 區段明確處理的錯誤。

Net.Data 錯誤日誌檔案大小及輪替

日誌檔的最大大小可爲 500 KB。這樣的大小大約可容納 3000 個日誌項目。

日誌檔到達最大大小時，會將檔案保存到 netdata.log*MMMDDYYYY_nn*

其中：

MMM 月 (Jan-Dec)

DD 日

YYYY 年

nn 01 到 99 的值，可用來單獨識別特定日期的每個保存檔案。

而會在原始檔案中繼續記錄。

Net.Data 錯誤日誌格式

日誌檔登錄有下列格式：

[*DD/MMM/YYYY:HH:MM:SS*] [*MACRO*] [*BLOCK*] [*PID#*] [*TID#*] *error_message*

參數：

DD 日

MMM 月 (Jan-Dec)

YYYY 年

HH 小時 (00-23)

MM 分 (00-59)

SS 秒數 (00-59)

MACRO
產生錯誤訊息的巨集

BLOCK
產生錯誤訊息的 HTML 區塊名稱。

PID# 產生錯誤訊息的處理的程序 ID 編號。可將多個 Net.Data 處理寫入日誌檔中，所以這個 ID 是必要的。

TID# 產生錯誤訊息的緒的緒 ID 編號。可將相同 Net.Data 處理的多個緒寫入到日誌檔中，所以這個 ID 是必要的。

error_message
錯誤訊息的文字

記錄現場連線 Cliette 及錯誤訊息

「現場連線」會將訊息及「現場連線」、Net.Data 及 DB2 資料庫之間的通信記錄在「現場連線」日誌檔中。日誌的最大大小是 Net.Data 所決定，固定爲 1 KB，大約爲 1200 個日誌項目。

您可定期瀏覽日誌檔案或保存的備份，來決定您的 cliette 或 DB2 資料庫是否有問題。

若要啓動「現場連線」日誌：

以 `-l` 屬性啓動「連線管理程式」：

```
dtwcm -l [層次]
```

其中*層次*是記錄的層次。 它可以具有下列值：

normal
「現場連線」會記錄所有 cliette 活動、相關的 DB2 SQL 陳述式及狀態訊息，以及「現場連線」錯誤訊息

minimal
「現場連線」僅記錄重要的資訊，如資料庫查詢及結果集合中的列數。

本段落將討論下列記錄主題：

- 『「現場連線」日誌的規劃』
- 『控制現場連線記錄層次』
- 『未記錄的現場連線訊息的類型』
- 『現場連線日誌檔名稱』
- 第180頁的『現場連線日誌檔大小及輪替』
- 第180頁的『現場連線日誌格式』

「現場連線」日誌的規劃

在記錄訊息時，您需要規劃下列議題：

- 決定磁碟空間：

  如果計畫使用錯誤記錄，您必須給予日誌檔額外的磁碟空間。
- 執行「連線管理程式」

  您可以在 dtwcm 指令上輸入一個屬性來啓動記錄。請參閱第177頁的『記錄現場連線 Cliette 及錯誤訊息』，取得語法。

  如果要使用記錄，則您可檢查這些日誌及保存檔，找出 cliette 中的錯誤。
- 調整：

  請注意記錄會影響執行效能。請參閱第171頁的『設定錯誤日誌層次』，取得執行效能議題及『控制現場連線記錄層次』的資訊。

控制現場連線記錄層次

當呼叫「連線管理程式」時，您可以在 dtwcm 指令上，指定記錄層次。dtwcm 指令的 -l 屬性具有兩個設定：

normal
「現場連線」會記錄所有 cliette 活動、相關的 DB2 SQL 陳述式及狀態訊息，以及「現場連線」錯誤訊息

minimal
「現場連線」僅記錄重要訊息。這個選項將在日誌中提供更少的訊息。

未記錄的現場連線訊息的類型

「現場連線」不會記錄 Net.Data 錯誤或由巨集中的 MESSAGE 區段明確處理的錯誤。

現場連線日誌檔名稱

「現場連線」會對「連線管理程式」及每一個 cliette 建立一個日誌檔。下列列示描述名稱格式：

連線管理程式檔案
格式：

conman-*process_id*-*DDMMMYYYYHHMMSS*.log

參數：

process_id
「連線管理程式」處理的識別字

DD
日

MMM
月 (Jan-Dec)

YYYY
年

HH
小時，24 時制

MM
分鐘

SS 秒

範例：

```
conman-513-01Feb1999095639.log
```

Cliette 檔案

格式：

cliett-*process_id-DDMMMYYYYHHMMSS*.log

參數：

process_id
cliette 緒的識別字

DD
日

MMM
月 (Jan-Dec)

YYYY
年

HH
小時，24 時制

MM
分鐘

SS 秒

範例：

```
cliett-592-01Feb1999095647.log
```

現場連線日誌檔大小及輪替

日誌檔的最大大小可爲 1 MB。這樣的大小大約可容納 6000 個日誌項目。當日誌檔達到最大大小時，處理將關閉原始的日誌檔、建立新的日誌檔，以及繼續記錄到新的檔案中。

日誌檔與 dtwcm 及 dtwcdb2 一樣位在同一個目錄中

現場連線日誌格式

日誌檔登錄有下列格式：

```
--process_type-DD/MMM/YYYY:HH:MM:SS-PID#--
message_text
```

參數：

process_type
dtwcm 或 cliet，視連線管理程式或 cliette 記錄訊息而定。

DD 日

MMM 月 (Jan-Dec)

YYYY 年

HH 小時 (00-23)

MM 分 (00-59)

SS 秒數 (00-59)

PID# 產生訊息的處理的程序 ID 編號。

message_text
訊息的文字。

範例 1：連線管理程式日誌登錄。

```
--dtwcm-02/Mar/1999:13:43:07-330--
Creating connection manager ...successfully
Reading configuration info ...
Completing initialization ...
Initializing cm server ... successfully
Initializing NLS environment ... successfully
Detecting cliette ./dtwcdb2 for DTW_SQL:CELDIAL:  Min process(es) = 1, Priv Port = 7100.
Starting 1 cliettes for DTW_SQL:CELDIAL.
Started: ./dtwcdb2 7128 7100 7200 DTW_SQL:CELDIAL  LOG_MAX , pid: 213
1 cliettes for DTW_SQL:CELDIAL started.
...
```

範例 2：cliette 日誌登錄。

```
--cliet-02/Mar/1999:13:43:08-335--
Cliette starting ...
Cliette: DTW_SQL:SAMPLE, database: SAMPLE, user: *USE_DEFAULT
Making a new connection to database: SAMPLE, user: *USE_DEFAULT.
Calling SQLAllocHandle for environment ...
Calling SQLAllocHandle for connection ...
Calling SQLSetConnectAttr ...
```

```
Calling SQLConnect ...
Connecting to database: SAMPLE sucessfully.
Telling CM the cliette is ready ...
Ready and waiting for command from CM ...
```

附錄A. 參考書目

本節會列示本書中所參考的文件。

- 『Net.Data 技術圖書庫』

Net.Data 技術圖書庫

可以從 Net.Data 網站存取「Net.Data 技術圖書庫」，網址：

`http://www.software.ibm.com/data/net.data/library.html`

文件	說明
• *Net.Data Administration and Programming Guide for OS/390* • *OS/2 版、Windows NT 版和 UNIX 版 Net.Data 管理及程式設計指南* • *OS/400 版 Net.Data 管理及程式設計指南*	含有關於如何安裝、配置及呼叫 Net.Data 的觀念及作業資訊。此外也描述如何撰寫 Net.Data 巨集、使用 Net.Data 效能技術、使用 Net.Data 語言環境、管理連線及使用 Net.Data 記錄及追蹤故障檢修及效能調整。
Net.Data 參考手冊	描述 Net.Data 巨集語言、變數及內建函數。
Net.Data 語言環境介面參考手冊	描述 Net.Data 語言環境介面。
Net.Data 訊息與訊息碼參考手冊	列示 Net.Data 錯誤訊息及回覆碼。

附錄B. Net.Data for AIX

AIX 的明細包括在 Net.Data 所附的 README 檔中。 README 檔包括下列資訊：

- 需求
- 安裝
- 架構
- 解除安裝

載入語言環境的共用程式庫

在 AIX 平台上建立語言環境時，您需要載入共用程式庫。在 AIX 上，需有語言環境，方可提供 Net.Data 所呼叫的常式，及傳回語言環境介面常式 (如 dtw_initialize() 與 dtw_execute()) 的位址。

Net.Data 會使用 dtw_fp 結構，從 AIX 中的語言環境取回語言環境介面常式的指標，且具有此種格式：

```
typedef struct dtw_fp {
    int (* dtw_initialize_fp)();   /* dtw_initialize function pointer */
    int (* dtw_execute_fp)();      /* dtw_execute function pointer    */
    int (* dtw_cleanup_fp)();      /* dtw_cleanup function pointer    */
} dtw_fp_t;
```

當載入共用程式庫時，Net.Data 會將這個結構傳遞到語言環境，作爲 dtw_getFp() 常式中的參數。

dtw_fp 結構將當成唯一的參數來傳遞。這個結構含有每一個支援介面的欄位，且設定這些欄位是語言環境的責任。如果語言環境提供設定的介面，它會將欄位設定爲該介面的函數指標。如果它未提供設定的介面，它會將欄位設定爲 NULL。程式模版中的 dtw_getFp() 常式會顯示此常式的正常實施方式。

當載入共用程式庫時，爲了讓 Net.Data 能夠取得這個常式的指標， dtw_getFp 常式須是共用程式庫的匯出檔案中所設定的第一個進入點。程式庫的樣本匯出檔案稱爲 dtwsampshr.o，它支援如下的所有可用的語言環境介面常式：

```
#!dtwsampshr.o
dtw_getFp
dtw_initialize
dtw_execute
dtw_cleanup
```

提高 REXX 環境中的執行效能

若您在 AIX 系統多次呼叫 REXX 語言環境，您可考慮將 RXQUEUE_OWNER_PID 環境變數設定爲 0。可多次呼叫 REXX 語言環境的巨集可以很容易地產生許多處理程序及大量的系統資源。

您可以將環境變數設定成下列三種方式的其中之一：

- 在巨集中，使用 DTW_SETENV 內建函數：

  ```
  @DTW_rSETENV("RXQUEUE_OWNER_PID", "0")
  ```

- 在 AIX 系統環境檔：

```
/etc/environment: RXQUEUE_OWNER_PID = 0
```

這個方法會影響 REXX 對整個機器的行爲。

- 在 HTTP Web 伺服器環境檔中；例如，對 Domino Go Webserver 插入下列陳述式：

```
InheritEnv RXQUEUE_OWNER_PID = 0
```

這個方法會影響 REXX 對 Web 伺服器的行爲。

NLS 注意事項

Net.Data 係使用與 Web 伺服器使用的同一字碼頁來執行。若要使得 Net.Data 能夠使用適合您的語言環境的字碼頁，則 Web 伺服器必須使用正確的字碼頁。例如，如果您將使用 IBM Internet Connection Server，且想要使用韓文字碼頁，請停止伺服器，並使用韓文語言環境來重新啓動它：

```
stopsrc httpd
startsrc -s httpd -e "LC_ALL=ko_KR"
```

如果您的巨集含有 UTF-8 字元，或您將連接到含有 Unicode 資料的 DB2 資料庫，請將 INI 檔中的 DTW_UNICODE 架構變數設定爲 YES。Net.Data 目前在巨集中支援 UTF-8 字元，非 UTF-16。不過，Net.Data 可以處理以 UTF-8 或 UTF-16 編碼的資料庫資料。 Net.Data 輸出恆是 UTF-8 格式。

執行效能要訣：如果您將在二位元組語言環境中執行，且您的字串或工作中的內建函數恆會處理單位元組字串，請將 DTW_MBMODE 設定爲 NO，以節省不必要的轉換。

附錄C. Net.Data SmartGuide

Net.Data SmartGuides 在於提供您一種快速又簡便方式，來建立自行設定的 Net.Data 應用程式。簡單地選取一個 SmartGuide，回答少許問題，然後 Net.Data 即會替您建立一個自行設定的應用程式。

Net.Data 會提供下列 SmartGuide，讓您學習如何建立巨集及使用 Net.Data 特性：

探索明細

這個 SmartGuide 將採用舊有的資料庫表格，並建立具有 Web 能力的探索明細應用程式，讓您可以存取不同明細層次的資料。「探索明細 SmartGuide」可以選擇是否要連線到您的資料庫，來收集您的資料庫資訊。您最多可自行設定 5 份 Web 報表。建立的巨集會使用您資料庫中所儲存的主要鍵與外來鍵，自動鏈結您的 Web 報表。

儲存程序

這個 SmartGuide 會連線到您的資料庫，並取回與您的資料庫一起登記的所有儲存程序的列示。選取一個儲存程序，SmartGuide 即會替您建立一個 Net.Data 巨集，來呼叫您的儲存程序。 然後，您可以修改建立的巨集，或在您的舊有 Net.Data 應用程式中整合它。

聯絡 這個 SmartGuide 會建立一個具有 Web 能力的通訊錄，來儲存名稱、地址、電話號碼及其他重要的聯絡資訊。它附有一個搜尋功能，可讓您快速存取您的聯絡資訊。建立的聯絡應用程式可以包括簡短或詳細的報表。此外，您可以包括一個自行設定的報表。

請查閱 Net.Data Web 網站 `http://www.software.ibm.com/data/net.data`，取得最新版的 Net.Data SmartGuide 套裝軟體。

這個附錄將討論下列主題：

開始之前

若要執行 SmartGuides 並建立 Net.Data 巨集，必須已安裝下列軟體：

- Java Development Kit (JDK) 或 Java Runtime Environment (JRE) 1.1.x
- Net.Data 版本 2 或更高的版本
- IBM Universal Database (UDB) 5.0 或更高的版本
- REXX (探索明細 SmartGuide 必備的軟體)

執行 SmartGuides

Net.Data SmartGuides 是從指令行中啓動且包含在檔案 NetDataSmartGuides.jar 中。

若要以 Java Development Kit (JDK) 啓動 SmartGuide：

1. 將下列字行新增到您的 CLASSPATH 環境變數中。

   *[Path]*NetDataSmartGuides.jar

   其中 *[Path]* 是 NetDataSmartGuides.jar 檔案的可選用路徑。

2. 如果您想要使用「探索明細 SmartGuide」與「儲存程序 SmartGuide」的資料庫連線特性，請將 UDB JDBC 驅動常式新增到您的 CLASSPATH 環境變數中。

   對於 Windows NT 與 OS/2 作業系統，請將下列一行新增到您的 CLASSPATH 環境變數中：

   *[Path]*NetDataSmartGuides.jar;*[UDBInstallationPath]*\java\db2java.zip

   對於 UNIX 作業系統，請將下列一行新增到您的 CLASSPATH 環境變數中：

   *[Path]*NetDataSmartGuides.jar:*[UDBInstallationPath]*\java\db2java.zip

   其中 *[Path]* 是 NetDataSmartGuides.jar 檔案的可選用路徑，而 *[UDBInstallationPath]* 則是您的 UDB 安裝路徑，例如，C:\SQLLIB。

3. 啓動 SmartGuide。
   - 對於 UNIX 作業系統，請輸入下列指令：

     java TaskGuide LaunchPad.class
   - 對於 Windows NT 與 OS/2 作業系統，請執行下列檔案：

     SmartGuides.cmd

   發射台會隨著如 第189頁的圖26 顯示的 SmartGuides 一起開啓。

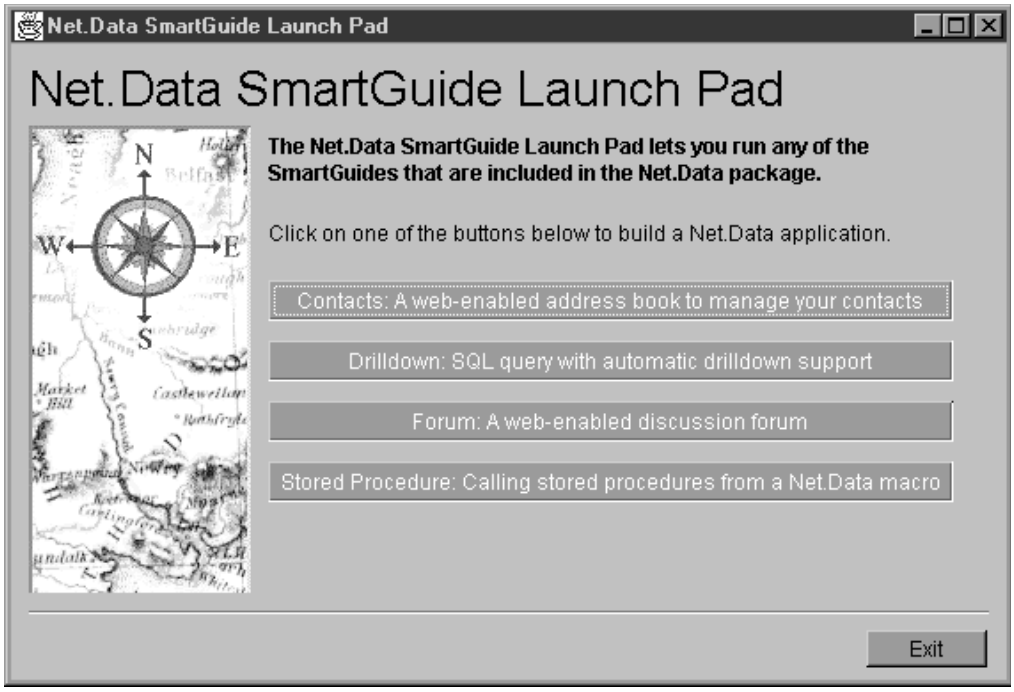


圖 26. SmartGuide 發射台

4. 按一下您想要執行的 SmartGuide 的名稱。

若要以 Java Runtime Environment (JRE) 啓動 SmartGuide：

1. 對於 Windows NT 作業系統，請選擇 **開始->程式集->Net.Data->SmartGuides**，然後會執行名爲 SMARTGUIDES.BAT 的批次檔。對於其他作業系統，請輸入下列指令來執行 SmartGuides：

   ```
   jre -cp [Path]NetDataSmartGuides.jar TaskGuide LaunchPad.class
   ```

   其中 *[Path]* 是 NetDataSmartGuides.jar 檔案的可選用路徑。

   發射台會隨著如 圖26 顯示的 SmartGuides 一起開啓。

2. 如果您想要使用「探索明細 SmartGuide」與「儲存程序 SmartGuide」的資料庫連線特性，請輸入下列指令：

   對於 Windows NT 與 OS/2 作業系統：

   ```
   jre -cp [Path]NetDataSmartGuides.jar;[UDBInstallationPath]
     \java\db2java.zip TaskGuide LaunchPad.class
   ```

   對於 UNIX 作業系統：

   ```
   jre -cp [Path]NetDataSmartGuides.jar:[UDBInstallationPath]
     \java\db2java.zip TaskGuide LaunchPad.class
   ```

   其中 *[Path]* 是 NetDataSmartGuides.jar 檔案的可選用路徑，而 *[UDBInstallationPath]* 則是您的 UDB 安裝路徑，例如，C:\SQLLIB。

3. 按一下您想要執行的 SmartGuide 的名稱。

附錄D. 使用 Net.Data SQL 輔助程式建置 SQL 陳述式

「Net.Data SQL 輔助程式」是一種以 Java 爲主的 SQL 陳述式建置器，它可以提供易於使用的 GUI，透過建置 SQL 陳述式的處理來指導您。透過「Net.Data SQL 輔助程式」，您可以：

- 建置 SELECT、INSERT、UPDATE 及 DELETE 陳述式 (包括 SELECT DISTINCT)
- 使用值查閱、AND 或 OR，以及類型感應輸入欄位，來建置多個條件
- 定義表格 JOINS (inner, right outer 及 left outer)
- 選取要檢視的直欄
- 選取排序次序
- 輸入要在條件、值及排序中使用的使用者定義的變數

在建置 SQL 陳述式後，您可以：

- 將 SQL 陳述式儲存爲檔案
- 建立及儲存含有 SQL 陳述式的巨集
- 將 SQL 陳述式或巨集複製到剪輯暫存區

這個附錄將討論下列主題：

- 『開始之前』
- 『執行 Net.Data SQL 輔助程式』

開始之前

若要執行「Net.Data SQL 輔助程式」，您必須已安裝了下列軟體：

- Java Development Kit (JDK) 或 Java Runtime Environment (JRE) 1.1.x
- 可使用 JDBC 的資料庫

請參閱您的資料庫文件，取得如何透過 JDBC 存取資料來源的詳細資訊，以及其他可能必要的伺服器啓動值。例如，在遠端存取 DB2 UDB v5.0 資料來源時，資料庫伺服器必須正在執行 JDBC 伺服器 (db2jstrt)。

執行 Net.Data SQL 輔助程式

「Net.Data SQL 輔助程式」是從指令行啓動，且位在檔案 {*inst_dir*}/assist/NetDataAssist.jar 中。

若要使用「Java 開發工具箱 (JDK)」啓動「Net.Data 輔助程式」：

輸入輸入下列來啓動「Net.Data 輔助程式」：

```
java -classpath %CLASSPATH%;{inst_dir}/assist/NetDataAssist.jar NetDataAssist
```

若要使用「Java 執行環境 (JRE)」啓動「Net.Data 輔助程式」：

輸入下列指令來啓動「Net.Data 輔助程式」：

```
jre -cp %CLASSPATH%;{inst_dir}/assist/NetDataAssist.jar NetDataAssist
```

按一下**下一步**按鈕，導覽用來登入、建構 SQL 陳述式及建立 Net.Data 巨集的視窗。

附錄E. 搭配使用 NetObjects Fusion NOF Plug-in 與 Net.Data servlet

Net.Data 會提供 NetObjects Fusion plug-in 供 Net.Data servlet 使用。 Net.Data servlet 會在第70頁的『Net.Data servlet』中予以說明。

您可以使用 NetObjects Fusion (NOF) 來整合舊有的 Net.Data 巨集，這將提供與 Web 站台管理與易於使用的圖形式使用者介面（GUI）的更佳整合。

這個附錄將討論下列主題：

- 『關於 NetObjects Fusion Plug-in』
- 第194頁的『安裝 NetObjects Fusion Plug-in』
- 第194頁的『設定 NetObjects Fusion 的 Net.Data Plug-in』
- 第196頁的『使用 NOF Plug-in 來發行 servlet』

關於 NetObjects Fusion Plug-in

NetDataservlet.NFX plug-in 可與 Net.Data servlet 搭配運作。這個 NOF plug-in 支援舊有 Net.Data 巨集，或單一 Net.Data 函數的呼叫。 plug-in 會建立 HTML，來呼叫 Net.Data 作爲 servlet 或 server-side-include (SSI)。 Web 伺服器呼叫 Net.Data 時，會執行 Net.Data 巨集或函數。 請使用 Net.Data servlet plug-in 將巨集及函數 servlet 內嵌到 NOF 管理的網站，其說明在圖27 中。plug-in 提供大部份的基本巨集或函數參數，並以選取的預設值來自動建置巨集。若要使用此 plug-in，您必須安裝與架構 NOF 及 plug-in 檔案。

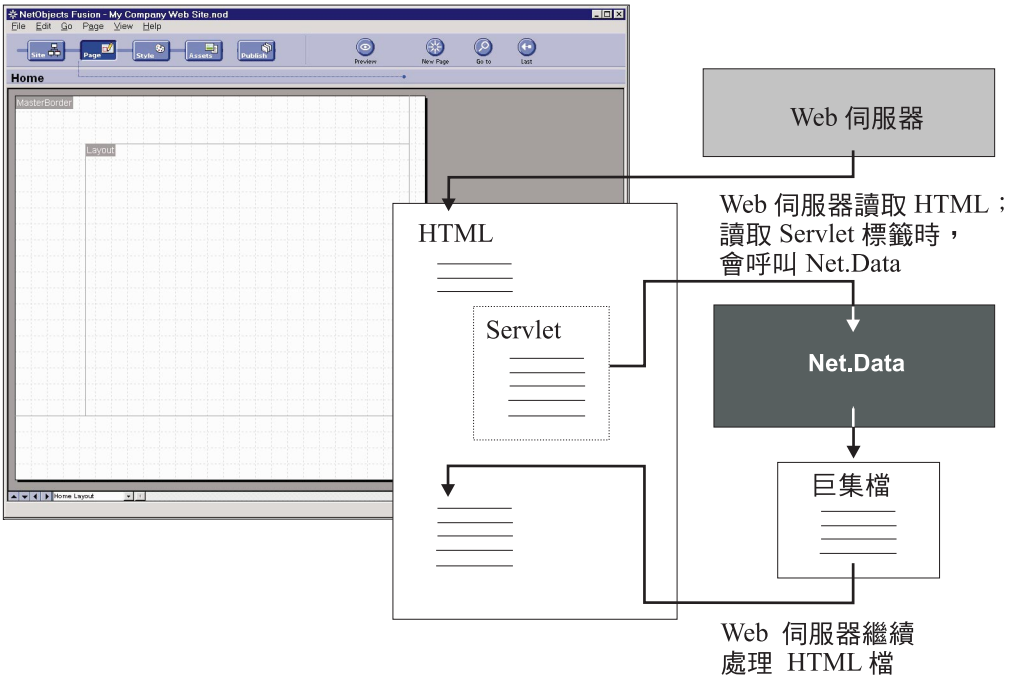


圖 27. Net.Data servlet Plug-in

安裝 NetObjects Fusion Plug-in

軟硬體需求：

NOF plug-in 需要有 NetObjects Fusion 版本 2.0 或更新版本。

若要安裝：請將 <inst_dir>\fx\NetDataservlet.nfx 與 <inst_dir>\fx\NetDataservlet.gif 複製到您的 <NetObjects Fusion>\components\ 目錄中。

設定 NetObjects Fusion 的 Net.Data Plug-in

使用 NOF，您可變更正在處理之 servlet 的性質。

1. 開啓 NetObjects Fusion。
2. 從 NetObjects Fusion (NOF) 的**工具**選用區中，選取 **NetObjects 構成要素**按鈕：

    plug-in 按鈕會顯示在**工具**選用區的底端。

3. 從這六個**工具**選用區按鈕中，選取 **NetObjects 構成要素**按鈕： 
4. 在 NOF 畫布上，請標出此區域以指定想要放置所選 plug-in 的位置。這是顯示 servlet 結果的位置。會開啓「安裝的構成要素」視窗，其中會顯示可選取的 plug-in 列示。如果 servlet plug-in 不在列示中，請使用路徑和檔案名稱欄位來指定 plug-in 檔名 `NetDataservlet.NFX`，以與巨集或函數 servlet 搭配使用
5. 從列示中選取 servlet plug-in，並按一下 **OK** 按鈕。

   此 plug-in 會變成 NOF 畫布上的物件。

修改 Plug-in 性質

您可使用 Net.Data servlet plug-in 來修改巨集及函數 servlet。

使用 NOF 來修改 Net.Data servlet：

1. 在 NOF 畫布上，請標出此區域以指定想要放置所選 plug-in 的位置。這是顯示 servlet 結果的位置。會開啓「安裝的構成要素」視窗，其中會顯示可選取的 plug-in 列示。如果 servlet plug-in 不在列示中，請使用路徑和檔案名稱欄位來指定 plug-in 檔名 `NetDataservlet.NFX`，以與巨集或函數 servlet 搭配使用。
2. 從列示中選取 Net.Data servlet plug-in，並按一下 **OK** 按鈕。

   此 plug-in 會變成 NOF 畫布上的物件。
3. 選取並自行設定 Net.Data servlet plug-in 的性質：
   a. 選取 NOF 畫布上的 Net.Data servlet plug-in。NOF **性質**選用區即會開啓，顯示第195頁的圖28 中所示的 plug-in 性質。

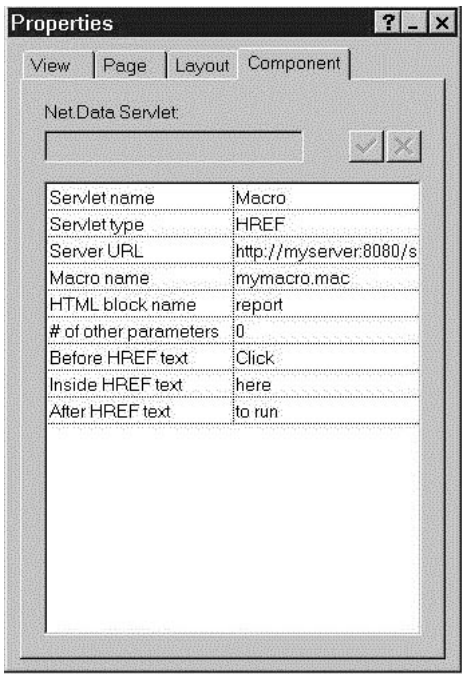


圖 28. Net.Data servlet 性質選用區

Net.Data servlet 的性質：

您可自行設定下列性質：

servlet 名稱

選取您想要呼叫的 servlet 名稱：函數或巨集。而依據您所選取的 servlet 名稱，會顯示不同的性質。

servlet 類型

選取您想要的 servlet 類型：SSI、HREF 或 FORM 提出按鈕而依據您想要的 servlet 類型，會顯示不同的性質。

提出標籤

如果選取 FORM 提出按鈕類型，請指定提出標籤的文字。否則，不會顯示這個性質。

伺服器 URL

如果選取 HREF servlet 類型，請將伺服器 URL 指定給已啓用 servlet 的 Web 伺服器。如果選取 SSI，則不會顯示這個性質。

巨集名稱

如果選取巨集 servlet 名稱，請指定要執行之舊有 Net.Data 巨集的名稱。否則，不會顯示這個性質。

HTML 區塊名稱

如果選取巨集 servlet 名稱，請指定要執行之 Net.Data 巨集中的 HTML 區塊名稱。 否則，不會顯示這個性質。

函數類型

如果選取函數 servlet 名稱，請選取要執行的函數類型：函數或 SQL。否則，不會顯示這個性質。

語言環境

如果選取函數 servlet 名稱，請指定要使用的 Net.Data 語言環境。否則，不會顯示這個性質。

陳述式 如果選取 函數 servlet 名稱，請指定要執行的陳述式。否則，不會顯示這個性質。

資料庫 如果您選取函數 servlet 名稱，請設定要使用的資料庫的名稱。否則，不會顯示這個性質。

其他參數數目

指定傳送給 Net.Data 的其他參數數目 (最大值：25)。請輸入每個參數的參數名稱及值 (選用)。

HREF 本文之前

如果選取 HREF servlet 類型，請將文字指定成出現在 <A HREF> HTML 標籤的文字前面。否則，不會顯示這個性質 (選用)。

HREF 本文內

如果選取 HREF servlet 類型，請將文字指定成出現在 <A HREF> HTML 標籤的內部。否則，不會顯示這個性質 (選用)。

HREF 本文之後

如果選取 HREF servlet 類型，請將文字指定成出現在 <A HREF> HTML 標籤的文字後面。否則，不會顯示這個性質 (選用)。

SQL 提示！

如果選取 HREF servlet 類型並指定 SQL 函數類型，則會出現含有提示的訊息，以告知 HREF SQL 陳述式應將加號 (+) 字元用於任意空格 ( ) 字元。不能變更這個文字，也不會在發行網頁後顯示。否則，不會顯示這個性質。

b. 定義網頁的性質後，請按一下**發行**按鈕，以使用 Net.Data servlet NOF plug-in 來建置與發行網頁。

註: 如果選取 SSI servlet 類型，您的網頁檔案副檔名應爲 .shtml。您可從 NOF **性質**選用區中將這個設定成您的預設網頁，方法是選取**頁**筆記本標籤、按一下**自訂名稱**按鈕，並在**副檔名類型**欄位中輸入 .shtml。

使用 NOF Plug-in 來發行 servlet

設定網頁的性質後，請按一下**發行**按鈕，以使用 plug-in 來建置與發行網頁。

附錄F. Net.Data 樣本巨集

這個樣本巨集應用程式顯示員工姓名的列示，而應用程式使用者只要從列示中選取員工的姓名，就可取得個別員工的其它資訊。巨集會使用 SQL 語言環境來查詢 EMPLOYEE 表格，以取得員工姓名及特定員工的相關資訊。

巨集檔會使用一個併入檔，來含有巨集的 DEFINE 區塊。

第198頁的圖29顯示樣本巨集。第200頁的圖30 顯示併入檔。

```
%{************************ 樣本巨集 ****************************
*   檔名 = sqlsamp1.d2w                                          *
*   說明：                                                       *
*     這個 Net.Data 巨集查詢...                                  *
*         - 員工表格，以建立要在瀏覽器上                         *
*          顯示的員工選項列示                                    *
*         - 員工表格以取得各別員工的                             *
*         其它資訊                                                    *
*                                                                *
******************************************************************%}
%{************************************************************************
*   廣域 DEFINE 的併入檔 -                                                *
************************************************************************%}
%INCLUDE "sqlsamp1.hti"
%{************************************************************************
*  函數：  queryDB              語言環境： SQL                           *
*  說明： 查詢 myTable 變數所表示的表格，及                              *
*  從結果建立選項列示。myTable 變數的值是                                *
*  指定於併入檔 sqlsamp1.hti 中。                                        *
************************************************************************%}
%FUNCTION(DTW_SQL) queryDB() {
 SELECT FIRSTNME FROM $(myTable)
  %MESSAGE {
    -204:  {<p><b>錯誤 -204：找不到表格 $(myTable)。</b>
           <p>請確定所用的併入檔無誤。</b>
           %} : exit
    +default: "警告 $(RETURN_CODE)" : continue
    -default: "意外的錯誤 $(RETURN_CODE)" : exit
%}

%REPORT {
<select name=emp_name>
%ROW{
<option>$(V1)
%}
</select>
%}
%}

%{************************************************************************
*  函數： fname         語言環境： SQL                                    *
*  說明： 查詢 myTable 變數所表示的表格，                                 *
*         以取得 emp_name 變數所識別出的員工                             *
*         之其它資料。                                                   *
************************************************************************%}
%FUNCTION(DTW_SQL) fname(){
SELECT FIRSTNME, PHONENO, JOB FROM $(myTable) WHERE FIRSTNME='$(emp_name)'
  %MESSAGE {
  -204: "錯誤 -204: 找不到表格 "
  -104: "錯誤 -104: 語法錯誤"
   100: "警告 100: 無記錄" : continue
  +default: "警告 $(RETURN_CODE)" : continue
  -default: "意外的 SQL 錯誤" : exit
%}
%}
```

圖 29. 樣本巨集 (1/3)

```
%{*****************************************************************************
*  HTML 區塊： INPUT            標題： 動態查詢選項                             *
*                                                                              *
*  說明： 查詢員工表格，以建立要在瀏覽器上                                      *
*         顯示的員工選項列示                                                   *
*****************************************************************************%}
%HTML(INPUT) {
<html>
<head>
<title>建立員工選取列示</title>
</head>
<body>
<h3>$(exampleTitle)</h3>
<p>此範例會查詢表格，並使用結果來建立使用 <em>%REPORT</em> 區塊的選項列示。
<hr>
<form method="post" action="report">
@queryDB()<input type="submit" value="Select Employee">
</form>
<hr>
  </body>
</html>
%}
```

圖 29. 樣本巨集 (2/3)

```
%{*****************************************************************************
*  HTML 區塊：     REPORT                                                      *
*  說明： 查詢員工表格，以取得各別員工的                                       *
*         其它資訊                                                             *
*****************************************************************************%}
%HTML(REPORT){
<html>
<head>
<title>取得員工資訊</title>
</head>
<body>
<h3>您所選的員工名稱 = $(emp_name)</h3>
<p>以下為該員工的相關資訊：
<PRE>
@fname()
</PRE>
<hr><a href="input">回到上一頁</a>
  </body>
</html>
%}

%{     Net.Data 巨集 1 結束 %}
```

圖 29. 樣本巨集 (3/3)

```
================================================================================
%{***************************     併入檔     ********************************
*   檔名 = sqlsamp1.hti                                                        *
*   說明：                                                                     *
*     此併入檔提供給 Net.Data 巨集 sqlsamp1.d2w                                *
*     巨集 DEFINE。                                                            *
********************************************************************************%}
%define {
    emp_name   =""
    reposition = sign
 exampleTitle = "樣本巨集"
    myTable = "員工"
    DATABASE = "樣本"
%}

%{      併入檔結束   %}
```

圖 30. 併入檔

附錄G. 注意事項

本書是針對 IBM 在美國所提供之產品與服務開發出來的。 而在其他國家中， IBM 不見得有提供本書中所提的各項產品、服務、或功能。要知道在您所在之區是否可用到這些產品與服務時，請向當地的 IBM 服務代表查詢。本書在提及 IBM 的產品、程式或服務時，不表示或提示只能使用 IBM 的產品、程式或服務。只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。不過，其他非 IBM 產品、程式、或服務在運作上的評價與驗證，其責任屬於使用者。

本文件中包含著 IBM 所擁有之專利或暫准專利。使用者不得享有本書內容之專利權。您可以用書面方式來查詢特許權限，來函請寄到：

臺灣國際商業機器股份有限公司
台北市基隆路一段 206 號
法務部

若要查詢有關二位元組 (DBCS) 資訊的特許權限事宜，請聯絡您國家的 IBM 智慧財產部門，或者用書面方式寄到：

臺灣國際商業機器股份有限公司
台北市基隆路一段 206 號
法務部

下列段落若與該國之法律條款抵觸，即視為不適用： IBM 就本書僅提供 『交附時之現況』 保證，而並不提供任何明示或默示之保證，如默示保證書籍之適售性或符合客戶之特殊使用目的； 有些地區在某些固定的異動上並不接受明示或默示保證的放棄聲明，因此此項聲明不見得適用於您。

本書中可能會有技術上或排版印刷上的訛誤。 因此，IBM 會定期修訂；並將修訂後的內容納入新版中。 同時，IBM 會隨時改進並 (或) 變動本書中所提及的產品及 (或) 程式。

在本書中，若有任何非 IBM Web 網站的參考資料，都只是爲了您的方便而提供，無論在何種情況下，都不能爲那些網站作保證。而那些網站上的資料都不能作爲此 IBM 產品的一部份，且使用那些網站的風險都必須由您自己承擔。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊者可洽詢 IBM。其下列資訊指的是： (1) 獨立建立的程式與其他程式 (包括此程式) 之間更換資訊的方式 (2) 相互使用已交換之資訊方法 若有任何問題請聯絡：

台北市基隆路一段 206 號
臺灣國際商業機器股份有限公司
法務部

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於雙方之 [IBM 客戶合約] 或 [IBM 國際程式授權合約] (或任何同等合約) 條款，提供本書中所說的授權程式與其所有適用的授權資料。

本書所提及之非 IBM 產品資訊，係一由產品的供應商，或其出版的聲明或其他公開管道取得。 IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性、或任何對產品的其他主張是否完全無誤。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

所有關於 IBM 未來方向或計劃的聲明，皆需視變更或取消的情形而定，但並不另行通知，而且這些聲明僅代表目標及目的。

本書僅限用於規劃目的。在所描述之產品上市前，本書所提供的資訊有可能變更。

本書包含日常事務作業所使用的資料及報告範例。爲了儘可能地完整呈現這些範例，這些範例會包含個人、公司及產品的名稱。所有這些名稱皆爲虛構的，若有任何名稱或地址與實際的公司企業相似，則純屬巧合。

商標

下列專有名詞是 IBM 公司在美國或 (及) 其它國家的商標：

AIX
AS/400
DB2
DB2 Universal Database
DRDA
DataJoiner
IBM
IMS
Language Environment
MVS/ESA
Net.Data
OS/2
OS/390
OS/400
OpenEdition

下列專有名詞是以下其它公司的商標：

Java 及所有 Java 型的商標或標誌是 Sun Microsystems, Inc. 在美國及 (或) 其它國家的商標。

UNIX 是透過 X/Open Company Limited 獨家授權，在美國及 (或) 其它國家的註冊商標。

Lotus 及 Domino Go Webserver 是 Lotus Development Corporation 在美國及 (或) 其它國家的商標。

Microsoft、Windows、Windows NT 及 Windows 標誌是 Microsoft Corporation 在美國及 (或) 其它國家的商標或註冊商標。

其它公司、產品與服務名稱 (以雙星號 ** 表示)，可能是其它公司的商標或服務標記。

名詞解釋

一劃

一致資源定址器 (uniform resource locator). 用來指出 HTTP 伺服器並可選擇性地指出目錄以及檔案名稱的位址，例如：
`http://www.software.ibm.com/data/net.data/index.html`。

三劃

工作單元 (unit of work). 被視爲原子作業的作業的可恢復順序。工作單元內的所有作業均可完成 (已確定) 或還原 (已回轉)，好似作業是單一作業一般。僅有資源上受到確定控制的作業可被確定或回轉。

七劃

快取. 記憶體或磁碟空間的一部份，含有最近存取過的資料，其設計目的在於當再次存取相同資料時加快存取速度。快取通常是用來保留可透過網路存取之經常使用之資料的副本。

快取 (caching). 儲存對 Web 伺服器之要求的經常使用結果的處理，以供區域性快速取出，直到復新這些資訊時。

快取管理程式. 管理一部機器之快取的程式。它可管理多個快取。

防火牆 (firewall). 一個含有軟體的電腦，其負責保護內部的網路不讓未經授權的外來者存取。

九劃

持續性 (persistence). 保留整個異動的指定值的狀態，在此異動橫跨多個 Net.Data 呼叫。 僅有變數可以持續。此外，受到確定控制影響的資源上的作業將保持件用中，直到執行明確的確定或回轉爲止，或直到完成異動爲止。

相對路徑名稱 (relative path name). 不是以最高層次或 ″根″ 目錄開頭的路徑名稱。系統會假設路徑名稱是以處理的現行工作目錄作爲開頭。

十劃

純本文檔介面 (flat file interface). 一組 Net.Data 內建式函數，可讓您讀取與寫入純文字檔中的資料。

十一劃

埠 (port). 一個 16 位元數字，用來供 TCP/IP 和高階通訊協定或應用程式通信用。

現行工作目錄 (current working directory). 將從其中解析所有相對路徑名稱的處理的預設目錄。

現場連線 (Live Connection). 由「連線管理程式」與多個 cliette 組成的 Net.Data 元件。「現場連線」會管理資料庫與 Java 虛擬機器連線的重用。

異動 (transaction). 一個 Net.Data 呼叫。如果使用持續的 Net.Data，則異動可以橫跨多個 Net.Data 呼叫。

通用閘道介面 (CGI). 一種標準的方式，可讓 Web 伺服器藉此將控制傳給應用程式並收回資料。

連線管理程式 (Connection Manager). Net.Data 中的可執行檔 `dtwcm`，支援「現場連線」時需用到這個檔案。

十二劃

登記 (registry). 可以儲存及擷取字串的儲存庫。

絕對路徑 (absolute path). 物件的完整路徑名稱。絕對路徑名稱開始於最高層或 ″根″ 目錄 (以正斜線 (/) 或反斜線字元來識別)。

超文字標示語言 (hypertext markup language). 一種用來撰寫 Web 文件的標籤語言。

超本文轉送通信協定 (hypertext markup language). 用於 Web 伺服器與瀏覽器間的通訊協定。

十三劃

傳輸控制通信協定 / 網際網路通信協定 (Transmission Control Protocol / Internet Protocol). 一組支援區域及廣域網路之對等連接功能的通訊協定。

資料庫 (database). 集結了許多表格而成的集合，也可以是表格空間及索引空間的集合。

資料庫管理系統 (database management system, DBMS). 一個控制資料庫的建立、組織和修改以及存取儲存於其內之資料的軟體系統。

資料類型 (data type). 直欄與文字的屬性。

路徑 (path name). 告訴系統如何找出物件的位置。路徑名稱會表示爲目錄名稱的順序，且其後會跟著物件名稱。個別目錄與物件名稱是以正斜線 (/) 或反斜線 (\) 字元來區隔。

路徑 (path). 用來尋找檔案的搜尋路徑。

十四劃

語言環境 (language environment). 提供 Net.Data 巨集與外部資料來源 (如 DB2) 或程式設計語言 (如 Perl) 之存取的模組。

十五劃

確定控制 (commitment control). 建立處理內 Net.Data 執行的界限，在此資源上的作業是工作單元的一部份。

十七劃

應用程式設計介面 (application programming interface, API). 爲一功能性介面，由作業系統或由一個可分開訂購的授權程式所提供，讓您可以使用高階語言撰寫應用程式來使用作業系統或授權程式之特定資料或函數。 Net.Data 可支援下列具有專利的 Web 伺服器 API，可讓您可在 CGI 處理上提高執行效能：ICAPI, GWAPI, ISAPI 與 NSAPI。

A

API. 應用程式設計介面。Net.Data 支援三種 Web 伺服器 API，來改善透過 CGI 處理的執行效能。

applet. 一種包含在 HTML 頁面中的 Java 程式。Applet 是與可使用 Java 的瀏覽器（如 Netscape Navigator）一起使用，且是在當 HTML 頁面被處理時載入。

B

BLOB. 二進位大型物件。

C

CGI. 通用閘道介面。

cliette. Net.Data「現場連線」中長時間執行的處理，用來伺服來自 Web 伺服器的要求。「連線管理程式」會排程 cliette 處理來處置這些需求。

CLOB. 字元大型物件。

cookie. HTTP 伺服器傳送給 Web 瀏覽器的資訊封包，然後每次瀏覽器存取那個伺服器時再由瀏覽器送回。cookie 可含有伺服器選擇的任意資訊，且是用來維護無狀態 HTTP 異動之間的狀態。*Free Online Dictionary of Computing*

D

DATALINK. 一種 DB2 資料類型，它可以從資料庫啓用儲存在資料庫外的檔案的參照。

DBCLOB. 雙位元組大型物件。

DBMS. 資料庫管理系統。

Domino *Go* Web 伺服器. Lotus 公司及 IBM 提供的 Web 伺服器，它提供一般及安全連線。 ICAPI 與 GWAPI 是這個伺服器所提供的介面。

G

GWAPI. *Go* Web 伺服器 API。

H

HTML. 超文字標示語言。

HTTP. 超本文轉送通信協定。

I

ICAPI. Internet Connection API。 *請參閱*。

Internet. 一個國際公用的 TCP/IP 電腦網路。

Intranet. 一個位於公司防火牆內的 TCP/IP 網路。

ISAPI. Microsoft Internet Server API。

J

Java. 一種不依附作業系統之物件導向型程式設計語言，特別適用於 Internet 的應用程式。

L

LOB. 大型物件。

M

middleware. 介於應用程式及網路之間的軟體。它會管理從屬站應用程式與伺服器透過網路的交談。

N

NSAPI. Netscape API。

null (空值). 一個代表沒有資訊的特殊值。

P

Perl. 一種解譯過的程式設計語言。

T

TCP/IP. 傳輸控制通信協定 / 網際網路通信協定。

U

URL. 一致資源定址器 (Uniform resource locator)。

W

Web 伺服器 (Web server). 一部執行 HTTP 伺服器軟體 (如：Internet Connection) 的電腦。

索引

索引順序以中文字，英文字，及特殊符號之次序排列。

〔七劃〕

〔八劃〕

〔九劃〕

〔十劃〕

〔十一劃〕

〔十二劃〕

〔十三劃〕

〔十四劃〕

〔十五劃〕

〔十六劃〕

〔十七劃〕

〔十八劃〕

〔十九劃〕

〔二十三劃〕

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

U

W

讀者意見表

爲使本書盡善盡美，本公司極需您寶貴的意見；懇請您使用過後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號（∨）；我們會在下一版中，作適當修訂，謝謝您的合作!

評估項目	評　估　意　見		備　註
正確性	內容說明與實際程序是否符合	□是□否	
	參考書目是否正確	□是□否	
一致性	文句用語及風格，前後是否一致	□是□否	
	實際畫面訊息與本書所提之畫面訊息是否一致	□是□否	
完整性	是否遺漏您想知道的項目	□是□否	
	字句、章節是否有遺漏	□是□否	
術語使用	術語之使用是否恰當	□是□否	
	術語之使用，前後是否一致	□是□否	
可讀性	文句用語是否通順	□是□否	
	有否不知所云之處	□是□否	
內容說明	內容說明是否詳盡	□是□否	
	例題說明是否詳盡	□是□否	
排版方式	本書的形狀大小，版面安排是否方便使用	□是□否	
	字體大小，顏色編排，是否有助於閱讀	□是□否	
目錄索引	目錄內容之編排，是否便於查考	□是□否	
	索引語錄之排定，是否便於查考	□是□否	
	※評估意見爲 "否" 者，請於備註欄說明。		

其他：（篇幅不夠時，請另紙說明。）

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。

Printed in Singapore