



OS/400 版的 Net.Data 管理與程式 設計手冊



OS/400 版的 Net.Data 管理與程式 設計手冊

注意事項

請務必先詳讀第105頁的『附錄C. 注意事項』中的資訊，再使用此資訊及其所支援的產品。

第四版 (1998 年 12 月)

本版本適用於：

- IBM Operating System/400 (程式 5763-SS1), 版本 3 版次 2 修訂 0
- IBM Operating System/400 (程式 5716-SS1), 版本 3 版次 7 修訂 0
- IBM TCP/IP Connectivity Utilities for AS/400 (程式 5763-TC1), 版本 3 版次 2 修訂 0
- IBM TCP/IP Connectivity Utilities for AS/400 (程式 5716-TC1), 版本 3 版次 7 修訂 0
- IBM HTTP Server for AS/400 (程式 5769-DG1), 版本 4 版次 3 修訂 0

及所有後續的版本、版次與修訂，除非在新版中另有指示。

© Copyright International Business Machines Corporation 1997, 1998. All rights reserved.

目錄

序文	vii
關於 Net.Data	vii
關於本書	viii
誰應閱讀本書	viii
關於本書中的範例	viii
 第1章 簡介	1
何謂 Net.Data ?	1
為什麼使用 Net.Data ?	2
 第2章 架構 Net.Data	5
將 Net.Data 程式物件複製到您的 CGI-BIN 程式庫	5
架構 Web 伺服器	6
關於 Net.Data 起始設定檔	7
自行設定 Net.Data 起始設定檔	7
建立起始設定檔案	8
架構變數陳述式	9
路徑架構陳述式	11
環境架構陳述式	15
授與 Net.Data 存取物件的存取權	17
 第3章 維持資產的安全	19
使用防火牆	19
加密網路上的資料	21
使用身份驗證	22
使用授權	23
使用 Net.Data 機制	23
 第4章 呼叫 Net.Data	25
使用巨集檔來呼叫 Net.Data (巨集要求)	25
HTML 鏈結	27
HTML 套表	27
呼叫持續巨集	28
持續巨集語法	28
範例	29
 第5章 開發 Net.Data 巨集	31
Net.Data 巨集檔的結構	32
DEFINE 區塊	34
FUNCTION 區塊	34
HTML 區塊	35
Net.Data 巨集變數	37
變數範圍	37
定義變數	38

參照變數	40
變數類型	40
Net.Data 函數	47
定義使用者定義的函數	48
呼叫函數	53
呼叫 Net.Data 的內建函數	54
在巨集中建立網頁	58
HTML 區塊	58
報告區塊	59
巨集檔中的條件式邏輯和迴路	65
條件式邏輯	65
迴路結構	67
第6章 使用語言環境	69
REXX 語言環境	70
架構 REXX 語言環境	70
呼叫外部 REXX 程式	70
傳送參數	70
使用 SAY REXX 指令與 OS/400 V3R2 或 V3R7	71
安全	72
提高執行效能	72
REXX 語言環境範例	72
SQL 語言環境	73
架構 SQL 語言環境	73
在確定控制下執行 SQL 語言環境	74
管理遠端資料庫的資料庫連線	74
儲存程序	75
SQL 語言環境限制	80
安全	80
提高執行效能	81
SQL 語言環境範例	82
系統語言環境	84
架構系統語言環境	84
傳送參數	84
安全	85
提高執行效能	85
系統語言環境範例	85
第7章 透過持續巨集的異動管理	87
關於持續巨集	87
定義異動	88
啟動異動	88
在異動中設定巨集 HTML 區塊	89
終止異動	93
定義異動中變數的範圍	93
在異動中設定 COMMIT 與 ROLLBACK	94

持續巨集的範例	94
附錄A. 問題分析	99
附錄B. Net.Data 樣本巨集	101
附錄C. 注意事項	105
商標	106
名詞解釋	109
索引	111

序文

感謝您選取 Net.Data 版本 2，此為用來建立動態網頁的 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。

Net.Data 版本 2 可明顯地提高執行效能，並可讓您建置與開發 Internet 業務解決方案的新特性。

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 (這些系統包括可透過 DRDA 存取的 DB2 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 是巨集處理器，在 Web 伺服器上當成 middleware 來執行。您可撰寫 Net.Data 應用程式 (稱為巨集)，Net.Data 會予以解譯，以利用使用者輸入的自定內容、資料庫的現行狀態、舊有企業邏輯及巨集中所設定的其他因數，來建立動態網頁。

由 Netscape Navigator 或 Internet Explorer 等瀏覽器，以 URL (通用資源位置) 形式發出的要求，到達 Web 伺服器，然後將要求轉送給 Net.Data 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：

- 將企業邏輯封裝在以 C、C++、RPG、COBOL、JAVA 或 REXX 程式設計語言 (但不限於這些語言) 撰寫的應用程式內。
- 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 支援工業標準介面，如「超本文轉送通信協定 (HTTP)」及「通用閘道介面 (CGI)」。HTTP 是在瀏覽器與 Web 伺服器之間使用，而 CGI 是在 Web 伺服器與 Net.Data 之間使用。這個支援可讓您選取所要的瀏覽器或 Web 伺服器，以與 Net.Data 搭配使用。Net.Data 系列產品會在 OS/400、OS/390、Windows NT、AIX、OS/2、HP-UX、Sun Solaris 與 SCO 作業系統上提供類似的功能。

Net.Data 版本 2 也併入一些執行效能與功能增強：

- 執行效能增強包括：
 - 可透過 SQL 語言環境來呼叫 SQL 儲存程序
 - 透過 SQL 語言環境，從 SQL 儲存程序中取回一個或多個結果集合
 - 支援使用持續 Net.Data 進行跨多個 Net.Data 呼叫的異動

- 巨集語言增強功能包括：
 - 任意放置備註的能力
 - 巢狀 IF 區塊
 - WHILE 區塊
 - MACRO_FUNCTION 區塊
 - 整數比較
 - 多重報表區塊

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，以開發巨集。您可以學到如何使用 Net.Data 提供的語言環境，存取 DB2 資料庫以及使用 RPG, COBOL 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，但尚未上市的产品或特性。

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>

<http://www.as400.ibm.com/netdata>

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。作業系統、Net.Data 訊息及其他相關資訊，請參閱 *Net.Data* 參考手冊。

爲了瞭解本書中所提及的各種概念，您必須熟悉 Web 伺服器的工作方式，並了解簡單的 SQL 陳述式與 HTML 標籤 (包括 HTML 套表標籤)。此外，您應該熟悉 *Net.Data* 參考手冊中的資訊。

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。

本章描述 Net.Data 及使用它的理由。

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

何謂 Net.Data？

使用 Net.Data 巨集，您可以使用邏輯、變數、函數呼叫及報告產生工具。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) 的文字檔。Net.Data 會處理巨集檔案，來產生可被 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。可從區域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。

第2頁的圖1 舉例說明 Net.Data for OS/400 與 Web 伺服器之間的關係，以及與所支援之資料與程式設計語言環境的關係。

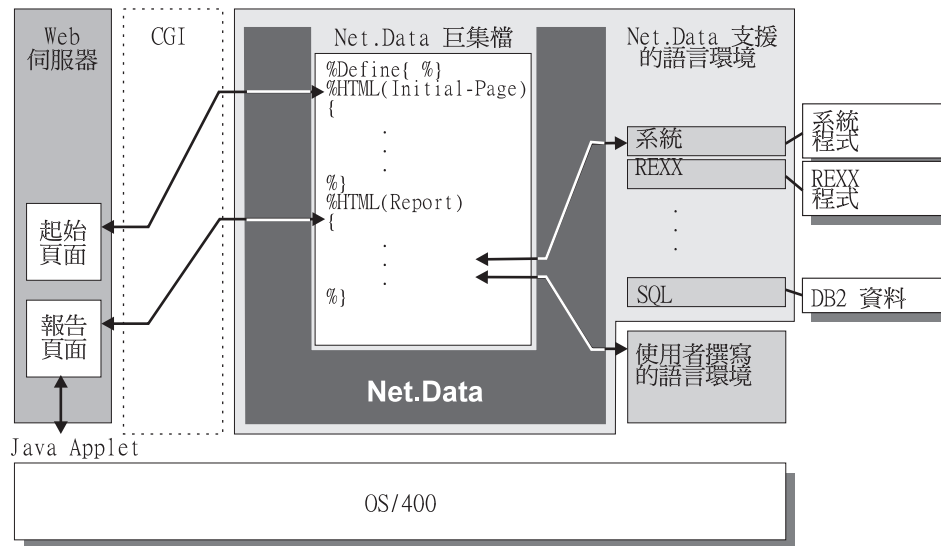


圖 1. Net.Data for OS/400、Web 伺服器及支援之資料與程式來源之間的關係

當 Web 伺服器收到請求 Net.Data 服務的要求時，Web 伺服器會呼叫 Net.Data 作為 CGI 應用程式。URL 包括 Net.Data 的特定資訊，包括將處理的巨集檔案。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，因為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、REXX 及 JavaScript）。Net.Data 也會提供存取 DB2 資料庫的語言環境，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檔案所做的變更。

Net.Data 會提供額外的能力來加強 OS/400 已強大的資料管理能力，這個能力會支援合併與網際網路資訊處理與呈現的未來格式。為了加惠您的應用程式，Net.Data：

- 提供資料建立與呈現的分隔方式。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。這容許使用者輕易地使用最新呈現趨勢來變更資料的呈現方式。
- 經由提供與 C, C++, RPG, COBOL, 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資料。
- 提供不需要會特定程式設計技巧便能開發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：

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。

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 是將所有資料全視為字串。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37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

遞送的 Net.Data for OS/400 包含下列幾種標準配備：

- IBM TCP/IP Connectivity Utilities/400 V3R2, V3R7, V4R1 與 V4R2
- IBM HTTP Server for AS/400 V4R3

不需額外購買；因為沒有 Net.Data 軟體需要您下載與安裝。

您需要的 AS/400 TCP/IP 與 HTTP Server 軟體是 OS/400 的標準配備，但您可選擇是否要安裝它。若為下列版本的 OS/400 作業系統，則下列可選用的軟體應安裝在您的系統上：

- IBM OS/400 作業系統版本 3 版次 2, 版本 3 版次 7，以及後續版本與版次 (57xx-SS1)：
 - IBM TCP/IP Connectivity Utilities/400 (57xx-TC1)
- IBM OS/400 作業系統版本 4 版次 3，以及後續版本與版次 (57xx-SS1)：
 - IBM HTTP Server for AS/400 (57xx-DG1)

安裝 Net.Data 後，請完成下列段落中所描述的步驟，來架構 Net.Data for OS/400。這些步驟包括：

- 『將 Net.Data 程式物件複製到您的 CGI-BIN 程式庫』
- 第6頁的『架構 Web 伺服器』
- 第7頁的『自行設定 Net.Data 起始設定檔』
- 第8頁的『建立起始設定檔案』
- 第17頁的『授與 Net.Data 存取的物件的存取權』

將 Net.Data 程式物件複製到您的 CGI-BIN 程式庫

使用 Net.Data 前，您必須先將 Net.Data 程式物件複製到 CGI-BIN 程式庫，並提供這個物件的存取權力。

欲複製 **Net.Data** 程式物件時：

1. 使用「建立重複物件 (CRTDUPOBJ)」命令，將 Net.Data 程式物件 (DB2WWW) 從 QTCP 程式庫複製到 CGI-BIN 程式庫。

OS/400 V4R3 使用者：使用程式庫 QHTTPSVR 中的程式物件；QTCP 程式庫中的程式物件會將 Net.Data 要求遞送到 QHTTPSVR 程式庫。

2. 變更 CGI-BIN 目錄中的 DB2WWW 程式物件，以便 CGI 程式執行時所依據的使用者設定檔具有程式物件的存取權。

依據預設值，*PUBLIC 使用者的 DB2WWW 程式物件權限會設定為 *EXCLUDE。若要提供程式物件的存取權，請將 *PUBLIC 使用者的程式物件權限變更為 *USE，或特別給與使用者設定檔 DB2WWW 程式物件的存取權。

您可以將 Net.Data 程式物件複製到多個程式庫中給不同應用程式使用。這容許您可具有多個版本的 Net.Data 起始設定檔案或多個保護計劃。有關 Net.Data 起始設定檔案的詳細資訊，請參閱第7頁的『自行設定 Net.Data 起始設定檔』；有關身份驗證的詳細資訊，請參閱第22頁的『使用身份驗證』。

若要將 Net.Data 程式物件複製到多個程式庫：

1. 使用上述所列的步驟，將 Net.Data 程式物件 (DB2WWW) 複製到程式庫中。
2. 使 Net.Data 程式物件與每一程式庫中的 CL 程式產生關聯。
 - a. 建立一個 CL 程式，來呼叫位於步驟 1 中所設定的程式庫中的 Net.Data 程式物件。
 - b. 將 CL 程式複製到每一程式庫中。

事實上，您建立的 CL 程式會變成 Net.Data 程式物件。如果未使程式物件與 CL 程式產生關聯，且將 Net.Data 程式物件 DB2WWW 複製到不同的程式庫中，則在使用 SQL 語言環境時，您將得到 -901 SQL 碼。

在下列段落中，如果您選擇建立 CL 程式來呼叫 Net.Data，則您建立的 CL 程式應視為 Net.Data 程式物件。

架構 Web 伺服器

通用開道介面 (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。

經由將指令新增到 HTTP 架構檔，以便呼叫 Net.Data，來架構 Web 伺服器呼叫 Net.Data。

例如，假定 Net.Data 程式物件常駐在程式庫 CGI 中，則下列指令會將 Net.Data 要求重新導向到 /QSYS.LIB/CGI.LIB/DB2WWW.PGM：

```
Map /cgi-bin/db2www/* /QSYS.LIB/CGI.LIB/DB2WWW.PGM/*
Map /CGI-BIN/DB2WWW/* /QSYS.LIB/CGI.LIB/DB2WWW.PGM/*
Exec /QSYS.LIB/CGI.LIB/*
```

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，因為指令會區分大小寫。在這個範例中，這兩個 Map 陳述式均指向同一位置。

Exec 指令將啟用 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)。

Net.Data 不會使用 Pass 指令。如果您想要簡化您的 URL，請在下面段落中討論的 Net.Data 起始設定檔案中，使用 MACRO_PATH 陳述式。

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。架構變數的設定值控制了多種 Net.Data 作業。

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，巨集及程式。

您可選擇是否要對 Net.Data for OS/400 建立 Net.Data 起始設定檔案。經由使用起始設定檔案，您可以使用較短的 URL 及較短的程式參照，以及在 Net.Data 巨集檔案中併入檔案。不過，如果您決定建立自己的語言環境，須具有一個起始設定檔案。

如果未建立一個起始設定檔案，Net.Data 在執行時，好似您已架構一個僅具有支援的語言環境陳述式的起始設定檔案 (請參閱 第69頁的『第6章 使用語言環境』，瞭解支援的語言環境)。在這種情況下，巨集檔案內的所有巨集、併入及可執行參照均須是完整的。

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：

- 第9頁的『架構變數陳述式』
- 第11頁的『路徑架構陳述式』
- 第15頁的『環境架構陳述式』

請參閱第8頁的『建立起始設定檔案』，學習如何建立起始設定檔案。

第8頁的圖2 中顯示的起始設定檔案範例，含有這些陳述式的例子。

1 DTWR_CLOSE_REGISTRIES YES	• 第 1 行設定架構變數的值
2 MACRO_PATH /WWW/MACRO;/QSYS.LIB/WWW.LIB/MACRO.FILE	• 第 2- 4 行定義 Net.Data 需要存取的檔案
3 INCLUDE_PATH /WWW/MACRO;/QSYS.LIB/WWW.LIB/MACRO.FILE	
4 EXEC_PATH /QSYS.LIB;/QSYS.LIB/WWW.LIB	• 第 5 - 7 行定義可用的環境陳述式。
5 ENVIRONMENT(DTW_REXX) /QSYS.LIB/QTCP.LIB/QTMRHREXX.SRVPGM ()	
6 ENVIRONMENT(DTW_SQL) /QSYS.LIB/QTCP.LIB/QTMSQL.SRVPGM (IN DATABASE, LOGIN, PASSWORD, TRANSACTION_SCOPE, SHOWSQL, DB_CASE, RPT_MAX_ROWS, START_ROW_NUM, DTW_SET_TOTAL_ROWS, OUT DTWTABLE, SQL_CODE, TOTAL_ROWS)	
7 ENVIRONMENT(DTW_SYSTEM) /QSYS.LIB/QTCP.LIB/QTMSYS.SRVPGM ()	

圖 2. Net.Data 起始設定檔

每一個別架構陳述式的文字均須在同一行上。(為了便於閱讀，ENVIRONMENT 陳述式會顯示在多行上。) 如果您不計劃在 Net.Data 巨集中呼叫任何語言環境，將不需要 ENVIRONMENT 陳述式。如果您已完整格定巨集檔案內的所有檔案參照，您也不必設定任何路徑架構陳述式。

下列段落描述如何建立起始設定檔案，以及如何自行設定 INI 檔中的架構陳述式。

建立起始設定檔案

當使用 Net.Data for OS/400 時，您可選擇是否要建立起始設定檔案。若為下列情況，您應該建立一個起始設定檔案：

- 您想要將任何 Net.Data 架構變數設定為非預設值。
- 您想要定義巨集、併入與可執行程式檔案的路徑陳述式，來縮短這些檔案的參照。
- 您將使用 Net.Data 不支援的語言環境。

建立起始設定檔案：

1. 在 DB2WWW 程式物件常駐的程式庫中，使用「建立來源實體檔 (CRTSRCPF)」命令，建立起始設定檔案。

檔名：	INI
成員名稱：	DB2WWW

建議您最好建立記錄長度為 240 的起始設定檔案，因為架構陳述式的文字均須在同一行上。

2. 使用「來源登錄公用程式 (SEU)」或工作站編輯程式，將架構陳述式新增到巨集範例與下列段落中所描述的檔案中。

如果您建立一個起始設定檔案，隨後又更新了它，則您不需終止或重新啟動 Web 伺服器，變更即會生效。在 HTTP 伺服器工作執行起始呼叫 Net.Data 期間，Net.Data 會讀取起始設

定檔案一次。架構資料會儲存起來，以便在後續的 Net.Data 呼叫上，Net.Data 不必讀取起始設定檔案。不過，如果對起始設定檔案做了變更，Net.Data 會偵測到已對起始設定檔案做了變更，然後會再新讀取起始設定檔案。

授權要訣：確定 Net.Data 執行時所依據的使用者 ID 具有此檔案的適當存取權。有關的詳細資訊，請參閱第17頁的『授與 Net.Data 存取的物件的存取權』。

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 Net.Data 提供的語言環境所使用的架構變數陳述式，您可以在起始設定檔案中設定這些陳述式：

- 『DTW_SQL_ISOLATION: DB2 隔離變數』
- 第10頁的『DTW_SQL_NAMING_MODE: SQL 表格命名變數』
- 第11頁的『DTWR_CLOSE_REGISTRIES: 開啓 Web 登記變數』

DTW_SQL_ISOLATION: DB2 隔離變數

DTW_SQL 語言環境使用 DTW_SQL_ISOLATION 架構陳述式，來決定 DTW_SQL 語言環境執行的資料庫作業與並行執行處理隔離的程度。

語法：

DTW_SQL_ISOLATION locking_method

其中 *locking_method* 是下列其中一值：

DTW_SQL_NO_COMMIT

設定不要使用確定控制。以 OS/400 作業系統而言，如果關聯式資料庫是設定在關聯式資料庫目錄中，且關聯式資料庫是在非 OS/400 系統上，則不要設定這個值。

DTW_SQL_READ_UNCOMMITTED

對於在 SQL ALTER、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、和 REVOKE 陳述式參照的物件，以及更新、刪除、和插入的列設定鎖定。這些物件一直鎖定到工作單元 (異動) 終止為止。可以看見其他處理中未確定的變更。

DTW_SQL_READ_COMMITTED

對於在 SQL ALTER、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、和 REVOKE 陳述式參照的物件，以及更新、刪除、和插入的列設定鎖定。這些物件一直鎖定到工作單元 (異動) 終止為止。已選取的列一直鎖定到選取下一列為止。不可以看見其他處理中未確定的變更。

DTW_SQL_REPEATABLE_READ

對於在 SQL ALTER、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、和 REVOKE 陳述式參照的物件，以及選取、更新、刪除、和插入的列設定鎖定。這些物件一直鎖定到工作單元 (異動) 終止為止。不可以看見其他處理中未確定的變更。

DTW_SQL_SERIALIZABLE

對於在 SQL ALTER、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、和 REVOKE 陳述式參照的物件，以及選取、更新、刪除、和插入的列設定鎖定。這些物件一直鎖定到工作單元 (異動) 終止為止。不可以看見其他處理中未確定的變更。在 SELECT、UPDATE、DELETE、和 INSERT 陳述式中參照的所有表格一直專用鎖定，直到工作單元 (異動) 終止為止。

DTW_SQL_NAMING_MODE: SQL 表格命名變數

DTW_SQL_NAMING_MODE 架構陳述式設定如何在 SQL 陳述式中設定表格名稱。

語法：

DTW_SQL_NAMING_MODE *mode*

其中 *mode* 是下列其中一值：

SQL_NAMING

設定表格由此格式的集合名稱來定義：

collection.table

其中 *collection* 是集合名稱，*table* 是表格名稱。預設定義子是執行處理的使用者 ID，此處理執行 SQL 陳述式，當表格名稱未明確定義且未設定預設集合名稱時，即使用此定義子。SQL_NAMING 是預設表格名稱。

SYS_NAMING

設定檔案是由此格式的常式庫名稱來定義：

library/file

其中 *library* 是程式庫名稱，*file* 是表格名稱。如果表格名稱 (檔案) 未明確定義且未設定預設集合名稱 (檔案庫) 的話，則預設搜尋路徑是不完整表格名稱的檔案庫列示 (*LIBL)。

DTWR_CLOSE_REGISTRIES: 開啓 Web 登記變數

設定要關閉或保持 Web 登記開啓。這個變數可讓您保持 Web 登記開啓，使存取相同「登記」的後續 Net.Data 巨集呼叫不必重新開啓登記。

語法：

```
DTWR_CLOSE_REGISTRIES YES|NO
```

其中：

YES 設定在已處理 Net.Data 巨集之後要關閉所有已開啓的 Web 登記。

NO 設定在已處理 Net.Data 巨集之後將所有已開啓的 Web 登記保持開啓。NO 是預設值。

執行效能要訣： 您可以使用 DTWR_CLOSE_REGISTRIES 架構陳述式來改進存取 Web 登記的執行效能 (使用 Web 登記內建式函數)，方法是把登記的開啓和關閉縮至最小。

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 巨集檔所使用之檔案和可執行程式的位置。路徑陳述式如下：

- 第12頁的『MACRO_PATH』
- 第13頁的『EXEC_PATH』
- 第14頁的『INCLUDE_PATH』
- 第15頁的『FFI_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：巨集檔、可執行檔、文字檔、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所有路徑陳述式。

- 每一個設定目錄都是以分號 (;) 區隔開來。
- 斜線 (/) 與反斜線 (\) 則視為一樣。
- 每一個路徑陳述式可以設定多重路徑。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。

要訣： Net.Data 搜尋所有設定的目錄，但不搜尋次目錄。例如，當 Net.Data 巨集存在於下列目錄中時，您必須在路徑陳述式中指出每一個次目錄：

```
/usr/test/client  
/usr/test/assoc  
/usr/test/partner
```

您的 MACRO_PATH 陳述式可能會像：

```
MACRO_PATH [=] /usr/test/client;usr/test/assoc;usr/test/partner
```

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。

MACRO_PATH

MACRO_PATH 架構陳述式定義 Net.Data 搜尋 Net.Data 巨集檔的目錄。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 找到巨集檔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25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">提出另一個查詢。</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 陳述式中設定的目錄中找不到檔案，則 Net.Data 將在根 (/) 目錄中搜尋檔案。例如，如果提出下列的 URL：

`http://myserver/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` 會嘗試在根 (`/`) 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。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，以及巨集檔中呼叫外部程式的 `EXEC` 陳述式。

`Net.Data` 起始設定檔：

```
EXEC_PATH = /qsys.lib/programs.lib;/qsys.lib/rexx.lib/rexxpgms.file;
```

`Net.Data` 巨集：

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

如果在 `/qsys.lib/rexx.lib/rexxprgms.file` 目錄中找到檔案 `myFunction.mbr`，則程式的完整名稱爲 `/qsys.lib/rexx.lib/rexxpgms.file/myFunction.mbr`。

如果在 `EXEC_PATH` 陳述式中設定的目錄中找不到檔案：

- 如果設定的路徑是絕對的，`Net.Data` 將在設定的路徑中搜尋檔案。例如，如果設定了下列 `EXEC` 陳述式：

```
%EXEC{/qsys.lib/programs.lib/rpg1.pgm %}
```

`Net.Data` 將在 `/qsys.lib/programs.lib` 目錄中搜尋檔案 `rpg1.pgm`。

- 如果設定的路徑是相對的，`Net.Data` 將搜尋現行工作目錄。例如，如果設定了下列 `EXEC` 陳述式：

```
%EXEC { rpg1.pgm %}
```

`Net.Data` 將嘗試在現行工作目錄中尋找檔案 `rpg1.pgm`。

INCLUDE_PATH

INCLUDE_PATH 架構陳述式定義 Net.Data 搜尋的一或多個目錄，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
```

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 陳述式以及用次目錄名稱定義的完整 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：

- 如果設定的路徑是絕對的，Net.Data 將在設定的路徑中搜尋檔案。例如，如果設定了下列 INCLUDE 陳述式：

```
%INCLUDE "/u/user1/includes/oeheader.inc"
```

Net.Data 將在 /u/user1/includes 目錄中搜尋檔案 *oeheader.inc*。

- 如果設定的路徑是相對的，Net.Data 將搜尋現行工作目錄。例如，如果設定了下列 INCLUDE 陳述式：


```
%INCLUDE "oeheader.inc"
```

Net.Data 將嘗試在現行工作目錄中尋找檔案 `oeheader.inc`。

FFI_PATH

FFI_PATH 架構陳述式定義 Net.Data 搜尋的一或多個目錄，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。

因為 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。

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69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* 語言環境參考手冊中加以描述。

在您可以呼叫語言環境之前，Net.Data 需要 ENVIRONMENT 陳述式存在於該語言環境的 Net.Data 起始設定檔中。

Net.Data 設定數個變數來影響 Net.Data 語言環境解譯函數呼叫的方式，這些函數定義於 FUNCTION 區塊。這些變數的設定必須傳遞至語言環境才有作用。

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*, ...)

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服務程式的名稱。在 OS/400，服務程式名稱是以 .SRVPGM 副檔名設定。

- *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。

參數在 *dtw_lei* 結構的 *parm_data_array* 欄位中傳遞，它們跟隨在 FUNCTION 區塊定義中設定的參數之後。(關於這些結構的資訊，請參閱 *Net.Data 語言環境參考手冊*。)

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為架構變數，或定義為巨集檔案中的變數。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。

當 Net.Data 處理 INI 檔時，它不會載入語言環境服務程式。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服務程式。只要載入 Net.Data，即會持續載入服務程式。

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，或為 Net.Data 巨集撰寫者在其巨集內所需設定或置換的。

```
1  MACRO_PATH    /WWW/MACRO;/QSYS.LIB/WWW.LIB/MACRO.FILE
2  INCLUDE_PATH  /WWW/MACRO;/QSYS.LIB/WWW.LIB/MACRO.FILE
3  EXEC_PATH     /QSYS.LIB;/QSYS.LIB/WWW.LIB

4  ENVIRONMENT(DTW_REXX) /QSYS.LIB/QTCP.LIB/QTMHREXX.SRVPGM ( )
5  ENVIRONMENT(DTW_SQL) /QSYS.LIB/QTCP.LIB/QTMSQL.SRVPGM (IN DATABASE,
    LOGIN, PASSWORD, TRANSACTION_SCOPE, SHOWSQL, DB_CASE,
    RPT_MAX_ROWS, START_ROW_NUM, DTW_SET_TOTAL_ROWS,
    OUT DTWTABLE, SQL_CODE, TOTAL_ROWS)
6  ENVIRONMENT(DTW_SYSTEM) /QSYS.LIB/QTCP.LIB/QTMSYS.SRVPGM ( )
```

每一個 ENVIRONMENT 陳述式必須由一行構成。

授與 Net.Data 存取物件的存取權

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 巨集中所參照的物件的適當存取權，以及具有 URL 參照的巨集的適當存取權。。

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 起始設定檔案 INI.FILE/DB2WWW.MBR
- 執行 Net.Data 可執行檔與服務程式，以及搜尋可執行檔與服務程式的路徑中的目錄（程式庫）
- 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

範例：

您需要授與 Net.Data CGI 執行程式時所依據的使用者設定檔的何種權限給 Net.Data 巨集，取決於您選擇來儲存您的 Net.Data 巨集的檔案系統而定。下列方法將給與 QTMHHTTP1 使用者設定檔權限（在 V3R2 與 V3R7 中，Internet Connection for AS/400 僅會在 QTMHHTTP1 使用者設定檔下執行 CGI 程式。）：

- 在根檔案系統中，使用「變更權限 (CHGAUT)」CL 命令，給與使用者設定檔的權限：

```
CHGAUT OBJ('/WWW') USER(QTMHHTTP1) DTAAUT(*RX)
CHGAUT OBJ('/WWW/macro') USER(QTMHHTTP1) DTAAUT(*RX)
CHGAUT OBJ('/WWW/macro/*') USER(QTMHHTTP1) DTAAUT(*RX)
```

您需要給與路徑中所有物件的權限。

- 在程式庫檔案系統 (QSYS.LIB) 中，使用「授與物件權限 (GRTOBJAUT)」CL 命令，給與使用者設定檔的權限：

```
GRTOBJAUT OBJ(WWW) OBJTYPE(*LIB) USER(QTMHHTTP1) AUT(*USE)
GRTOBJAUT OBJ(WWW/MACRO) OBJTYPE(*FILE) USER(QTMHHTTP1) AUT(*USE)
```

您僅需給與程式庫與來源實體檔的權限。

您可以按照下列，使用 CHGAUT CL 命令，給與 QSYS.LIB 檔案系統中物件的權限：

```
CHGAUT OBJ('/QSYS.LIB/WWW.LIB') USER(QTMHHTTP1) DTAAUT(*RX)
CHGAUT OBJ('/QSYS.LIB/WWW.LIB/MACRO.FILE') USER(QTMHHTTP1) DTAAUT(*RX)
```

特定的語言權限環境權限注意事項會在第69頁的『第6章 使用語言環境』中的每一語言環境段落中加以描述。

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 (為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21頁的『加密網路上的資料』
- 第22頁的『使用身份驗證』
- 第23頁的『使用授權』
- 第23頁的『使用 Net.Data 機制』

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20頁的圖3 會顯示這個架構。

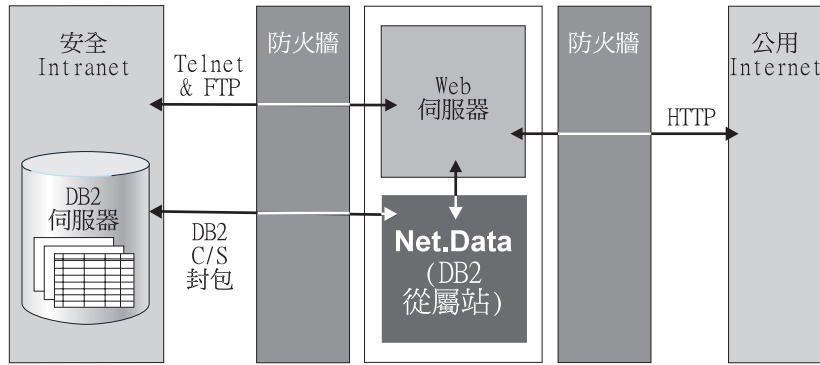


圖 3. 高安全架構

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 可經由架構防火牆，容許 DB2 資料通過防火牆，來存取 Intranet 內的 DB2 伺服器。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。第21頁的圖4 顯示這個架構。

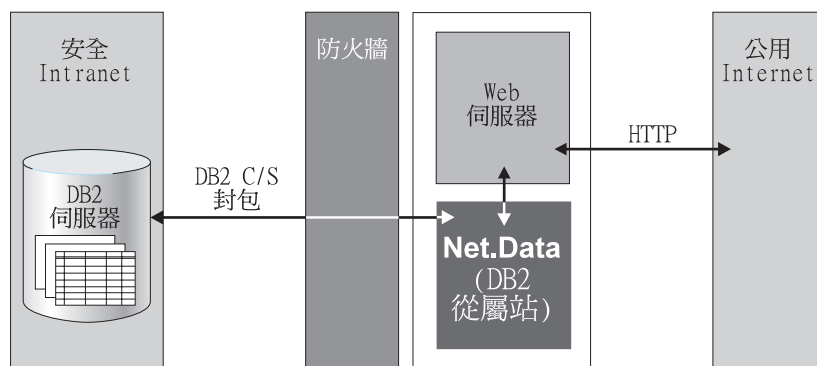


圖 4. 中安全架構：

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5 顯示這個架構。

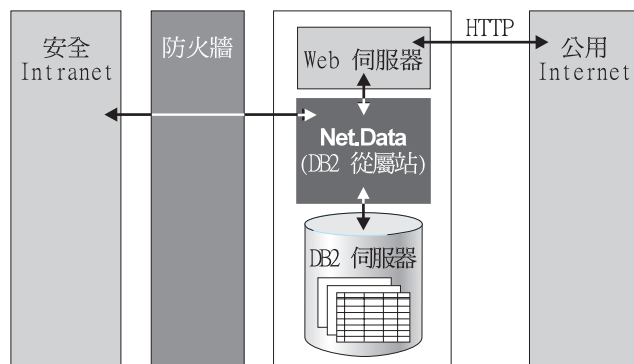


圖 5. 低安全架構：

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。

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。

在 OS/400 環境中，使用下列其中一種方法 (有三種方法)，使用者 ID 可與處理 Net.Data 要求的緒或處理產生關聯：

以從屬站為基礎的身份驗證

在從屬站上，會提示使用者輸入區域 OS/400 使用者 ID 和密碼。然後，Web 伺服器會鑑定使用者。如果鑑定順利，則提供的使用者 ID 會與要求產生關聯。特殊 Web 伺服器 `%%CLIENT%%` 存取控制使用者 ID 的使用，可啟用這類型的身份驗證。

從 OS/400 V4R1 開始，IBM 的 HTTP 伺服器支援以從屬站為主的身份驗證。

以伺服器為基礎的身份驗證

Web 伺服器的使用者 ID 會與每個要求產生關聯，且不會提示使用者輸入使用者 ID 或密碼。特殊 Web 伺服器 `%%SERVER%%` 存取控制使用者 ID 的使用，可啟用這類型的身份驗證。

依據預設值，IBM 的 HTTP 伺服器係在 QTMHHTTP1 使用者 ID (使用者設定檔) 下執行 CGI 程式。不過，如果 UserID 指令有效或是在已設定 UserID 次指令的保護設定之內，將在設定的使用者 ID 下執行程式。

代理身份驗證

有權存取某些預設資源之集成的代理使用者 ID，與從屬站要求有關。這類型的身份驗證需要建立含有存取權的代理使用者 ID，而此存取權是適用於使用者群組或要求類別。具有代理使用者 ID 的身份驗證通常會使用在 V4R1 中第一次引進的驗證列示。有關詳細資訊與範例，請參閱 *OS/400 System API Reference*。

架構 Web 伺服器時，會指定 Web 伺服器用來將使用者 ID 與從屬站要求產生關聯的方法。若需存取控制使用者 ID、安裝 Web 伺服器，與使用 Protect、Protection、DefProt 及 UserId 指令來架構 Web 伺服器的其它明細，請參閱 HTTP 伺服器的說明文件。

要訣：若要保護 Net.Data 巨集檔案，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17頁的『授與 Net.Data 存取的物件的存取權』。

及

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22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

除上述方法之外，您也可使用其他 Net.Data 提供的機制，如路徑陳述式及隱藏變數，以及使用 HTML 套表或 SQL 陳述式的方法。

路徑陳述式

Net.Data 會評估路徑架構陳述式的設定，以決定 Net.Data 巨集檔所用的檔案及可執程式位置。這些路徑陳述式會指定當嘗試尋找巨集檔案、可執行檔、純本文檔或併入檔時，Net.Data 將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。

隱藏變數

您可以使用隱藏變數來隱藏 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43頁的『隱藏變數』，以取得有關詳細資訊。

您也可以使用下列方法，設定保護計劃：

HTML 套表

使用 Net.Data 建立自己的保護體制。例如，您可以透過 HTML 套表向使用者要求驗證資訊，並使用資料庫中的資料或者透過由 Net.Data 巨集來呼叫外部程式，來驗證之。

SQL 陳述式

使用允許使用者傳送給資料庫的 SQL 陳述式，來保護您的資產；例如，將 SELECT 陳述式限制在兩個表格。

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

Net.Data for OS/400 是使用「通用閘道介面 (CGI)」及巨集檔所呼叫的。這類型的呼叫方法稱為巨集要求。

此外，您可以呼叫持續巨集，或含有設定異動界限的函數的巨集。有關持續巨集的詳細資訊，請參閱第87頁的『第7章 透過持續巨集的異動管理』。

Net.Data 巨集檔名稱及要在 Net.Data 巨集中執行的 HTML 區塊，都是指定在鏈結、套表或 URL 中。

這章說明如何使用巨集檔來呼叫 Net.Data。

- 『使用巨集檔來呼叫 Net.Data (巨集要求)』
- 第28頁的『呼叫持續巨集』

使用巨集檔來呼叫 Net.Data (巨集要求)

這節說明如何透過指定巨集檔來呼叫 Net.Data。欲取得呼叫的巨集要求樣式，您可以使用 HTML 鏈結、HTML 套表或 URL 來呼叫 Net.Data。

下列語法顯示可用來呼叫 Net.Data 的不同方式。

- HTML 鏈結：

```
<A HREF="http://server/Net.Data_invocation_path/filename/block/
[?name=val&...]">any text</A>
```

- HTML 套表：

```
<FORM METHOD=method
ACTION="http://server/Net.Data_invocation_path/
filename/block/[?name=val&...]">any text</FORM>
```

- URL：

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

參數：

server 指定 Web 伺服器的名稱。若伺服器為區域伺服器，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12頁的『MACRO_PATH』，以取得有關的詳細資訊。

block 指定所參考之 Net.Data 巨集檔中的 HTML 區塊名稱。

method 指定與此套表搭配使用的 HTML 方法。建議使用 METHOD=POST。

?name=val&...

指定要傳送給 Net.Data 的一或多個選用性參數。

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://MyServer/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</A>
```

範例 2：使用套表呼叫 Net.Data

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

範例 3：使用 HTML 鏈結呼叫在 qsys.lib 檔案系統中的 Net.Data：

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

範例 4：使用套表呼叫在 qsys.lib 檔案系統中的 Net.Data：

```
<FORM METHOD=POST
ACTION="http://MyServer/cgi-bin/db2www/
qsys.lib/mylib.lib/myfile.file/myMacro.mbr/report">
.
.
.
</FORM>
```

注意事項：URL 雖須是您的程式碼中的連續字串，但為了便於閱讀，它們將分成兩行。

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

您可在網頁中建立鏈結，以在巨集檔中使用 HTML <a> 標籤來執行 HTML 區塊。使用 HREF 屬性來指定巨集及 HTML 區塊，並將一些文字甚至影像包括在鏈結標籤中，就可完成這項作業。瀏覽器顯示網頁時，這個方法可將文字或影像定義成『熱點』。使用者在瀏覽器上按一下文字或影像時，Net.Data 會執行巨集中的 HTML 巨集。

下列範例顯示當使用者在網頁上選取文字 List all monitors 時，可執行 SQL 查詢的鏈結。

```
<a href="http://server/cgi-bin/db2www/listA.d2w/report">
List all monitors</a>
```

鏈結會呼叫下列的巨集：

```
%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59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/equip1st.d2w/report">
<P>您要查閱哪類硬體？
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="MON" checked> 監視器
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PNT"> 指標裝置
```

```

<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PRT"> 印表機
<LI><INPUT TYPE="RADIO" NAME="hardware" 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 equip1st.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='$(hardware)'
%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=\$(hardware) 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。

呼叫持續巨集

本段落將告訴您如何呼叫持續巨集。這些巨集含有用於異動處理的函數。呼叫這些巨集類似於一般巨集要求，在其中您要設定一個伺服器、巨集檔案及 HTML 區塊。對於持續巨集，您亦要設定一個異動 handle，將 HTML 區塊指明為異動的一部份。

有關持續巨集與異動處理的詳細資訊，請參閱第87頁的『第7章 透過持續巨集的異動管理』。

持續巨集語法

您可以使用下列語法，呼叫持續巨集：

- HTML 鏈結：

```
<A HREF="http://server/Net.Data_invocation_path/transaction_handle/filename/
block/[?name=val&...]">any text</A>
```

- HTML 套表：

```
<FORM METHOD=method
ACTION="http://server/Net.Data_invocation_path/
transaction_handle/filename/block/
[?name=val&...]">any text</FORM>
```

- URL：

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

參數：

server 指定 Web 伺服器的名稱。若伺服器為區域伺服器，您可以省略伺服器名稱，而使用相關的 URL。

Net.Data_invocation_path

Net.Data 可執行檔的路徑與檔名。例如，/cgi-bin/db2www/。

transaction_handle

設定將為 Net.Data 巨集所起始的異動一部份的 URL。可經由呼叫 DTW_RTVHANDLE 內建函數來取得這個識別字，且它須跟在 Net.Data_invocation_path 後。

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12頁的『MACRO_PATH』，以取得有關的詳細資訊。

block 指定所參考之 Net.Data 巨集檔中的 HTML 區塊名稱。

method 指定與此套表搭配使用的 HTML 方法。建議使用 METHOD=POST。

?name=val&...

指定要傳送給 Net.Data 的一或多個選用性參數。

範例

下列範例描述如何呼叫持續巨集。

範例 1：巨集檔案中的 URL：

```
http://www.mycompany.com/cgi-bin/db2www/$(handle)/mymacro.mac/report1
```

範例 2：具有在同一異動中執行的其他巨集呼叫的鏈結的典型 HTML 區塊

```
@DTW_STATIC()
...
%define handle = ""
@DTW_RTVHANDLE(handle)
```

```
%html(report) {  
@DTW_ACCEPT(handle)  
...  
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/  
  pcgil.mbr/report2">continue</a><br>  
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/  
  pcgil.mbr/quit">quit</a><br>  
%}
```


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檔案的函數
- 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6 中所顯示一般。

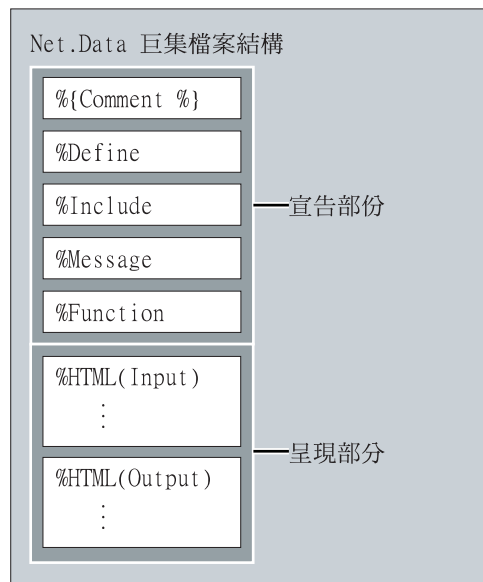


圖 6. 巨集檔結構

- 宣告部份包含巨集檔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。關於巨集檔部份和結構的語法，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17頁的『授與 Net.Data 存取的物件的存取權』。

本章找出組成 Net.Data 巨集檔的不同區塊，並說明撰寫巨集檔所用的方法。

- 『Net.Data 巨集檔的結構』
- 第37頁的『Net.Data 巨集變數』
- 第47頁的『Net.Data 函數』
- 第58頁的『在巨集中建立網頁』
- 第65頁的『巨集檔中的條件式邏輯和迴路』

Net.Data 巨集檔的結構

巨集檔案由兩個部份所組成：

- 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17頁的『授與 Net.Data 存取的物件的存取權』。

- 呈現部份定義網頁的配置、參照變數，以及使用作為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 區塊名稱作為進入點，來處理巨集檔案。HTML 區塊說明於第35頁的『HTML 區塊』。

在本節中，我們會舉一個簡單的 Net.Data 巨集為例，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為 OMPSAMP.MBR 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.mbr %}
}%

%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>@rex1(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，透過 Net.Data 來存取它。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。

當呼叫巨集檔時，Net.Data 會從頭開始處理之。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。備註可在巨集檔中的任意處。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為簡單。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，它們會定義一個對外部 REXX 程式的函數呼叫，以及一個對巨集檔內所含函數的函數呼叫。

```
%{ ***** FUNCTION 區塊 *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- 此函數接受
                                                         一個參數，並傳回一個
                                                         替代了相關函數的
                                                         呼叫
                                                         }
|
| %EXEC{ompsamp.mbr %} <-- 函數會執行一個名為
|                          "ompsamp.mbr" 的外部 REXX 程式
| %}
|
| %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`) 為一個 REXX 函數宣告，它會執行名為 `ompsamp.mbr` 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 命令。REXX 命令也會傳回一個稱為 `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.mbr` 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 或)，括住要求的 @rexsl() 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 (一整行) 都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，這是因為它們皆為解譯過的語言，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 區塊會作為巨集的進入與跳出點。HTML 區塊恆會設定在 Net.Data 呼叫要求中，且每一巨集至少須有一個 HTML 區塊。

範例巨集檔案中的第一個 HTML 區塊名為 INPUT。INPUT 區塊含有具有一個輸入欄位的簡單套表的 HTML。

```
%{ ***** HTML 區塊：輸入 *****%}
%HTML (INPUT) {
  <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"> <--- 當提出套表時，
                                     會呼叫 "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。您可以為其指定任何名稱。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。當提出格式後，即會將此欄位中的資料透過 input_data 傳遞給 HTML 輸出區塊。

只要巨集檔是位在同一 Web 伺服器上，您便可以使用相關參照方式，來存取其他巨集檔中的 HTML 區塊。例如，ACTION 參數 ACTION="../othermacro.d2w/main" 可存取巨集檔 othermacro.d2w 中稱為 main 的 HTML 區塊。同樣地，此套表中鍵入的任何資料，皆會以 input_data 變數傳送這個巨集。

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>
```

您不必像使用大部份的 CGI 程式一樣，需要定義環境變數才能接收輸入資料。Net.Data 會為您處理環境變數。您只需參照變數名稱即可。

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

```
%{ ***** 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>@rex1(input_data) <--- 此行包含呼叫函數 rex1，
                           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 INPUT 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 rex1，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。函數定義決定要傳遞的變數之數目與類型。

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 符記 (如變數名稱與值，以及文字字串) 最多可含有 256 KB 的資料。在 OS/400 方面，最大符記大小是由作業系統來決定。不同的語言環境對於值的大小，有額外的限制。

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，分為下列類型：

- 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：
 - 此類的變數雖非明確地定義，不過在首次參照時可被個案化。
 - 參數變數，為 FUNCTION 區塊定義的一部份，且僅能在 FUNCTION 區塊中。
 - 此類變數可由 Net.Data 個案化，且會對應到套表資料或 URL 資料 name-value 配對。
 - 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：

- 『變數範圍』
- 第38頁的『定義變數』
- 第40頁的『參照變數』
- 第40頁的『變數類型』

變數範圍

識別字 (變數或函數呼叫) 變成可見的，表示被宣告或被個案化時，即可被參考。可看見識別字的區域，即稱為其範圍。五種範圍類型如下：

- 廣域
 - 若您可在巨集檔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
 - URL 資料
 - 自 HTML 區塊中被個案化的變數
- 巨集檔

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 區塊乃被視為獨立的 DEFINE 陳述式。) 含有巨集檔範圍的識別字，從宣告處一直到巨集檔尾端，皆為可見的狀態。

- **FUNCTION 區塊或 MACRO_FUNCTION 區塊**

若為下列情況，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。

- 在函數區塊中製作識別字的案例，並且未在函數呼叫之前宣告或起始設定該識別字。

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。若識別字使用於函數區塊之內，則會保留在函數呼叫之前已指定的值。若在函數區塊之內更新，則在函數呼叫之後，識別字會保留新值。

- **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、...、V_n)，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。

識別字被參考到時，會被置換成其值。若對變數的參考沒有相關的值，或若函數呼叫沒有回覆值，則參照會被置換成空字串。

定義變數

定義 Net 巨集中變數的方法有下列 3 種：

- **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。這個語法是 Net.Data 專用的：

```
%DEFINE variable_name="variable value"
```

```
%DEFINE variable_name={ variable value on multiple  
                           lines of text %}
```

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或是底線符號。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。

請使用連續兩個雙引號來併入字串中的引號。只鍵入兩個連續的引號等於空字串。例如：


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

變數 HI 顯示 say "hello"。

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

變數 reply 顯示 hello,

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

變數 empty 是空值。

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 SELECT 和 INPUT 標籤**

您可以使用 HTML 套表所用的 SELECT 和 INPUT 標籤。下列範例使用標準 HTML 套表標籤來定義變數：

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

或

```
<SELECT NAME="variable_name">
```

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27頁的『HTML 套表』中的範例。

自 INPUT 或 SELECT 陳述式中接收的變數值，會置換掉 Net.Data 巨集中 DEFINE 陳述式所設定的變數值。

- **URL 資料**

您可以將 Net.Data 巨集當成 URL 來呼叫，並且在 URL 中併入變數 (例如使用者 ID) 來傳給 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 從輸入陳述式中接收的其它資料。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，它會以其值取代該變數參照。

若要使用變數作為文字呈現陳述式的一部份，請在您的 HTML 區塊中參照它們。例如，若您已定義變數 homeURL：

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

您可以使用 \$(homeURL) 來參照首頁並建立鏈結：

```
<A href="$(homeURL)">首頁</A>
```

您可以在 Net.Data 巨集的任何部份中參照變數。如果在參照變數時，尚未定義它們，Net.Data 將傳回空的字串。Net.Data 不會定義變數。

限制：不容許循環參照 (或迴路)。例如，下列的 DEFINE 陳述式，在參照變數及計算最終的值時，會出現錯誤：

```
%DEFINE a="$(b)"  
%DEFINE b="$(a)"
```

變數類型

您可以在巨集檔中使用下列類型的變數參照。

- 第41頁的『條件式變數』
- 第41頁的『環境變數』
- 第42頁的『執行變數』
- 第43頁的『隱藏變數』
- 第44頁的『列示變數』
- 第45頁的『表格變數』
- 第45頁的『雜項變數』
- 第46頁的『表格處理程序變數』
- 第46頁的『報告變數』
- 第47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與範例。

條件式變數

透過類似於 IF THEN 結構的方法，條件式變數可讓您為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。條件變數的語法為：

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

若 varB 已定義，則 varA="value_1"，否則為 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44頁的『列示變數』中的範例。

環境變數

您可以參照執行 *Net.Data* 之處理內的 *Net.Data* 環境變數。

定義環境變數的語法如下：

```
%define var=%ENVVAR
```

其中 *var* 是被定義的變數之名稱。

例如，變數 SERVER_NAME 可定義為環境變數：

```
%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
```

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

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* 中的語言結構這一章。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` 被定義為執行變數，而 `dateRpt` 被定義為變數參照，它包含執行變數。

```
%DEFINE date=%exec "date"  
%DEFINE dateRpt="今天是 $(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 在巨集檔中發現執行變數，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13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為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為報告的一部份：

```
%DEFINE database="ce1dial"  
%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，為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真正變數名稱已隱藏。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 為空值，因此查詢中的 whereClause 亦為空值。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。它裡面有一個相同記錄或橫列的陣列、以及一個說明各列欄位的直欄名稱陣列。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)
```

`TABLE` 後面的數字，是這個表格可以包含的列數。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，且沒有配置儲存體。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。

至於其他所有的語言環境，例如 `DTW_REXX` 或 `DTW_PERL`，語言環境不會自動地設定表格值。相反地，表格的個別元素可供原始語言直譯器作為輸出參數使用，然後必須由要執行的 `script` 或程式來設定。請參閱第69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。表格的個別元素，可以在函數的 `REPORT` 區塊中被加以參照。有關詳細資料，請參閱第46頁的『表格處理程序變數』。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。

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，而非以 0 起始陣列的標準 `C` 和 `C++` 語言慣例。

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檔內的變數參照，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：

<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 處理巨集檔的方式。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為了用於 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。

請使用表格處理程序變數，做為巨集檔內的變數參照。例如，若要取回被處理之現行直欄的名稱，您可以使用：

<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檔案所產生的網頁輸出。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` 設為 `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。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。請在適當的 `Net.Data` 巨集區塊中設定或參照語言環境變數。

透過這些變數，您可以執行的作業包括：建立資料庫的連線及啟用 `NLS` 支援。

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)
%}
```

Net.Data 函數

您可以在 `Net.Data` 巨集中使用兩種類型的函數：

使用者定義的函數 (UDF)

例如，您定義要與應用程式一起使用的那些函數會呼叫外部程式或儲存程序。

Net.Data 的內建函數

Net.Data 提供要在應用程式中使用的那些函數，如字組與字串操作函數，以及取得與設定表格變數函數的函數。

本段落描述下列主題：

- 『定義使用者定義的函數』
- 第53頁的『呼叫函數』
- 第54頁的『呼叫 Net.Data 的內建函數』

定義使用者定義的函數

若要在巨集檔案中定義自己的函數，請使用 `FUNCTION` 區塊或 `MACRO_FUNCTION` 區塊。

FUNCTION 區塊

定義可從 Net.Data 巨集來呼叫的次常式，該次常式可由語言環境來處理或呼叫外部程式。

MACRO_FUNCTION 區塊

定義可從 Net.Data 巨集來呼叫的次常式，該次常式必須由 Net.Data 來處理，而不是語言環境。區塊中的陳述式必須是 Net.Data 巨集語言原始陳述式。

`MACRO_FUNCTION` 區塊是可以改善執行效能的 `FUNCTION` 區塊的替代區塊。`MACRO_FUNCTION` 區塊僅是由 Net.Data 來處理，且不會呼叫語言環境。關於這兩個結構的詳細資訊在 *Net.Data* 參考手冊中可找到。

語法： 使用下列語法來定義函數：

FUNCTION 區塊：

```
%FUNCTION(type) function-name(usage 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。

Net.Data 會提供幾個預設的語言環境。

function-name

指定 FUNCTION 或 MACRO_FUNCTION 區塊的名稱。請以巨集檔案其他處的函數呼叫來執行 FUNCTION 區塊或 MACRO_FUNCTION 區塊。函數呼叫以前置的 at (@) 符號來參照 *function-name*。有關詳細資料，請參閱第53頁的『呼叫函數』。

您可以使用相同名稱來定義多重 FUNCTION 或 MACRO_FUNCTION 區塊，以便同時處理。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，會以定義於 Net.Data 巨集中的次序來執行。

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、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為止。預設類型為 IN。

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。參數變數參照在 FUNCTION 區塊或 MACRO_FUNCTION 區塊之外即失去效用。

return-var

請在 RETURNS 關鍵字後指定此參數，來識別特殊的 OUT 參數。在 Net.Data 巨集的處理上，指派給函數呼叫之 return 變數的值，可以替代函數呼叫。如果未設定 RETURNS 子句，函數呼叫的值為：

- NULL，若從呼叫到語言環境的回覆碼為零的話
- 回覆碼的值，若回覆碼為非零的話。

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。將可執行的陳述式傳遞到語言環境之前，Net.Data 將以實際值置換這些變數參照或函數呼叫。

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42頁的『執行變數』。

對於 MACRO_FUNCTION 區塊，可執行的陳述式是 HTML 陳述式和 Net.Data 巨集語言結構的組合。此情況下，不會呼叫任何語言環境，因為 Net.Data 扮演語言處理器的角色，且評估和執行可執行的陳述式。

report-block

定義一個或多個 REPORT 區塊，來處理 FUNCTION 區塊的輸出。請參閱第59頁的『報告區塊』。

message-block

定義 MESSAGE 區塊，用以處理 FUNCTION 所傳回的任何訊息。請參閱第51頁的『訊息區塊』。

請在最外層的 Net.Data 巨集上定義函數，供 Net.Data 巨集呼叫。

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，作為語法有效的內含程式碼 (如 REXX 或 Perl) 的一部份時，它們可能會被誤譯為 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，%{ 字元會被解譯為 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，會認為找到它。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，作為內含程式碼的一部份，不使它們被 Net.Data 解譯為特殊字元：

- 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 %{ 作為 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 %{ 字元解譯為 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 bynumber 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。Net.Data 使用下列訊息區塊處理：

1. Net.Data 會為呼叫 FUNCTION 區塊的每一個函數，設定一個語言環境變數 RETURN_CODE。對 MACRO_FUNCTION 區塊的函數呼叫，則不設定 RETURN_CODE。
2. 當語言環境將回覆碼傳回 Net.Data 時，Net.Data 會將 RETURN_CODE 的值設為回覆碼值。
3. 當函數呼叫完成後，MESSAGE 區塊會透過 RETURN_CODE 值來判斷該如何繼續處理。

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 區塊是否為 +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 不為 0，則檢查區域 MESSAGE 區塊是否為預設值，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 不為 0，則根據 RETURN_CODE 的符號，檢查廣域 MESSAGE 區塊是否為 +default 或 -default，然後依指定跳出或繼續。
6. 若 RETURN_CODE 不為 0，則檢查廣域 MESSAGE 區塊是否為 default，然後依指示跳出或繼續。
7. 若 RETURN_CODE 不為 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.mbr %}
    %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 值為 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 區塊中是否為 +default。
3. 檢查區域 MESSAGE 區塊中是否為 default。
4. 檢查廣域 MESSAGE 區塊中是否有完全相符的。
5. 檢查廣域 MESSAGE 區塊中是否為 +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。使用 at (@) 字元，其後跟著 FUNCTION 區塊名稱或 MACRO_FUNCTION 區塊名稱：

```
@function_name([ argument,... ])
```

function_name

這是要呼叫的 FUNCTION 區塊或 MACRO_FUNCTION 區塊的名稱。除非該函數為內建函數，否則此函數必須已定義在 Net.Data 巨集中。

argument

這是已定義的變數、文字字串、變數參照或函數呼叫的名稱。函數呼叫上的引數符合 FUNCTION 區塊或 MACRO_FUNCTION 區塊的參數，而在處理 FUNCTION 區塊或 MACRO_FUNCTION 區塊時，會指定對應的引數給每一個參數。引數的數目和類型，必須與相對應參數的相同。

Net.Data 以下列次序，來處理 FUNCTION 區塊、MACRO_FUNCTION 區塊、或與函數呼叫有關的內建式函數：

1. Net.Data 在 FUNCTION 區塊之可執行的陳述式區段內，處理變數參照和函數呼叫。Net.Data 使用變數的現行值來置換所有的變數參照，且以函數呼叫的回覆值來執行和置換所有函數呼叫。變數參照和函數呼叫是以被指定的次序來處理。Net.Data 在這個步驟中，不處理內建式函數或 MACRO_FUNCTION 區塊。
2. 原始語言處理器可處理可執行的陳述式區段。以 FUNCTION 區塊而言，處理器對應到於 FUNCTION 區塊上指定的語言環境，例如 SQL、REXX 或 Perl。以 MACRO_FUNCTION 區塊而言，Net.Data 扮演語言處理器的角色，且執行可執行的陳述式。內建式函數沒有可執行的陳述式。Net.Data 根據函數名稱來處理內建式函數。Net.Data 將函數的參數傳送至原始語言處理器。Net.Data 只將 IN 和 INPUT 參數的值傳遞給原始語言處理器，且只接受原始語言處理器傳回 OUT 和 INOUT 參數的值。
3. Net.Data 基於來自語言處理器的回覆碼與回覆訊息，來設定隱含的 RETURN_CODE 和 DTW_DEFAULT_MESSAGE 變數。Net.Data 在 MACRO_FUNCTION 區塊上不設定這些變數。
4. 如果 FUNCTION 區塊或 MACRO_FUNCTION 區塊含有一個或多個 REPORT 區塊，或設定要產生的預設報表，Net.Data 將使用任何參照的輸出參數的新值來處理報表。Net.Data 不會產生內建式函數的報告。
5. 若 FUNCTION 區塊包含區域 MESSAGE 區塊，則 Net.Data 會處理該 MESSAGE 區塊。下列其中一項條件發生時，Net.Data 會處理整體 MESSAGE 區塊：
 - 整體 MESSAGE 區塊已指定，且回覆碼不是由區域 MESSAGE 區塊來處理。
 - 內建函數被呼叫。Net.Data 不會處理 MACRO_FUNCTION 區塊的 MESSAGE 區塊。
6. Net.Data 會以函數的回覆值置換函數呼叫。以 FUNCTION 區塊而言，此值是下列其中一項：

RETURNS 參數值

置換含有 RETURNS 關鍵字的 FUNCTION 區塊。

空字符串 ("")

當 RETURN_CODE 為 0 時，置換不含 RETURNS 關鍵字的 FUNCTION 區塊。

RETURN CODE

當 RETRUN_CODE 不是 0 時，置換不含 RETURNS 關鍵字的 FUNCTION 區塊。

以 MACRO FUNCTION 區塊而言，處理可執行的陳述式區段之結果，會置換函數呼叫。

以內建式函數而言，該值根據內建函數的格式而定。

呼叫 Net.Data 的内建函數

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，您不必再用 FUNCTION 區塊來定義。您只要在巨集中可呼叫使用者定義函數的任何地方，呼叫這些函數即可。

您可以使用您用來呼叫使用者定義的函數 (Net.Data 函數呼叫) 的同一個方法，來呼叫內建函數。圖7 顯示 Net.Data 函數與巨集檔的互動方式。

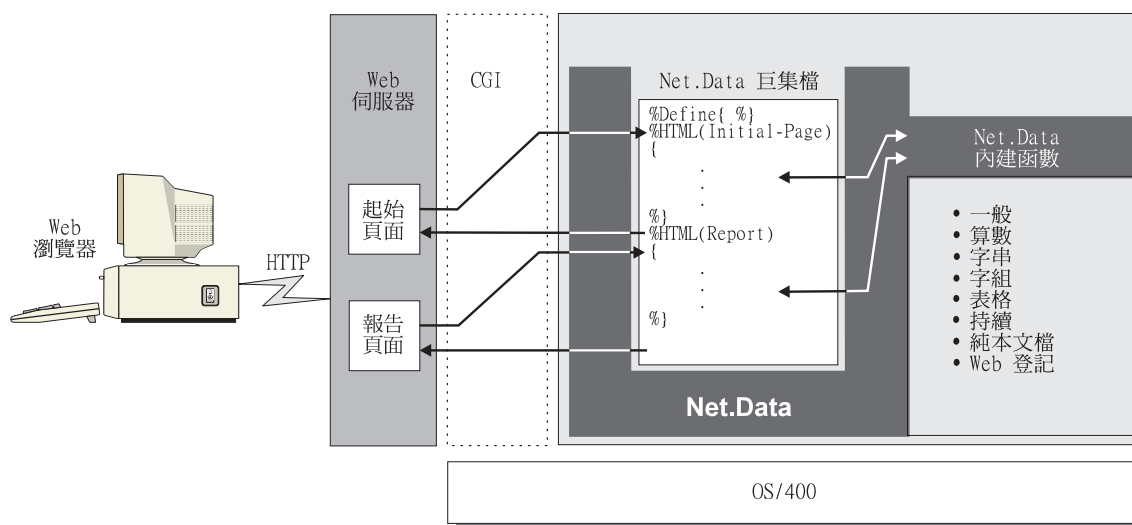


圖 7. *Net.Data* 內建函數

Net.Data 從 HTML 區塊內處理內建函數的函數呼叫。Net.Data 會處理函數，然後在巨集檔案中的 HTML 區塊內傳回任何結果。

內建函數可採三種方式傳回其結果。您可以從字首便可看出它是以何種方式傳回結果的：

- **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* 參考手冊中的 Net.Data 內建函數章節。

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。此外，您可以使用持續函數來進行異動處理。關於每一個函數的語法和範例說明，請參閱 *Net.Data* 參考手冊。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。

- 『一般目的函數』
- 第56頁的『算術函數』
- 第56頁的『字串函數』
- 第57頁的『字組函數』
- 第57頁的『表格函數』
- 第57頁的『純本文檔函數』
- 第57頁的『Web 登記函數』
- 第58頁的『持續函數』

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。您可以使用此組函數，來查詢與設定環境變數，來使用 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，不處理巨集檔案的其餘部份，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>
%}

```

DTW_URLESCSEQ 將會用它們的 `escape` 值置換 URL 中不容許的字元。例如，如果輸入變數 `string1` 等於 `"Guys & Dolls"`，則 DTW_URLESCSEQ 會將輸出變數指定為值 `"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_rPOWER("2", "-3", result)
```

DTW_POWER 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` 被指定為 `zero`。

```
@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相同，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元中的字元數的 DTW_WORDLENGTH 及傳回字串內字元位置的 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，來取得與設定 Net.Data 表格變數中的值。在表格變數中，乃含有一個陣列的值，以及其相關的直欄名稱。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_DUMP 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，以便另一個處理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，以容許其他處理存取檔案。

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_DELENTY 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。

持續函數

持續巨集函數會協助您定義哪些巨集區塊在單一異動內是持續的，來支援 Net.Data 中異動的處理。使用這些函數來定義異動的開頭與結尾、哪些 HTML 塊在整個異動中是持續的、異動內變數的範圍，以及是否要確定或取消異動內的變更。

例如，DTW_ACCEPT 會識別異動的異動 handle，而 DTW_TERMINATE 則會識別異動中的最後一個 HTML 區塊。DTW_RTVHANDLE 會對異動中的區塊建立唯一的異動 handle。在異動期間，您可以使用 DTW_COMMIT 與 DTW_ROLLBACK，來起始確定與取消。

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* 參考手冊中語言結構一章。

HTML 區塊

Net.Data 巨集檔包含 HTML 區塊及 HTML 區塊中的結構，這些結構可建立 Web 瀏覽器的文字呈現陳述式，如 HTML。在巨集檔案中，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，且 HTML 區塊的內容會控制 Net.Data 其餘的呼叫部份。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 HTML 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/equip1st.d2w/report">
<dl>
<dt>您要列出什麼硬體?
<dd><input type="radio" name="hardware" value="MON" checked> 監視器
<dd><input type="radio" name="hardware" value="PNT">指標裝置
<dd><input type="radio" name="hardware" value="PRT">印表機
<dd><input type="radio" name="hardware" 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=${hardware)
%REPORT{
<B>此為您要的列示:</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，Net.Data 會解析巨集檔。當 Net.Data 開始處理呼叫時指定的 HTML 區塊後，在本例中為 HTML(INPUT) 區塊，就會開始處理區塊中的文字。Net.Data 無法辨識為 Net.Data 巨集語言結構的任何事物都會被假設成 HTML 陳述式並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(REPORT) 區塊。之後，Net.Data 會以 HTML(INPUT) 區塊的方式來處理 HTML(REPORT) 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 \${hardware) 變數參照，被換成輸入格式中傳回的值後，Net.Data 即開始查詢。此時，Net.Data 會再度將 HTML 報表傳給瀏覽器，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(REPORT) 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 HTML 標籤、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。若您未指定表格名稱，則 Net.Data 使用 FUNCTION 區塊的參數列示中第一個輸出表格的套表資料。若您未在 FUNCTION 區塊中指定表格，則 Net.Data 使用預設套表資料。

REPORT 區塊是由三個可選用的部份所組成：

- 標題資訊，內含 HTML 資料，出現在表格各列資料前，僅顯示一次
- ROW 區塊，內含 HTML 和表格變數，出現在結果表格的各列中，僅顯示一次
- 註腳資訊，內含的資料乃出現在表格各列資料後，僅顯示一次。

範例：

```
%REPORT{
<H2>查詢結果</H2>
<P>選取名稱以顯示明細。
<TABLE BORDER=1>
<TR><TD>姓名</TD><TD>位置</TD>
%ROW{
  <TR>
  <TD>
  <a href="/cgi-bin/db2www/name.mac/details?name=$(V1)&location;=$(V2)">$(V1)</a></TD>
  <TD>$(V2)</TD>
  %}
  </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46頁的『表格處理程序變數』。而進一步的詳細資訊，請參閱 *Net.Data* 參考手冊中的「報告變數」一節。
- 若要提供標題和註腳資訊，請在 ROW 區塊的前後提供本文。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為巨集語言結構的一切，都視為文字呈現陳述式，並將這些陳述式傳給瀏覽器。
- 您可以在 REPORT 區塊中使用使用者定義的函數和變數。
- 若要讓 Net.Data 使用預先格式化本文來列印預設報告，請勿在巨集檔中併入 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本文，請設定 DTW_HTML_TABLE 為 YES。

- 若要停用列印預設報告，請設定 DTW_DEFAULT_REPORT 為 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。它自表格 CustomerTbl 中，顯示名稱、電話號碼、以及傳真號碼：

```
%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： $(NUM_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，分別提供「姓名」、「電話」及「傳真」的值。這個本文是 REPORT 區塊的註腳部份。
3. 在取回每一列之後，就繪製一條直線，以便閱讀。
4. 在報告尾端列印一次字串取回的記錄總數：和 NUM_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75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，您必須在函數參數列示中傳遞 Net.Data 表格變數。如果您在參數列示中傳遞的表格的數目比您設定的報表區塊數目還要多，將對與報表區塊沒有關聯的每一表格建立預設報表。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 (OUT table1, table2) {
    CALL myproc %}
```

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。因為未設定任何 REPORT 區塊，將顯示的這兩個表格為預設報表，首先 table1，然後 table2。

範例 2： 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。既然設定了 DTW_DEFAULT_REPORT="YES"，Net.Data 僅會顯示第一個表格的預設報表。

範例 3： MACRO_FUNCTION 區塊


```
%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。

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 (OUT table1, table2) {
    CALL myproc

    %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 (OUT table1, table2) {
    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。因為沒有在 REPORT 區塊上設定任何表格名稱，所以 Net.Data 會依據這兩個表格出現在 FUNCTION 區塊參數列示中的次序，來顯示它們的報表，首先 table1，然後 table2。

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) {
  %EXEC{ /qsys.lib/mylib.lib/mypgm.pgm %}
  %REPORT(table2) {
    ...
    %row { .... %}
    ...
  %}
%}
```

在這個範例中，僅設定了一個 **REPORT** 區塊，且因為它設定了表格名稱 **table2**，所以它會使用此表格來顯示它的報表。因為設定的 **REPORT** 區塊的數目少於 **FUNCTION** 參數列示中所傳遞的表格的數目，所以將依據其餘表格出現在 **FUNCTION** 區塊參數列示中的次序，來顯示它們的報表；首先顯示 **table1** 的預設報表，然後 **table3** 的預設報表。

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：

- 每一表格設定一個 **REPORT** 區塊。對 **REPORT** 區塊設定的表格名稱須符合 **FUNCTION** 區塊參數列示上的對應表格名稱參數。
- 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。請注意，設定 **DTW_DEFAULT_REPORT** = "YES" 將造成在未設定 **REPORT** 區塊時，僅設定一個表格的預設報表。在 **SQL** 語言環境中會有例外狀況發生，在此 **YES** 值將造成同於 **MULTIPLE** 的處理。
- 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為 **DTW_SAVE_TABLE_IN** 表格。
- 多個報表區塊可以搭配 **Net.Data** for **OS/400** 支援的任何語言環境一起使用。

限制：

- 報表變數是針對整個函數所設定的，會影響所有 **REPORT** 區塊的處理程序，及它們處理的表格。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。
- 表格變數在使用於函數之前，必須以 **TABLE** 陳述式來定義。

巨集檔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。

- 『條件式邏輯』
- 第67頁的『迴路結構』

條件式邏輯

在 Net.Data 巨集中，請使用 IF 區塊來執行條件式處理程序。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* 中語言結構一章。

IF 區塊語法的規則，是由區塊在巨集檔中的位置所決定。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
```

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
```

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 區塊
...
%IF 區塊
```

例外狀況：當 IF 區塊位於 REPORT 區塊之內時，請勿在 IF 區塊中設定 ROW 區塊。

Net.Data 基於構成條件的詞彙內容所產生的方式之一，來處理 IF 區塊條件列示。預設動作是將所有詞彙視為字串，並依條件所示執行字串比較。然而，若下列二條件符合，則 Net.Data 會執行數字比較：

- 若條件為二進位運算 (<, >, <=, >=, !=, ==)。
- 若條件中的兩個詞彙皆代表整數。這表示詞彙是數位字串，有時會前置 '+' 或 '-' 字元。字串不能包含 '+' 或 '-' 以外的任何非數字字元。

有效字串的範例：

```
+1234567890
-47
000812
92000
```

無效字串的範例：

```
- 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 區塊時評估該區塊，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。

範例：含有其他區塊內的 IF 區塊的巨集檔案

```
%{ 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)
    %}
%}

```

限制：Net.Data 不支援非整數數字的數值比較。

迴路結構

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* 參考手冊中語言結構一章。

Net.Data 處理 **WHILE** 區塊的方式，與處理 **IF** 區塊的方式完全相同，但會每次會透過迴路來重新評估條件。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** 區塊的巨集檔案

```

%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，而且必須當作服務程式來實施。您必須將 ENVIRONMENT 陳述式新增到 Net.Data 起始設定檔案中，方可使服務程式與您撰寫的語言環境產生關聯。當 Net.Data 第一次遇到設定語言環境名稱之 FUNCTION 區塊的函數呼叫時，Net.Data 會載入並執行服務程式一次。設定同一語言環境名稱之 FUNCTION 區塊的後續函數呼叫，只會使 Net.Data 執行已載入的服務程式。

圖8 顯示 Web 伺服器、Net.Data 及 Net.Data 語言環境之間的關係。

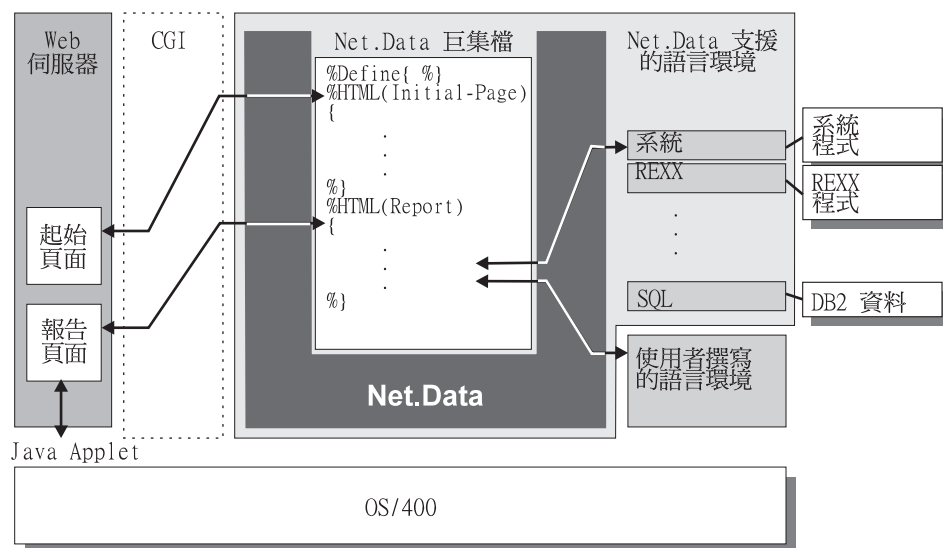


圖 8. Net.Data 語言環境

關於如何建立使用者撰寫的語言環境的詳細資訊，請參閱Net.Data 語言環境參考手冊。

下列段落描述 Net.Data 所包括的每一語言環境，包括如何架構及使用語言環境。

- 第70頁的『REXX 語言環境』
- 第73頁的『SQL 語言環境』
- 第84頁的『系統語言環境』

REXX 語言環境

REXX 語言環境可解譯 Net.Data 巨集 FUNCTION 區塊中設定的內部 REXX 程式，也可以執行儲存於個別檔案中的外部 REXX 程式。

架構 REXX 語言環境

請使用下列步驟來架構 REXX 語言環境。如果您選擇不建立 Net.Data 起始設定檔案，依據預設值將啟用 REXX 語言環境。沒有額外的架構需要進行。

1. 如果您已建立一個起始設定檔案，且您想要使用 REXX 語言環境，請將下列架構陳述式新增到起始設定檔案中：

```
ENVIRONMENT(DTW_REXX)/QSYS.LIB/QTCP.LIB/QTMHREXX.SRVPGM ( )
```

關於 Net.Data 起始設定檔案 (包括環境架構陳述式) 的詳細資訊，請參閱第5頁的『第2章 架構 Net.Data』。

2. 確定外部 REXX 程式常駐在 QSYS.LIB 檔案系統中。

呼叫外部 REXX 程式

若要呼叫外部 REXX 程式，請使用具有下列格式的 FUNCTION 區塊：

```
%EXEC{ REXX-file-name [可選用的參數] %}
```

例如：

```
%FUNCTION(DTW_REXX) rexx1() {  
%EXEC{ /QSYS.LIB/REXX.LIB/REXXSRC.FILE/TREXX.MBR %}  
%}
```

傳送參數

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{  
/QSYS.LIB/NETDATA.LIB/QREXXSRC.FILE/CALL1.MBR ${INPARM1}  
%}  
%}
```

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。使用這種方法傳遞的參數被視為輸入類型參數 (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反映回 Net.Data)。

間接

經由 REXX 程式變數儲存池方式間接傳遞參數。當啟動 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。這個空間稱為變數儲存池。

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 inpl, OUT outpl){
    outpl = 2*inpl
}%

%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，作為 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，直欄標題為 `myTable_N.j`，而欄位值為 `myTable_N.i.j`，其中 *i* 是橫列號碼，*j* 為直欄號碼。表格中的橫列數目為 `myTable_ROWS`，表格中的直欄數目則為 `myTable_COLS`。

使用 SAY REXX 指令與 OS/400 V3R2 或 V3R7

如果您將執行 OS/400 V3R2 或 V3R7，且 REXX 程式將使用 SAY REXX 指令，將資料寫入到 `stdout`，請在字串開頭插入 12 個空格。

例如：

```
SAY '          STARTOFDATA'
```

不處理 12 個空格，但若未插入，可能會發生無法預期的結果。

安全

確定 Net.Data 執行時所依據的使用者 ID 具有含有外部 REXX 程式的檔案的讀取權及寫入權，以及 REXX 程式使用的任何物件的讀取權及寫入權。

提高執行效能

請使用下列要訣，改善 Net.Data 應用程式的執行效能：

- 限制結合的 REXX 程式呼叫的 REXX 程式的數目。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，因為 REXX 直譯器恆會在新的啟動群組中執行。每次 Net.Data 呼叫直譯器來執行 REXX 程式時，即須啟動直譯器。
- 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。
- 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。

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
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]

假定 Web 巨集儲存在程式庫 NETDATA, 檔案 REXXMAC 與成員 REXX1 中, 則會經由從瀏覽器中載入下列 URL 來參照巨集:

```
http://hostname/cgi-bin/db2www/qsys.lib/netdata.lib/rexxmac.file/
rex1.mbr/report
```

SQL 語言環境

您可以使用 SQL 語言環境, 透過 DB2 來執行 SQL 陳述式。只要 SQL 陳述式對 DB2 for OS/400 命令有效, 則任何有效的 SQL 陳述式可傳遞到 SQL 語言環境。

架構 SQL 語言環境

請使用下列步驟來架構 SQL 語言環境。

1. 在關聯式資料庫目錄中建立本端資料庫的目錄登錄 (亦即, 具有遠端位置 *LOCAL 的目錄登錄), 以及建立 SQL 語言環境將存取的任何遠端資料庫。經由使用「新增關聯式資料庫目錄登錄 (ADDRDBDIRE)」命令來新增登錄。

如果您選擇不建立 Net.Data 起始設定檔案, 依據預設值將啟用 SQL 語言環境。沒有必需的額外架構。

2. 修改 Net.Data 起始設定檔案。

- a. 如果您已建立一個起始設定檔案, 且您想要使用 SQL 語言環境, 請在起始設定檔案中新增下列架構陳述式: 此環境陳述式的文字均須在起始設定檔案中的同一行上。為了便於閱讀, 在此會分成好幾行來顯示它。

```
ENVIRONMENT(DTW_SQL) /QSYS.LIB/QTCP.LIB/QTMSQL.SRVPGM ( )
(IN DATABASE, LOGIN, PASSWORD, TRANSACTION_SCOPE, SHOWSQL,
DTW_SET_TOTAL_ROWS, DB_CASE, START_ROW_NUM, RPT_MAX_ROWS,
OUT DTWTABLE, SQL_CODE, TOTAL_ROWS)
```

您不需要在環境陳述式中設定上面設定的所有變數。如果您未使用如 DTW_SET_TOTAL_ROWS 與 TOTAL_ROWS 的變數, 您可以從環境陳述式中除去它。另一種將這些變數傳遞到 SQL 語言環境的方式, 在您的巨集中的 Net.Data 函數呼叫上傳變數。關於語言環境變數的資訊, 請參閱 *Net.Data 參考手冊*。

- b. 新增或更新架構變數。SQL 語言環境支援下列可在 第74頁的表1 中顯示的 Net.Data 起始設定檔案中設定的架構變數:

表 1. SQL 語言環境架構變數

架構變數	說明
DTW_SQL_ISOLATION	決定 SQL 語言環境執行的資料庫作業與並行執行處理隔離的程度。可能值如下： DTW_SQL_NO_COMMIT DTW_SQL_READ_UNCOMMITTED DTW_SQL_READ_COMMITTED DTW_SQL_REPEATABLE_READ DTW_SQL_SERIALIZABLE 預設值為 DTW_SQL_READ_UNCOMMITTED。
DTW_SQL_NAMING_MODE	決定如何在 SQL 陳述式中設定表格名稱。可能值如下： SQL_NAMING SYS_NAMING 預設值為 SQL_NAMING。

關於 Net.Data 起始設定檔案 (包括環境架構陳述式與架構變數陳述式) 的詳細資訊，請參閱第5頁的『第2章 架構 Net.Data』。

在確定控制下執行 SQL 語言環境

依據預設值，SQL 語言環境會在確定控制下執行，並遵循所有管理確定控制的規則。

- 登載所有透過 DTW_SQL 存取的檔案或表格，但當 SQL 陳述式為 SELECT 時除外。
如果您想存取集合中的 SQL 表格，將透過 AS/400 上的本機 SQL 支援，自動替您登載它，且不需要任何明確的動作。
- 您可經由在 Net.Data 起始設定檔案中設定 DTW_SQL_ISOLATION，選擇是否要變更確定層次。請參閱第73頁的『架構 SQL 語言環境』，取得關於 SQL 語言環境支援的隔離層次的詳細資訊。

管理遠端資料庫的資料庫連線

您一次最多可連接到 50 個本端或遠端資料庫。只要 Net.Data 執行時所依據的 HTTP 伺服器處理一直進行，則 SQL 語言環境將保持連線中。在起始連線到資料庫後，保持連線將提供快速的資料庫存取。您可以經由將下列事項列入考慮中，來避免錯誤及改善執行效能：

- 確定依據異動範圍設定來規劃資料庫連線：
Net.Data 不容許同一遠端資料庫有並行連線。如果已存在使用某個使用者 ID (LOGIN SQL 語言環境參數) 的遠端資料庫連線，第二個使用者 ID 要求連線到同一個遠端資料庫，則 SQL 語言環境首先將中斷舊有連線，執行一個確定 (若使用確定控制的話)，然後使用 '新' 使用者 ID 與密碼來重新建立連線。需要確定，因為如果連線中斷，萬一稍後在巨集中發生錯誤，將沒有方法來進行取消。請依據下列規則來規劃您的連線：

- 如果 TRANSACTION_SCOPE=SINGLE，則在存取了遠端資料庫後，您可以變更登入 ID。SQL 語言環境會中斷舊有連線，執行確定，並使用新的使用者 ID 和通行碼重新建立連線。
- 如果 TRANSACTION_SCOPE=MULTIPLE (這是預設值)，則在存取了遠端資料庫後，不要變更登入 ID。SQL 語言環境會自動取消且會傳回 -752 的 SQL_CODE，指出連線可能已變更。
- 經由減少連線到同一遠端資料庫的使用者 ID 數目，來改善執行效能。
當 SQL 語言環境建立遠端系統連線時，它會使使用者 ID 與連線產生關聯。如果在後續的 Net.Data 查詢上使用使用者 ID 不符合與連線有關聯的那些 ID，將終止連線並建立新的資料庫連線 (如果異動範圍為 SINGLE，才會發生這種情況)。
若要改善執行效能，在向遠端資料庫發出 SQL 陳述式時，撰寫使用者 ID 的程式碼，或使用同一使用者 ID。對於本端資料庫，將不處理使用者 ID 和通行碼。

儲存程序

儲存程序是一種已編譯的程式，儲存於 DB2 本端或遠端伺服器上，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 SQL 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。

本段落說明下列主題：

- 『儲存程序語法』
- 第76頁的『呼叫儲存程序』
- 第77頁的『傳送參數』
- 第78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  
[%REPORT(resultsetname...)]
```

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2 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

指與透過報表區塊從儲存程序傳回的結果集合有關聯的名稱。這個參數須符合 *tablename* 的關聯參數名稱，若設定的話。

表 2. 儲存程序的資料類型

CHAR	FLOAT	TIME
DATE	GRAPHIC	TIMESTAMP
DECIMAL	INTEGER	VARCHAR
DOUBLE	REAL	VARGRAPHIC
DOUBLEPRECISION	SMALLINT	

呼叫儲存程序

欲呼叫儲存程序時：

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，包括從儲存程序傳回的任何結果集合的結果集合名稱。

```
%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 {  
...  
%}
```

範例：

```
%function (dtw_sql) mystoredproc (IN CHAR(30)  
arg1 OUT mytable) {  
    CALL myproc  
    %report {  
        ...  
    }
```

```

%row { ... %}
...
%}
%}

```

5. 如果儲存程序將建立多個結果集合：

- 將結果集合設定為 FUNCTION 陳述式的 OUT 參數。結果集合會儲存為區域表格。

```

%function (dtw_sql) function_name (OUT tablename, ...)

```

- 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, table2) {
    CALL myproc
    %report (table1) {
        ...
        %row { ... %}
        ...
    %}
    %report (table1) {
        ...
        %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。您可以經由以 IN 參數設定參數，將參數傳遞給儲存程序。如果儲存程序將更新參數，則您須以 INOUT 或 OUT 關鍵字，傳遞回覆值的參數。針對參數設定的資料類型須符合儲存程序所期望的資料類型。

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** 區塊顯示它們。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 **FUNCTION** 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，使結果顯示在 **Net.Data** 的預設報表樣式中。
- 使結果集合與 **REPORT** 區塊產生關聯，讓 **Net.Data** 在報表中顯示結果集合。然後，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。

結果集合恆會儲存在區域表格中，以便另一個函數稍後可以在巨集中使用資料。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64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，並使用預設報表顯示它：

可使用下列語法：

```
%function (dtw_sql) function_name
(OUT tablename) {
    CALL stored_procedure
}%
```

例如：

```
%function (dtw_sql) mystoredproc(OUT mytable1) {
    CALL myproc
}%
```

若要傳回單一結果集合並設定一個 **REPORT 區塊進行顯示處理：**

可使用下列語法：

```
%function (dtw_sql) function_name
(OUT tablename) {
    CALL stored_procedure
    %REPORT (resultsetname) {
        ...
    }
}%
```

例如：


```
%function (dtw_sql) mystoredproc (OUT mytable1) {
    CALL myproc
    %REPORT (mytable1) {
        ...
        %row { ... }
        ...
    }
}
```

另一種方式即是使用下列語法：

```
%function (dtw_sql) function_name () {
    CALL stored_procedure

    %REPORT () {
        ...
    }
}
```

例如：

```
%function (dtw_sql) mystoredproc () {
    CALL myproc
    %REPORT {
        ...
        %row { ... }
        ...
    }
}
```

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
(OUT tablename1, tablename2) {
    CALL stored_procedure
}
```

其中沒有報表區塊。

例如：

```
%define DTW_DEFAULT_REPORT = "MULTIPLE"
%function (dtw_sql) mystoredproc (OUT mytable1, mytable2) {
    CALL myproc
}
```

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自己的 **REPORT** 區塊產生關聯。可使用下列語法：

```
%function (dtw_sql) function_name (OUT tablename1, tablename2) {
    CALL stored_procedure
    %REPORT (resultsetname1)
```

```

        ...
        %row { ... %}
        ...
    %}
    %REPORT (resultsetname2)
        ...
        %row { ... %}
        ...
    %}
    %}

```

例如：

```

%function (dtw_sql) mystoredproc (OUT mytable1, mytable2) {
    CALL myproc

    %REPORT(mytable1) {
        ...
        %row { ... %}
        ...
    %}

    %REPORT(mytable2) {
        ...
        %row { ... %}
        ...
    %}
    %}

```

SQL 語言環境限制

當規劃您的環境時，請考慮下列限制：

- 若為下列情況，請不要使用 SQL 語言環境：
 - 建立使用者定義的語言環境，它會使用資料庫存取類別程式庫或 SQL 呼叫層次介面
 - 以及在巨集中參照使用者定義的語言環境
- 不要將 SQL 陳述式傳遞給 EXEC 陳述式上的 SQL 語言環境。

安全

確定 Net.Data 執行時所依據的使用者 ID 具有 HTTP 伺服器常駐的機器上的資料庫的存取權，包括資料庫檔案與異動日誌（若確定控制在作用中的話）。

當存取遠端資料庫時，Net.Data 變數 LOGIN（使用者 ID）與 PASSWORD 係用來決定可以存取哪些資料庫資源。當存取 HTTP 伺服器常駐的機器上的資料庫時，將不處理這些變數，因為物件的存取權是由 Net.Data 執行時所依據的使用者設定檔來決定。

提高執行效能

若要瞭解 DB2 執行效能的注意事項，請參閱 *DB2 for AS/400 SQL Programming Guide*。本書具有豐富的資訊，如有效率地使用 SQL 索引、改善結合查詢的執行效能，以及從兩個以上的表格中選取資料時如何改善執行效能。在下列段落中，將重點描述資料庫與 SQL 語言環境的特定技術：

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：

- 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 for AS/400 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 for AS/400 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（因為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 for AS/400 將不會使用索引。

例如，EMPNO 為 CHAR(6)，DEPTNO 為 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 for AS/400 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。

- 減少連線到資料庫的使用者 ID 數目，來避免重新連到資料庫。SQL 語言環境會使使用者設定檔及密碼與它建立的資料庫的任何遠端連線產生關聯。如果 LOGIN 與 PASSWORD 變數不符合與已開啓的連線有關聯的使用者設定檔與密碼，將關閉連線並重新建立連線，而且 LOGIN 與 PASSWORD 值將與重新開啓的連線產生關聯。
- 使用 START_ROW_NUM 與 RPT_MAX_ROWS Net.Data 變數，減少傳回的表格的大小。在 SELECT SQL 陳述式上，若結果集合含有數百筆記錄，將經由使用類似可捲動的游標的 START_ROW_NUM 與 RPT_MAX_ROWS，把結果集合的次集傳回到瀏覽器，來限制傳回的記錄的數目。您應該明白 Net.Data 每次將重新發出查詢，因為沒有狀態的概念。不過，您可以使用持續巨集的 Net.Data 支援，將結果集合儲存在 Net.Data 表格中，使得它在異動期間保持不變。請參閱第87頁的『第7章 透過持續巨集的異動管理』，學習更多關於持續 Net.Data 巨集的資訊。
- 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。因為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。

請注意，從 OS/400 V4R2 開始，SQL 引擎有一個已備妥的陳述式快取記憶體。使用快取記憶體，SQL 引擎會將關於已備妥的陳述式的資訊儲存在別處，並將此資訊保存在全系統的儲存體中。然後，當再次執行相同陳述式時，即使在不同的使用者與不同工作下，陳述式的執行速度將更快。全系統的備妥陳述式快取記憶體為正常 SQL 處理的一部份，不需要任何使用者動作，即可架構或啓用它。快取記憶體可能會減少靜態 SQL 蓋過動態 SQL 的任何執行效能好處。

SQL 語言環境範例

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 語言環境會將每一參數中的字串值轉換為正確的內部格式，並依據 SQL 儲存程序的參照傳遞每一參數。當 SQL 儲存程序完成處理時，更新的內部呈現將轉換為字串並置於對應的參數中。

```
%{*****}
/*  DEFINE BLOCK
/*****%}
DEFINE {
  MACRO_NAME      = "TEST ALL TYPES"
  DTW_HTML_TABLE  = "YES"
  Procedure       = "NDLIB.TESTTYPE"
  parm1           = "1"                %{SMALLINT      %}
  parm2           = "11"               %{INT          %}
  parm3           = "1.1"              %{DECIMAL (2,1) %}
  %{
%FUNCTION(DTW_SQL) CRTPROC(){
  CREATE PROCEDURE $(Procedure)
  ( INOUT SMALLINT,
    INOUT INT,
    INOUT DECIMAL(2,1))
```

```

EXTERNAL NAME $(Procedure) LANGUAGE C SIMPLE CALL
%MESSAGE{
    default : "$(DTW_DEFAULT_MESSAGE) : continuing.<br>": continue
}%
}%

%FUNCTION(DTW_SQL)    myProc
    (INOUT SMALLINT    parm1,
     INOUT INT          parm2,
     INOUT DECIMAL(2,1) parm3){
CALL $(Procedure)
}%

```

```

%HTML(REPORT){
<HEAD>
<TITLE>Net.Data : SQL 儲存程序：範例 '$(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>
}%

```

假定 Web 巨集儲存在程式庫 NETDATA, 檔案 SQLMAC 與成員 SQL1 中, 則會經由從瀏覽器中載入下列 URL 來參照巨集:

```
http://hostname/cgi-bin/db2www/qsys.lib/netdata.lib/sqlmac.file/  
sql1.mbr/report
```

系統語言環境

「系統」語言環境支援外部程式的呼叫, 如 RPG、COBOL 與 C, 以及支援執行 FUNCTION 區塊的 EXEC 陳述式中所指定的 CL 命令。「系統」語言環境會透過下列方法來解譯 EXEC 陳述式: 將設定的程式名稱或命令與參數傳遞給作業系統, 以便使用 C 語言 system() 函數呼叫來執行它。

架構系統語言環境

如果您選擇不建立 Net.Data 起始設定檔案, 依據預設值將啟用「系統」語言環境。沒有必需的額外架構。

如果您已建立一個起始設定檔案, 且您想要使用「系統」語言環境, 請在起始設定檔案中新增下列架構陳述式:

```
ENVIRONMENT(DTW_SYSTEM) /QSYS.LIB/QTCP.LIB/QTMSYS.SRVPGM ( )
```

關於 Net.Data 起始設定檔案 (包括環境架構陳述式) 的詳細資訊, 請參閱第5頁的『第2章 架構 Net.Data』。

傳送參數

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 INPARAM1 = "SWITCH1"  
  
%function(DTW_SYSTEM) sys1() {  
  %EXEC{  
    /QSYS.LIB/NETDATA.LIB/RPGCALL1.PGM  
    ('$(INPARAM1)' 'LITERALSTRING')  
  }  
}
```

Net.Data 變數 INPARAM1 會被解除參照, 並傳遞到程式。參數傳遞到程式的方式將同於從「命令登錄」顯示畫面中呼叫程式時參數傳遞到程式的方式。使用這種方法傳遞的參數被視為輸入類型參數 (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

使用環境變數間接傳遞參數。環境變數即是儲存在程式外的環境空間中的套表 "name=value"的字串。這些字串會儲存在與處理有關聯的暫時空間中。

當 Net.Data 呼叫 DTW_SYSTEM 語言環境函數時，在執行 %EXEC 區塊內的陳述式之前，語言環境會先將輸入 (IN) 或輸入/輸出 (INOUT) 的任何函數參數儲存在環境空間中。在順利完成陳述式後，DTW_SYSTEM 語言環境便會判斷是否有任何輸出 (OUT 或 INOUT) 函數參數。若有，語言環境則會從環境空間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 取得控制時，它會輪流以從 DTW_SYSTEM 語言環境中取得的新值，更新所有 OUT 或 INOUT 參數。

使用 表3 中描述的 API 設定與取回環境變數：

表 3. 環境變數 API

ILE 程式設計語言	若要取回，請使用...	若要設定，請使用...
C, C++	getenv()	putenv()
CL(1), RPG, COBOL	QtmhGetEnv()(2)	QtmhPutEnv()(3)

- 1. 對於 OS/400 V3R7，您也可以使用 CHGENVVAR 與 ADDENVVAR CL 命令，來設定環境變數。
- 2. QtmhGetEnv() 會隨附於 IBM TCP/IP Connectivity Utilities/400，作為它的一部份。
- 3. QtmhPutEnv() 原先不會隨附於 IBM TCP/IP ConnectivityUtilities/400 for V3R2 與 V3R7，作為它的一部份。是在後期產品修正時加入的，可透過 V3R2 PTF 5763TC1-SF40953 或 V3R7 PTF 5716TC1-SF40954 取得它。

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，直欄標題為 myTable_N_j，而欄位值為 myTable_V_i_j，其中 i 是橫列號碼，j 為直欄號碼。表格的橫列與直欄數目為 myTable_ROWS 與 myTable_COLS。

不建議您傳遞具有多列的表格，因為處理的環境變數的數目受到限制。

安全

確定 Net.Data 執行時所依據的使用者 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。

提高執行效能

直接將僅輸入參數傳遞到「系統」語言環境經由定義廣域 Net.Data 變數及參照變數來呼叫的程式。

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 為輸入 (IN) 參數，而 P2 與 P3 則為輸出 (OUT) 參數。函數呼叫程式 UPDPGM，以 P1 的值更新參

數 P2，並將 P3 設定為字串。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 {  
        /QSYS.LIB/NETDATA.LIB/UPDPGM.PGM  
    %}  
%}  
  
%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}"。  
%}
```

假定 Web 巨集儲存在程式庫 NETDATA，檔案 SYSMAC 與成員 SYS1 中，則會經由從瀏覽器中載入下列 URL 來參照巨集：

```
http://hostname/cgi-bin/db2www/qsys.lib/netdata.lib/sysmac.file/  
sys1.mbr/upd1
```


第7章 透過持續巨集的異動管理

Net.Data 提供透過持續巨集的異動處理的支援。持續巨集是一種含有內建函數的巨集，這些函數可讓巨集當作 Web 伺服器中的持續 CGI 處理一部份來執行。這表示巨集的多個區塊或多個巨集可當作單一邏輯異動的一部份來執行。

透過非持續的巨集，Net.Data 會將每一個巨集呼叫視為一個完整的異動。這表示在每一個回應傳送到瀏覽器後，將確定資料庫、釋放資源，以及將任何事物設定為起始狀態。下次呼叫同一個巨集時，將依據傳遞給巨集作為套表資料的資訊或巨集本身的資訊，重新建立應用程式的狀態。無法儲存跨呼叫的巨集變數、當不能明確還原所做的變更時將無法取消資料庫變更，以及無法將跨多個瀏覽器階段作業的資料庫變更視為一個完整的異動。

透過持續巨集，身為應用程式開發者可以建立異動層次的應用程式，如此當維持一個持續連線時可呼叫一個或多個巨集。這表示變數資料在跨呼叫間是持續的，以便您不再需要在當作隱藏變數的巨集呼叫之間傳遞資訊 (如使用者登入 ID)。這包括在非持續巨集中無法跨呼叫傳遞的 Net.Data 表格變數。最重要的是，在異動期間，如果使用者決定取消，應用程式可以取消所有工作。

請參閱第28頁的『呼叫持續巨集』，學習如何呼叫持續巨集。

本章描述下列主題：

- 『關於持續巨集』
- 第88頁的『定義異動』
- 第94頁的『持續巨集的範例』

關於持續巨集

當使用持續巨集時，Net.Data 將在 Web 伺服器的特殊持續 CGI 處理中執行、收到透過標準輸入與環境變數的輸入，以及透過標準輸出提供資料。不過，在輸出傳回到 Web 伺服器之後，Web 伺服器不必終止 Net.Data 處理。相反地，處理仍會保持作用中，等待使用者透過 Web 瀏覽器傳送的回應。因為處理不會終止，Net.Data 便可以維護巨集的狀態資訊，並讓異動開啓。

Net.Data 會經由傳送一個新的 HTTP 表頭給伺服器，與 Web 伺服器通信，使這個伺服器在持續的 CGI 處理中執行。新表頭 (『Accept-HTSession』) 的支援已新增到版本 4、版次 3 (V4R3) 的 AS/400 HTTP Server 中。當 Net.Data 第一次傳送它的輸出時，Net.Data 會決定哪些 HTTP 表頭要傳送到伺服器，因為表頭須在輸出之前。當您開發持續巨集時，這會提供下列暗示：

- 從巨集建立第一個輸出時，Net.Data 必須知道這個巨集是否為持續巨集。
- 使用新的持續巨集的內建函數，您必須在建立任何輸出之前，先設定巨集是持續的。

這些限制將會在下列的文件中提到。

持續 Net.Data 處理的性質非常類似於具有下列例外狀況的標準 Net.Data 處理的那些性質。

- 它們係在假連線導向的環境中執行。Net.Data 與 Web 伺服器之間的連線是持續的，但瀏覽器與 Web 伺服器之間仍沒有連線。
- 它們可具有長時間執行的異動。因為單一 Net.Data 處理可以橫跨多個瀏覽器要求，所以可留下開啓的異動，並依據後續的瀏覽器要求或錯誤狀況，於必要時確定或取消這些異動。
- 持續 Net.Data 處理可以消耗更多的系統資源，因為它可以保持相當長時間的作用中。在管理那些資源時要特別小心。
- 可移轉性將減少，因為 Web 伺服器須含有持續性的支援。

定義異動

一個異動可以橫跨一個 HTML 區塊、多個 HTML 區塊或多個巨集。當您設定想要巨集在異動內是持續的時候，您需要定義異動的開頭與結尾，以及哪些 HTML 區塊將包括在異動中。Net.Data 會提供內建函數，來協助您完成下列持續巨集作業：

- 『啓動異動』
- 第89頁的『在異動中設定巨集 HTML 區塊』
- 第93頁的『終止異動』
- 第93頁的『定義異動中變數的範圍』
- 第94頁的『在異動中設定 COMMIT 與 ROLLBACK』

啓動異動

您可以在任何輸出傳送到瀏覽器之前，經由向 Net.Data 指出在您的巨集中巨集是持續的，來啓動一個異動。然後，Net.Data 會傳送一個特殊 HTTP 表頭給 Web 伺服器，告訴它巨集需要持續 CGI 支援。

啓動異動：

在任何輸出傳送到 Web 瀏覽器之前，您可以在巨集中使用下列其中一種方法：

- 呼叫 DTW_STATIC() 內建函數。

DTW_STATIC() 函數告訴 Net.Data 現行巨集是持續的。

語法： @DTW_STATIC (["*timeout*"])

其中 *timeout* 是可選用的參數，它設定在結束異動之前，Web 伺服器應等待來自瀏覽器的回應的秒數。

範例：

```

@DTW_STATIC("60")
%DEFINE {
    var1 = "val1"
    var2 = "val2"
}%
...

%HTML(input){
    ...
}%

%HTML(report){
    ...
}%

```

會對這個異動設定逾時值 60 秒。如果在 60 秒內未收到瀏覽器的回應，Web 伺服器將結束異動。這不會影響瀏覽器上的現行頁面。不過，將為異動一部份的下一頁面現在將是新異動的一部份。

- 定義具有 `STATIC` 屬性的變數。

語法： `%DEFINE(STATIC) var1 = "val1"`

範例：

```

%DEFINE(STATIC) var1 = "val1"
%DEFINE var2 = "val2"
...
%HTML(input){
    ...
}%
%HTML(report){
    ...
}%

```

在整個異動期間，靜態定義的變數會保留它的值，這個異動可以橫跨多個 `Net.Data` 呼叫。

在異動中設定巨集 HTML 區塊

您可以經由在呼叫 HTML 區塊的 URL 要求中使用名為異動 *handle* 的識別字，定義哪些 HTML 區塊是異動的一部份。定義與使用異動 *handle* 有三個步驟：

1. 在您的巨集中定義異動 *handle*。
2. 呼叫 `DTW_ACCEPT` 內建函數，將 *handle* 名稱傳遞給 `Net.Data` 與 Web 伺服器。
3. 在 URL 要求中設定 *handle* 來呼叫您的下一個 HTML 區塊。

定義異動 *handle*：

1. 在 `DEFINE` 區段中定義異動 *handle* 的變數。例如：

```
%DEFINE handle=""
```

2. 您可以經由在 DEFINE 區段中設定 DTW_RTVHANDLE() 內建函數，選擇是否要建立唯一的異動 handle。

語法：@DTW_RTVHANDLE(*handle_name*)

範例：

```
@DTW_STATIC()

%DEFINE handle = ""
@DTW_RTVHANDLE(handle)
```

異動 handle 可以是任何有效的字串。不過，DTW_RTVHANDLE() 函數會經由建立唯一異動 handle，提供一個安全措施，阻止其他函數呼叫將在您的異動中執行的巨集。

設定 *Net.Data* 的異動 *handle*：

以 DTW_ACCEPT() 內建函數設定 Net.Data 的異動 handle 的值。因為這個 handle 是傳送到伺服器的 HTTP 表頭中所含資訊的一部份，所以在巨集建立任何輸出之前，須先呼叫 DTW_ACCEPT() 函數。一般而言，它將是您的 HTML 區塊中的第一個元素。

語法：@DTW_ACCEPT(*handle_name*, ["*timeout*"])

其中 *timeout* 是可選用的參數，它設定在結束異動之前，Web 伺服器應等待來自瀏覽器的回應的秒數。

您可以在 HTML 區塊內或任何 HTML 區塊內呼叫 DTW_ACCEPT()。如果在任何 HTML 區塊外呼叫函數，異動 handle 與可選用的逾時值將適用於巨集內的所有 HTML 區塊。

範例 1：設定將在這個異動中執行的後續 URL 要求的異動 handle

```
@DTW_STATIC()

%DEFINE handle = ""
@DTW_RTVHANDLE(handle)

%HTML(Block1){
@DTW_ACCEPT(handle)
...
%}
```

重要事項：當您呼叫 DTW_ACCEPT() 作為 HTML 區塊中的第一個元素，請確定在設定 %HTML 陳述式的那一行上沒有空格，且 DTW_ACCEPT() 會呼叫本身。Net.Data 會將空格視為要傳送到瀏覽器的文字，並發出一個錯誤，因為在資料傳送到瀏覽器之前，找不到 DTW_ACCEPT() 呼叫。

範例 2：設定將適用於巨集中所有 HTML 區塊的異動 handle

```

@DTW_STATIC()

%DEFINE handle = ""
@DTW_RTVHANDLE(handle)

@DTW_ACCEPT(handle)

%HTML(Block1){
...
%}

%HTML(Block2){
...
%}

```

設定當您呼叫 **HTML** 區塊時要在您的異動中執行的 **handle**：

在您建立了異動 **handle** 並呼叫了 **DTW_ACCEPT()** 函數後，僅有具有該異動 **handle** 的 URL 才能在您的異動中執行。異動 **handle** 須緊跟在 URL 中的 CGI 程式名稱之後。

注意事項： 下列每一 URL 須設定為您的程式碼中的連續字串；不過，為了便於閱讀，在此會將它們分開。

- HTML 鏈結：

```
<A HREF="http://server/Net.Data_invocation_path/transaction_handle/
filename/block/[?name=val&...]">any text</A>
```

- HTML 套表：

```
<FORM METHOD=method
ACTION="http://server/Net.Data_invocation_path/transaction_handle/
filename/block/[?name=val&...]">any text</FORM>
```

- URL：

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

參數：

server 指定 Web 伺服器的名稱。若伺服器為區域伺服器，您可以省略伺服器名稱，而使用相關的 URL。

Net.Data_invocation_path

Net.Data 可執行檔的路徑與檔名。例如，/cgi-bin/db2www/。

transaction_handle

設定將為 Net.Data 巨集所起始的異動一部份的 URL。可經由呼叫 **DTW_RTVHANDLE** 內建函數來取得識別字，且須跟在 **Net.Data_invocation_path** 後。

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12頁的『MACRO_PATH』，以取得有關的詳細資訊。

block 指定所參考之 Net.Data 巨集檔中的 HTML 區塊名稱。

method 指定與此套表搭配使用的 HTML 方法。建議使用 METHOD=POST。

?name=val&...

指定要傳送給 Net.Data 的一或多個選用性參數。

一般而言，您將提供這些 URL 的 HTML 鏈結，或在您的巨集中的套表動作標籤上設定 URL。

範例 1：具有在同一異動中執行的其他巨集呼叫的鏈結的典型 HTML 區塊

```
@DTW_STATIC()
...
%define handle = ""
@DTW_RTVHANDLE(handle)

%html(report) {
@DTW_ACCEPT(handle)
...
<a href="/cgi-bin/db2www/${handle}/qsys.lib/mylib.lib/
  macros.file/pcgil.mbr/report2">continue</a><br>
<a href="/cgi-bin/db2www/${handle}/qsys.lib/mylib.lib/
  macros.file/pcgil.mbr/quit">quit</a><br>
%}
```

範例 2：具有另一個巨集的 FORM ACTION 鏈結的典型 HTML 區塊

```
@DTW_STATIC()
...
%define handle = ""
@DTW_RTVHANDLE(handle)

%html(input) {
@DTW_ACCEPT(handle)
...
<form method=post action="/cgi-bin/db2www/${handle}/qsys.lib/
mylib.lib/macros.file/pcgil.mbr/report2">
<p>您想要看到何種類型的硬體？
<menu>
<li><input type="radio" name="hardware" value="MON" checked>監視器
<li><input type="radio" name="hardware" value="PNT">指標裝置
<li><input type="radio" name="hardware" value="PRT">印表機
<li><input type="radio" name="hardware" value="SCN">掃描器
</menu>
</form>
%}
```

終止異動

您可以經由向 `Net.Data` 指出您不再要您的巨集是持續的，來終止一個異動。

終止異動：

您可以使用 `DTW_TERMINATE()` 內建函數，設定異動的結尾。類似於 `DTW_ACCEPT()` 函數，在巨集建立任何輸出之前，必須先呼叫這個函數，且一般會將它設定為 HTML 區塊中的第一個元素。`DTW_TERMINATE` 告訴 `Net.Data` 這個呼叫是現行異動中的最後一個呼叫。

語法：`@DTW_TERMINATE()`

這個函數不接受任何參數。

範例：

```
%html(quit) {  
  @DTW_TERMINATE()  
  ...  
%}
```

定義異動中變數的範圍

您可以經由將範圍設定為 `%DEFINE` 陳述式的屬性，決定您想要變數在異動中具有的范围。

您可以設定：

異動範圍

變數範圍適用於整個異動。

單一呼叫範圍

變數範圍適用於單一 `Net.Data` 呼叫。

設定變數的異動範圍：

設定屬性 `STATIC` 指出變數具有異動範圍，表示將橫跨異動中的所有呼叫來儲存變數的值。`STATIC` 是持續巨集的預設值。例如：

```
@dtw_static()  
%define(static) var1 = "val1"
```

設定變數的單一呼叫範圍：

設定屬性 `TRANSIENT` 指出變數具有單一呼叫範圍，表示在每一次呼叫時，將重新起始設定變數的值。`TRANSIENT` 是非持續巨集的預設值。例如：

```
@dtw_static()  
%define(transient) var1 = "val1"
```

在持續巨集中：

- 所有在 DTW_STATIC() 呼叫之後的變數若未明確定義為 TRANSIENT，均為 STATIC。
- 所有在 DTW_STATIC() 呼叫之前的變數若未明確定義為 STATIC，均為 TRANSIENT。

在異動中設定 COMMIT 與 ROLLBACK

在非持續巨集中，於結束巨集呼叫時，將依據呼叫的成功或失敗，Net.Data 會以隱含方式執行確定或取消。透過持續巨集，現在於異動終止時，將進行確定或取消。不過，因為異動可以橫跨多個呼叫，所以您可能想要逐漸確定或取消異動內的變更。

在異動期間確定擱置中的變更：

設定 DTW_COMMIT() 內建函數。

這個函數不會採用任何參數，且會執行異動中所有擱置的變更。

例如：

```
%html(report) {
@dtw_accept(handle)
...
%IF (action="Enter")
    @dtw_commit()
%ENDIF

%}
```

取消異動中的擱置變更：

設定 DTW_ROLLBACK() 內建函數。

這個函數不會採用任何參數，且會捨棄異動中所有擱置的變更。

例如：

```
%html(report) {
@dtw_accept(handle)
...
%IF (action="Cancel")
    @dtw_rollback()
%ENDIF

%}
```

持續巨集的範例

下列簡單巨集含有會在單一異動中執行的多個 HTML 區塊：

```
@dtw_static()
#define a = "0"
#define(transient) b = "0"
```



```

%define handle = ""
@dtw_rtvhandle(handle)

%html(report) {
@dtw_accept(handle)
a = $(a)<br>
b = $(b)<br>
@dtw_add(a, "2", a)
@dtw_add(b, "2", b)
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/report2">
click here to continue</a><br>
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/quit">
click here to quit</a><br>
%}

%html(report2) {
@dtw_accept(handle)
a = $(a)<br>
b = $(b)<br>
@dtw_add(a, "2", a)
@dtw_add(b, "2", b)
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/report3">
Click here to continue</a><br>
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/quit">
Click here to quit</a><br>
%}

%html(report3) {
@dtw_accept(handle)
a = $(a)<br>
b = $(b)<br>
@dtw_add(a, "2", a)
@dtw_add(b, "2", b)
<a href="/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/quit">
Click here to quit</a><br>
%}

%html(quit) {
@dtw_terminate()
a = $(a)<br>
b = $(b)<br>
done
%}

```

假定第一個呼叫是要呼叫 HTML 區塊 report，則 Net.Data 將：

1. 呼叫 DTW_STATIC() 函數，指出這個巨集是持續的。
2. 建立變數 a 作為 STATIC 變數，因為持續巨集的預設值為 STATIC。
3. 建立變數 b 作為 TRANSIENT 變數，因為將透過 TRANSIENT 屬性，以明確方式定義它。
4. 呼叫 DTW_RTVHANDLE()，產生一個異動 handle 並將它置於變數 handle 中。

5. 開始處理 HTML 區塊 report 並呼叫 DTW_ACCEPT(), 告訴 Net.Data 哪一個異動 handle 是供這個異動使用。
6. 尋找要傳送到瀏覽器的輸出, 這使得 Net.Data 會將 HTTP 表頭傳送到 Web 伺服器, 指出異動正在啟動中。
7. 顯示 HTML 頁面。變數 a 與 b 均具有 0 值。

在第一個頁面輸出傳送到瀏覽器之後, 使用者可以選擇要繼續異動或退出。如果他們選擇繼續, Web 伺服器將呼叫 URL:

```
/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/report2
```

Web 伺服器會將異動 handle 識別為 Net.Data 在 HTTP 表頭中設定的 handle。它會呼叫 Net.Data 作為持續 CGI 程式, 表示巨集呼叫是現行異動的一部份。

當呼叫 HTML 區塊 report2 時, Net.Data 將:

1. 呼叫 DTW_STATIC() 函數, 指出這個巨集是持續的。
2. 承認變數 a 是一個 STATIC 變數, 且保存現行值, 而不是將它重新起始設定為 0。
3. 承認變數 b 是 TRANSIENT 變數, 建立變數的新案例, 然後將它起始設定為 0。
4. 呼叫 DTW_RTVHANDLE(), 產生一個異動 handle 並將其置於變數 handle 中。
5. 開始處理 HTML 區塊 report2 並呼叫 DTW_ACCEPT(), 告訴 Net.Data 哪一個異動 handle 是供這個異動使用。
6. 尋找要傳送到瀏覽器的輸出, 這使得 Net.Data 會將 HTTP 表頭傳送到伺服器, 指出異動正在作用中。
7. 顯示 HTML 頁面。變數 a 將具有值 2, 而變數 b 將具有值 0。會從先前呼叫中儲存變數 a 的值, 因為它是靜態變數。變數 b 的值將重設為 0。

在第兩個頁面傳送到瀏覽器之後, 使用者可以選擇要繼續異動或退出。如果他們選擇退出, Web 伺服器將呼叫下列 URL:

```
/cgi-bin/db2www/$(handle)/qsys.lib/mylib.lib/macros.file/pcgil.mbr/quit
```

Web 伺服器會將異動 handle 識別為 Net.Data 在 HTTP 表頭中設定的 handle, 且會呼叫 Net.Data 作為持續 CGI 程式, 表示巨集呼叫是現行異動的一部份。

當呼叫 HTML 區塊 quit 時, Net.Data 將:

1. 呼叫 DTW_STATIC() 函數, 指出這個巨集是持續的。
2. 承認變數 a 是一個 STATIC 變數, 且保存現行值, 而不是將它重新起始設定為 0。
3. 承認變數 b 是 TRANSIENT 變數, 建立變數的新案例, 然後將它起始設定為 0。
4. 呼叫 DTW_RTVHANDLE(), 產生一個異動 handle 並將其置於變數 handle 中。
5. 開始處理 HTML 區塊 quit 並呼叫 DTW_TERMINATE(), 告訴 Net.Data 這是這個異動中的最後一個呼叫。

- 6. 尋找要傳送到瀏覽器的輸出，這使得 Net.Data 會將 HTTP 表頭傳送到伺服器，指出正在終止異動。
- 7. 顯示 HTML 頁面。變數 a 具有值 4，而變數 b 具有值 0。
- 8. 清除所有變數及其他具有異動層次範圍的資源，因為已執行了 DTW_TERMINATE() 呼叫。

附錄A. 問題分析

本段落描述對 Net.Data 進行除錯時您可以使用的問題分析技術。

對於下列問題，將假定 Net.Data CGI-BIN 程式物件 (DB2WWW) 已移到名為 WWWCGI 的程式庫，此程式庫正是 CGI-BIN 程式常駐之所。

- 症狀：錯誤 500：

不好的 script 要求 -- '/QSYS.LIB/WWWCGI.LIB/DB2WWW.PGM/
QSYS.LIB' 不是可執行檔

原因：Exec 規則不正確。

```
Exec /QSYS.LIB/WWWCGI.LIB/DB2WWW.PGM/*  
Exec /qsys.lib/wwwcgi.lib/db2www.pgm/*
```

解決方法：設定僅提供 DB2WWW 程式路徑的 Exec 規則。例如：

```
Exec /QSYS.LIB/WWWCGI.LIB/*  
Exec /qsys.lib/wwwcgi.lib/*
```

- 症狀：錯誤 404：

找不到 - 檔案不存在或防讀
已試過多次"

原因：遺失 Exec 規則。

解決方法：設定一個以大寫及小寫字體提供 DB2WWW 程式路徑的 Exec 規則。例如：

```
Exec /QSYS.LIB/WWWCGI.LIB/*  
Exec /qsys.lib/wwwcgi.lib/*
```

- 症狀：錯誤 403：

禁止 - 依據規則

原因：Map 或 Exec 規則遺失或不正確。

解決方法：設定大寫及小寫字體的 DB2WWW 程式的 Map 與 Exec 規則。例如：

```
Map /cgi-bin/db2www/* /QSYS.LIB/WWWCGI.LIB/DB2WWW.PGM/*  
Map /CGI-BIN/DB2WWW/* /QSYS.LIB/WWWCGI.LIB/DB2WWW.PGM/*  
Exec /QSYS.LIB/WWWCGI.LIB/*
```

- 症狀：Web 伺服器架構檔中的架構陳述式是正確的，但 Net.Data 並未正確地處理巨集檔案。有數個可能的原因與解決方法：

- 原因：Map、Exec 或 Pass 規則的次序不正確。

解決方法：當對 Map、Exec 或 Pass 規則評估 URL 時，會依據第一個相配的規則來處理它。請確定在到達想要的規則之前，不重新對映或更新將評估的陳述式。另外，請確定使用者在架構檔中沒有 Pass /* 陳述式。

- **原因：**使用者設定檔 QTMHHTP1 沒有適當的權限，所以無法存取 Net.Data 巨集。
解決方法：所有 CGI-BIN 程式均在使用者設定檔 QTMHHTP1 下執行。QTMHHTP1 使用者設定檔須被授與當處理 Net.Data 巨集時 Net.Data 存取的所有物件的權限。
- **原因：**Net.Data 起始設定檔案中的「路徑」陳述式不正確。
解決方法：請確定物件參照是完整的，或 Net.Data 起始設定檔案具有適當的路徑陳述式。Net.Data 將使用起始設定檔案中的路徑陳述式 (若有一個的話)，解析為任何 Net.Data 巨集，或將處理的 Net.Data 巨集中的可執行檔參照。如果物件參照不是完整的，且起始設定檔案中的路徑陳述式不正確，Net.Data 將指出找不到要參照的物件。

附錄B. 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。

```

%{***** 樣本巨集 *****}
* 檔名 = sqlsamp1.d2w *
* 說明： *
* 此 Net.Data 巨集檔查詢... *
* - 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 * FROM $(myTable)
%MESSAGE {
  -204: {<p><b>錯誤 -204: 找不到表格 $(myTable) 。</b><p>
        <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>$(V2)
%}
</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 EMPNME, PHONENO, JOB FROM $(myTable) WHERE EMPNME='$(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
%}
%}

```



```

%{ *****
*   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>
%}

```

```

%{*****
*   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 %}
=====
%{*****          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     = "MRZ.EMPLOYEE"
%}

%{      併入檔結束      %}

```

附錄C. 注意事項

本書是針對 IBM 在美國所提供之產品及服務程式開發出來的。在其他國家中，IBM 不見得有提供本書中所提及的各項產品、服務或特性。請連絡您當地的 IBM 業務代表，以取得產品、服務或特性的相關資訊。凡提及 IBM 產品、程式或服務時，並不表示或暗示只可使用 IBM 產品、程式或服務。只要不違反 IBM 的智慧財產權，任何功能上相等的產品、程式或服務，都可以用來代替 IBM 產品、程式或服務。但是，使用者對於任意非 IBM 產品、程式或服務所作的評估與驗證，須自行負責。

本文件中包含著 IBM 所擁有之專利或暫准專利。因此修改本文件，並不會讓您享有這些專利權的使用權。您可以用書面方式，將授權要求寄到：

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

欲取得二位元組字元集 (DBCS) 的相關資訊，請洽當地的「IBM 智慧財產權部門」，或是將您的要求寄至下列地址：

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

下列段落不適用於「英國」或任何法律與其相抵觸之其他國家：國際商業機器股份有限公司 (IBM) 係以『交付時之現狀』提供本書，而不提供任何明示或默示之保證，其中包括 (但不限於) 為特定目的而無傷害性、適售性的保證。有些國家並不允許在某些交易中不作明示或暗示的保證，因此，此聲明可能亦不適用。

本書中可能有技術上或排版印刷上的訛誤。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，但恕不另行通知。

在本書中，若有任何非 IBM Web 網站的參考資料，都只是為了您的方便而提供，無論在何種情況下，都不能為那些網站作保證。而那些網站上的資料都不能作為此 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。

獲得本程式的授權者，如需其相關資訊作為下列用途：(i) 在獨立創作的程式與其他程式 (包括本程式在內) 之間交換資訊 (ii) 互相使用彼此交換的資訊，請洽：

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

上述資訊之取得有其特殊要件，在某些情況下，必須付費方得使用。

IBM 提供本資訊所描述的授權程式以及其所能使用所有授權資料，皆受「IBM 客戶合約」、「IBM 國際程式授權合約」或與客戶之間的任何同等合約的管束。

有關非 IBM 產品的資訊皆取自該產品的提供者、其出版聲明或其他公開的可用來源。IBM 並無測試那些產品，也無法確認執行效能的精確度、相容性或任何與非 IBM 產品相關的要求。若有關非 IBM 產品的功能問題，您應向該產品的提供者申訴。

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。為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。所有這些名稱皆為虛構的，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

下列詞彙是 IBM 公司在美國或其它國家或此兩者的商標：

AIX	IMS
AS/400	Language Environment
CBIDO	MVS/ESA
CBPDO	Net.Data
CICS	OpenEdition
CustomPac	Operating System/400
DB2	OS/2
DB2 Universal Databas	OS/390
DataJoiner	OS/400
Distributed Relational Database Architecture	RACF
DRDA	SystemPac
IBM	

下列詞彙是以下其它公司的商標：

Java 和所有 Java 的商標與標誌是 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，在美國以及 (或) 其它國家的註冊商標。

Lotus 和 Domino Go Webserver 是 Lotus Development Corporation 在美國及 (或) 其它國家的商標。

Microsoft、Windows、Windows NT 以及 Windows 95 標誌是 Microsoft Corporation 在美國及 (或) 其它國家的商標或註冊商標。

其它以雙星號（**）標示的公司、產品與服務名稱可能為其它公司的註冊商標或服務標記。

名詞解釋

七劃

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。

八劃

空值 (null). 一個代表沒有資訊的特殊值。

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。

通用資源位置 (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>

通用閘道介面 (Common Gateway Interface). 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。

十二劃

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). 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。有些語言環境是由 Net.Data (如 REXX、Perl 及 Oracle) 所提供。您也可建立您自己的語言環境。

十七劃

應用程式設計介面 (application programming interface, API). 為一功能性介面，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 及 NSAPI。

A

API. 應用程式設計介面。Net.Data 支援三種週邊 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）一起使用，且是在當 HTML 頁面被載入時載入。

C

CGI. 通用閘道介面。

D

DBMS. 資料庫管理系統。

H

HTML. 超文字標示語言。

HTTP. 超本文轉送通信協定。

I

ICAPI. Internet Connection API。

ICS. Internet Connection Server。

ICSS. Internet Connection Secure Server。

Internet. 一個國際公用的 TCP/IP 電腦網路。

Internet Connection Secure Server. IBM 之具安全特性的 Web 伺服器。

Internet Connection Server. IBM 之無安全特性的 Web 伺服器。

Intranet. 一個位於公司防火牆內的 TCP/IP 網路。

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。

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. 通用資源位置。

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。

〔五劃〕

- 加密, 網路 21
- 巨集要求
 - 語法 25
 - 說明 25
 - 範例 25
- 巨集檔
 - 之內和之間導引 36
 - 呈現部份 31
 - 函數 47
 - 宣告部分 31
 - 持續 87
 - 迴路 67
 - 區塊 33
 - 條件邏輯 65
 - 產生 HTML 58
 - 結構 32
 - 開發 31
 - 說明 1
 - 樣本 32
 - 識別字範圍 37
 - 變數 37
 - DEFINE 區塊 34
 - FUNCTION 區塊 34
 - HTML 區塊 35
 - IF 區塊 65
 - WHILE 區塊 67
- 巨集檔的部份
 - 呈現 31
 - 宣告 31

〔六劃〕

- 列示變數 44
- 列印, 停用預設報告 60
- 名詞解釋 107
- 存取權, 設定 Net.Data 檔案的 17
- 安全
 - 身份驗證 21

- 安全 (繼續)
 - 防火牆 19
 - 授權 23
 - 設定存取權 17
 - 概觀 19
 - 網路加密 21
 - Net.Data 機制 23

〔七劃〕

- 身份驗證, 安全 21
- 防火牆 19

〔八劃〕

- 使用者定義函數 48
- 函數
 - 使用者定義 48
 - 呼叫 53
 - 呼叫儲存程序 75
 - 定義 48
 - 說明 47
 - FUNCTION 區塊語法 48
 - MACRO_FUNCTION 區塊語法 48
- 函數呼叫
 - 處理程序次序 53
 - 語法 53
- 呼叫 Net.Data
 - 巨集要求 25
 - 使用 CGI 25
 - 使用巨集檔 25
 - 直接要求 25
 - 套表 25, 29, 91
 - 概觀 25
 - 語法 25
 - 鏈結 25, 28, 91
 - HTML 區塊 58
 - URL 25, 29, 91
- 呼叫函數 53
- 呼叫儲存程序 75, 76
- 定義變數
 - DEFINE 陳述式或區塊 38
 - HTML 套表之 SELECT 與 INPUT 標籤 39
 - URL 資料 39

注意事項 105
表尾資訊, REPORT 區塊 60
表格處理程序變數 46
表格變數 45

〔九劃〕

保護資產 19
宣告部份, 巨集檔結構 31
建立起始設定檔案 8
持續巨集檔案 87
架構 Net.Data
 起始設定檔
 更新 7
 建立 7
 架構變數陳述式 9
 路徑陳述式 11
 說明 7
 ENVIRONMENT 陳述式 15
 概觀 5
 對 Net.Data 檔案的存取權 17
架構變數陳述式
 架構起始設定檔案中的 9
 說明 9
 DTWR_CLOSE_REGISTRIES 11
 DTW_SQL_ISOLATION 9
 DTW_SQL_NAMING_MODE 10

〔十劃〕

套表
 位在網頁中可呼叫 Net.Data 27
 呼叫 Net.Data 25, 29, 91
格式化資料輸出 59
起始設定檔
 更新 7
 建立 7, 8
 架構變數陳述式 9
 路徑陳述式 11
 說明 7
 ENVIRONMENT 陳述式 15
 format 7
迴路, WHILE 區塊 67

〔十一劃〕

區塊, 巨集檔 33
參照變數 40

執行變數 42
授權
 安全 23
 設定 Net.Data 檔案的存取權 17
啓動 Net.Data 25
條件
 變數 41
 邏輯, IF 區塊 65
異動處理 87
符記大小 37
處理結果集合, 儲存程序 78

〔十二劃〕

報告格式, 自行設定 61
報告變數 46
結果集合
 多重 79
 多個, 指南與限制 64
 處理, 儲存程序 78
 單一 78

〔十三劃〕

傳送參數, 儲存程序 77
資料類型, 對於儲存程序是有效的 76
路徑陳述式
 更新準則 11
 保護資產 23
 架構起始設定檔案中的 11
 EXEC_PATH 13
 FFI_PATH 15
 INCLUDE_PATH 14
 MACRO_PATH 12
預設值報告
 列印 60
 設定儲存程序的 78, 79

〔十四劃〕

語言環境 69
 架構 ENVIRONMENT 陳述式 15
 架構起始設定檔案中的 15
 範例 15
 變數 47

〔十五劃〕

廣域識別字範圍 37
樣本巨集 101
標題資訊, REPORT 區塊 60
範圍。
 變數 37
 REPORT 區塊 38
範圍, 識別字
 巨集檔 37
 廣域 37
 FUNCTION 區塊 38
 ROW 區塊 38
複製 Net.Data 程式物件
 到 CGI-BIN 程式庫 5
 到多個程式庫 6

〔十六劃〕

導引, 在巨集之內和之間 36

〔十七劃〕

儲存程序
 多重結果集合 79
 有效資料類型 76
 步驟 76
 從巨集檔呼叫 75
 處理結果集合 78
 單一結果集合 78
 傳送參數 77
 預設值報告 78, 79
 REPORT 區塊 78, 79
檔案, 設定 Net.Data 的存取權 17
環境變數 41
隱藏變數
 保護資產 23
 隱藏變數名稱 43

〔十八劃〕

雜項變數 45

〔十九劃〕

鏈結
 位在網頁中可呼叫 Net.Data 27

鏈結 (繼續)

 呼叫 Net.Data 25, 28, 91
類型, 變數 40

〔二十三劃〕

變數
 列示 44
 定義 38
 表格處理程序 46
 架構, 陳述式
 起始設定檔 9
 說明 9
 SQL 命名模式 (DTW_SQL_NAMING_MODE) 10
 SQL 隔離 (DTW_SQL_ISOLATION) 9
 Web 登記關閉 (DTWR_CLOSE_REGISTRIES) 11
 參照 40
 執行 42
 條件 41
 符記大小 37
 報告 46
 語言環境 47
 說明 37
 範圍。 37
 環境 41
 隱藏 43
 雜項 45
 類型 37, 40
 table 45
變數參照處理程序, 處理程序次序 53

C

CGI-BIN 程式庫, 複製 Net.Data 程式物件 5

D

DEFINE 區塊
 定義變數 38
 說明 34
DTWR_CLOSE_REGISTRIES 11
DTW_SQL_ISOLATION 9
DTW_SQL_NAMING_MODE 10

E

ENVIRONMENT 陳述式
 服務程式程式 16

ENVIRONMENT 陳述式 (繼續)

- 架構起始設定檔案中的 15
- 參數列示 16
- 語言環境類型 16
- 語法 15
- 說明 15
- 範例 16

F

FUNCTION 區塊

- 呼叫函數 53
- 格式化輸出 59
- 處理程序函數呼叫 53
- 處理程序變數參照 53
- 說明 34
- 識別字範圍 38

H

HTML

- 在巨集檔中產生 58
- 表格的標籤 60
- 套表
 - 呼叫 Net.Data 25, 29, 91
 - 關於 27
 - SELECT 和 INPUT 標籤, 定義變數 39
- 區塊
 - 呼叫 Net.Data 58
 - 處理程序 59
 - 說明 35
 - 範例 58
- 無法識別資料 59
- 鏈結
 - 呼叫 Net.Data 25, 28, 91
 - 關於 27
- FORM「提出」按鈕 59

I

- IF 區塊 65
- IN 參數 53
- INOUT 參數 53

M

MACRO_FUNCTION 區塊

- 呼叫函數 53
- 處理程序函數呼叫 53
- 處理程序變數參照 53
- 語法 48

MESSAGE 區塊

- 處理程序 51
- 語法 51
- 說明 51
- 範例 52
- 範圍。 51

N

Net.Data

- 巨集檔, 開發 31
- 安全機制 23
- 呼叫 25
- 架構 5
- 概觀 1
- 檔案, 存取權 17

Net.Data 巨集。請參閱巨集檔。 1

Net.Data 程式物件

- 複製到 CGI-BIN 程式庫 5
- 複製到多個程式庫 6

O

OUT 參數 53

R

REPORT 區塊

- 指南 64
- 限制 64
- 格式化資料輸出 59
- 說明 59
- 標題和表尾資訊 60
- 範圍。 38
- 儲存程序 78, 79

RETURNS 參數 53

RETURN_CODE 變數 51, 53

ROW 區塊, 識別字範圍 38

S

SQL

- 命名模式架構變數 10
- 隔離架構變數 9

U

URL

- 呼叫 Net.Data 25, 29, 91
- 定義變數 39

W

- Web 登記, 關閉變數 11
- WHILE 區塊 67

折疊線

台北市敦化南路一段二號十二樓

臺灣國際商業機器股份有限公司
中文支援中心 啟

廣告回信
台灣地區郵政管理局 登記
北台字第 0587 號

(免貼郵票)

收件人 姓名：
地址：

寄

折疊線

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 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	參考書目是否正確 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
一 致 性	文句用語及風格，前後是否一致 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	實際畫面訊息與本書所提之畫面訊息是否一致 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
完 整 性	是否遺漏您想知道的項目 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	字句、章節是否有遺漏 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
術語使用	術語之使用是否恰當 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	術語之使用，前後是否一致 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
可 讀 性	文句用語是否通順 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	有否不知所云之處 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
內容說明	內容說明是否詳盡 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	例題說明是否詳盡 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
排版方式	本書的形狀大小，版面安排是否方便使用 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	字體大小，顏色編排，是否有助於閱讀 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
目錄索引	目錄內容之編排，是否便於查考 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	索引語錄之排定，是否便於查考 <input type="checkbox"/> 是 <input type="checkbox"/> 否	
	※評估意見為 "否" 者，請於備註欄說明。	

其他：（篇幅不夠時，請另紙說明。）

[illegible]

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