

Net.Data



# OS/400 版管理及程式設計指南



Net.Data



# OS/400 版管理及程式設計指南

**注意事項**

請務必先詳讀第113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。

**第六版 (1999 年 5 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, 1999. All rights reserved.

# 目錄

序文	vii
關於 Net.Data	vii
本版中新增的特性？	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
關於 Net.Data 起始設定檔	6
自行設定 Net.Data 起始設定檔	6
建立起始設定檔案	7
架構變數陳述式	8
路徑架構陳述式	13
環境架構陳述式	17
設置語言環境	18
設置 Java 應用程式語言環境	18
設置 SQL 語言環境	19
架構 Web 伺服器	19
授與 Net.Data 存取的物件的存取權	20
第3章 維持資產的安全	23
使用防火牆	23
加密網路上的資料	25
使用身份驗證	25
使用授權	26
使用 Net.Data 機制	26
Net.Data 架構變數	26
巨集開發技術	27
第4章 呼叫 Net.Data	31
使用巨集呼叫 Net.Data (巨集要求)	31
HTML 鏈結	32
HTML 套表	33
呼叫持續巨集	34
持續巨集語法	34
範例	35
第5章 開發 Net.Data 巨集	37
Net.Data 巨集的結構	38
DEFINE 區塊	39
FUNCTION 區塊	39
HTML 區塊	40
Net.Data 巨集變數	42
識別字範圍	42
定義變數	43

參照變數 . . . . .	45
變數類型 . . . . .	46
Net.Data 函數 . . . . .	52
定義函數 . . . . .	53
呼叫函數 . . . . .	57
呼叫 Net.Data 的內建函數 . . . . .	57
在巨集中建立網頁 . . . . .	61
HTML 區塊 . . . . .	61
報告區塊 . . . . .	62
巨集中的條件式邏輯和迴路 . . . . .	67
條件性邏輯：IF 區塊 . . . . .	67
迴路結構：WHILE 區塊 . . . . .	69
<b>第6章 使用語言環境 . . . . .</b>	<b>71</b>
Net.Data 提供的語言環境的概觀 . . . . .	72
呼叫語言環境 . . . . .	72
處理錯誤狀況 . . . . .	72
安全 . . . . .	73
直接呼叫語言環境 . . . . .	73
呼叫程式 . . . . .	73
傳遞參數到程式 . . . . .	73
從程式傳回值 . . . . .	75
直接呼叫語言環境範例 . . . . .	76
Java 應用程式語言環境 . . . . .	76
呼叫 Java 程式 . . . . .	76
傳遞參數到 Java 程式 . . . . .	77
Java 應用程式語言環境範例 . . . . .	77
REXX 語言環境 . . . . .	77
執行 REXX 程式 . . . . .	77
傳送參數到 REXX 程式 . . . . .	78
REXX 語言環境範例 . . . . .	79
SQL 語言環境 . . . . .	80
執行 SQL 陳述式 . . . . .	80
資料類型注意事項 . . . . .	82
管理 Net.Data 應用程式中的異動 . . . . .	85
管理多個資料庫連線 . . . . .	86
儲存程序 . . . . .	86
SQL 語言環境範例 . . . . .	91
系統語言環境 . . . . .	92
發出指令及呼叫程式 . . . . .	92
傳遞參數到程式 . . . . .	92
系統語言環境範例 . . . . .	93
<b>第7章 透過持續巨集的異動管理 . . . . .</b>	<b>95</b>
關於持續巨集 . . . . .	95
定義異動 . . . . .	96
啟動異動 . . . . .	96
在異動中設定巨集 HTML 區塊 . . . . .	97
終止異動 . . . . .	100
定義異動中變數的範圍 . . . . .	100
在異動中設定 COMMIT 與 ROLLBACK . . . . .	101
持續巨集的範例 . . . . .	101

<b>第8章 提高執行效能</b>	105
Net.Data 巨集快取	105
將語言環境最佳化	105
REXX 語言環境	105
SQL 語言環境	106
系統語言環境	106
<b>附錄A. 參考書目</b>	107
Net.Data 技術圖書庫	107
相關文件	107
<b>附錄B. Net.Data 樣本巨集</b>	109
<b>附錄C. 注意事項</b>	113
商標	114
名詞解釋	115
索引	117





---

## 序文

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

---

## 關於 Net.Data

透過 IBM 的 Net.Data 產品，您便可以使用來自關聯式與非關聯式資料庫管理系統 (DBMS) 的資料 (這些系統包括可透過 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 是一個巨集處理器，它以 Web 伺服器機器上的中間軟體之身份來執行。您可撰寫 Net.Data 應用程式 (稱為 *macro*)，Net.Data 解譯該應用程式以自行設定的內容建立動態網頁，這些自行設定的內容是根據使用者的輸入、資料庫的現行狀態、其它資料來源、現有的企業邏輯以及您指定給巨集的其它因數。

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

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

---

## 本版中新增的特性？

Net.Data for OS/400 會在本版次中提供下列新的特性：

- 提高快取巨集及併入檔的執行效能
- 語言環境增強功能包括：
  - 兩個新的語言環境：
  - Java 應用程式

- 直接呼叫
- 可在 SQL 語言環境中使用大型物件 (LOB)
- 巨集語言增強功能包括：
  - 能夠從巨集中使用 DTW\_SENMAIL 內建函數建立及傳送電子郵件訊息
  - 能夠以 DTW\_SETCOOKIE 及 DTW\_GETCOOKIE 內建函數取得及設定 HTTP cookie
  - 能夠以 DTW\_REPLACE 函數置換字串
  - 可在 DTW\_TIME 函數中使用毫秒
  - 可動態建置變數參照
  - 能夠在 DTW\_SELECT() 的 OPTION 元素中設定 VALUE 值
  - 可在 MACRO\_FUNCTION 語言結構中使用 RETURNS 關鍵字
  - 可在變數名稱中使用 # 字元。
- 架構增強功能包括：
  - 在 Net.Data 起始設定檔中 Net.Data 提供的語言環境不再需要 ENVIRONMENT 陳述式
  - 能夠以 DTW\_SHOWSQL 架構變數停用 SHOWSQL 變數 (預設值為停用)

---

## 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。爲了瞭解本書中所提及的各種概念，您必須熟悉 Web 伺服器的工作，並了解簡單的 SQL 陳述式與 HTML 標籤 (包括 HTML 套表標籤)。

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

## 關於本書中的範例

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

## 第1章 簡介

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

透過巨集，Net.Data 可簡化交談式 Web 應用程式的撰寫。

本章將描述 Net.Data 及為什麼要為您的 Web 應用程式使用它的理由。

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

圖1 會描述 Net.Data for OS/400、Web 伺服器及支援的資料與程式設計語言環境之間的關係。

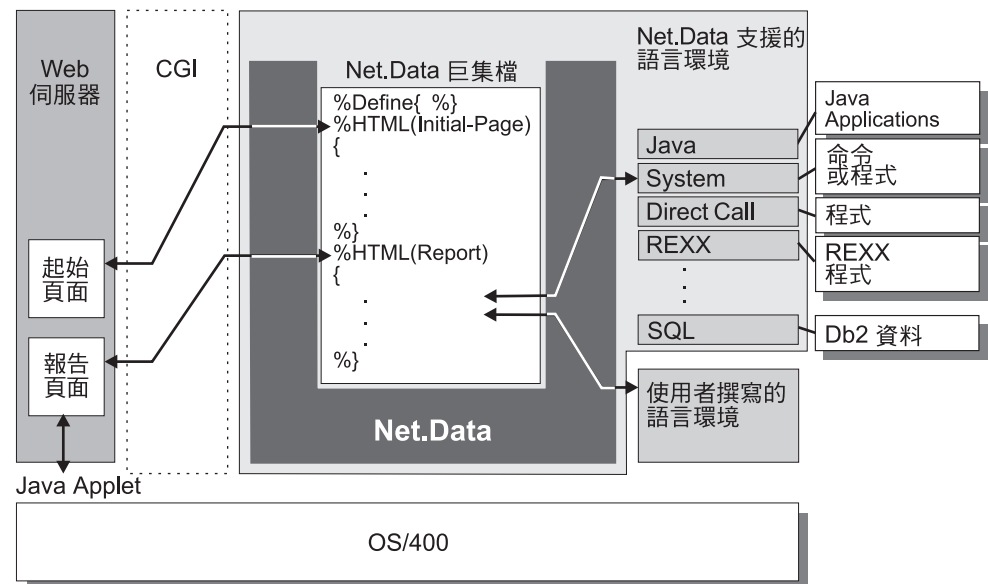


圖 1. Net.Data for OS/400、Web 伺服器及支援之資料與程式來源之間的關係

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。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所做的變更。

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

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

### 解譯的巨集語言

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

### 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42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 及後續版次

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7頁的『建立起始設定檔案』
- 第6頁的『自行設定 Net.Data 起始設定檔』
- 第18頁的『設置語言環境』
- 第19頁的『架構 Web 伺服器』
- 第20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)」指令，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 QHTTSPVR 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 QHTTSPVR 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『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25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。

---

## 關於 Net.Data 起始設定檔

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

- 指定傳送電子郵件時將使用的 SMTP 伺服器及字集
- 啟用 SQL 語言環境及變數 SHOWSQL

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71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8頁的『架構變數陳述式』
- 第13頁的『路徑架構陳述式』
- 第17頁的『環境架構陳述式』

請參閱第7頁的『建立起始設定檔案』，學習如何建立起始設定檔案。



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 DTW SMTP_SERVER 9.5.5.78	• 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	• 第 5 行指定使用者定 義的 ENVIRONMENT 陳 述式
4 EXEC_PATH /QSYS.LIB;/QSYS.LIB/WWW.LIB	
5 ENVIRONMENT(MYLE1) /QSYS.LIB/LELIB.LIB/MYLE1.SRVPGM (IN VAR1, OUT VAR2)	

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。) 確定起始設定檔案對您可以從巨集呼叫的每一使用者定義的語言環境均含有 ENVIRONMENT 陳述式。如果您已完整限定巨集內的所有檔案參照，您就不需要設定任何路徑架構陳述式。

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

## 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)」指令，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20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。

下列子段落描述您可以在起始設定檔案中設定的架構變數陳述式：

- 『DTW\_MACRO\_CACHE\_SIZE：巨集快取大小變數』
- 第9頁的『DTW\_PAD\_PGM\_PARMS：參數填補架構變數』
- 第9頁的『DTW\_SHOWSQL：啓用或停用 SHOWSQL 架構變數』
- 第10頁的『DTW\_SMTP\_CCSSID：電子郵件 SMTP CCSID 變數』
- 第10頁的『DTW\_SMTP\_CHARSET：電子郵件 SMTP 字集變數』
- 第11頁的『DTW\_SMTP\_SERVER：電子郵件 SMTP 伺服器變數』
- 第11頁的『DTW\_SQL\_ISOLATION：DB2 隔離變數』
- 第12頁的『DTW\_SQL\_NAMING\_MODE：SQL 表格命名變數』
- 第12頁的『DTWR\_CLOSE\_REGISTRIES：開啓 Web 登記變數』

### DTW\_MACRO\_CACHE\_SIZE：巨集快取大小變數

指定當快取巨集時，Net.Data 應該使用的記憶體大小 (MB)。當超出快取大小時，Net.Data 會除去舊的快取巨集，在快取中挪出空間。Net.Data 會除去最早使用過的巨集。

#### 語法：

DTW\_MACRO\_CACHE\_SIZE [=] 大小

#### 其中：

大小 指定快取大小 (MB)。預設值為 5 MB，且恆啟動快取。如果大小為 0，將不會快取巨集。如果大小為 1 - 4，將使用預設值 5。

**範例：**指定 16 MB 的快取大小。

DTW\_MACRO\_CACHE\_SIZE 16

## DTW\_PAD\_PGM\_PARMS：參數填補架構變數

向語言環境指出將傳遞給程式或儲存程序的字元參數是否將以空白填補。字元參數具有 CHARACTER 或 CHAR 的資料類型。

對於 IN 或 INOUT 參數，如果參數值的長度少於指定的精確度，將在參數值右邊插入空白，直到參數值的長度同於精確度為止。

對於 OUT 參數，參數參數將設定為精確度空白。

在呼叫程式或儲存程序後，所有尾隨空白將從 OUT 及 INOUT 參數值中除去。

在 Net.Data 起始設定檔中設定這個變數，可指定一個值給您的所有巨集。您可以經由在巨集中定義此值，來置換它。如果 DTW\_PAD\_PGM\_PARMS 未定義在巨集中，它將使用起始設定檔中的值。

「直接呼叫」及 SQL 語言環境支援 DTW\_PAD\_PGM\_PARMS。

語法：

DTW\_PAD\_PGM\_PARMS [=] YES|NO

其中：

**YES** 指定所有 IN 及 INOUT 字元參數均向左對齊，且在傳遞參數給程式或儲存程序之前，先以空白填補，以符合參數的精確度定義。在呼叫程式或儲存程序後，將除去尾隨空白。

**NO** 指定當傳遞參數給程式或儲存程序時，沒有填補字元將新增到字元參數值 (值為以 NULL 終止的值)。在呼叫程式或儲存程序後，將不會除去尾隨的空白。這是預設值。

## DTW\_SHOWSQL：啓用或停用 SHOWSQL 架構變數

置換您的 Net.Data 巨集內的設定 SHOWSQL 的效果。

語法：

DTW\_SHOWSQL YES|NO

其中：

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

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

表1描述 Net.Data 起始設定檔及巨集中的設定如何決定是否要對特殊巨集啓用或停用 SHOWSQL 變數。

表 1. SHOWSQL 的 Net.Data 起始設定檔及巨集中的設定之間的關係

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

表 1. SHOWSQL 的 Net.Data 起始設定檔及巨集中的設定之間的關係 (繼續)

DTW_SHOWSQL 的設定	設定 SHOWSQL	顯示 SQL 陳述式
YES	YES	YES

## DTW SMTP CCSID : 電子郵件 SMTP CCSID 變數

指定與 DTW SMTP CHARSET 中指定的「多目的網際網路郵件擴充 (MIME)」字集有關聯的 ASCII 編碼字集識別字。將 DTW\_SENDMAIL 函數上指定的資料從 EBCDIC 轉換為 ASCII 時，將使用 CCSID。

如果指定 DTW SMTP CCSID，您亦須指定 DTW SMTP CHARSET。當指定 CCSID 時，請確定它適合 DTW SMTP CHARSET 中指定的 MIME 字集，且系統支援 CCSID。表2會列示共用 MIME 字集，以及關聯的 ASCII CCSID。如果未設定 DTW SMTP CCSID，Net.Data 會使用與 MIME 字集 ISO-8859-1 有關聯的 CCSID，亦即 819。

語法：

```
DTW SMTP CCSID [=] ascii_ccsid
```

其中 *ascii\_ccsid* 是從 EBCDIC 轉換為 ASCII 時將使用的 ASCII CCSID (介於 1-65534 之間的數字)。

範例：

```
DTW SMTP CCSID 912
```

這個 ASCII CCSID 對應到 MIME 字集 ISO-8859-2

## DTW SMTP CHARSET : 電子郵件 SMTP 字集變數

指定 DTW\_SENDMAIL 將在電子郵件訊息中使用的「多目的網際網路郵件擴充 (MIME)」字集。如果指定 DTW SMTP CHARSET，您亦須指定 DTW SMTP CCSID。當指定 MIME 字集時，請確定字集有效，因為 Net.Data 不會驗證針對這個變數指定的值。如果未設定 DTW SMTP CHARSET，Net.Data 會使用 MIME 字集 ISO-8859-1，它具有關聯的 CCSID 819。

表2會列示共用 MIME 字集，以及關聯的 ASCII CCSID。

表 2. Net.Data 支援的字集

MIME 標準字集	ASCII CCSID	說明
US-ASCII	367	美式英文
ISO-2022-JP	5052	日文 MBCS
ISO-8859-1	819	拉丁語-1
ISO-8859-2	912	拉丁語-2
ISO-8859-5	915	斯拉夫語
ISO-8859-6	1089	阿拉伯語
ISO-8859-7	813	希臘語
ISO-8859-8	916	希伯來語
ISO-8859-9	920	拉丁語-5

語法：

```
DTW SMTP_CHARSET character_set
```

其中 *character\_set* 是將使用的 MIME 字集。

範例：

```
DTW SMTP_CHARSET iso-8859-2
```

這個 MIME 字集會對應到 912 ASCII CCSID。

## **DTW SMTP\_SERVER: 電子郵件 SMTP 伺服器變數**

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

語法：

```
DTW SMTP_SERVER server_name
```

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

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

範例：

```
DTW SMTP_SERVER 9.5.34.5
```

## **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 (交易) 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。

#### **SYSTEM\_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啟的 Web 登記保持開啟。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)，方法是把登記的開啟和關閉縮至最小。如果多個處理可以同時存取登記 (如在同時瀏覽器要求的情況中)，請將 DTWR\_CLOSE\_REGISTRIES 設定為 YES。

## 路徑架構陳述式

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

- 『MACRO\_PATH』
- 第14頁的『EXEC\_PATH』
- 第15頁的『INCLUDE\_PATH』
- 第16頁的『FFI\_PATH』
- 第16頁的『HTML\_PATH』
- 第16頁的『DTW\_JAVA\_CLASSPATH』

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

### 更新準則：

許多一般準則適用於路徑陳述式。例外狀況將會在每一路徑陳述式的說明中提到。

- 每一個指定的目錄都是以分號 (;) 作為結尾。
- 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。

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

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

### 語法：

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

等號 (=) 是可選用性的，由方括弧 ([]) 表示。

Net.Data 會將路徑 /macro/sqlm.d2w 附加到 MACRO\_PATH 架構陳述式的路徑中 (從左到右)，直到 Net.Data 找到巨集或搜尋過所有路徑為止。關於呼叫 Net.Data 巨集的資訊，請參閱第31頁的『第4章 呼叫 Net.Data』。



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

Net.Data 起始設定檔：

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

HTML 鏈結：

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

如果在目錄 `/u/user1/macros` 中找到檔案 `query.d2w`，則完整的路徑為 `/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/rexxpgms.file` 目錄中找到檔案 `myFunction.mbr`，則程式的完整名稱為 `/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：

- 如果指定的路徑是絕對的，Net.Data 將在指定的路徑中搜尋檔案。例如，如果設定了下列 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 陳述式以及具有次目錄名稱的 INCLUDE 檔。

Net.Data 起始設定檔：

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

Net.Data 巨集：

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

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

如果在 INCLUDE\_PATH 陳述式中設定的目錄中找不到檔案：

- 如果指定的路徑是絕對的，Net.Data 將在指定的路徑中搜尋檔案。例如，如果設定了下列 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。

## HTML\_PATH

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

HTML\_PATH 須指定不在 QSYS.LIB 檔案系統的 IFS 目錄。

語法：

```
HTML_PATH [=] path
```

範例： 下列範例顯示在起始設定檔中的 HTML\_PATH 陳述式。

Net.Data 起始設定檔：

```
HTML_PATH /db2/lobs
```

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

**執行效能要訣：** 由於 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮到系統限制問題。詳細資訊，請參閱第82頁的『使用大型物件』。

## DTW\_JAVA\_CLASSPATH

DTW\_JAVA\_CLASSPATH 架構陳述式可指定用來找出 Java 類別的路徑。目錄是以冒號來分隔。

語法：

```
DTW_JAVA_CLASSPATH [=] path
```

範例： 下列範例顯示在起始設定檔中的 DTW\_JAVA\_CLASSPATH 陳述式。

Net.Data 起始設定檔：

DTW\_JAVA\_CLASSPATH /directory1/directory2:/QIBM/ProdData/Java400

## 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71頁的『第6章 使用語言環境』中加以描述，而語言環境介面則會在 *Net.Data 語言環境介面參考手冊* 中加以描述。

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

Net.Data for OS/400 不需要 Net.Data 所附的語言環境的 ENVIRONMENT 陳述式。不過，如果發現語言環境陳述式，它將會置換 Net.Data 使用的預設值。建議您 Net.Data 提供的語言環境的 ENVIRONMENT 陳述式最好不要新增到 Net.Data 架構檔。

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

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

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

### 新增或更新 ENVIRONMENT 陳述式：

ENVIRONMENT 陳述式具有下列語法：

```
ENVIRONMENT(type) library_name (parameter_list, ...)
```

#### 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。

服務程式名稱是以 .SRVPGM 副檔名來指定的。

- *parameter\_list*

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

若要設定及傳遞參數列示中的變數，請在巨集中定義變數。

在執行由語言環境所處理的函數之前，您必須先將這些參數定義為架構變數，或定義為巨集中的變數。如果函數修改其中一個輸出參數，則函數完成之後參數會保留修改過的值。

當 Net.Data 處理起始設定檔時，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。

在 OS/400 上，ENVIRONMENT 陳述式需要 Net.Data 語言環境，且不建議使用它。不過，這個範例會顯示 Net.Data 使用的某些預設 ENVIRONMENT 陳述式。

```
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/QTMRHREXX.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 ( )
```

**要求**每一 ENVIRONMENT 陳述式須在單一行上。

---

## 設置語言環境

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

- 『設置 Java 應用程式語言環境』
- 第19頁的『設置 SQL 語言環境』

### 設置 Java 應用程式語言環境

在您使用 Java 應用程式語言環境 (首先在 OS/400 V4R4 中引進它) 之前，請先完成下列步驟：

1. 安裝『AS/400 Developer Kit for Java』授權程式，產品識別字為 5769JV1。必須安裝『AS/400 Developer Kit for Java』，方可在 AS/400 上執行 Java 應用程式。
2. 在 Net.Data 起始設定檔中設定 DTW\_JAVA\_CLASSPATH 路徑架構變數，以便 Java 可以找到 Java 應用程式類別。關於這個路徑架構陳述式的詳細資訊，請參閱第16頁的『DTW\_JAVA\_CLASSPATH』。

在設置 Java 應用程式語言環境後，請參閱第76頁的『Java 應用程式語言環境』，學習如何使用 Java 應用程式語言環境。

## 設置 SQL 語言環境

在使用 SQL 語言環境之前，請先完成下列步驟：

1. 在關聯式資料庫目錄中建立本端資料庫的目錄登錄 (具有遠端位置 \*LOCAL 的目錄登錄)，以及建立 SQL 語言環境需要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)」指令來新增登錄。  
如果您將存取遠端資料庫，請完成其他架構步驟，如設置本端系統及遠端系統之間的通信。分散式資料庫支援的詳細資訊，請參閱 *OS/400 Distributed Database Programming*。
2. 如果您將使用 DataLinks，請確定已在使用的系統上架構了 TCP/IP，以及確定「DataLink 檔案管理程式」已啟動，且在將含有要鏈結的物件的所有系統上架構妥當。關於 DataLinks 的詳細資訊，請參閱 *DB2 for OS/400 SQL Programming*。
3. 如果 SQL 語言環境將傳回大型物件 (LOB)，請設定 HTML\_PATH 架構變數。若要更瞭解這個架構變數，請參閱第16頁的『HTML\_PATH』。
4. 新增或更新架構變數。SQL 語言環境支援可在 Net.Data 起始設定檔案中指定的下列架構變數：

### 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

### DTW\_SQL\_NAMING\_MODE

決定如何在 SQL 陳述式中指定表格名稱

### DTW\_SHOWSQL

啓用巨集變數 SHOWSQL 的使用

若要更瞭解 Net.Data 架構變數陳述式，請參閱第8頁的『架構變數陳述式』。

在設置 SQL 語言環境，請參閱第80頁的『SQL 語言環境』，學習如何使用 SQL 語言環境。

---

## 架構 Web 伺服器

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

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

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/*
```

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

```
Pass /*
```

### Map 指令

Map 指令會將使用格式 `/cgi-bin/db2www/*` 的登錄對映到您系統上 Net.Data 程式常駐的程式庫。(字串尾端的星號 (\*) 代表該字串後的任何字元。) 大寫與小寫 map 陳述式均包括在內，因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

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)。

### Pass 指令

如果您想要以 SQL 語言環境使用大型物件 (LOB)，請建立 Pass 指令，來建立 SQL 語言環境將在其中儲存 LOB 檔的目錄。例如：

```
Pass /tmplobs/* /html_path/*
```

其中 `html_path` 是 HTML\_PATH 架構變數中指定的目錄名稱，這個架構變數會指定將儲存 LOB 的預設目錄。詳細資訊，請參閱第16頁的『HTML\_PATH』。

Net.Data 不會使用 Pass 指令。如果您想要簡化您的 URL，請在第13頁的『MACRO\_PATH』中討論的 Net.Data 起始設定檔案中，使用 MACRO\_PATH 陳述式。

---

## 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 `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
- 存取可被語言環境陳述式目標參照的任何物件。例如，SQL 語言環境會執行 SQL 陳述式，而 SQL 陳述式會存取資料庫檔案，所以 Net.Data 執行時所用的使用者 ID 須具有資料庫檔案的權限。

### 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 QTMHHTP1 使用者設定檔權限 (在 V3R2 與 V3R7 中，Internet Connection for AS/400 僅會在 QTMHHTP1 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 指令，給與使用者設定檔的權限：



```
CHGAUT OBJ('/WWW') USER(QTMHHTP1) DTAAUT(*RX)
CHGAUT OBJ('/WWW/macro') USER(QTMHHTP1) DTAAUT(*RX)
CHGAUT OBJ('/WWW/macro/*') USER(QTMHHTP1) 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 指令，給與使用者設定檔的權限：

```
GRTOBJAUT OBJ(WWW) OBJTYPE(*LIB) USER(QTMHHTP1) AUT(*USE)
GRTOBJAUT OBJ(WWW/MACRO) OBJTYPE(*FILE) USER(QTMHHTP1) 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 指令，給與 QSYS.LIB 檔案系統中物件的權限：

```
CHGAUT OBJ('/QSYS.LIB/WWW.LIB') USER(QTMHHTP1) DTAAUT(*RX)
CHGAUT OBJ('/QSYS.LIB/WWW.LIB/MACRO.FILE') USER(QTMHHTP1) DTAAUT(*RX)
```

特定的語言權限環境權限注意事項會在第71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25頁的『加密網路上的資料』
- 第25頁的『使用身份驗證』
- 第26頁的『使用授權』
- 第26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24頁的圖3 會顯示這個架構。

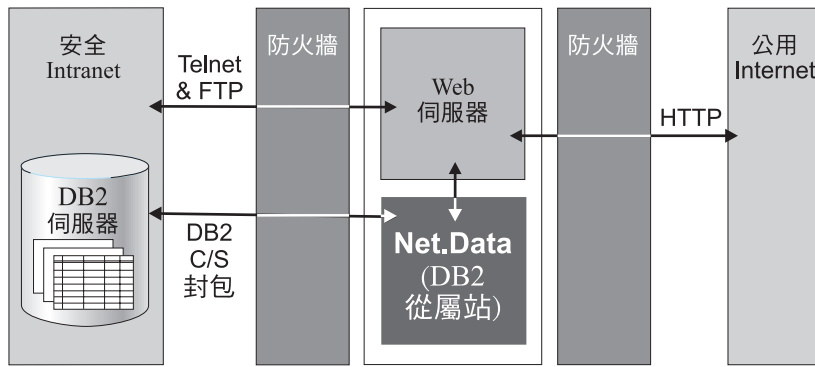


圖 3. 高安全架構

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。圖4 顯示這個架構。

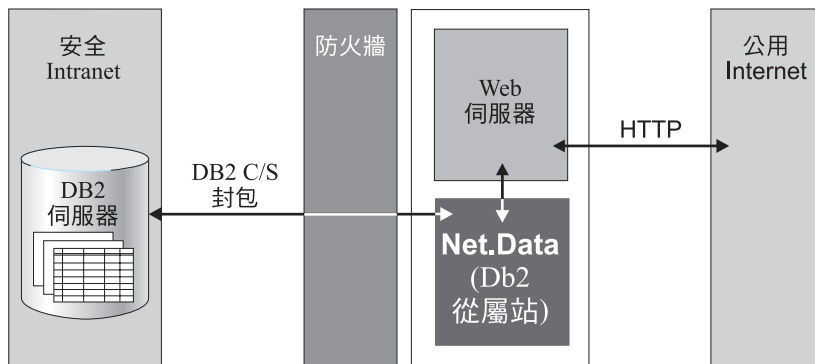


圖 4. 中安全架構：

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。

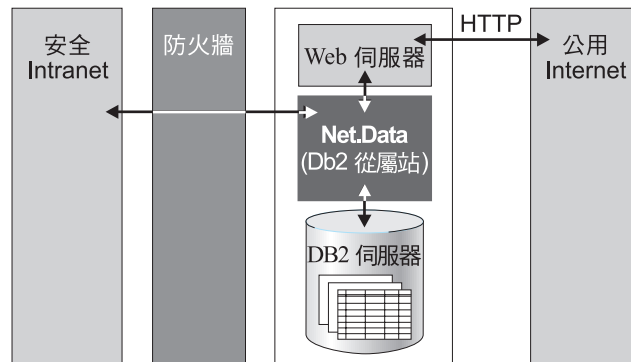


圖 5. 低安全架構：

## 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，可啓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，可啓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，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20頁的『授與 Net.Data 存取的物件的存取權』。

---

## 使用授權

授權提供使用者對於物件、資源或函數的完整或有限存取權。如 DB2 的資料來源會提供它們自己的授權機制，來保護它們管理的資訊。這些授權機制會假定與執行 Net.Data 要求的處理有關聯的使用者 ID 已經過適當的身份驗證，如第25頁的『使用身份驗證』中描述一般。然後，這些資料來源的舊有存取控制機制，會依據授權使用者 ID 所保留的授權來允許或拒絕存取。

---

## 使用 Net.Data 機制

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

## Net.Data 架構變數

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

### 使用路徑陳述式控制檔案存取

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

您也應該使用『使用授權』中所描述的授權檢查，以及驗證無法在第27頁的『巨集開發技術』中所描述的 INCLUDE 陳述式中變更檔名。

### 停用生產系統的 SHOWSQL

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

您可以使用下列方法之一，讓 SQL 陳述式無法顯示在生產系統中。

- 當使用支援 DTW\_SHOWSQL 架構變數的 Net.Data 版本時，請在起始設定檔中使用這個變數，來置換在您的 Net.Data 巨集中設定 SHOWSQL 的效果。請參閱第9頁的『DTW\_SHOWSQL：啓用或停用 SHOWSQL 架構變數』，取得語法及其他資訊。
- 使用『巨集開發技術』中描述的 DTW\_ASSIGN() 函數。

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

## 巨集開發技術

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

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

### 確定無法在 URL 中置換 Net.Data 變數

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

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

您可以使用下列方法之一，讓 SQL 陳述式無法顯示在生產系統中。

- 當使用支援 DTW\_SHOWSQL 架構變數的 Net.Data 版本時，請在 Net.Data 起始設定檔中使用這個變數，來置換在您的 Net.Data 巨集中設定 SHOWSQL 的效果。請參閱第9頁的『DTW\_SHOWSQL：啓用或停用 SHOWSQL 架構變數』，取得語法及其他資訊。
- 請使用上面範例中所描述的 DTW\_ASSIGN() 函數，來指定 SHOWSQL 的值，防止它被置換。

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

您也可以使用 DTW\_ASSIGN，來確保其他 Net.Data 變數（如 RPT\_MAX\_ROWS 或 START\_ROW\_NUM）不會被置換。請參閱 *Net.Data* 參考手冊中有關變數的那一章，取得這些變數的詳細資訊。

### 確認無法以更改應用程式行為的方式來修改您的 SQL 陳述式。

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

1. 識別有效輸入的語法；例如，客戶 ID 須以字母開頭，且僅能含有英數字元。

2. 判斷容許不正確輸入、有意的破壞輸入或想取得 Net.Data 應用程式的內部資產的輸入，可能帶來的傷害。
3. 在巨集中包括必須符合應用程式需要的輸入驗證陳述式。如此的驗證取決於輸入的語法及使用方式。在較簡單的情況中，它足以檢查出輸入中的無效內容，或呼叫 Net.Data 來驗證輸入類型。如果輸入的語法相當複雜，則巨集開發者可能必須局部或完整剖析輸入，方可驗證它是否有效。

#### 範例 1：使用 DTW\_POS() 字串函數來驗證 SQL 陳述式

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

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

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

這個使用者修改過的原始 SQL SELECT 陳述式將傳回整個購物者表格。

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

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

#### 範例 2：使用 DTW\_TRANSLATE()

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

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

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

**確認無法以更改應用程式行為的方式來修改 INCLUDE 陳述式中的檔名。**

如果您使用 Net.Data 變數，透過 INCLUDE 陳述式指定檔名的值，則將不會決定要併入的檔案，直到執行 INCLUDE 檔為止。如果您打算在巨集內設定這個變數的值，但不容許瀏覽器的使用者置換巨集提供的值，則您應該使用



DTW\_ASSIGN 而非使用 DEFINE 來設定變數值。如果您打算允許瀏覽器的使用者提供檔名，則您的巨集應該驗證提供的值。

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

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

且您的 Net.Data 巨集含有下列 INCLUDE 陳述式：

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

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

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

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

**設計您的資料庫及查詢，以便使用者要求無權存取其他使用者的敏感資料。**

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

**範例：**下列 SQL 陳述式將傳回變數 order\_rn 所識別的訂單的訂單資訊：

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

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

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

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

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

### 使用 Net.Data 隱藏變數

您可以使用 Net.Data 隱藏變數來隱藏 Net.Data 巨集的不同性質，讓使用者無法用他們的 Web 瀏覽器來顯示您的 HTML 原始檔案。例如，您可以隱藏您資料庫的內部結構。請參閱第48頁的『隱藏變數』，取得隱藏變數的詳細資訊。

### 自使用者要求驗證資訊

您可以依據使用者提供的輸入來建立自己的保護方案。例如，您可以透過 HTML 套表向使用者要求驗證資訊，並使用您的 `Net.Data` 巨集從資料庫取回的資料，或者從 `Net.Data` 巨集中定義的函數呼叫外部程式，來驗證它。

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

<http://www.w3.org/Security/Faq>



---

## 第4章 呼叫 Net.Data

Net.Data for OS/400 是使用「通用閘道介面 (CGI)」及巨集所呼叫的。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95頁的『第7章 透過持續巨集的異動管理』。

本章將描述如何使用巨集呼叫 Net.Data。

- 『使用巨集呼叫 Net.Data (巨集要求)』
- 第34頁的『呼叫持續巨集』

---

### 使用巨集呼叫 Net.Data (巨集要求)

這節說明如何透過指定巨集來呼叫 Net.Data。

下列語法陳述式顯示如何呼叫 Net.Data。

- URL：

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

參數：

*server* 指定 Web 伺服器的名稱及路徑。若伺服器為區域伺服器，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13頁的『MACRO\_PATH』，以取得有關的詳細資訊。

*block* 指定參照的 Net.Data 巨集中的 HTML 區塊名稱。

*method* 指定與此套表搭配使用的 HTML 方法。

*?name=val&...*

指定要傳送給 Net.Data 的一或多個選用性參數。

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

- HTML 鏈結：

```
<A HREF="URL">任何文字</A>
```

- HTML 套表：

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

參數：

*method* 指定與此套表搭配使用的 HTML 方法。

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

範例

下列範例列出呼叫 Net.Data 的各種方法：

**範例 1：**使用 HTML 鏈結呼叫 Net.Data：

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

**範例 2：**使用套表呼叫 Net.Data

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

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

```
<A HREF="http://server/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://server/cgi-bin/db2www/  
  qsys.lib/mylib.lib/myfile.file/myMacro.mbr/report">  
.  
.  
.  
</FORM>
```

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

- 『HTML 鏈結』
- 第33頁的『HTML 套表』

## HTML 鏈結

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

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

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

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

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

```
%DEFINE DATABASE="MNS97"

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

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

此查詢會傳回一個表格，內容包含由 EQPTABLE 表格內所描述之每種監視器的型號、成本及說明資訊。這個範例會藉由因產生預設報告來顯示查詢結果。請參閱第62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，來建置第32頁的『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95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13頁的『`MACRO_PATH`』，以取得有關的詳細資訊。

*block* 指定參照的 Net.Data 巨集中的 HTML 區塊名稱。

*method* 指定與此套表搭配使用的 HTML 方法。

`?name=val&...`

指定要傳送給 Net.Data 的一或多個選用性參數。

## 範例

下列範例描述如何呼叫持續巨集。

**範例 1：**巨集中的 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。

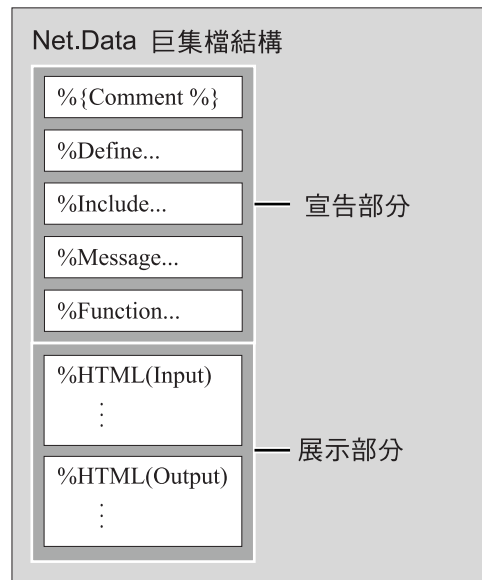


圖 6. 巨集結構

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

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

**授權要訣：**確定 Web 伺服器有權存取此檔。有關的詳細資訊，請參閱第20頁的『授與 Net.Data 存取的物件的存取權』。

本章找出組成 Net.Data 巨集的不同區塊，並說明撰寫巨集所用的方法。

- 第38頁的『Net.Data 巨集的結構』
- 第42頁的『Net.Data 巨集變數』
- 第52頁的『Net.Data 函數』
- 第61頁的『在巨集中建立網頁』
- 第67頁的『巨集中的條件式邏輯和迴路』

## Net.Data 巨集的結構

巨集是由兩個部份所組成：

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

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

**授權要訣：**確定 Web 伺服器有權存取 EXEC 和 INCLUDE 陳述式所參照的檔案。有關的詳細資訊，請參閱第20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。HTML 區塊說明於第40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>@rexsl(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。若要呼叫巨集，Net.Data 需用到兩個參數：要處理的巨集名稱，以及在該巨集中要顯示的 HTML 區段。

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

## DEFINE 區塊

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

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

第一行是一個備註。所謂備註，是指位於 %{ 和 %} 內的本文。備註可在巨集中的任意處。下一個陳述式會開始 DEFINE 區段。您可以在一個定義區塊中定義多個變數。在這個範例中，只可定義一個變數 page\_title。一旦定義之後，您可以使用語法 \$(page\_title)，在巨集中的任意處參照此變數。透過變數的使用，可讓您以後在對您的巨集進行整體的變更時較為簡單。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。一個定義外部 REXX 程式的呼叫，另一個則含有列入 REXX 陳述式。

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

%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 指令。REXX 指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 <b> 或 <em>)，括住要求的 @rexx1() 函數呼叫，來按照自己的意思製作結果本文的格式。不使用 result 變數，而是使用 REXX SAY 陳述式，REXX 程式即可將寫好的 HTML 陳述式以標準格式輸出。

第二個函數區塊，也參照 REXX 程式 today。不過，本例中的整個 REXX 程式包含在其本身的函數宣告中。不需要一個外部程式。REXX 和 Perl 函數都可接受列入 (inline) 程式，這是因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 區塊恆會在 Net.Data 巨集要求中指定，且每一巨集至少須有一個 HTML 區塊。

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

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

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

<p>
```

```

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

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

```

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

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

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

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

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

```

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

```

您可以經由參照套表中指定的變數名稱，來存取或操作套表資料。

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

```

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

</head><body>
<h1>輸出頁</h1>
<p>@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 HTML(OUTPUT) 區塊，此區塊是一個標準的 HTML，再加上用來替代變數的巨集陳述式以及一個函數呼叫。同樣地，page\_title 變數被取代成標題陳述式。一如前述，本區塊中含有一個函數呼叫。在此例中，它會呼叫函數 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。函數定義會指定要傳遞的變數之數目與用法。

---

## Net.Data 巨集變數

Net.Data 可讓您在 Net 巨集內定義及參照變數。另外，您可以將巨集中的這些變數傳遞給語言環境後再傳回。Net.Data 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：
  - 雖未明確地定義，不過在首次指定值時可被案例化的變數。
  - 參數變數，為 FUNCTION 區塊定義的一部份，且僅能在 FUNCTION 區塊中。
  - 可由 Net.Data 案例化，且會對應到套表資料或查詢字串資料的變數。
  - 此類變數乃和 Net.Data 表格相連結，且僅能在 ROW 區塊或 REPORT 區塊中被參考到。

下列段落描述：

- 『識別字範圍』
- 第43頁的『定義變數』
- 第45頁的『參照變數』
- 第46頁的『變數類型』

## 識別字範圍

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

- 廣域  
若您可在巨集的任一處參照到此識別字，則該識別字即具有廣域範圍。具有廣域範圍的識別字如下：
  - Net.Data 的內建函數
  - 套表資料
  - 查詢字串資料
  - 自 HTML 區塊中被案例化的變數
- 巨集  
當識別字的宣告是出現在任一區塊的外頭時，則該 ID 的範圍即屬此種。一個區塊開始於一個左方括弧 ({)，終止於一個百分比符號和右方括弧 (%)。(請注意：本定義中不包括 DEFINE 區塊，且 DEFINE 區塊乃被視為獨立的 DEFINE 陳述式。) 不同於具有廣域範圍的識別字，具有巨集範圍的識別字僅能被巨集中其後有識別字宣告的項目所參照。

- FUNCTION 區塊或 MACRO\_FUNCTION 區塊

若為下列情況，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。

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

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

- REPORT 區塊

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

- ROW 區塊

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

## 定義變數

定義 Net.Data 巨集中變數的方法有下列 3 種：

- 定義陳述式或區塊
- HTML 套表標籤
- 查詢字串資料

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

- **DEFINE 陳述式或區塊**

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

```
%DEFINE variable_name="variable value"

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

%DEFINE {
    variable_name1="variable value 1"
    variable_name2="variable value 2"
}%
```

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

例如：

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

變數 *reply* 具有值 *hello*。

只鍵入兩個連續的引號等於空字串。例如：

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

變數 `empty` 具有空字串。

如果您的變數含有特殊字元 (如行尾)，請使用區塊大括弧括住值：

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

若要在字串中包括雙引號，您可以連續使用兩個雙引號。

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

您也可以使用區塊大括弧來跳出雙引號：

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

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

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

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

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

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

或

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

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

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

`variable_name` 是您提供給變數的名稱，而變數的值是由套表中接收的輸入所決定。關於此種類型的變數定義如何使用於 `Net.Data` 巨集中，請參閱第33頁的『`HTML` 套表』中的範例。

- **查詢字串資料**

您可以透過查詢字串，傳遞變數給 `Net.Data`。例如：

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

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



## 參照變數

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

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

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

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

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

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

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

### 範例 1：鏈結中的變數參照

如果您已定義變數 `homeURL`：

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

您可以用 `$(homeURL)` 來參照首頁，並建立一個鏈結：

```
<A href="$(homeURL)">首頁</A>
```

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

### 範例 2：動態建立變數參照

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

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

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

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

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

**範例 3：**具有巢狀變數參照及函數呼叫的變數參照

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

變數參照會傳回 `hey` 的值。

## 變數類型

您可以在巨集中使用下列類型的變數。

- 『條件式變數』
- 第47頁的『環境變數』
- 第47頁的『執行變數』
- 第48頁的『隱藏變數』
- 第49頁的『列示變數』
- 第50頁的『表格變數』
- 第50頁的『雜項變數』
- 第51頁的『表格處理程序變數』
- 第51頁的『報告變數』
- 第52頁的『語言環境變數』

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

請參閱 *Net.Data* 參考手冊，取得每一變數的語法及範例。

### 條件式變數

透過類似於 `IF THEN` 結構的方法，條件式變數可讓您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49頁的『列示變數』中的範例。



## 環境變數

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

定義環境變數的語法如下：

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

其中 *var* 是將定義的環境變數的名稱。

例如，變數 SERVER\_NAME 可定義為環境變數：

```
%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

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

## 執行變數

透過執行變數，可讓您從一個變數參照呼叫其他程式。

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

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

*runit* 變成執行變數。

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

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

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

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

今天是 Tue 11-07-1999

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

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14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 `NULL` (空值)。若您在 `DTW_ASSIGN` 函數呼叫中使用此變數，來指定其值給另一個變數，則指定後新變數的值也會是 `NULL` (空值)。執行變數的唯一目的，是呼叫其定義的程式。

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

```
%DEFINE mph=%EXEC "calcMPH ${distance} ${time}"
```

下一範例會傳回系統日期作為報告的一部份：

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

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

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

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

## 隱藏變數

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

1. 在 `HTML` 區塊中最後一次參照變數的後面，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 `NULL` (空值)，因此查詢中的 `whereClause` 亦為 `NULL` (空值)。否則，`whereClause` 有 `OR` 所分隔的選取值。例如，若三個都市皆選取，則 SQL 查詢如下：

```

SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'

```

本範例顯示選取了 `Seattle`，其產生的 SQL 查詢如下：

```

SELECT name, city FROM citylist
WHERE cond1='Seattle'

```

## 表格變數

表格變數可以定義一套相關資料。它含有一組包括一列直欄表頭的橫列及直欄。在 Net.Data 巨集中定義表格的方式，如下列陳述式所示：

```
%DEFINE myTable=%TABLE(30)
```

位在 %TABLE 後面的數字會限制這個表格變數可含有的列數。如果您想指定一個沒有列數限制的表格，您可以如下列範例所示，使用預設值或指定 ALL：

```
%DEFINE myTable2=%TABLE  
%DEFINE myTable3=%TABLE(ALL)
```

定義表格時，並無橫列與直欄。把值移入表格內的唯一方法，是將它當做 OUT 或 INOUT 參數來傳給函數，或使用 Net.Data 提供的內建表格函數。DTW\_SQL 語言環境自動地將 SELECT 陳述式的結果放入表格中。

對於非資料庫語言環境，例如 DTW\_REXX 或 DTW\_PERL，語言環境也會負責設定表格值。不過，語言環境 script 或 program 會按著一個資料格接著一個資料格，來定義表格值。請參閱第71頁的『第6章 使用語言環境』，取得語言環境如何使用表格變數的詳細資訊。

您可以參照表格變數名稱，在函數之間傳遞表格。在函數的 REPORT 區塊中可以參照表格的個別元素，或使用 Net.Data 表格函數來參照它們。請參閱第51頁的『表格處理程序變數』，瞭解如何在 REPORT 區塊內，存取表格中的個別元素，以及參閱第60頁的『表格函數』，瞭解如何使用表格函數來存取表格的個別元素。表格變數通常是在 SQL 函數中移入值，然後當做參數傳遞給 SQL 函數或另一個函數，以用來當做報告的輸入。您可以將表格變數當做 IN、OUT 或 INOUT 參數，傳遞給任何非 SQL 函數。表格只能當作 OUT 參數傳給 SQL 函數。

如果您參照一個表格變數，將依據 DTW\_HTML\_TABLE 變數的設定，來顯示表格內容並製作它的格式。在下列範例中，將顯示 myTable 的內容：

```
%HTML (output) {  
  $(myTable)  
}
```

表格中的直欄名稱和欄位值，是以 1 起始的陣列元素來定址。

## 雜項變數

這些變數是 Net.Data 定義的變數，用途如下：

- 影響 Net.Data 處理程序
- 找出函數呼叫的狀態
- 取得資料庫查詢之結果集的相關資訊
- 決定有關檔案位置和日期的資訊

雜項變數可以有 Net.Data 決定的預定值，或您所設定的值。例如，Net.Data 基於目前所處理的檔案，來決定 DTW\_CURRENT\_FILENAME 變數值，而您可以設定 Net.Data 是否要除去定位字元和換行字元所造成的額外空白。

預設變數可用來當做巨集內的變數參照，並提供有關檔案現行狀態、日期、或函數呼叫狀態之資訊。例如，若要取回目前的檔案名稱，您可以使用：

```
%REPORT {
<p>此檔案為 <i>$(DTW_CURRENT_FILENAME)</i>。</P>
}
```

可更改的變數值通常是使用 `DEFINE` 陳述式或 `@DTW_ASSIGN()` 函數來設定，且可讓您影響 `Net.Data` 處理巨集的方式。例如，若要設定是否除去空白，您可以使用下列 `DEFINE` 陳述式：

```
%DEFINE DTW_REMOVE_WS="YES"
```

## 表格處理程序變數

`Net.Data` 定義表格處理變數乃是為了用於 `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。

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

```
%REPORT {
<p>直欄 1 為 <i>$(N1)</i>。</P>
}
```

表格處理程序變數亦提供有關查詢結果的資訊。您可以參照巨集中的變數 `TOTAL_ROWS`，來顯示 `SQL` 查詢傳回的列數，如下列範例所示：

```
所找到的名稱：$(TOTAL_ROWS)
```

有些表格處理程序變數，會被其他變數或內建式函數所影響。例如，`TOTAL_ROWS` 需要啟用 `DTW_SET_TOTAL_ROWS` `SQL` 語言環境變數，以便 `Net.Data` 在處理 `SQL` 查詢的結果集或函數呼叫時，能夠指定 `TOTAL_ROWS` 的值，如下列範例所示：

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
...
```

```
所找到的名稱：$(TOTAL_ROWS)
```

## 報告變數

`Net.Data` 會以預設報表格式顯示巨集所產生的網頁輸出。預設報表格式會顯示在使用 `<PRE>` `</PRE>` 標籤的表格格式中。您可以經由下列方式置換預設報表：以顯示輸出的指示來定義 `REPORT` 區塊，或使用其中一個報表變數來阻止預設報表的產生。

報表變數會協助您自行設定網頁輸出的顯示方式，以及如何搭配預設報表與 `Net.Data` 表格一起使用。您必須先使用 `DEFINE` 陳述式或 `@DTW_ASSIGN()` 函數來定義這些變數，才能使用它們。

報表變數會設定間隔、置換預設報表格式、設定 HTML 表格輸出對預設表格輸出，以及設定其他顯示特性。例如，您可以使用 ALIGN 變數，來控制表格處理程序變數的前端和尾端空間。下列範例使用 ALIGN 變數，對查詢所傳回的列示，以空格來分隔其中每一個直欄名稱。

```
%DEFINE ALIGN="YES"
...
<p>您的查詢是針對這些直欄： $(NLIST)
```

START\_ROW\_NUM 報告變數可讓您決定從何列開始顯示查詢的結果。例如，下列變數值指定 Net.Data 從第三列開始顯示查詢的結果。

```
%DEFINE START_ROW_NUM = "3"
```

您也可以決定 Net.Data 是否要對預設格式使用 HTML 標籤。若 DTW\_HTML\_TABLE 設為 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。

透過這些變數，您可以執行如下的作業：建立資料庫連線、啟用 NLS 支援，以及決定是否已順利執行 SQL 陳述式。

例如，您可以使用 SQL\_STATE 變數，來存取或顯示資料庫傳回的 SQL 狀態值。

```
%FUNCTION (DTW_SQL) val1() {
  select * from customer
%REPORT {
  ...
  %ROW {
  ...
%}
  SQLSTATE=$(SQL_STATE)
%}
```

---

## Net.Data 函數

Net.Data 會提供要在應用程式中使用的內建函數，如字組與字串操作函數，以及取回與設定表格變數函數的函數。例如，您也可以定義要與應用程式一起使用，來呼叫外部程式或儲存程序的函數。

### 使用者定義的函數

例如，您定義要與應用程式一起使用，來呼叫外部程式或儲存程序的那些函數。

### Net.Data 的內建函數

Net.Data 會提供將在應用程式中使用的那些函數，如操作字組與字串的函數，以及取得與設定表格變數的函數。

這些段落將描述下列主題：

- 第53頁的『定義函數』



- 第57頁的『呼叫函數』
- 第57頁的『呼叫 Net.Data 的內建函數』

## 定義函數

若要在巨集中定義自己的函數，請使用 **FUNCTION** 區塊或 **MACRO\_FUNCTION** 區塊：

### FUNCTION 區塊

定義可從 Net.Data 巨集呼叫，且由語言環境來處理的次常式。FUNCTION 區塊必須含有語言陳述式或外部程式的呼叫。

### MACRO\_FUNCTION 區塊

定義可從 Net.Data 巨集呼叫，且透過 Net.Data 而非語言環境來處理的次常式。MACRO\_FUNCTION 區塊可以含有 HTML 區塊中所容許的任何陳述式。

**語法：**使用下列語法來定義函數：

#### FUNCTION 區塊：

```
%FUNCTION(type) function-name([usage] [datatype] parameter, ...) [RETURNS(return-var)] {
    executable-statements
    [report-block]
    ...
    [report-block]
    [message-block]
%}
```

#### MACRO\_FUNCTION 區塊：

```
%MACRO_FUNCTION function-name([usage] parameter, ...) [RETURNS(return-var)] {
    executable-statements
    report-block
    ...
    report-block
%}
```

其中：

*type* 指出架構在起始設定檔中之語言環境。語言環境會呼叫某個特定的語言處理器（處理可執行的陳述式），並提供 Net.Data 和語言處理器之間的標準介面。

#### *function-name*

指定 FUNCTION 或 MACRO\_FUNCTION 區塊的名稱。函數呼叫會指定 *function-name*，它的前面會有一個 @ 符號。有關詳細資料，請參閱第57頁的『呼叫函數』。

您可以使用相同名稱來定義多個 FUNCTION 或 MACRO\_FUNCTION 區塊，以便同時處理它們。每一個區塊皆必須有相同的參數列示。當 Net.Data 呼叫函數時，相同名稱的所有 FUNCTION 區塊或相同名稱的 MACRO\_FUNCTION 區塊，會按它們在 Net.Data 巨集中定義時的次序來執行。

*usage* 指定參數是輸入 (IN) 參數、輸出 (OUT) 參數、或同時兼具這兩種類型 (INOUT)。此指定表示參數是否在 FUNCTION 區塊或 MACRO\_FUNCTION 區塊或兩者上來回傳遞。用法類型適用於參數列示中所有後續的參數，直到被另一個用法類型改變為止。預設類型為 IN。

#### *datatype*

參數的資料類型。有些語言環境預期傳遞的參數是資料類型。例如，當呼叫儲

存程序時，SQL 語言環境會預期它們，這同於呼叫程式時，「直接呼叫」語言環境所預期的一般。請參閱第71頁的『第6章 使用語言環境』，瞭解您正在使用的語言環境的支援資料類型。

#### *parameter*

區域變數的名稱，可換成指定於函數呼叫中之相對應引數的值。例如，可執行之陳述式或 REPORT 區塊中的參數參照 \$(parm1)，可換成參數的實際值。另外，使用該語言的自然語法或當作環境變數使用時，即可將這些參數傳送至語言環境，供可執行的陳述式存取。參數變數參照無法在 FUNCTION 或 MACRO\_FUNCTION 區塊外使用。

#### *return-var*

請在 RETURNS 關鍵字後指定此參數，來識別特殊的 OUT 參數。傳回變數的值是在函數區塊中指定的，且它的值會傳回到巨集中呼叫該函數之處。例如，在下列句子中，`<p>My name is @my_name()., @my_name()` 會被傳回變數的值所取代。如果未設定 RETURNS 子句，函數呼叫的值為：

- 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。

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47頁的『執行變數』。

對於 MACRO\_FUNCTION 區塊，可執行的陳述式是文字和 Net.Data 巨集語言結構的組合。此情況下，不會包括任何語言環境，因為 Net.Data 扮演語言處理器的角色，且會處理可執行的陳述式。

#### *report-block*

定義一個或多個 REPORT 區塊，來處理 FUNCTION 或 MACRO\_FUNCTION 區塊的輸出。請參閱第62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55頁的『訊息區塊』。

在 Net.Data 巨集中呼叫函數之前，請先在任何其他區塊外定義它們。

## 在函數中使用特殊字元

當符合 Net.Data 語言結構語法的字元在函數區塊的語言陳述式區段中使用，作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.pr1 %}  
%}
```

- 使用變數參照來設定 %{ 字元：

```
%define percent_openbrace = "%{"  
  
%FUNCTION(DTW_PERL) func() {  
    ...  
    for $num_words (sort by number keys $(percent_openbrace) $Rtitles{$num} ) {  
        &make_links($Rtitles{$num}{$num_words});  
    }  
    ...  
%}
```

## 訊息區塊

MESSAGE 區塊可讓您決定在順利完成 (或未順利完成) 函數呼叫後，要如何繼續處理，並且可讓您顯示相關資訊給函數呼叫者。當處理訊息時，Net.Data 會為呼叫 FUNCTION 區塊的每一個函數，設定一個語言環境變數 RETURN\_CODE。不會在 MACRO\_FUNCTION 區塊的函數呼叫上設定 RETURN\_CODE。

MESSAGE 區塊是由一系列的訊息陳述式所組成，每個陳述式都設有回覆碼值、訊息文字與要執行的動作。有關 MESSAGE 區塊的語法，請參閱 *Net.Data 參考手冊* 中語言結構一章。

MESSAGE 區塊的範圍可以是廣域或區域。若 MESSAGE 區塊是定義在 FUNCTION 區塊中，則對該 FUNCTION 區塊來說其範圍是區域性的。若它是指定在最外層的巨集上，則 MESSAGE 區塊的範圍是廣域的，且可供在 Net.Data 巨集中執行的所有函數呼叫使用。如果您定義多個廣域 MESSAGE 區塊，則會採用最近一次定義的。

Net.Data 會採用下列的規則，來處理從某函數呼叫所傳回的 RETURN\_CODE 變數之值：

1. 檢查區域 MESSAGE 區塊是否有完全相符的；然後根據所指定的，看是要跳出或繼續。

2. 如果 RETURN\_CODE 不是 0，請檢查區域 MESSAGE 區塊是否為 +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。請在函數名稱或巨集函數名稱之後使用 `@` 字元：

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

*function\_name*

這是要呼叫的函數或巨集函數的名稱。除非該函數為內建函數，否則此函數必須已定義在 `Net.Data` 巨集中。

*argument*

這是變數、引號內的字串、變數參照或函數呼叫的名稱。函數呼叫上的引數會符合函數或巨集函數參數列示上的參數。此外，當處理函數或巨集函數時，每一個參數均會指定為它的對應引數的值。引數的數目和類型，必須與相對應參數的相同。

作為引數的引號內的字串可以含有變數參照及函數呼叫。

**範例 1：**具有字串引數的函數呼叫

```
@myFunction("abc")
```

**範例 2：**具有變數及函數呼叫引數的函數呼叫

```
@myFunction(myvar, @DTW_rADD("2","3"))
```

**範例 3：**具有含有變數參照及函數呼叫的字串的函數呼叫

```
@myFunction("abc$(myvar)def@DTW_rADD("2","3")ghi")
```

## 呼叫 `Net.Data` 的內建函數

`Net.Data` 提供一大組內建函數，可幫助您簡化 Web 頁面的開發。這些函數已經過 `Net.Data` 的定義，所以您不需要定義它們。您可以如同呼叫其他函數一般，來呼叫這些函數。

第58頁的圖7 顯示 `Net.Data` 內建函數與巨集的互動方式。

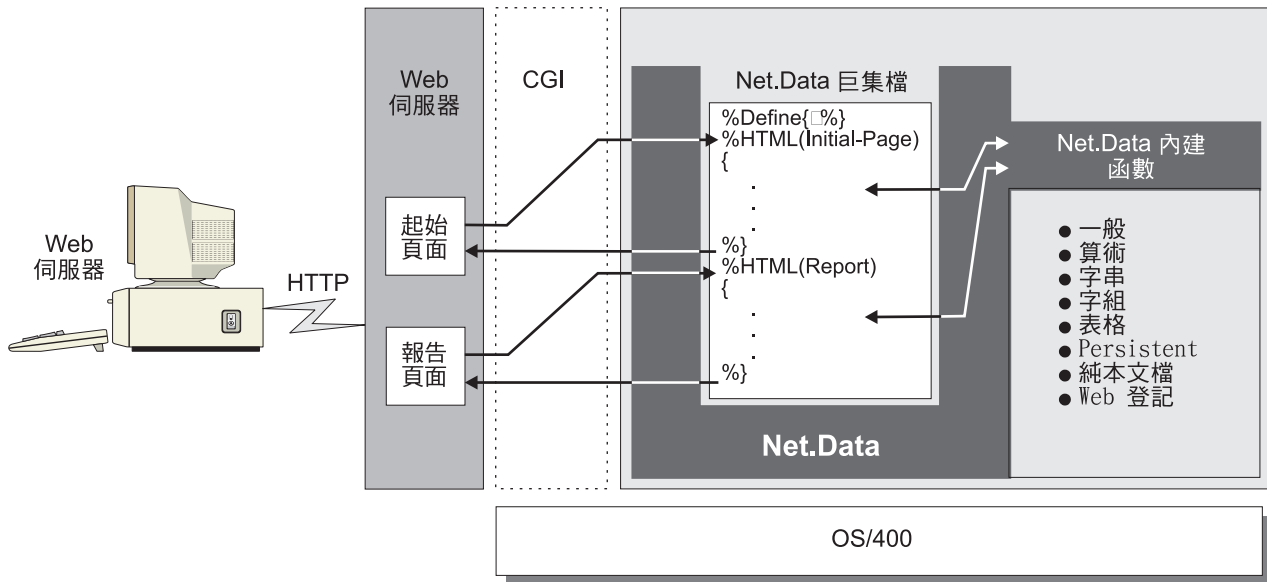


圖 7. Net.Data 內建函數

內建函數可採三種方式傳回其結果，取決於它的字首：

- **DTW\_、DTWF\_ 及 DTWR\_**：以輸出參數傳回呼叫的結果，或未傳回任何結果。(DTWF\_ 是指純本文檔函數的字首。DTWR\_ 是指 Web 登記函數的字首。)
- **DTW\_r、DTWF\_r 及 DTWR\_r**：函數呼叫的結果會置換巨集中的函數呼叫，同樣地，RETURNS 關鍵字值也會置換使用者定義函數的函數呼叫，此函數已指定 RETURNS 關鍵字。
- **DTW\_m**：透過傳給函數的每一個參數傳回多個結果。

有些內建函數沒有所屬類型。若要判斷具有哪種類型的特殊內建函數，請參閱 *Net.Data* 參考手冊中有關 Net.Data 內建函數的那一章。

下列段落將提供 Net.Data 內建函數的高階概觀。請使用這些函數來執行一般目的、算術、字串、字組或表格操作函數。此外，您可以使用持續函數來進行異動處理。請參閱 *Net.Data* 參考手冊，取得每一函數的說明及其語法和範例。這些函數中有一些要求變數在它們被使用前就被設定，方可使用它們，或須在特定上下文中使用它們。並非所有作業系統均支援每一個內建函數。請參閱 *Net.Data* 參考手冊，您的作業系統支援哪些函數。

- 第59頁的『一般目的函數』
- 第59頁的『算術函數』
- 第59頁的『字串函數』
- 第60頁的『字組函數』
- 第60頁的『表格函數』
- 第60頁的『純本文檔函數』
- 第60頁的『Web 登記函數』
- 第61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。 您可以使用它們來傳送郵件、處理 HTTP cookie、建立 HTML 跳出碼 (escape code)，以及從系統中取得其他有用的資訊。

例如，若要設定當發生特定狀況時，Net.Data 應跳出巨集，不處理巨集的其餘部份，請使用 DTW\_EXIT 函數：

```
%HTML(cache_example) {  
  
<html>  
<head>  
  <title>這是頁面標題</title>  
</head>  
<body>  
  <center>  
    <h3>這是主要標題</h3>  
    <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
    <! Joe Smith 看到一篇非常短的頁面 !>  
    <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
    %IF (customer == "Joe Smith")  
      </body>  
</html>  
  
@DTW_EXIT()  
  
%ENDIF  
  
...  
  
  </body>  
</html>  
%}
```

另一個有用函數就是 DTW URLESCSEQ 函數，它會以它們的跳出值來置換 URL 中不容許的字元。例如，如果輸入變數 string1 等於 "Guys & Dolls"，則 DTW URLESCSEQ 會將輸出變數指定為值 "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_POWER("2", "-3", result)
```

DTW\_POWER 會在變數 result 中傳回 ".125"

## 字串函數

這些函數可讓您處理字串中的字元。您可以變更字串的大小寫，插入或刪除字元，指定一個字串值給另一個變數，以及其他有用的函數。

例如，您可以使用 DTW\_ASSIGN 將輸入變數的值指定給輸出變數。您亦可以使用這個函數在巨集中變更變數。在下列範例中，變數 RC 被指定為 0。

```
@DTW_ASSIGN(RC, "0")
```

其他字串函數包括連接字串的 `DTW_CONCAT` 及在特定位置插入字串的 `DTW_INSERT`，以及許多其他字串操作函數。

## 字組函數

這些函數可讓您處理字串中的字組。這些函數中的絕大部份，乃與字串函數的作用類似，只不過其是以整個字組運作。例如，可讓您計算字串中的字組數、除去字組、從字串中找出某個字組。

例如，使用 `DTW_DELWORD` 從字串中刪除設定的字數：

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

`DTW_DELWORD` 傳回字串 "Now time"。

其他字組函數包括傳回字組中的字元數的 `DTW_WORDLENGTH` 及傳回字串內字組位置的 `DTW_WORDPOS`。

## 表格函數

您可以使用這些函數來產生報告或套表，以便使用 `Net.Data table` 變數中的資料。您也可以使用這些函數，來建立 `Net.Data` 表格，以及操作及取回那些表格中的值。表格變數含有一組值，以及相關的直欄名稱。這些函數方便您將整群的值傳給函數。

例如，`DTW_TB_APPENDROW` 會附加一列到表格中。在下列範例中，`Net.Data` 會附加 10 列到表格 `myTable` 中：

```
@DTW_TB_APPENDROW(myTable, "10")
```

此外，`DTW_TB_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，以便另一個要求無法讀取或更新檔案。當 `Net.Data` 完成該檔案時，`DTWF_CLOSE` 即會釋放它，以容許其他要求存取檔案。

## Web 登記函數

使用 `Web` 登記函數來維護登記和其所包含之登錄。`Web` 登記是一個由 `Net.Data` 維護密碼的檔案，可讓您輕易地新增、取回與刪除登錄。

例如，`DTWR_ADDENTRY` 可新增登錄，而 `DTWR_DELEENTRY` 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交易內的變更。

例如，DTW\_ACCEPT 會識別交易的交易 handle，而 DTW\_TERMINATE 則會識別交易中的最後一個 HTML 區塊。DTW\_RTVHANDLE 會對交易中的區塊建立唯一的交易 handle。在交易期間，您可以使用 DTW\_COMMIT 與 DTW\_ROLLBACK，來起始確定與取消。

詳細資訊，請參閱第95頁的『第7章 透過持續巨集的異動管理』。亦請參閱 *Net.Data* 參考手冊中有關內建函數的那一章，來取得有效持續函數列示及其語法和範例。

---

## 在巨集中建立網頁

*Net.Data* 可讓您輕易地將標準網頁呈現在應用程式使用者的瀏覽器上。下列段落描述巨集的 HTML 和 REPORT 區塊，並顯示如何在 *Net.Data* 巨集中製作網頁的格式。有關這些區塊的語法資訊，請參閱 *Net.Data* 參考手冊中有關語言結構的那一章。

## HTML 區塊

*Net.Data* 巨集包含 HTML 區塊，這些區塊可建立 Web 瀏覽器的文字呈現陳述式，如 HTML。在巨集中，您必須至少設定一個 HTML 區塊，但若您想要的話，可以設定多個。每一個 HTML 區塊會在瀏覽器中建立單一網頁。每一次呼叫 *Net.Data* 時，它僅會處理一個 HTML 區塊。若要建立一個由多個網頁組成的應用程式，您可以呼叫 *Net.Data* 多次，使用標準的導引技術（如鏈結與套表）來處理 HTML 區塊。

任何有效的文字呈現陳述式（如 HTML 或 JavaScript）可以出現在 HTML 區塊中。另外，您可在 HTML 區塊中使用 INCLUDE 陳述式、函數呼叫及變數參照。下列範例顯示在 *Net.Data* 巨集中使用 HTML 區塊的一般方式：

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT) {
<H1>硬體查詢套表</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/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>這是您要求的列示：</B><BR>
%ROW{
<HR>
```



```
$(N1): $(V1)    $(N2): $(V2)
<P>
$(V3)
%}
%}
%}

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

您可以從類似下面的 HTML 鏈結中來呼叫 Net.Data 巨集：

```
<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.d2w/input">List of hardware</a>
```

當應用程式使用者按一下此鏈結時，Web 瀏覽器會呼叫 Net.Data，且 Net.Data 會解析巨集。當 Net.Data 開始處理呼叫時指定的 HTML 區塊後，在本例中為 HTML(INPUT) 區塊，就會開始處理區塊中的文字。若有任何事物無法被 Net.Data 辨識為 Net.Data 巨集語言結構，它將傳送至瀏覽器來顯示。

當使用者做好選擇並按「提出」按鈕後，Net.Data 即開始執行 HTML FORM 元素的 ACTION 部份，它指定呼叫 Net.Data 巨集的 HTML(OUTPUT) 區塊。然後，Net.Data 會以 HTML(INPUT) 區塊的方式來處理 HTML(OUTPUT) 區塊。

然後，Net.Data 會處理 myQuery() 函數呼叫，該呼叫接著再呼叫 SQL FUNCTION 區塊。當 SQL 陳述式中的 \$(hardware) 變數參照，被換成從輸入格式中傳回的值後，Net.Data 即開始查詢。此時，Net.Data 會再度處理報表，並根據 REPORT 區塊中設定的呈現文字陳述式來顯示查詢結果。

在 Net.Data 完成 REPORT 區塊處理程序之後，即會傳回 HTML(OUTPUT) 區塊，並完成處理程序。

## 報告區塊

REPORT 區塊可用來顯示 FUNCTION 區塊所輸出的資料並加以格式化。雖然您所指定的可能是文字、巨集變數參照與函數呼叫等任何有效的組合，但一般來說，此種輸出類型通常是套表資料。您可以選擇在 REPORT 區塊中指定表格名稱。如果您未指定表格名稱，則 Net.Data 將使用來自 FUNCTION 參數列示中第一個輸出表格的表格資料。

REPORT 區塊是由三個可選用的部份所組成：

- 表頭資訊，它含有在表格橫列資料之前顯示一次的文字。
- ROW 區塊，它含有對結果表格的每一列顯示一次的文字和表格變數。
- 標底資訊，它含有在表格橫列資料之後顯示一次的文字。

範例：

```
%REPORT{
<H2>查詢結果</H2>
<P>選取名稱以顯示明細。
<TABLE BORDER=1>
  <TR>
    <TD>Name</TD>
    <TD>Location</TD></TR>
%ROW{
  <TR>
    <TD>
```



```

        <a href="/cgi-bin/db2www/name.d2w/details?name=$(V1)&location;=$(V2)">$(V1)</a>
    </TD>
</TD>$(V2)</TD>
</TR>
%}
</TABLE>
%}

```

## REPORT 區塊指南

當建立 REPORT 區塊時，請使用下列指南：

- 若要避免顯示來自 ROW 區塊的任何表格輸出，請將 ROW 區塊空白，或乾脆整個省略掉。
- 請在 REPORT 區塊內使用 Net.Data 提供的變數，來存取 Net.Data 巨集結果表格中的資料。這些變數說明於 第51頁的『表格處理程序變數』。詳細資訊，請參閱 *Net.Data* 參考手冊中有關「報告變數」的那一節。
- 若要提供標題和標底資訊，請在 ROW 區塊的前後提供文字。Net.Data 將在 ROW 區塊之前找到的一切事物都當成標題資訊來處理。Net.Data 將在 ROW 區塊之後找到的一切事物都當成註腳資訊來處理。如同 HTML 區塊一樣，Net.Data 會將標題、ROW 及註腳區塊中無法辨識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。
- 您可以呼叫 REPORT 區塊中的函數及參照函數。
- 若要讓 Net.Data 使用預先格式化文字來列印預設報告，請勿在巨集中併入 REPORT 區塊。下列範例顯示預設報告格式：

```

SHIPDATE | RECDATE | SHIPNO |
-----|-----|-----|
25/05/1997 | 30/05/1997 | 1495194B |
-----|-----|-----|
25/05/1997 | 28/05/1997 | 2942821G |
-----|-----|-----|

```

- 若要使用 HTML 標籤來代替預先格式化文字，請設定 DTW\_HTML\_TABLE 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。它會在表格 CustomerTbl 中，顯示姓名、電話號碼、以及傳真號碼：

```

%DEFINE SET_TOTAL_ROWS="YES"
...
%FUNCTION(DTW_SQL) custlist() {
    SELECT Name, Phone, Fax FROM CustomerTbl
%REPORT{
<I>電話查詢結果：</I>
<BR>
=====
<BR>
%ROW{
    姓名： <B>$(V1)</B>
<BR>
    電話： $(V2)
<BR>

```

```

傳真：$(V3)
<BR>
-----
<BR>
    %}
    取回的記錄總數：$(TOTAL_ROWS)
    %}
    %}
    %}

```

查詢報告在瀏覽器中看來如下：

電話查詢結果：

=====

姓名： **Doen, David**  
 電話：422-245-1293  
 傳真：422-245-7383

-----

姓名： **Ramirez, Paolo**  
 電話：955-768-3489  
 傳真：955-768-3974

-----

姓名： **Wu, Jianli**  
 電話：525-472-1234  
 傳真：525-472-1234

-----

取回的記錄總數：3

Net.Data 會以下列方式來產生報告：

1. 在報告開頭列印一次電話查詢結果：。此本文以及區隔符號行均是 REPORT 區塊的標題部份。
2. 在取回的每一橫列上，對變數 V1、V2 及 V3，分別以「姓名」、「電話」及「傳真」的值來置換它們。
3. 在報告尾端列印一次字串取回的記錄總數：和 TOTAL\_ROWS 的值。（這個本文是 REPORT 區塊的註腳部份。）

## 多個 REPORT 區塊

您可以在單一 FUNCTION 或 MACRO FUNCTION 區塊內，設定多個 REPORT 區塊，建立多個具有一個函數呼叫的報表。

一般而言，您將搭配 DTW\_SQL 語言環境與多個 REPORT 區塊一起使用，而這個語言環境具有一個函數，可呼叫會傳回多個結果集合的儲存程序 (請參閱 第86頁的『儲存程序』)。不過，多個 REPORT 區塊可與任一語言環境一起用來建立多個報表。

若要使用多個 REPORT 區塊，請將 Net.Data 表格變數置於函數參數列示中。如果從儲存程序傳回的結果集合的數目比您設定的 REPORT 報表區塊數目還要多，且如果 Net.Data 內建函數 DTW\_DEFAULT\_REPORT = "MULTIPLE"，將對與 REPORT 區塊沒有關聯的每一結果集合建立預設報表。如果未設定任何報表區塊，且如果 DTW\_DEFAULT\_REPORT = "YES"，將僅建立一個預設報表。請注意，僅適用於 SQL 語言環境，DTW\_DEFAULT\_REPORT 值 "YES" 等於 "MULTIPLE" 的值。

**範例：** 下列範例描述您可以用哪些方式使用多個報表區塊。

**使用預設報表格式顯示多個報表：**

**範例 1：** DTW\_SQL 語言環境

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION (dtw_sql) myStoredProc (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：** MACRO\_FUNCTION 區塊。在這個範例中，這兩個表格會傳遞到 MACRO\_FUNCTION 區塊。因為設定了 DTW\_DEFAULT\_REPORT="MULTIPLE"，所以 Net.Data 僅會顯示第一個表格的預設報表。當指定 MULTIPLE 時，將會對這兩個表格建立預設報表。

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%MACRO_FUNCTION multReport (INOUT tablename1, tablename2) {
%}
```

在這個範例中，這兩個表格會傳遞到 MACRO\_FUNCTION multReport。再一次，Net.Data 會依據這兩個表格出現在 MACRO FUNCTION 區塊參數列示中的次序，來顯示它們的預設報表，首先 table1，然後 table2。

**範例 3：** DTW\_REXX 語言環境

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (dtw_rexx) multReport (INOUT table1, table2) {
    SAY '正在建立多個預設報表...<BR>'
%}
```

在這個範例中，這兩個表格會傳遞到 REXX 函數 multReport。因為設定了 DTW\_DEFAULT\_REPORT="YES"，所以 Net.Data 僅會顯示第一個表格的預設報表。

**經由設定要進行顯示處理的 REPORT 區塊來顯示多個報表：**

**範例 1：** 已命名的 REPORT 區塊

```
%FUNCTION(dtw_sql) myStoredProc (OUT table1, table2) {
    CALL myproc (table1, table2)

    %REPORT(table2) {
        ...
        %ROW { .... %}
        ...
    %}

    %REPORT(table1) {
        ...
        %row { .... %}
        ...
    %}
%}
```

在這個範例中，已對 FUNCTION 區塊參數列示中傳遞的這兩個表格設定了 REPORT 區塊。表格會依據它們在 REPORT 區塊上設定的次序來顯示，首先 table2，然後 table1。經由在 REPORT 區塊上設定表格名稱，您可以控制報表的顯示次序。

**範例 2：** 沒有名稱的 REPORT 區塊

```
%FUNCTION(dtw_sql) myStoredProc (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，所以會依據這兩個表格從儲存程序傳回的次序，來顯示它們的報表。

**使用預設報表與 REPORT 區塊的結合來顯示多個報表：**

**範例：**預設報表與 REPORT 區塊的結合

```

%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION(dtw_system) editTables (INOUT table1, table2, table3) {
    %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 區塊的數目少於從儲存程序傳回的結果集合數目，所以將顯示剩餘的結果集合的報表：首先顯示 table1 的預設報表，然後顯示 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：**

- 您可以對每一結果集合或表格名稱指定一個或多個 REPORT 區塊。對 REPORT 區塊指定的名稱必須符合 FUNCTION 區塊參數列示中的對應結果集合名稱或表格名稱參數。
- 依多個表格的處理次序來設定它們的 REPORT 區塊。
- 當未設定表格的 REPORT 區塊時，若要設定預設處理，請定義 DTW\_DEFAULT\_REPORT = "MULTIPLE"。當 Net.Data 建置網頁時，它會先顯示含有 REPORT 區塊的表格的報表，再顯示表格的預設報表。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。
- 多個報表區塊可以與任何語言環境一起使用。

**限制：**

- 函數中所有報表變數的值將套用到該函數中的所有 REPORT 區塊。您無法替個別的 REPORT 區塊來修改報告變數的值。
- MESSAGE 區塊必須位於 REPORT 區塊的前或後，不可位於 REPORT 區塊之間。
- 表格變數在傳遞到函數之前，必須在 TABLE 陳述式內定義它們。
- 如果第一個報表區塊指定表格名稱，則所有報表區塊均須指定表格名稱。
- 如果第一個報表區塊未指定表格名稱，則所有報表區塊不可指定表格名稱。

## 巨集中的條件式邏輯和迴路

Net.Data 可讓您使用 IF 和 WHILE 區塊，在您的 Net.Data 巨集中納入條件式邏輯和迴路。

IF 區塊及 WHILE 區塊會使用一個條件列示，協助您測試一個或數個條件，然後依據條件測試的結果，來執行陳述式區塊。條件列示含有邏輯運算子，如 = 及 <+，以及函數項，這些會構成引號內的字串、變數、變數參照及函數呼叫。引號內的字串可以含有變數參照及函數呼叫。您可以將條件列示集狀化。

下列段落將描述條件性邏輯及迴路：

- 『條件性邏輯：IF 區塊』
- 第69頁的『迴路結構：WHILE 區塊』

### 條件性邏輯：IF 區塊

在 Net.Data 巨集中，請使用 IF 區塊來執行條件式處理。該 IF 區塊類似於大部份高階語言中的 IF 陳述式，因其提供測試一個或數個條件，然後基於條件測試的結果，來執行陳述式區塊的能力。

您幾乎可於巨集中的任意處設定 IF 區塊，且可以使用巢狀。有關 IF 區塊的語法，請參閱 *Net.Data 參考手冊*中有關語言結構的那一章。

**IF 區塊規則：**IF 區塊語法的規則，是由區塊在巨集中的位置所決定。IF 區塊的可執行區塊陳述式中，容許的元素是基於 IF 區塊本身的位置而定。

- 在含有 IF 區塊的區塊中有效之任何元素，在該 IF 區塊之內亦有效。例如，若您在 HTML 區塊中設定 IF 區塊，則 HTML 區塊中容許的任何元素，在 IF 區塊中也容許，例如 INCLUDE 陳述式和 WHILE 區塊。

```
%HTML 區塊
...
%IF 區塊
...
%INCLUDE
...
%WHILE
...
%ENDIF
%}
```

- 同樣地，若於 Net.Data 巨集之宣告部份的其他區塊以外設定 IF 區塊，則只有其他區塊 (例如 DEFINE 區塊或 FUNCTION 區塊) 之外容許的那些元素，在 IF 區塊中才容許。

```
%IF
...
%DEFINE
...
%FUNCTION
...
%ENDIF
```

- 當某個 IF 區塊巢狀於另一個 IF 區塊內，而後者位於宣告部份中其他區塊之外時，可以使用外部區塊所使用的任何元素。當某個 IF 區塊巢狀於另一個區塊內，而後者位於某個 IF 區塊中時，請使用所在區塊的語法規則。

例如，巢狀 IF 區塊必須遵循當它在 HTML 區塊之中時的規則。

```
%IF
...
%HTML {
...
%IF
...
%ENDIF
%}
...
%ENDIF
```

**例外狀況：**請勿在 IF 區塊中設定 ROW 區塊。

## IF 區塊字串比較

Net.Data 基於構成條件的函數項內容所產生的方式之一，來處理 IF 區塊條件列示。預設動作是將所有函數項視為字串，並依條件所示執行字串比較。不過，如果在兩個代表整數的字串之間作比較，這是一種數值比較。如果字串僅含有數字，且選用性在前面有 '+' 或 '-' 字元，則 Net.Data 將假設它為數字。字串不能包含 '+' 或 '-' 以外的任何非數字字元。Net.Data 不支援非整數數字的數值比較。

**有效整數字串的範例：**

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

**無效整數字串的範例：**

```
- 20      (包含空白字元)
234,000   (包含逗號)
57.987    (包含小數點)
```

Net.Data 於執行 IF 條件時會評估該區塊，這個時間可不同於 Net.Data 原來讀取它的時間。例如，若您在 REPORT 區塊中設定 IF 區塊，則 Net.Data 在讀取含有 REPORT 區塊的 FUNCTION 區塊定義時，不會評估與 IF 區塊相關的條件列示，而是在呼叫並執行函數時才評估。這在 IF 區塊的條件列示部份和要執行的區塊陳述式兩者皆是如此。

**IF 區塊範例：**在其他區塊內含有 IF 區塊的巨集

```
{ 此巨集是從另一個巨集被呼叫，在此套表資料中
  傳送作業系統及版本變數。
}
```

```
%IF (platform == "AS400")
%IF (version == "V3R2")
%INCLUDE "as400v3r2_def.hti"
%ELIF (version == "V3R7")
```

```

        %INCLUDE "as400v3r7_def.hti"
    %ELIF (version == "V4R1")
        %INCLUDE "as400v4r1_def.hti"
    %ENDIF
    %ELSE
        %INCLUDE "default_def.hti"
    %ENDIF
    %MACRO_FUNCTION numericCompare(IN term1, term2, OUT result) {
    %IF (term1 < term2)
        @dtw_assign(result, "-1")
    %ELIF (term1 > term2)
        @dtw_assign(result, "1")
    %ELSE
        @dtw_assign(result, "0")
    %ENDIF
    %}

    %HTML(report){
        %WHILE (a < "10") {
            外部 while 迴路 #$(a)<BR>
            %IF (@dtw_rdivrem(a,"2") == "0")
                這是奇數迴路<BR>
            %ENDIF
            @DTW_ADD(a, "1", a)
        %}
    %}

```

## 迴路結構：WHILE 區塊

使用 WHILE 區塊，於 Net.Data 巨集中執行迴路。如同 IF 區塊，WHILE 區塊提供測試數個條件，然後基於條件測試的結果，來執行陳述式區塊的能力。不像 IF 區塊，陳述式區塊可基於條件測試的結果，不限次數地執行。

您可以在 HTML 區塊、REPORT 區塊、ROW 區塊、MACRO\_FUNCTION 區塊及 IF 區塊內指定 WHILE 區塊，並且可以使用巢狀。有關 WHILE 區塊的語法，請參閱 *Net.Data* 參考手冊中有關語言結構的那一章。

Net.Data 處理 WHILE 區塊的方式，與處理 IF 區塊的方式完全相同，但會每次執行區塊後會重新評估條件。而且，如同其他條件式迴路結構一樣，若撰寫條件不正確，則處理程序也可能會陷入無限迴路內。

**範例：**具有 WHILE 區塊的巨集

```

%DEFINE loopCounter = "1"

%HTML(build_table) {
%WHILE (loopCounter <= "100") {
    %{ generate table tag and column headings %}
    %IF (loopCounter == "1")
        <TABLE BORDER>
        <TR>
        <TH>Item #
        <TH>說明
    %ENDIF

    %{ 產生個別橫列 %}
    <TR>
    <TD>$(loopCounter)

```

```
<TD>@getDescription(loopCounter)

%{ 產生結束表格標籤 %}
%IF (loopCounter == "100")
%ENDIF

%{ 增加迴路計數 %}
@DTW_ADD(loopCounter, "1", loopCounter)
%}
%}
```



## 第6章 使用語言環境

Net.Data 會提供一些語言環境，您可以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。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 語言環境介面參考手冊*。

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

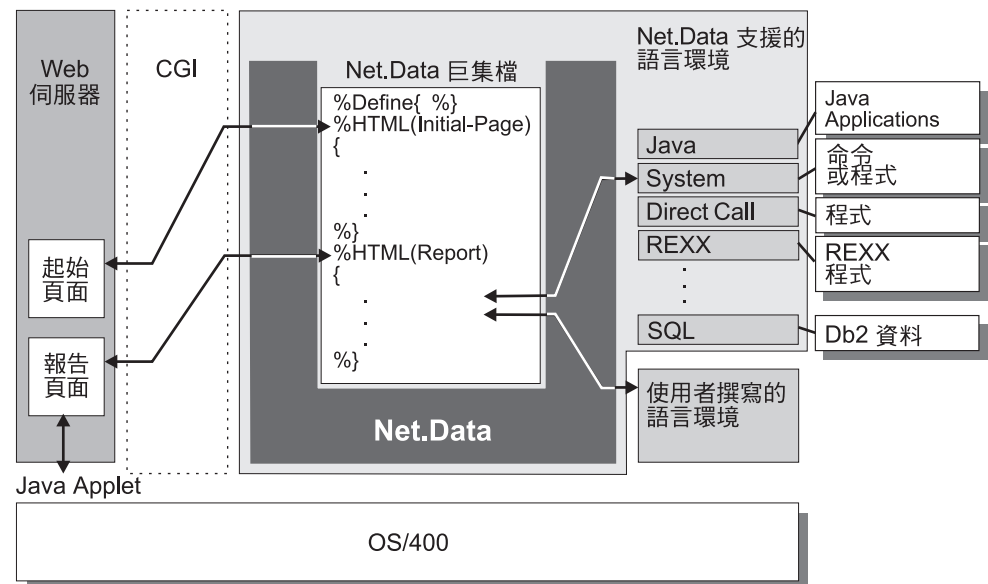


圖 8. Net.Data 語言環境

下列段落將描述 Net.Data 語言環境，以及如何在您的巨集中使用它們：

- 第72頁的『Net.Data 提供的語言環境的概觀』
- 第72頁的『呼叫語言環境』
- 第73頁的『直接呼叫語言環境』
- 第76頁的『Java 應用程式語言環境』
- 第77頁的『REXX 語言環境』
- 第80頁的『SQL 語言環境』
- 第92頁的『系統語言環境』

關於使用語言環境時如何增進執行效能的資訊，請參閱第105頁的『將語言環境最佳化』。

## Net.Data 提供的語言環境的概觀

Net.Data 會提供語言環境，讓您存取資料及程式設計資源，供您的應用程式使用。

表3提供每一語言環境的簡短說明。

表 3. *Net.Data* 語言環境

語言環境	環境名稱	說明
直接呼叫	DTW_DIRECTCALL	「直接呼叫」語言環境支援外部程式的呼叫，這些程式是使用高階程式設計語言，如 RPG、COBOL 及 C/C++ 所撰寫而成的。
Java 應用程式	DTW_JAVAPPS	Net.Data 會支援具有 Java 語言環境的舊有 Java 應用程式。
REXX	DTW_REXX	REXX 語言環境可解譯 Net.Data 巨集的 FUNCTION 區塊中指定的內部 REXX 程式，或可以執行儲存於個別檔案中的外部 REXX 程式。
SQL	DTW_SQL	SQL 語言環境會透過 DB2 來執行 SQL 陳述式。可在表格變數中傳回 SQL 陳述式的結果。
系統	DTW_SYSTEM	系統語言環境支援執行指令及呼叫外部程式。

## 呼叫語言環境

若要呼叫語言環境：

- 使用 FUNCTION 陳述式，來定義一個可呼叫語言環境的函數。
- 使用語言環境的函數呼叫。

例如：

```
%FUNCTION(DTW_SQL) custinfo() {  
  select customer, custno from customer.data  
  %}  
...  
%HTML(REPORT){  
  @custinfo()  
  %}
```

## 處理錯誤狀況

在語言環境函數中偵測到錯誤時，語言環境將會用錯誤碼設定 Net.Data RETURN\_CODE 變數。

您可以使用下列資源，來處理錯誤狀況：

- Net.Data 提供的語言環境會傳回錯誤碼，它的說明在 *Net.Data 訊息與訊息碼參考手冊*中。
- 資料庫語言環境 (如 SQL 語言環境) 會將 RETURN\_CODE 設定為資料庫管理系統 (DBMS) 傳回的錯誤碼，稱為 SQLCODE。請參閱 DBMS 的訊息與訊息碼文件，來瞭解 DBMS 所使用的 SQLCODE。

## 安全

確定 *Net.Data* 執行時所用的使用者 ID 具有適當的權限，可存取語言環境陳述式的目標所參照的任何物件。例如，SQL 語言環境會執行 SQL 陳述式，而 SQL 陳述式會存取資料庫檔案，所以 *Net.Data* 執行時所用的使用者 ID 須具有資料庫檔案的權限。

## 直接呼叫語言環境

「直接呼叫」語言環境可讓您呼叫以高階程式設計語言 (如 C、RPG、COBOL 及 CL) 撰寫而成的程式。參數可傳遞到程式，而參數值可從程式接收，如此使得程式輕易地與 *Net.Data* 整合在一起，且容許使用者使用舊有程式設計技巧，來編寫複雜的企業邏輯。

## 呼叫程式

若要呼叫一個指令，請定義一個函數，它將使用「直接呼叫 (DTW\_DIRECTCALL)」語言環境，並包括將在 EXEC 陳述式中呼叫的程式的路徑。例如：

```
%function(DTW_DIRECTCALL) dc1() {  
    %EXEC { /QSYS.LIB/NETDATA.LIB/MYPGM.PGM %}  
%}
```

如果您使用 EXEC\_PATH 架構變數來定義含有程式的目錄的路徑，則您可以縮短程式的路徑。請參閱第14頁的『EXEC\_PATH』，以學習如何定義 EXEC\_PATH 架構變數。

### 支援的語言環境變數

「直接呼叫」語言環境支援 DTW\_PAD\_PGM\_PARMS 變數，這個變數指出將傳遞到程式的參數是否要以空白填補，直到符合指定的精確度。請參閱 *Net.Data* 參考手冊，取得這個變數的說明、語法及範例。請參閱『傳遞參數到程式』，取得如何傳遞參數到程式的詳細資訊。

## 傳遞參數到程式

經由在函數定義上指定參數類型，以及指定將傳遞的參數是僅輸入 (IN)、僅輸出 (OUT) 或輸入/輸出 (INOUT) 參數，將參數傳遞到程式。例如：

```
%function(DTW_DIRECTCALL) dc2(IN CHAR(3) p1,  
                                INOUT INTEGER p2,  
                                OUT DECIMAL(7,2) p3) {  
    %EXEC { /QSYS.LIB/NETDATA.LIB/MYPGM.PGM %}  
%}
```

在上面範例中，「直接呼叫」語言環境將傳遞這三個參數到程式 MYPGM：字元變數、整數及聚集十進位變數。您最多可傳遞 50 個參數到呼叫的程式。僅有以資料類型指定的參數才能傳遞到程式。「直接呼叫」語言環境會將對應到參數的字串，轉換為資料類型的內部表示式。然後，語言環境會按照函數定義上指定的次序，將變數的內部表示式的指標傳遞到呼叫的程式。

因為變數的指標將傳遞到程式，所以程式可以變更變數的值。不過，僅有 OUT 或 INOUT 變數，在經過程式變更後，才會向呼叫語言環境的巨集反映。

# 支援的資料類型

表4會列示「直接呼叫」語言環境支援的資料類型。並非所有資料類型均受到每一高階語言的支援。

表 4. 直接呼叫資料類型

資料類型	用法注意事項
CHAR( <i>n</i> ) CHARACTER( <i>n</i> ) CHARACTER	字串。如果指定 <i>n</i> ，它必須大於 0 的值。如果未指定字串，將假設為一個字元。因為所有從「直接呼叫」語言環境傳遞的字串均是以空值終止，所以語言環境將配置 <i>n</i> +1 個位元組 ( 1 個位元組供 NULL (空值) 終止符使用)。超出 <i>n</i> 的字串將被截斷。
VARCHAR( <i>n</i> )	可變長度字串，其中 <i>n</i> 是大於 0，小於或等於 32740 的值。字串是空值終止，所以語言環境會配置 <i>n</i> +2+1 個位元組 (2 個位元組用於儲存字串長度， 1 個位元組用於儲存空值終止符)。超出 <i>n</i> 的字串將被截斷。 字串的前兩個位元組含有字串長度 (二進位值)。如果參數定義為 OUT (僅輸出)，則在變數傳遞到呼叫的程式之前，字串長度將設定為 0。
INTEGER INT	帶正負號的二進位整數，長度為 4 個位元組。
SMALLINT	帶正負號的二進位整數，長度為 2 個位元組。
FLOAT( <i>p,s</i> )	單精準度或倍精準度浮點數。對於單精準度， <i>p</i> 必須大於 0，小於 25。對於倍精準度， <i>p</i> 必須大於 24，小於 54。僅在轉換資料為可顯示的格式 (如轉換為字串) 時，才會使用精確度 ( <i>p</i> ) 及小數位數 ( <i>s</i> )。
REAL( <i>p,s</i> )	單精準度浮點數。 <i>p</i> 必須大於 0，小於 25。僅在轉換資料為可顯示的格式 (如轉換為字串) 時，才會使用精確度 ( <i>p</i> ) 及小數位數 ( <i>s</i> )。
DOUBLE ( <i>p,s</i> ) DOUBLEPRECISION( <i>p,s</i> )	倍精準度浮點數。 <i>p</i> 必須大於 0，小於 53。僅在轉換資料為可顯示的格式 (如轉換為字串) 時，才會使用精確度 ( <i>p</i> ) 及小數位數 ( <i>s</i> )。
NUMERIC( <i>p,s</i> )	區化十進位數，具有精確度 <i>p</i> 及小數位數 <i>s</i> 。 <i>p</i> 的值須大於 0，小於 32。
DEC( <i>p,s</i> ) DECIMAL( <i>p,s</i> )	聚集十進位數，具有精確度 <i>p</i> 及小數位數 <i>s</i> 。 <i>p</i> 的值須大於 0，小於 32。
DTWTABLE	用來將 Net.Data 表格傳遞到呼叫的程式的特殊資料類型。「直接呼叫」語言環境會將指標傳遞到表格，然後可以使用 Net.Data 語言環境介面表格函數，來操作它。

定義為數值的參數可以包括貨幣符號及三位數的區隔符號。在傳遞變數到程式之前，當數值變數從字串格式轉換為它的內部格式時，「直接呼叫」語言環境會除去貨幣符號及三位數的區隔符號。 Net.Data 會從 Net.Data 執行所在的處理的處理屬性中，取回貨幣符號、小數格式及三位數的分隔字元。

## 以空值終止的字串參數

在架構檔或巨集內，若 DTW\_PAD\_PGM\_PARMS 設定為 NO，則「直接呼叫」語言環境將使用空值終止符字元 (值 x'00')，將字串值傳遞到您的程式。這需要您撰寫程式碼，來處理字串 (除非您正在使用 C 或 C++，因為它會預期以空值終止的字串)。

例如，如果您將參數欄位定義為 `CHAR(10)`，但您傳遞一個長度為 5 個位元組的字串值，則 `Net.Data` 會在第 5 個位元組之後放置空值終止符。在 `CHAR(10)` 欄位中，以字串形式傳遞值 "12345" 將產生：

```
x'F1F2F3F4F500.....'
```

空值終止符後的位元組沒有定義（您無法假設位元組是空值或空白）。

因為字串是以空值終止，且在空值終止符後含有未起始設定的位元組，所以您無法在 `RPG` 或 `COBOL` 程式中使用字串。例如，如果您在比較作業中使用字串，則作業將不會產生有效的結果。程式不會預期字串含有空值終止符，而是預期將在尾端以空白填補的字串。

您可以在程式內使用字串處理函數，取回字串值或使用 `VARCHAR` 資料類型。這種方法會在前兩個位元組中給與字串的長度。

在架構檔或巨集內，若 `DTW_PAD_PGM_PARMS` 設定為 `YES`，則「直接呼叫」語言環境會將字串值傳遞到您的程式，而這個值的右邊將填補空白，直到符合精確度長度。使用上面的同一範例，但 `DTW_PAD_PGM_PARMS` 設定為 `YES`，在 `CHAR(10)` 欄位中，以字串形式傳遞值 "12345" 將產生：

```
x'F1F2F3F4F5404040404000'
```

因為字串長度為 5，少於指定的精確度，所以將在值的後面插入空白，直到符合精確度長度。以如 `RPG` 的語言撰寫的程式現在可以使用參數，不需要處理以空值終止的字串。

## 傳遞參數時常見的錯誤

下列列示描述當使用「直接呼叫」語言環境呼叫程式，以及將參數傳遞到程式時，可能發生的錯誤。也會提供避免這些錯誤的秘訣。

### 參數不符錯誤

確定參數的數目及次序符合它們出現在呼叫的程式的參數列示中的數目及次序。

### 資料類型錯誤

確定針對參數指定的資料類型符合呼叫的程式所預期的資料類型。可能是「直接呼叫」語言環境支援的資料類型不被用來建立呼叫程式的高階程式設計語言所支援。

### 長度錯誤

確定針對參數定義的長度是正確的，且符合在呼叫程式中指定的長度。若指定的長度短於呼叫程式的宣告長度，可能會損毀儲存體，使得 `Net.Data` 無法正常運作。

## 從程式傳回值

有些高階程式設計語言（如 `C`）可以在程式呼叫中傳回一個整數。您可以在函數定義中指定 `RETURNS` 關鍵字來取回這整數，例如：

```
%function(DTW_DIRECTCALL) dc3(IN CHAR(3) p1) RETURNS(retval) {  
  %EXEC { /QSYS.LIB/NETDATA.LIB/MYPGM.PGM %}  
  %}
```

當函數呼叫順利完成時，參數 `retval` 將含有程式所傳回的值。

## 直接呼叫語言環境範例

在這個範例中，巨集會呼叫程式並傳遞數個參數。程式的原始檔會位在巨集之後，它是以 RPG 及 CL 撰寫而成的。呼叫的程式會接受兩個整數參數。它會將第一個參數（輸入參數）複製到第二個參數（輸出參數）。

巨集：

```
%define ilepgm = "/QSYS.LIB/NETDATADEV.LIB/TDCCLI01.PGM"
%define out1 = "0"
%FUNCTION(DTW_DIRECTCALL) dcFunction(IN INT inp1,
                                     OUT INT outp2)
{ %EXEC { ${ilepgm} %} %}
%HTML(REPORT){
  @dcFunction("123", out1)
  The value of out1 is: "${out1}"
%}
```

ILE RPG 程式：

```
DINP1          S          10I00
DOUTP2         S          10I00
C*
C      *ENTRY          PLIST
C                      PARM          INP1
C                      PARM          OUTP2
C*
C                      Z-ADD      INP1      OUTP2
C*
C                      SETON                      LR
```

CL 程式：

```
PGM PARM(&INP1; &OUTP2;)
DCL VAR(&INP1;) TYPE(*CHAR) LEN(4)
DCL VAR(&OUTP2;) TYPE(*CHAR) LEN(4)
CHGVAR VAR(&OUTP2;) VALUE(&INP1;)
ENDPGM
```

---

## Java 應用程式語言環境

Java 應用程式語言環境可讓您呼叫 Java 程式，輕易地使 Java 應用程式與 Net.Data 整合在一起。首先在 OS/400 V4R4 中引進 Java 應用程式語言環境。

若要使用 Java 應用程式語言環境，請完成第18頁的『設置 Java 應用程式語言環境』中所描述的架構步驟。

## 呼叫 Java 程式

若要呼叫 Java 程式，請定義一個函數，來使用 Java 應用程式 (DTW\_JAVAPPS) 語言環境。指定一個代表 Java 程式的類別名稱的函數名稱。

範例：呼叫 Java 程式 helloWorld.java：

```
%function(DTW_JAVAPPS) helloWorld() { %}
```

Java 應用程式語言環境預典 Java 程式含有 'main' 的方法識別字，這是在 Java 程式中執行的第一個方法。當語言環境呼叫應用程式時，應用程式將有權存取 stdin 及 stdout。在 stdin 中沒有任何套表資料，因為 Net.Data 已讀取了資料。

**重要事項：**在呼叫 Java 應用程式之前，請設定 DTW\_JAVA\_CLASSPATH 路徑架構變數，以便可以找到 Java 類別。請參閱第16頁的『DTW\_JAVA\_CLASSPATH』，取得這個變數的語法。

## 傳遞參數到 Java 程式

您可以在函數定義上指定將傳遞的參數，將參數傳遞到 Java 程式。僅指定為僅輸入 (IN) 或輸入或輸出 (INOUT) 的字串參數。

**範例：**將在函數呼叫上傳遞 IN 參數 p1

```
%function(DTW_JAVAPPS) jv1(IN p1) { %}
```

Java 應用程式語言環境不支援更新變數的 Java 程式，因為它無法將更新過的值傳回到巨集。

## Java 應用程式語言環境範例

在這個範例中，Net.Data 巨集會呼叫 Java 程式 echoString。巨集會傳遞兩個字串參數到 Java 語言環境。在將第二個參數列印到標準輸出 (stdout) 之前，第一個字串將告訴 Java 程式要使用斜體或粗體來高亮度標示第二個參數 (字串)。因為程式會傳遞 "I" (代表斜體)，所以 Web 伺服器會在瀏覽器上以斜體方式顯示字串 *Hello World*。Java 程式的原始檔會位在巨集之後。

巨集：

```
%FUNCTION(DTW_JAVAPPS) echoString(textAttribute, text){ %}  
  %HTML(runjava){  
    @echoString("I","Hello World")  
  %}
```

Java 程式：

```
class echoString {  
  public static void main (String args[]) {  
    if (args[0].equals("I"))  
      System.out.println("<I>" + args[1] + "</I>");  
    else  
      System.out.println("<B>" + args[1] + "</B>");  
  }  
}
```

---

## REXX 語言環境

REXX 語言環境可讓您執行 REXX 程式。

## 執行 REXX 程式

透過 REXX 語言環境，您可以執行列入 REXX 程式或外部 REXX 程式。列入 REXX 程式是一種在巨集中具有 REXX 程式來源的 REXX 程式。外部 REXX 程式則在外部檔中具有 REXX 程式來源。

**若要執行列入 REXX 程式：**



定義一個函數，它須使用 REXX (DTW\_REXX) 語言環境，且在函數的中須含有 REXX 程式碼。

**範例：**含有列入 REXX 程式的函數

```
%function(DTW_REXX) helloWorld() {  
    SAY 'Hello World'  
%}
```

**若要執行外部 REXX 程式：**

定義一個函數，它須使用 REXX (DTW\_REXX) 語言環境，且須包括將在 EXEC 陳述式中執行 REXX 程式的路徑。

**範例：**含有指向外部程式的 EXEC 陳述式的函數

```
%function(DTW_REXX) externalHelloWorld() {  
%EXEC{ /QSYS.LIB/REXX.LIB/REXXSRC.FILE/HELLOWORLD.MBR%}  
%}
```

如果您使用 EXEC\_PATH 架構變數來定義含有程式的目錄的路徑，則您可以縮短程式的路徑。請參閱第14頁的『EXEC\_PATH』，以學習如何定義 EXEC\_PATH 架構變數。

**限制：**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，但若未插入它們，可能會發生無法預期的結果。

## 傳送參數到 REXX 程式

有兩種方式可將資訊傳遞給 REXX (DTW\_REXX) 語言環境所呼叫的 REXX 程式：直接與間接。

**直接** 使用 %EXEC 陳述式直接將參數傳遞給外部 REXX 程式。例如：

```
%FUNCTION(DTW_REXX) rexx1() {  
    %EXEC{  
        /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。使用這種方法傳遞給程式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**

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`。

## REXX 語言環境範例

下列範例顯示一個巨集，它會呼叫一個 REXX 函數，來建立一個含有兩欄三列的 Net.Data 表格。在 REXX 函數的呼叫之後是一個內建函數 `DTW_TB_TABLE()`，被呼叫來建立將傳回給瀏覽器的 HTML 套表。

```
%DEFINE myTable = %TABLE
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION(DTW_REXX) genTable(out out_table) {
    out_table_ROWS = 3
    out_table_COLS = 2

    /* 設定直欄標題 */
    do j=1 to out_table_COLS
        out_table_N.j = 'COL'j
    end

    /* 設定橫列中的欄位 */
    do i = 1 to out_table_ROWS
        do j = 1 to out_table_COLS
            out_table_V.i.j = '[' i j ']'
        end
    end
%}

%HTML(REPORT){
    @genTable(myTable)
    @DTW_TB_TABLE(myTable)
%}
```

結果：

COL1	COL2
[ 1 1 ]	[ 1 2 ]
[ 2 1 ]	[ 2 2 ]
[ 3 1 ]	[ 3 2 ]

## SQL 語言環境

SQL 語言環境可讓您經由將 SQL 陳述式傳送到資料庫管理系統 (DBMS)，來執行 SQL 陳述式。

若要使用 SQL 語言環境，請確定您遵循第18頁的『設置語言環境』中所描述的架構步驟。

### 執行 SQL 陳述式

您可以執行動態 SQL 支援的任何 SQL 陳述式。

若要執行 SQL 陳述式，請定義一個函數，來使用 SQL (DTW\_SQL) 語言環境，並在函數的語言環境可執行區段中含有 SQL 陳述式。

**範例：**執行 SQL SELECT 陳述式的 SQL 函數：

```
%function(DTW_SQL) getOrders() {  
    SELECT cust, custid, custorder FROM mylibrary.customers  
}%
```

#### 確定控制

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。
- 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11頁的『DTW\_SQL\_ISOLATION: DB2 隔離變數』，取得關於 SQL 語言環境支援的隔離層次的詳細資訊

異動管理的詳細資訊，請參閱第85頁的『管理 Net.Data 應用程式中的異動』。

#### OUT 及 INOUT 表格

如果您在函數定義上指定 OUT 或 INOUT Net.Data 表格，且 SQL 陳述式傳回結果集合，則 SQL 語言環境會將每一結果集合儲存在指定的表格中。然後，您可以在巨集中使用表格。如果未指定 OUT 表格，則 SQL 語言環境將使用預設表格。

#### 巢狀 SQL 陳述式

您可以從另一個 SQL 函數的 ROW 區塊內呼叫其他 SQL 函數。在每一 SQL 函數中使用唯一的 Net.Data 表格名稱，否則，可能會發生無法預期的結果。

**範例：**從另一個 SQL 函數的 ROW 區塊呼叫 SQL 函數

```
%define mytable1 = %TABLE  
%define mytable2 = %TABLE  
%FUNCTION(DTW_SQL) sql2 (IN p1, OUT t2) {  
    select * from NETDATA.STAFFINF where projno='${p1}'  
%REPORT {  
    %ROW { $(N1) is $(V1) %}  
    %}  
%}  
%FUNCTION(DTW_SQL) sql1 (OUT t1) {  
    select * from NETDATA.STAFFINF  
%REPORT {
```

```
%ROW { @sql2(V1, mytable2) %}
%}
%}
%HTML(netcall1) { @sql1(mytable1) %}
```

## 支援的語言環境變數

SQL 語言環境支援爲了支援 DB2 而設計的變數。例如，DATABASE 變數指定當執行 SQL 陳述式時，SQL 語言環境將連接的資料來源。下列列示指定 SQL 語言環境支援的變數。請參閱 *Net.Data* 參考手冊，取得這些變數的說明、語法及範例。

- DATABASE
- DB\_CASE
- DTW\_EDIT\_CODES
- DTW\_PAD\_PGM\_PARMS
- DTW\_SET\_TOTAL\_ROWS
- LOGIN
- NULL\_RPT\_FIELD
- PASSWORD
- SHOWSQL
- SQL\_STATE
- TRANSACTION\_SCOPE

## 支援的資料類型

SQL 語言環境支援表5中所示的資料類型

表 5. 資料類型

BLOB(1)	DOUBLE	SMALLINT
CHAR	DOUBLEPRECISION	TIME
CLOB(1)	FLOAT	TIMESTAMP
DATE	GRAPHIC	VARCHAR
DBCLOB(1)	INTEGER	VARGRAPHIC
DECIMAL	REAL	

(1) 這些資料類型無法以參數形式傳遞到儲存程序。若要瞭解儲存程序支援的資料類型，請參閱第86頁的『儲存程序語法』

請參閱第82頁的『資料類型注意事項』，瞭解 LOB 及 DATALINK 資料類型的特殊注意事項。

## SQL 語言環境限制

當規劃您的環境時，請考慮下列限制：

- 若下列情況中至少有一個存在，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
  - 使用 SQL CLI 的應用程式將在 Net.Data 所在的同一處理中執行
- 列入陳述式區塊中的 SQL 陳述式最多可達到 32KB。

- 您最多可以使用 50 個區域或遠端資料庫連線。當使用多個連線，請考慮下列限制：
  - Net.Data 不容許同一遠端資料庫有並行連線。
  - 如果 TRANSACTION\_SCOPE=MULTIPLE (這是預設值)，則在存取了遠端資料庫後，無法變更登入 ID。請參閱 第85頁的『管理 Net.Data 應用程式中的異動』。
- 請參閱第86頁的『管理多個資料庫連線』，取得這些限制的詳細資訊。

## 資料類型注意事項

SQL 語言環境支援的下列資料類型需要特殊考量。

- 『使用大型物件』
- 第84頁的『編碼結果集合中的 DataLink URL』

### 使用大型物件

您可以將大型物件檔案 (LOB) 儲存在 DB2 資料庫中，然後使用 SQL 語言環境來存取它們，供您的 Web 應用程式使用。

當 SQL 查詢在結果集合中傳回 LOB 時，SQL 語言環境不會將大型物件儲存在處理變數 (如 V1 或 V2) 的 Net.Data 表格中，或儲存在 Net.Data 表格欄位中。相反地，當 Net.Data 發現 LOB 時，它會將 LOB 儲存在 Net.Data 建立的檔案中。這個檔案位在 HTML\_PATH 路徑架構變數所指定的目錄中。Net.Data 表格欄位及處理變數的表格將設定為該檔案的路徑。僅執行 Net.Data 時所用的使用者 ID 方可存取這個檔案。

儲存 LOB 的檔案名稱是以動態方式建構的，具有下列格式：

名稱[.副檔名]

其中：

名稱 是識別大型物件的唯一字串

副檔名 是識別物件類型的字串。對於 CLOB 及 DBCLOB，副檔名為 'txt'。對於 BLOB，SQL 語言環境會嘗試決定副檔名，其方法是尋找 LOB 檔的前幾個位元組中的標記，因為它會指出 LOB 代表什麼。SQL 語言環境可識別下列類型的資料 (括弧中的內容即是使用的副檔名)：

- 位元圖形影像 (.bmp)
- 圖形式影像格式 (.gif)
- JPEG 影像檔 (.jpg)
- Tagged 影像檔格式 (.tif)
- Postscript (.ps)
- 樂器數位介面音效檔 (.mid)
- AIFF 音效檔 (.aif)
- 音效視覺交錯音效檔 (.avi)
- 基本音效檔 (.au)
- 真實音效檔 (.ra)
- Windows 音效視覺檔 (.wav)
- 可攜性文件格式 (.pdf)

- Midi 順序檔 (.rmi)

如果無法識別 BLOB 的物件類型，將不會對檔名新增副檔名。

在巨集檔中參照 LOB 時，SQL 語言環境將使用下列語法傳回檔案名稱，這個名稱會具有 LOB 檔案名稱前加的 /tmplobs/ 字串：

/tmplobs/名稱.[副檔名]

**規劃要訣：**當每一個查詢傳回 LOB 時，會造成檔案將在 HTML\_PATH 路徑架構變數指定的目錄中建立。由於 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮到系統限制問題。您可能想要建立批次程式，定期清除目錄建議您最好使用 DataLinks，因為它不需要透過 SQL 語言環境，將檔案儲存在目錄中，因而造成更佳的效能，且使用更少的系統資源。

**範例：**應用程式使用者必須按一下檔名，方可呼叫檢視器，因為應用程式使用 MPEG 音效 (.MPA) 檔。SQL 語言環境不認得此種檔案類型，因此會使用 EXEC 變數，將副檔名附加至檔案中。

```
%DEFINE{
lobpath = "@DTW_RGETINIDATA("HTML_PATH")"
filename = "@DTW_RREPLACE($(V3), "/tmplobs/", "", "1", "F")"
myFile=%EXEC "REN '$(lobpath)/$(filename)' '$(filename).mpa'"
%}
%{ 其中更名是您的作業系統中的更名指令 %}
%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
%REPORT{
<P>底下是您選取的資訊：<P>
%ROW{
$(myFile)
$(V1) Voice sample <IMG SRC="$(V2)">
<A HREF="$(V3).mpa">Voice sample</A><P>
%}
%}
%}

%HTML(REPORT){
@queryData()
%}
```

queryData 函數會傳回下列 HTML 輸出：

```
<P>Here are the images you selected:<P>
Kinson Yamamoto
<IMG SRC="/tmplobs/p2345n1.gif">
<A HREF="/tmplobs/p2345n2.mpa">Voice sample</A><P>
Merilee Lau
<IMG SRC="/tmplobs/p2345n3.gif">
<A HREF="/tmplobs/p2345n4.mpa">Voice sample</A><P>
```

前一個範例中的 REPORT 區塊，使用隱含的表格變數 V1、V2、與 V3。

- V1 代表個人的名稱，為純文字。
- V2 代表 .GIF 格式的個人照片檔。此圖片會以列入方式顯示。SQL 語言環境會自動併入字首 /tmplobs/ 及 .GIF 副檔名。
- V3 是 .mpa 格式的個人聲音樣本檔。當 SQL 語言環境遇到不認得的檔案格式時，(如：.mpa 檔)，其會將檔案寫入 HTML\_PATH 架構變數中指定的目錄中，且不會加上副檔名。此範例是說明在處理此種檔案類型時，該如何使用 EXEC 變數來新增副檔名。解析變數 \$(V3) 時，會於檔名之前加上路徑 /tmplobs/。例

如，/tmplobs/sound2a。在這個範例中，EXEC 變數會 REN 指令更改檔案的名稱，並新增副檔名 .mpa 到檔案。在可以更改檔案的名稱之前，/tmplobs/ 會從檔名中除去，而且會使用 DTW\_RGETINIDATA 函數來取回 HTML\_PATH 中指定的路徑，來取回將更名的檔案的完整路徑。當應用程式的使用者按一下「聲音樣本」時，就會播放聲音樣本。

**LOB 的存取權：** 確定 Web 伺服器執行時所用的使用者 ID 對 HTML\_PATH 指定的目錄具有寫入權。

## 編碼結果集中的 DataLink URL

DataLink 資料類型是擴充可儲存在資料庫檔案的資料類型的基本建置區塊之一。透過 DataLink，儲存在直欄中的真正資料僅是檔案的指標。這個檔案可以是任一檔案類型；影像檔、聲音記錄檔或文字檔。DataLinks 會儲存一個 URL，來解析檔案的位置。

DATALINK 資料類型需要使用「DataLink 檔案管理程式」。「DataLink 檔案管理程式」的詳細資訊，請參閱您的作業系統的 DataLink 文件。在您使用 DATALINK 資料類型之前，您必須確定 Web 伺服器有權存取「DB2 檔案管理程式伺服器」管理的檔案系統。

當 SQL 查詢透過 DataLink 來傳回結果集合，且 DataLink 直欄是以具有 READ PERMISSION DB DataLink 選項的 FILE LINK CONTROL 來建立時，DataLink 直欄中的檔案路徑將含有存取符記。DB2 會使用這個存取符記，來鑑定檔案的存取權。沒有這個存取符記，所有存取檔案的嘗試將因違反權限而失敗。不過，存取符記可能含有在將傳回給瀏覽器的 URL 中無法使用的字元，如分號 (;) 字元。例如：

```
/datalink/pics/UN1B;0YPVKG346KEB;baibien.jpg
```

URL 無效，因為它含有分號 (;) 字元。若要使得 URL 有效，則必須使用 Net.Data 內建函數 DTW\_URLESCSEQ 來編碼分號。不過，在引用這個函數之前，必須先執行某些字串操作，因為這個函數也會編碼斜線 (/)。

您可以撰寫 Net.Data MACRO\_FUNCTION，使字串操作自動化，並使用 DTW\_URLESCSEQ 函數。請在每一個從 DATALINK 資料類型直欄取回資料的巨集中使用這個技術。

### 範例 1：自動編碼從 DB2 UDB 傳回的 URL 的 MACRO\_FUNCTION

%{ 執行：將 DTW\_URLESCSEQ 引用到 DATALINK URL，使它成為有效的 URL。

輸入：來自 DB2 檔案管理程式直欄的 DATALINK URL。

傳回：具有符記部份的 URL 是已編碼的 URL

```
%}  
%MACRO_FUNCTION encodeDataLink(in DLURL) {  
  @DTW_rCONCAT( @DTW_rDELSTR( DLURL,  
    @DTW_rADD(@DTW_rLASTPOS("/", DLURL), "1" ) ),  
    @DTW_rURLESCSEQ( @DTW_rSUBSTR(DLURL,  
    @DTW_rADD( @DTW_rLASTPOS("/", DLURL), "1" ) ) ) )  
  )  
%}
```

在使用這個 MACRO\_FUNCTION 後，已適當編碼了 URL，且可以在 Web 瀏覽器上參照 DATALINK 直欄中指定的檔案。

### 範例 2：指定將傳回 DATALINK URL 的 SQL 查詢的 Net.Data 巨集



```

%FUNCTION(DTW_SQL) myQuery(){
  select name, DLURLCOMPLETE(picture) from myTable where name like '%river%'
%REPORT{
%ROW{
  <p> $(V1) <br>
  Before Encoding: $(V2) <br>
  After Encoding: @encodeDataLink($(V2)) <br>
  Make HREF: <a href="@encodeDataLink($(V2))"> click here </a> <br> <p>
  %}
%}
%}

```

請注意，將使用「DataLink 檔案管理程式」函數。這個函數 `dlurlcomplete` 會傳回完整的 URL。

## 管理 Net.Data 應用程式中的異動

當您使用插入、刪除或更新陳述式來修改資料庫的內容時，修改的內容將不會變成永久的，除非資料庫從 Net.Data 收到一個確定陳述式。如果發生錯誤，則 Net.Data 將傳送一個回轉陳述式給資料庫，反轉自上次確定後所做的任何修改。

Net.Data 傳送確定及可能回轉的方式取決於您如何設定 `TRANSACTION_SCOPE`，以及您是否明確地在巨集中指定確定。`TRANSACTION_SCOPE` 的值是 `MULTIPLE` 及 `SINGLE`。

### MULTIPLE

指定在發出確定及可能回轉陳述式之前，Net.Data 將執行所有 SQL 陳述式。在要求結束時，Net.Data 會傳送確定，而且如果順利發出了每一個 SQL 陳述式，則確定將使得資料庫中的所有修改內容變成永久的。如果任一陳述式傳回錯誤，則 Net.Data 將發出回轉陳述式，將資料庫設定回它的原始狀態。如果未設定 `TRANSACTION_SCOPE`，`MULTIPLE` 將是預設值。

若要啟動這個確定方法，請將 `TRANSACTION_SCOPE` 設定為 `MULTIPLE`。

例如：

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"MULTIPLE")
```

### SINGLE

指定在每一個 SQL 陳述式順利完成後，Net.Data 將發出一個確定陳述式。如果 SQL 陳述式傳回錯誤，將發出回轉陳述式。單一異動範圍可立即保障資料庫的修改；不過，使用這個範圍後，稍後可能無法使用回轉陳述式來還原修改的內容。

若要啟動這個確定方法，請將 `TRANSACTION_SCOPE` 設定為 `SINGLE`。例如：

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"SINGLE")
```

您可以使用 `COMMIT SQL` 陳述式，在巨集中的任一 SQL 陳述式結束時，發出一個確定陳述式。經由使 `TRANSACTION_SCOPE` 設定為 `MULTIPLE`，並在那些您覺得已合格作為異動的陳述式群組結束時發出確定陳述式，您（應用程式軟體開發者）便可以完全控制應用程式中的確定及回轉行為。

若要發出 SQL 確定陳述式，您可以定義一個可在您的 HTML 區塊中任一點呼叫的函數：

```
%FUNCTION(DTW_SQL) user_commit() {
    commit
}%
...
%HTML {
    ...
    @user_commit()
    ...
}%
```

## 管理多個資料庫連線

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 Web 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：

- 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。
- 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。

## 儲存程序

儲存程序是一種已編譯的程式，儲存於 DB2 中，可執行 SQL 陳述式。在 Net.Data 中，儲存程序是從 Net.Data 函數上使用 CALL 陳述式來呼叫。儲存程序參數是從 Net.Data 函數參數列示中來傳入。您可以使用儲存程序，經由保存編譯過的 SQL 陳述式和資料庫伺服器，來改善執行效能和提升完整性。Net.Data 支援透過 SQL 與 ODBC 語言環境，以 DB2 使用儲存程序。

本段落說明下列主題：

- 『儲存程序語法』
- 第87頁的『呼叫儲存程序』
- 第88頁的『傳送參數』
- 第88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 function_name ([IN datatype arg1, INOUT datatype arg2,
    OUT tablename, ...]) {
    CALL stored_procedure
    [%REPORT [(resultsetname)] { %}]
```



```
...
[%REPORT [(resultsetname)] { %}]
[%MESSAGE %}]
%}
```

其中：

*function\_name*

指起始儲存程序呼叫的 Net.Data 函數的名稱

*stored\_procedure*

儲存程序名稱

*datatype*

指 Net.Data 支援的其中一種資料庫資料類型，顯示在表6中。參數列示中指定的資料類型必須與儲存程序中的資料類型相配。有關這些資料類型的說明，請參閱您的資料庫文件。

*tablename*

指將儲存結果集合的 Net.Data 表格的名稱 (僅在結果集合將儲存在 Net.Data 表格時才會使用)。若設定的話，這個參數須符合 *resultsetname* 的關聯參數名稱

*resultsetname*

指與透過 REPORT 區塊及函數參數列示上的表格名稱 (或兩者) 從儲存程序傳回的結果有關聯的名稱。REPORT 區塊上的 *resultsetname* 須符合函數參數列示上的 *tablename*。

表 6. 儲存程序的資料類型

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

## 呼叫儲存程序

1. 定義起始儲存程序呼叫的函數。

```
%FUNCTION (DTW_SQL) function_name()
```

2. 可選擇是否要對儲存程序設定任何 IN、INOUT 或 OUT 參數，包括從儲存程序傳回的任何結果集合的結果集合名稱。您也可以從另一個儲存程序，指定表格名稱或結果集合同於 IN 或 INOUT 參數值。

```
%FUNCTION (DTW_SQL) function_name (IN datatype
arg1, INOUT datatype arg2, OUT tablename...)
```

3. 使用 CALL 陳述式來識別儲存程序名稱。

```
CALL stored_procedure
```

4. 如果儲存程序將建立一個結果集合，則可選擇是否要設定一個 REPORT 區塊，來定義 Net.Data 如何顯示結果集合。

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

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。函數參數列示上的參數數目及類型必須符合針對儲存程序定義的數目及類型。例如，如果針對儲存程序定義的參數列示上的參數為 INOUT，則函數參數列示上對應的參數必須是 INOUT。例如，如果針對儲存程序定義的參數列示上的參數為類型 CHAR(30)，則函數參數列示上對應的參數也必須是 CHAR(30)。

範例：傳送參數值至儲存程序

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) valuein) {
    CALL myproc
    ...
```

範例：從儲存程序傳回值

```
%FUNCTION (DTW_SQL) mystoredproc (OUT VARCHAR(9) retvalue) {
    CALL myproc
    ...
```

## 處理結果集合

您可以使用 SQL 或 ODBC 語言環境，從儲存程序。結果集合可以儲存在 Net.Data 表格中，以便在您的巨集內進一步處理它們，或使用 REPORT 區塊處理它們。如果儲存程序建立多個結果集合，您必須使名稱與儲存程序所建立的每一個結果集合產生關

聯。這是經由在 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，使結果在 Net.Data 的預設報表樣式中處理。
- 使結果集合與 REPORT 區塊產生關聯，來引用自己的報告樣式。在 REPORT 區塊中，您可以使用 Net.Data 變數、文字處理陳述式 (如 HTML 或 JavaScript)，或其他函數，來設定報表資料如何顯示在瀏覽器上。

結果集合恆會儲存在區域表格中，以便巨集中的另一個函數也可以存取資料。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66頁的『多個 REPORT 區塊的指南與限制』，取得當使用多個報表區塊時的指南與限制。

#### 若要傳回單一結果集合及使用預設報告：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name (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及指定 REPORT 區塊：

可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name (OUT tablename) {  
    CALL stored_procedure [(resultsetname)]  
    %REPORT [(resultsetname)] {  
        ...  
    %}  
%}
```

例如：

```
%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1) {  
    CALL myproc  
    %REPORT {  
        ...  
        %ROW { ... %}  
        ...  
    %}  
%}
```

另一種方式即是使用下列語法：

```
%FUNCTION (DTW_SQL) function_name () {  
    CALL stored_procedure  
  
    %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 = "YES"  
%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的一個或多個 **REPORT** 區塊 產生關聯。可使用下列語法：

```
%FUNCTION (DTW_SQL) function_name (OUT tablename1, tablename2, ...) {  
    CALL stored_procedure  
    %REPORT (tablename1)  
        ...  
        %ROW { ... %}  
        ...  
    %}  
    %REPORT (tablename2)  
        ...  
        %ROW { ... %}  
        ...  
    %}  
    ...  
%}
```

例如：

```
%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1, mytable2) {  
    CALL myproc  
  
    %REPORT(mytable1) {  
        ...  
        %ROW { ... %}  
        ...  
    %}  
  
    %REPORT(mytable2) {  
        ...  
        %ROW { ... %}  
        ...  
    %}  
%}
```

## 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 {
  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>
%}
```

## 系統語言環境

系統語言環境支援執行指令及呼叫外部程式。

### 發出指令及呼叫程式

若要發出一個指令，請定義一個使用系統 (DTW\_SYSTEM) 語言環境的函數，而這個語言環境須包括將在 EXEC 陳述式中發出的指令的路徑。例如：

```
%FUNCTION(DTW_SYSTEM) sys1() {
    %EXEC { /QSYS.LIB/ADDLIBLE.CMD LIB(MYLIBRARY) %}
%}
```

如果您使用 EXEC\_PATH 架構變數來定義含有物件 (如指令與程式) 的目錄的路徑，則您可以縮短可執行物件的路徑。請參閱第14頁的『EXEC\_PATH』，以學習如何定義 EXEC\_PATH 架構變數。

#### 範例 1：發出指令

```
%FUNCTION(DTW_SYSTEM) sys2() {
    %EXEC { /QSYS.LIB/CALL.CMD MYLIB/MYPGM %}
%}
```

#### 範例 2：呼叫程式

```
%FUNCTION(DTW_SYSTEM) sys3() {
    %EXEC { /QSYS.LIB/MYLIB.LIB/MYPGM.PGM %}
%}
```

**要訣：**當呼叫程式時，請使用「直接呼叫」語言環境，因為它更有效率且易於使用。

### 傳遞參數到程式

有兩種方式可將資訊傳遞給「系統」(DTW\_REXX) 語言環境所呼叫的程式：直接與間接。

**直接** 在程式的呼叫上直接傳遞參數。例如：

```
%DEFINE INPARAM1 = "SWITCH1"

%FUNCTION(DTW_SYSTEM) sys1() {
    %EXEC{
        /QSYS.LIB/NETDATA.LIB/RPGCALL1.PGM ('$(INPARAM1)' 'literalstring')
    %}
%}
```

Net.Data 變數 INPARAM1 將被參照並傳遞到程式。參數傳遞到程式的方式將同於從指令行中呼叫程式時參數傳遞到程式的方式。使用這種方法傳遞給程式的參數被視為輸入類型參數 (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7 中描述的 API 設定與取回環境變數：

表 7. 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 指令，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。

系統語言環境範例

下列範例顯示一個巨集，它使用「系統」語言環境，向所有工作站訊息佇列發出「傳送中斷訊息 (SNDBRKMSG)」指令。將傳送的訊息的文字是依據格式資料而建置的 (msgToSend)。

```
%FUNCTION(DTW_SYSTEM) sndbrkmsg () {
  %EXEC { /QSYS.LIB/SNDBRKMSG.CMD MSG('$(msgToSend)') TOMSGQ(*ALLWS) %}
}%
%HTML(sndbrkmsg) {
  @sndbrkmsg()
}%
```





---

## 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34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96頁的『定義異動』
- 第101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，這會提供下列提示：

- 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97頁的『在異動中設定巨集 HTML 區塊』
- 第100頁的『終止異動』
- 第100頁的『定義異動中變數的範圍』
- 第101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：**在您的程式碼中輸入陳述式時，URL 應該是在一行上，且沒有空格，但為了便於顯示，在此它將分成兩行。

- 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13頁的『MACRO\_PATH』，以取得有關的詳細資訊。

*block* 指定參照的 Net.Data 巨集中的 HTML 區塊名稱。

*method* 指定與此套表搭配使用的 HTML 方法。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 並將它置於變數 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。

在第二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 並將它置於變數 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。



---

## 第8章 提高執行效能

提高執行效能是調整系統的重要部份。本章將討論提高 Net.Data 執行效能的策略。討論的主題如下：

- 『Net.Data 巨集快取』
- 『將語言環境最佳化』

此外，請確定已正確地調整了您的 Web 伺服器。Web 伺服器的執行效能對回應時間有直接的影響，與 Net.Data 處理巨集或直接要求的快慢無關。

---

### Net.Data 巨集快取

在 Net.Data for OS/400 中，依據預設值將啟用巨集快取，且會使用來改善產量及減少 CPU 使用率。當啟用巨集快取時，會在第一次呼叫巨集時，於記憶體中快取已預先處理的巨集。然後，這些已預先處理的版本將可以重複使用，因此能夠消除與從 HFS 讀取巨集有關聯的成本，以及每次要求它們均要處理它們的成本。快取版本的巨集可供對含有巨集的檔案具有讀取權的要求器使用。

預先處理版本的巨集所使用的記憶體數量大約是巨集檔大小的兩倍。您可以經由使用快取架構變數，來控制快取巨集時將使用的記憶體數量。如何使用這個變數的詳細資訊，請參閱第8頁的『DTW\_MACRO\_CACHE\_SIZE：巨集快取大小變數』。

---

### 將語言環境最佳化

下列段落將描述一些技術，當使用 Net.Data 提供的語言環境時，您可以使用它們來提高執行效能。

- 『REXX 語言環境』
- 第106頁的『SQL 語言環境』
- 第106頁的『系統語言環境』

### REXX 語言環境

請使用下列要訣，改善 Net.Data 應用程式的執行效能：

- 儘可能結合您的 REXX 程式。具有更少、更大的程式會比較小的程式提供更好的執行效能，因為每次在巨集中呼叫 REXX 語言環境函數時，將起始設定 REXX 直譯器。
- 將 REXX 程式儲存在外部檔案中，而不是將 REXX 程式包括在 Net.Data 巨集的行內。
- 對於外部 REXX 程式，請參照 %EXEC 陳述式中的指令行上的廣域變數。
- 經由定義廣域 Net.Data 變數並參照變數，將僅輸入參數直接傳遞給 REXX 程式。對於列入 REXX 程式，請直接參照 REXX 來源中的廣域變數。

## SQL 語言環境

若要瞭解 DB2 執行效能的注意事項，請參閱*DB2 for OS/400 SQL Programming*。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。

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95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。使用快取，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。快取可能會減少靜態 SQL 蓋過動態 SQL 的任何執行效能好處。

## 系統語言環境

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。

Net.Data for OS/400 已引進名為「直接呼叫」的新語言環境，它可以提供更易於使用及效率的介面，來呼叫程式。使用「系統」語言環境發出指令；使用「直接呼叫」語言環境呼叫程式

---

## 附錄A. 參考書目

本節會列示本書中所參考的文件。

- 『Net.Data 技術圖書館』
- 『相關文件』

---

### Net.Data 技術圖書館

可以從 Net.Data 網站存取「Net.Data 技術圖書館」，網址：

<http://www.software.ibm.com/data/net.data/library.html>

文件	說明
<ul style="list-style-type: none"><li>• <i>Net.Data Administration and Programming Guide for OS/390</i></li><li>• <i>OS/2 版、Windows NT 版和 UNIX 版 Net.Data 管理及程式設計指南</i></li><li>• <i>OS/400 版 Net.Data 管理及程式設計指南</i></li></ul>	含有關於如何安裝、配置及呼叫 Net.Data 的觀念及作業資訊。此外也描述如何撰寫 Net.Data 巨集、使用 Net.Data 效能技術、使用 Net.Data 語言環境、管理連線及使用 Net.Data 記錄及追蹤故障檢修及效能調整。
<i>Net.Data 參考手冊</i>	描述 Net.Data 巨集語言、變數及內建函數。
<i>Net.Data 語言環境介面參考手冊</i>	描述 Net.Data 語言環境介面。
<i>Net.Data 訊息與訊息碼參考手冊</i>	列示 Net.Data 錯誤訊息及回覆碼。

---

### 相關文件

當使用 Net.Data 及相關產品時，下列文件可能會很有用：

- *DB2 for OS/400 SQL Programming*, SC41-5611
- *OS/400 Distributed Database Programming*, SC41-5702

此外，OS/400 文件及紅皮書，包括關於 DB2 的書籍，可在下列 URL 取得：

<http://publib.boulder.ibm.com/html/as400/infocenter.html>



## 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。

巨集檔會使用一個併入檔，來含有巨集的 DEFINE 區塊。

第110頁的圖9顯示樣本巨集。第112頁的圖10 顯示併入檔。

```

%{***** 樣本巨集 *****}
* 檔名 = sqlsamp1.d2w *
* 說明： *
* 這個 Net.Data 巨集查詢... *
* - 員工表格，以建立要在瀏覽器上 *
* 顯示的員工選項列示 *
* - 員工表格以取得各別員工的 *
* 其它資訊 *
* *
*****%}
%{*****}
* 廣域 DEFINE 的併入檔 - *
*****%}
%INCLUDE "sqlsamp1.hti"
%{*****}
* 函數： queryDB 語言環境： SQL *
* 說明： 查詢 myTable 變數所表示的表格，及 *
* 從結果建立選項列示。myTable 變數的值是 *
* 指定於併入檔 sqlsamp1.hti 中。 *
*****%}
%FUNCTION(DTW SQL) queryDB() {
    SELECT FIRSTNME FROM $(myTable)
    %MESSAGE {
        -204: {<p><b>錯誤 -204: 找不到表格 $(myTable) 。</b>
               <p>請確定所用的併入檔無誤。</b>
        } : exit
        +default: "警告 $(RETURN_CODE)" : continue
        -default: "意外的錯誤 $(RETURN_CODE)" : exit
    }
}%

%REPORT {
<select name=emp_name>
%ROW{
<option>$(V1)
}%
</select>
}%
}%

%{*****}
* 函數： fname 語言環境： SQL *
* 說明： 查詢 myTable 變數所表示的表格， *
* 以取得 emp_name 變數所識別出的員工 *
* 之其它資料。 *
*****%}
%FUNCTION(DTW SQL) fname(){
    SELECT FIRSTNME, PHONENO, JOB FROM $(myTable) WHERE FIRSTNME='$(emp_name)'
    %MESSAGE {
        -204: "錯誤 -204: 找不到表格 "
        -104: "錯誤 -104: 語法錯誤"
        100: "警告 100: 無記錄" : continue
        +default: "警告 $(RETURN_CODE)" : continue
        -default: "意外的 SQL 錯誤" : exit
    }
}%
}%

```

圖 9. 樣本巨集 (1/3)



```
%{ *****
*   HTML 區塊： INPUT                標題： 動態查詢選項                *
*                                                                           *
*   說明： 查詢員工表格，以建立要在瀏覽器上                        *
*           顯示的員工選項列示                                *
*                                                                           *
*****%}
%HTML(INPUT) {
<html>
<head>
<title>建立員工選取列示</title>
</head>
<body>
<h3>$(exampleTitle)</h3>
<p>此範例會查詢表格，並使用結果來建立使用 <em>%REPORT</em> 區塊的選項列示。
<hr>
<form method="post" action="report">
@queryDB()<input type="submit" value="Select Employee">
</form>
<hr>
</body>
</html>
%}
```

圖 9. 樣本巨集 (2/3)

```
%{ *****
*   HTML 區塊： REPORT                *
*   說明： 查詢員工表格，以取得各別員工的                *
*           其它資訊                                *
*                                                                           *
*****%}
%HTML(REPORT){
<html>
<head>
<title>取得員工資訊</title>
</head>
<body>
<h3>您所選的員工名稱 = $(emp_name)</h3>
<p>以下為該員工的相關資訊：
<PRE>
@fname()
</PRE>
<hr><a href="input">回到上一頁</a>
</body>
</html>
%}

%{      Net.Data 巨集 1 結束 %}
```

圖 9. 樣本巨集 (3/3)

```

=====
%{*****          併入檔          *****}
*   檔名 = sqlsamp1.hti                      *
*   說明：                                  *
*       此併入檔提供給 Net.Data 巨集 sqlsamp1.d2w          *
*       巨集 DEFINE。                                  *
*****%}
%define {
    emp_name    = ""
    reposition  = sign
    exampleTitle = "樣本巨集"
    myTable     = "員工"
    DATABASE    = "樣本"
%}

%{      併入檔結束  %}

```

圖 10. 併入檔

---

## 附錄C. 注意事項

本書是針對 IBM 在美國所提供之產品與服務開發出來的。而在其他國家中，IBM 不見得有提供本書中所提的各項產品、服務、或功能。要知道在您所在之區是否可用到這些產品與服務時，請向當地的 IBM 服務代表查詢。本書在提及 IBM 的產品、程式或服務時，不表示或提示只能使用 IBM 的產品、程式或服務。只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。不過，其他非 IBM 產品、程式、或服務在運作上的評價與驗證，其責任屬於使用者。

本文件中包含著 IBM 所擁有之專利或暫准專利。使用者不得享有本書內容之專利權。您可以用書面方式來查詢特許權限，來函請寄到：

臺灣國際商業機器股份有限公司  
台北市基隆路一段 206 號  
法務部

若要查詢有關二位元組 (DBCS) 資訊的特許權限事宜，請聯絡您國家的 IBM 智慧財產部門，或者用書面方式寄到：

臺灣國際商業機器股份有限公司  
台北市基隆路一段 206 號  
法務部

下列段落若與該國之法律條款抵觸，即視為不適用：IBM 就本書僅提供『交附時之現況』保證，而並不提供任何明示或默示之保證，如默示保證書籍之適售性或符合客戶之特殊使用目的；有些地區在某些固定的異動上並不接受明示或默示保證的放棄聲明，因此此項聲明不見得適用於您。

本書中可能有技術上或排版印刷上的訛誤。因此，IBM 會定期修訂；並將修訂後的內容納入新版中。同時，IBM 會隨時改進並 (或) 變動本書中所提及的產品及 (或) 程式。

在本書中，若有任何非 IBM Web 網站的參考資料，都只是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。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊者可洽詢 IBM。其下列資訊指的是：(1) 獨立建立的程式與其他程式 (包括此程式) 之間更換資訊的方式 (2) 相互使用已交換之資訊方法 若有任何問題請聯絡：

台北市基隆路一段 206 號  
臺灣國際商業機器股份有限公司  
法務部

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於雙方之 [IBM 客戶合約] 或 [IBM 國際程式授權合約] (或任何同等合約) 條款，提供本書中所說的授權程式與其所有適用的授權資料。

本書所提及之非 IBM 產品資訊，係一由產品的供應商，或其出版的聲明或其他公開管道取得。IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性、或任何對產品的其他主張是否完全無誤。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

所有關於 IBM 未來方向或計劃的聲明，皆需視變更或取消的情形而定，但並不另行通知，而且這些聲明僅代表目標及目的。

本書僅限用於規劃目的。在所描述之產品上市前，本書所提供的資訊有可能變更。

本書包含日常事務作業所使用的資料及報告範例。爲了儘可能地完整呈現這些範例，這些範例會包含個人、公司及產品的名稱。所有這些名稱皆爲虛構的，若有任何名稱或地址與實際的公司企業相似，則純屬巧合。

---

## 商標

下列專有名詞是 IBM 公司在美國或 (及) 其它國家的商標：

AIX	Language Environment
AS/400	MVS/ESA
DB2	Net.Data
DB2 Universal Database	OS/2
DRDA	OS/390
DataJoiner	OS/400
IBM	OpenEdition
IMS	

下列專有名詞是以下其它公司的商標：

Java 及所有 Java 型的商標或標誌是 Sun Microsystems, Inc. 在美國及 (或) 其它國家的商標。

UNIX 是透過 X/Open Company Limited 獨家授權，在美國及 (或) 其它國家的註冊商標。

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

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

其它公司、產品與服務名稱 (以雙星號 \*\* 表示)，可能是其它公司的商標或服務標記。

---

## 名詞解釋

### 一劃

**一致資源定址器 (uniform resource locator).** 用來指出 HTTP 伺服器並可選擇性地指出目錄以及檔案名稱的位址，例如：

`http://www.software.ibm.com/data/net.data/index.html`。

### 三劃

**工作單元 (unit of work).** 被視為原子作業的作業的可恢復順序。工作單元內的所有作業均可完成 (已確定) 或還原 (已回轉)，好似作業是單一作業一般。僅有資源上受到確定控制的作業可被確定或回轉。

### 七劃

**防火牆 (firewall).** 一個含有軟體的電腦，其負責保護內部的網路不讓未經授權的外來者存取。

### 九劃

**持續性 (persistence).** 保留整個異動的指定值的狀態，在此異動橫跨多個 Net.Data 呼叫。僅有變數可以持續。此外，受到確定控制影響的資源上的作業將保持件用中，直到執行明確的確定或回轉為止，或直到完成異動為止。

**相對路徑名稱 (relative path name).** 不是以最高層次或“根”目錄開頭的路徑名稱。系統會假設路徑名稱是以處理的現行工作目錄作為開頭。

### 十劃

**純本文檔介面 (flat file interface).** 一組 Net.Data 內建式函數，可讓您讀取與寫入純文字檔中的資料。

### 十一劃

**埠 (port).** 一個 16 位元數字，用來供 TCP/IP 和高階通訊協定或應用程式通信用。

**現行工作目錄 (current working directory).** 將從其中解析所有相對路徑名稱的處理的預設目錄。

**異動 (transaction).** 一個 Net.Data 呼叫。如果使用持續的 Net.Data，則異動可以橫跨多個 Net.Data 呼叫。

**通用閘道介面 (CGI).** 一種標準的方式，可讓 Web 伺服器藉此將控制傳給應用程式並收回資料。

### 十二劃

**登記 (registry).** 可以儲存及擷取字串的儲存庫。

**絕對路徑 (absolute path).** 物件的完整路徑名稱。絕對路徑名稱開始於最高層或“根”目錄 (以正斜線 (/) 或反斜線字元來識別)。

**超文字標示語言 (hypertext markup language).** 一種用來撰寫 Web 文件的標籤語言。

**超本文轉送通信協定 (hypertext markup language).** 用於 Web 伺服器與瀏覽器間的通訊協定。

### 十三劃

**傳輸控制通信協定 / 網際網路通信協定 (Transmission Control Protocol / Internet Protocol).** 一組支援區域及廣域網路之對等連接功能的通訊協定。

**資料庫 (database).** 集結了許多表格而成的集合，也可以是表格空間及索引空間的集合。

**資料庫管理系統 (database management system, DBMS).** 一個控制資料庫的建立、組織和修改以及存取儲存於其內之資料的軟體系統。

**資料類型 (data type).** 直欄與文字的屬性。

**路徑 (path name).** 告訴系統如何找出物件的位置。路徑名稱會表示為目錄名稱的順序，且其後會跟著物件名稱。個別目錄與物件名稱是以正斜線 (/) 或反斜線 (\) 字元來區隔。

**路徑 (path).** 用來尋找檔案的搜尋路徑。

### 十四劃

**語言環境 (language environment).** 提供 Net.Data 巨集與外部資料來源 (如 DB2) 或程式設計語言 (如 Perl) 之存取的模組。

### 十五劃

**確定控制 (commitment control).** 建立處理內 Net.Data 執行的界限，在此資源上的作業是工作單元的一部份。

## 十七劃

**應用程式設計介面 (application programming interface, API).** 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 及 GWAPI。

## A

**API.** 應用程式設計介面。Net.Data 支援三種 Web 伺服器 API，來改善透過 CGI 處理的執行效能。

**applet.** 一種包含在 HTML 頁面中的 Java 程式。Applet 是與可使用 Java 的瀏覽器（如 Netscape Navigator）一起使用，且是在當 HTML 頁面被處理時載入。

## B

**BLOB.** 二進位大型物件。

## C

**CGI.** 通用閘道介面。

**CLOB.** 字元大型物件。

## D

**DATALINK.** 一種 DB2 資料類型，它可以從資料庫啓用儲存在資料庫外的檔案的參照。

**DBCLOB.** 雙位元組大型物件。

**DBMS.** 資料庫管理系統。

**Domino Go Web 伺服器.** Lotus 公司及 IBM 提供的 Web 伺服器，它提供一般及安全連線。 ICAPI 與 GWAPI 是這個伺服器所提供的介面。

## G

**GWAPI.** Go Web 伺服器 API。

## H

**HTML.** 超文字標示語言。

**HTTP.** 超本文轉送通信協定。

## I

**ICAPI.** Internet Connection API。請參閱。

**Internet.** 一個國際公用的 TCP/IP 電腦網路。

**Intranet.** 一個位於公司防火牆內的 TCP/IP 網路。

## J

**Java.** 一種不依附作業系統之物件導向型程式設計語言，特別適用於 Internet 的應用程式。

## L

**LOB.** 大型物件。

## M

**middleware.** 介於應用程式及網路之間的軟體。它會管理從屬站應用程式與伺服器透過網路的交談。

## N

**null (空值).** 一個代表沒有資訊的特殊值。

## P

**Perl.** 一種解譯過的程式設計語言。

## T

**TCP/IP.** 傳輸控制通信協定 / 網際網路通信協定。

## U

**URL.** 一致資源定址器 (Uniform resource locator)。

## W

**Web 伺服器 (Web server).** 一部執行 HTTP 伺服器軟體 (如：Internet Connection) 的電腦。

# 索引

索引順序以中文字，英文字，及特殊符號之次序排列。

## 〔一劃〕

一般目的函數 59

## 〔三劃〕

大型物件 (LOB) 82, 83

有效格式 83

說明 82

## 〔五劃〕

加密, 網路 25

巨集

之內和之間導引 41

呈現部份 37

函數 52

宣告部分 37

持續 95

迴路 69

區塊 39

條件邏輯 67

產生 HTML 61

結構 38

開發 37

說明 1

樣本 38

識別字範圍 42

變數 42

DEFINE 區塊 39

FUNCTION 區塊 39

HTML 區塊 40

IF 區塊 67

WHILE 區塊 69

巨集的部份

呈現 37

宣告 37

巨集要求 31

語法 31

範例 31

## 〔六劃〕

列示變數 49

列印, 停用預設報告 63

名詞解釋 115

多個報表區塊 64

字串函數 59

字組函數 60

存取 DB2 80

存取權

語言環境的 73

Net.Data 檔案的 20

安全

身份驗證 25

防火牆 23

授權 26

設定存取權 20, 73

概觀 23

網路加密 25

語言環境 73

Net.Data 機制 26

## 〔七劃〕

快取巨集, 快取大小 8

系統語言環境 92

呼叫程式 92

發出指令 92

傳遞參數 92

概觀 92

身份驗證, 安全 25

防火牆 23

## 〔八劃〕

使用者定義函數 53

使用者定義的語言環境, ENVIRONMENT 陳述式 7

函數 86

一般目的 59

字串 59

字組 60

使用者定義 53

呼叫 57

呼叫儲存程序 86

定義 53

持續 61

純本文檔 60

算術 59

說明 52

FUNCTION 區塊語法 53

MACRO\_FUNCTION 區塊語法 53

table 60

Web 登記 60

函數呼叫

內建 57

語法 57

- 呼叫 77, 86, 87, 92
  - 函數 57
  - 程式, 系統 92
  - 程式, 直接呼叫 73
  - 語言環境 72
  - 儲存程序 86, 87
  - Java 應用程式 76
  - REXX 程式 77
- 呼叫 Net.Data 31
  - 使用 CGI 31
  - 使用巨集 31
  - 套表 31, 34, 99
  - 概觀 31
  - 鏈結 31, 34, 98
  - HTML 區塊 61
  - URL 31, 34, 99

- 定義變數
  - 查詢字串資料 44
  - DEFINE 陳述式或區塊 43
  - HTML 套表 SELECT, INPUT 及 TEXTAREA 標籤 44
- 注意事項 113
- 直接呼叫語言環境
  - 支援的資料類型 74
  - 呼叫程式 73
  - 從程式傳回值 75
  - 傳遞參數 73
  - 傳遞參數時常見的錯誤 75
  - 概觀 73
- 表格函數 60
- 表格處理程序變數 51
- 表格變數 50
- 表頭資訊, REPORT 區塊 63

## 〔九劃〕

- 保護資產 23
- 宣告部份, 巨集結構 37
- 建立起始設定檔案 7
- 持續巨集 95
- 持續函數 61
- 架構 Net.Data
  - 起始設定檔
    - 更新 6
    - 建立 6
  - 架構變數陳述式 8
  - 路徑陳述式 13
  - 說明 6
  - ENVIRONMENT 陳述式 17
- 設置語言環境 18
- 概觀 5
- Net.Data 檔 20

- 架構變數陳述式
  - 架構起始設定檔案中的 8
  - 說明 8
  - DTWR\_CLOSE\_REGISTRIES 12
  - DTW\_MACRO\_CACHE\_SIZE 8
  - DTW\_PAD\_PGM\_PARMS 9
  - DTW\_SHOWSQL 9
  - DTW\_SMTP\_CCSSID 10
  - DTW\_SMTP\_CHARSET 10
  - DTW\_SMTP\_SERVER 11
  - DTW\_SQL\_ISOLATION 11
  - DTW\_SQL\_NAMING\_MODE 12

## 〔十劃〕

- 套表 31, 33
  - 位在網頁中可呼叫 Net.Data 33
  - 呼叫 Net.Data 31, 34, 99
- 格式化資料輸出 62
- 純本文檔函數 60
- 起始設定檔
  - 更新 6
  - 建立 6, 7
- 架構變數陳述式 8
- 格式 7
- 路徑陳述式 13
- 說明 6
- ENVIRONMENT 陳述式 17
- 迴路, WHILE 區塊 69

## 〔十一劃〕

- 動態建立變數名稱 45
- 區塊, 巨集 39
- 參照變數 45
- 執行 SQL 陳述式 80
- 執行指令 92
- 執行效能
  - 系統語言環境 106
  - 將語言環境最佳化 105
  - REXX 語言環境 105
  - SQL 語言環境 106
- 執行變數 47
- 從程式傳回值 75
- 授權
  - 安全 26
  - 設定 Net.Data 檔案的存取權 20
- 啟動 Net.Data 31
- 條件
  - 變數 46
  - 邏輯, IF 區塊 67
- 異動處理 95
- 符記大小 42



處理結果集合, 儲存程序 88

## 〔十二劃〕

報告格式, 自行設定 63

報告變數 51

報表

多個具有一個函數呼叫的 64

預設值 64

提高執行效能 105

結果集合 88, 89, 90

多重 90

指南及限制 66

預設值報告 90

處理, 儲存程序 88

單一 89

## 〔十三劃〕

傳送參數 78, 88

儲存程序 88

REXX 程式 78

傳遞參數 92

系統語言環境 92

直接呼叫語言環境 73

Java 應用程式語言環境 77

傳遞參數時常見的錯誤 75, 86

資料類型 82, 84, 87

直接呼叫的 74

儲存程序的 87

DATALINK 84

LOB 82

路徑陳述式

更新準則 13

保護資產 26

架構起始設定檔案中的 13

DTW\_JAVA\_CLASSPATH 16

EXEC\_PATH 14

FFI\_PATH 16

HTML\_PATH 16

INCLUDE\_PATH 15

MACRO\_PATH 13

預設值報告 89, 90

列印 63

設定儲存程序的 89, 90

## 〔十四劃〕

算術函數 59

語言環境 77, 92

支援的 72

安全 73

語言環境 77, 92 (繼續)

系統 92

呼叫 72

直接呼叫 73

架構 ENVIRONMENT 陳述式 17

架構起始設定檔案中的 17

處理錯誤狀況 72

設定 18

範例 17

變數 52

Java 應用程式 76

REXX 77

SQL 80

## 〔十五劃〕

廣域識別字範圍 42

樣本巨集 109

標底資訊, REPORT 區塊 63

範圍, 識別字

巨集 42

廣域 42

FUNCTION 區塊 42

REPORT 區塊 43

ROW 區塊 43

編碼結果集合中的 DataLink URL 84

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 41

錯誤狀況, 語言環境 72

## 〔十七劃〕

儲存程序 86, 87, 88, 89, 90

多重結果集合 90

有效資料類型 87

步驟 87

從巨集呼叫 86

處理結果集合 88

單一結果集合 89

傳送參數 88

預設值報告 89, 90

REPORT 區塊 89, 90

檔案, 設定 Net.Data 的存取權 20

環境變數 47

隱藏變數

保護資產 26

隱藏變數名稱 48

## 〔十八劃〕

雜項變數 50

## 〔十九劃〕

識別字範圍 42

鏈結 31, 32

位在網頁中可呼叫 Net.Data 32

呼叫 Net.Data 31, 34, 98

類型, 變數 46

## 〔二十三劃〕

變數

列示 49

定義 43

表格處理程序 51

架構, 陳述式

以空白填補參數 (DTW\_PAD\_PGM\_PARMS) 9

巨集快取大小 (DTW\_MACRO\_CACHE\_SIZE) 8

起始設定檔 8

停用 SHOWSQL (DTW\_SHOWSQL) 9

啓用 SHOWSQL (DTW\_SHOWSQL) 9

電子郵件 SMTP CCSID (DTW\_SMTP\_CCSSID) 10

電子郵件 SMTP 字集

(DTW\_SMTP\_CHARSET) 10

電子郵件 SMTP 伺服器

(DTW\_SMTP\_SERVER) 11

說明 8

SMTP 字集 (DTW\_SMTP\_CHARSET) 10

SMTP 伺服器 (DTW\_SMTP\_SERVER) 11

SQL 命名模式 (DTW\_SQL\_NAMING\_MODE) 12

SQL 隔離 (DTW\_SQL\_ISOLATION) 11

Web 登記關閉 (DTWR\_CLOSE\_REGISTRIES) 12

動態建立名稱 45

動態建立的參照 45

參照 45

執行 47

條件 46

符記大小 42

報告 51

語言環境 52

說明 42

範圍。 42

環境 47

隱藏 48

雜項 50

類型 42, 46

table 50

## B

BLOB 82

## C

CGI-BIN 程式庫, 複製 Net.Data 程式物件 5

CLOB 82

## D

DATALINK 資料類型 84

編碼 URL 84

DataLink 檔案管理程式 84

DBCLOB 82

DEFINE 區塊

定義變數 43

說明 39

DTWR\_CLOSE\_REGISTRIES 12

DTW\_DEFAULT\_REPORT 64

DTW\_DIRECTCALL 73

DTW\_JAVAPPS 76

DTW\_JAVA\_CLASSPATH 16

DTW\_MACRO\_CACHE\_SIZE 8

DTW\_PAD\_PGM\_PARMS 9

DTW\_REXX 77

DTW\_SHOWSQL 9

DTW\_SMTP\_CCSSID 10

DTW\_SMTP\_CHARSET 10

DTW\_SMTP\_SERVER 11

DTW\_SQL 80

DTW\_SQL\_ISOLATION 11

DTW\_SQL\_NAMING\_MODE 12

DTW\_SYSTEM 92

## E

ENVIRONMENT 陳述式

使用者定義的語言環境的 7

服務程式程式 17

架構起始設定檔案中的 17

參數列示 17

語言環境類型 17

語法 17

說明 17

範例 18

## F

FFI\_PATH 16

FUNCTION 區塊

呼叫函數 57

格式化輸出 62

說明 39

識別字範圍 42

## H

HTML 31, 32, 33  
    在巨集中建立 61  
    表格的標籤 63  
    套表 31, 33  
        呼叫 Net.Data 31, 34, 99  
        關於 33  
        SELECT, INPUT 與 TEXTAREA 標籤, 定義變數 44  
    區塊  
        呼叫 Net.Data 61  
        處理程序 62  
        說明 40  
        範例 61  
    無法識別資料 62  
    鏈結 31, 32  
        呼叫 Net.Data 31, 34, 98  
        關於 32  
    FORM「提出」按鈕 62  
HTML\_PATH 16

## I

IF 區塊 67  
INCLUDE\_PATH 15

## J

Java 應用程式語言環境  
    呼叫程式 76  
    設定 18  
    傳遞參數 77  
    概觀 76

## L

LOB (大型物件) 82  
    支援的類型 82  
    具有 SQL 及 ODBC 語言環境 82

## M

MACRO\_FUNCTION 區塊  
    呼叫函數 57  
    語法 53  
MACRO\_PATH 13  
MESSAGE 區塊  
    處理程序 55  
    語法 55  
    說明 55  
    範例 56  
    範圍。 55

## N

Net.Data  
    巨集, 開發 37  
    安全機制 26  
    呼叫 31  
    架構 5  
    概觀 1  
    檔案, 存取權 20  
Net.Data 巨集。請參閱巨集。 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

## R

REPORT 區塊 89, 90  
    多重 64  
    多個的指南 66  
    表頭及標底資訊 63  
    限制 66  
    格式化資料輸出 62  
    預設值報告 64  
    說明 62  
    範例 64  
    範圍。 43  
    儲存程序 89, 90  
RETURN\_CODE 變數 55, 72  
REXX 語言環境 77, 78  
    呼叫程式 77  
    傳送參數 78  
    概觀 77  
ROW 區塊, 識別字範圍 43

## S

SQL  
    命名模式架構變數 12  
    隔離架構變數 11  
SQL 語言環境  
    執行 SQL 陳述式 80  
    設定 19  
    傳遞參數時常見的錯誤 86  
    概觀 80  
SQLCODE 72, 73  
SSQL 陳述式, 執行 80

## U

URL 31  
    呼叫 Net.Data 31, 34, 99  
    定義變數 44

## W

Web 登記函數 60

Web 登記, 關閉變數 12

WHILE 區塊 69

# 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。

Net.Data  
OS/400 版管理及程式設計指南

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

台灣北區郵政管理局 登記
北台字第 0587 號

(免貼郵票)

寄件人 姓名：  
地址：

寄

折疊線





Printed in Singapore