



Net.Data 語言環境參考手冊



Net.Data 語言環境參考手冊

注意事項

請務必先詳讀第85頁的『附錄B. 注意事項』中的資訊，再使用此資訊及其所支援的產品。

第四版 (1998 年 9 月)

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

目錄

序文	vii
關於 Net.Data	vii
關於這本書	vii
誰應閱讀本書	viii
關於本書中的範例	viii
關於 Net.Data 語言環境	ix
第1篇 Net.Data 提供的語言環境	1
第1章 Net.Data 提供之語言環境的概觀	3
第2章 使用 Net.Data 提供的語言環境	5
純本文檔介面語言環境	5
IMS Web 語言環境	5
Java Applet 語言環境	6
建立 Java Applet	6
產生 Applet 標籤	6
Java Applet 範例	10
使用 Net.Data Java Applet 介面	11
Java 應用程式語言環境	13
Java 語言環境檔案結構	13
建立 Java 函數	14
定義 Java 語言環境 Cliette	14
架構 Java 語言環境的 Net.Data	15
建立及執行巨集檔	16
ODBC 語言環境	16
Oracle 語言環境	17
Perl 語言環境	20
REXX 語言環境	21
SQL 語言環境	22
Sybase 語言環境	22
系統語言環境	24
Web 登記語言環境	25
架構 Net.Data 語言環境	26
第2篇 非 IBM 語言環境	27
第3章 建立新的語言環境	29
設計 DLL 或共用程式庫	29
我應該提供哪一種語言環境介面？	30
處理輸入參數	30
處理使用者要求	31

處理輸出參數	31
通信錯誤狀況	31
語言環境通信結構	31
dtw_lei 結構	32
dtw_parm_data 結構	33
語言環境介面函數	35
dtw_initialize()	35
dtw_execute()	36
dtw_getNextRow().	36
dtw_cleanup()	37
設計語言環境陳述式.	37
ENVIRONMENT 陳述式語法	38
ENVIRONMENT 陳述式範例	39
第4章 語言環境程式設計介面公用程式函數	41
語言環境公用程式函數	41
用來管理記憶體之公用程式函數	41
管理架構變數用的公用程式函數	41
表格操作的公用程式函數	42
橫列操作的公用程式函數	43
公用程式函數語法參考手冊	43
dtw_free()	44
dtw_getvar()	45
dtw_malloc()	46
dtw_row_SetCols()	47
dtw_row_SetV()	48
dtw_strdup()	49
dtw_table_AppendRow()	50
dtw_table_Cols()	51
dtw_table_Delete().	52
dtw_table_DeleteCol()	53
dtw_table_DeleteRow()	54
dtw_table_GetN()	55
dtw_table_GetV()	56
dtw_table_InsertCol()	57
dtw_table_InsertRow()	58
dtw_table_MaxRows()	59
dtw_table_New()	60
dtw_table_QueryColnoNj()	61
dtw_table_Rows()	62
dtw_table_SetCols()	63
dtw_table_SetN()	64
dtw_table_SetV()	65

第3篇 附錄與後記	67
----------------------------	-----------

附錄A. 語言環境模版	69
附錄B. 注意事項	85
商標	86
名詞解釋	87
索引	89

序文

感謝您選購 Net.Data 第 2 版 -- 用來建立動態 Web 網頁的 IBM 開發工具！有了 Net.Data，您就可以納入來自各種資料來源的資料，並使用您已熟悉的程式設計語言功能，進而可以快速地製作出帶有動態內容的 Web 網頁。

Net.Data 第 2 版的執行效能有明顯地改進，並且提供新的功能，讓您擁有建置及設計「網際網路」事務解決方案的能力。

關於 Net.Data

有了 IBM 的 Net.Data 產品，您就可以利用關聯式及非關聯式的資料庫管理系統 (DBMS) 的資料，來建立動態的 Web 網頁，資料庫管理系統包括 DB2、IMS 及使用 ODBC 的資料庫，另外，您也可以利用以 Java、JavaScript、Perl、C、C++ 及 REXX 等程式設計語言所撰寫的應用程式。

您可以把 Net.Data 當作在巨集處理器，在 Web 伺服器上當作中繼器來執行。您也可以撰寫名為巨集的 Net.Data 應用程式，Net.Data 可將它解譯，以建立動態的 Web 網頁，並依使用者的輸入、資料庫的現行狀態、舊有的企業邏輯及其他您設計在巨集中的因數，來自行設定內容。

URL (通用資源位置) 形式的要求會從瀏覽器 (例如，Netscape 或 Internet Explorer) 傳輸到 Web 伺服器，Web 伺服器會將要求轉送到 Net.Data 以供執行。Net.Data 會尋找並執行巨集，並且會建置一個 Web 網頁，而該 Web 網頁是依據您所寫入的函數來自行設定的。這些函數可以：

- 將企業邏輯封裝在 Perl scripts、C 及 C++ 應用程式或 REXX 程式中
- 存取資料庫，例如 DB2

Net.Data 支援工業標準介面，例如「超文本轉送通信協定 (HTTP)」及「通用閘道介面 (CGI)」。HTTP 用在瀏覽器及 Web 伺服器之間，而 CGI 則用在 Web 伺服器及 Net.Data 之間。這樣可讓您選用您最喜歡的瀏覽器或 Web 伺服器來配合 Net.Data 使用。Net.Data 也可在各種作業系統上支援 FastCGI 及主要的 Web 伺服器 APIs。

關於這本書

本書討論在當您從 Net.Data 巨集檔呼叫程式或函數，或資料來源如 DB2、Oracle 或 Sybase 資料庫時，所會用到的 Net.Data 語言環境。它說明 Net.Data 所提供的每一個語言環境，同時也說明您可用來設計及建置您自己的語言環境的語言環境介面。

本書可能會提及一些已發表，但還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)」站台：

- <http://www.software.ibm.com/data/net.data>
- <http://www.as400.ibm.com/netdata>

誰應閱讀本書

撰寫 *Net.Data* 巨集的人可使用這個資訊來了解 *Net.Data* 提供給您的語言環境能力。本書也包含一些資訊，可供想要撰寫自己的 *Net.Data* 語言環境的使用者參考。

若要了解本書所討論的概念，您必須先熟悉 C 程式設計語言及 *Net.Data* 管理及程式設計手冊與 *Net.Data* 參考手冊中的資訊。

關於本書中的範例

本書中使用的範例力求簡單以便展示特定的概念，因此並未考慮所有可能的情況。有些並不完整，無法單獨使用。

關於 Net.Data 語言環境

Net.Data 的設計是爲了要能夠將新的程式設計語言及資料庫介面新增至可插用的形態。這些介面稱爲語言環境，並且會被當作 DLL 或共用程式庫來存取。語言環境讓您可存取支援您的動態 Web 網頁的應用程式及資料庫。若您用函數呼叫或 SQL 陳述式來呼叫語言環境，您就可以存取這些語言環境所提供的函數及公用程式，以配合您所使用的應用程式來運作。例如，您可以直接存取您的 ODBC 資料庫、使用 Perl 語言環境來呼叫 Perl scripts，或是呼叫 Java Applet 語言環境來執行 Java applet。

Net.Data 起始設定檔可將每一個語言環境名稱連結至 DLL 或共用程式庫。每一種語言環境必須支援一組由 Net.Data 定義的標準介面。第一次呼叫指定該語言環境之 FUNCTION 區塊的函數時，Net.Data 會載入起始設定檔中所指定的 DLL 或共用程式庫。

Net.Data 會解析 Net.Data 巨集、維護 Net.Data 變數、與語言環境通訊、以及根據 REPORT 及 MESSAGE 區塊規格將輸出格式化。語言環境支援針對 Net.Data 定義的介面、使 Net.Data 參數可被語言處理器透過與語言相關的方式被存取、呼叫語言直譯器、以及以某種與語言相關的方式從語言直譯器接收傳回的變數。

第x頁的圖1示範 Net.Data 與語言環境互動的情況。

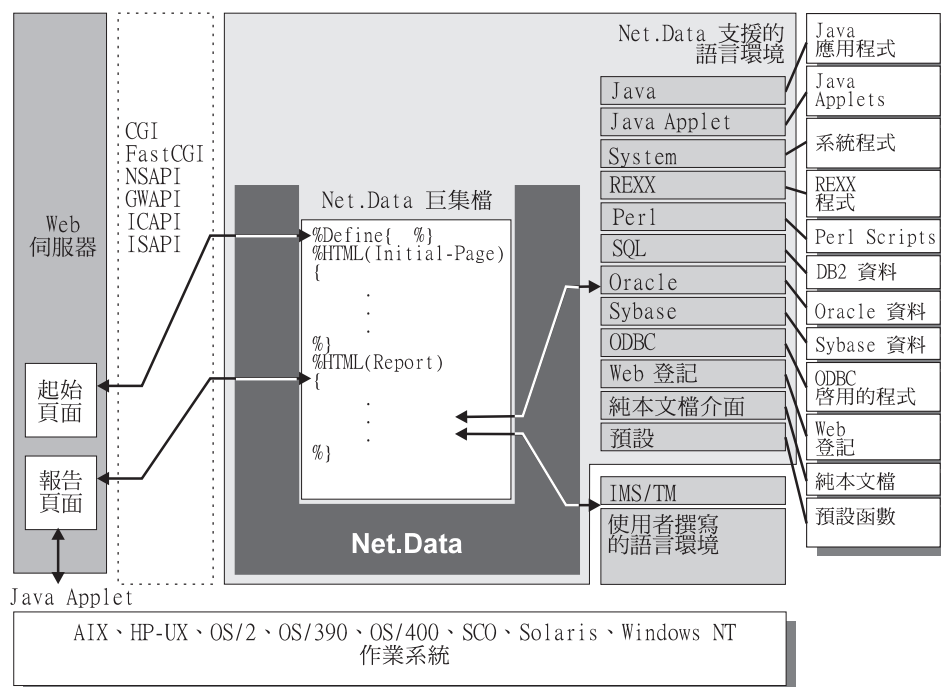


圖 1. Net.Data 及語言環境

在 Net.Data 應用程式中使用語言環境會用到二種作業。

- 使用 Net.Data 提供的語言環境來開發 Net.Data 應用程式。
- 開發新的語言或資料庫環境，以供其他使用者在開發 Net.Data 應用程式時使用。

本書的製作是爲了幫助您進行下列這二種作業：

- 第1頁的『第1篇 Net.Data 提供的語言環境』說明 Net.Data 所提供，可用來配合您的 Net.Data 應用程式的 Net.Data 提供的語言環境。
- 第27頁的『第2篇 非 IBM 語言環境』說明如何建立新的語言及資料庫環境。

第1篇 Net.Data 提供的語言環境

Net.Data 提供數種語言環境，可讓您從資料來源來回傳遞資訊。例如，SQL 的語言環境可讓您將原來的 SQL 查詢傳遞到資料庫。同樣地，REXX 語言環境可讓您呼叫 REXX 程式。

本段將說明每一種語言環境，以及如何在 Net.Data 起始設定檔中架構語言環境。

第3頁的『第1章 Net.Data 提供之語言環境的概觀』

第5頁的『純本文檔介面語言環境』

第5頁的『IMS Web 語言環境』

第6頁的『Java Applet 語言環境』

第13頁的『Java 應用程式語言環境』

第16頁的『ODBC 語言環境』

第17頁的『Oracle 語言環境』

第20頁的『Perl 語言環境』

第21頁的『REXX 語言環境』

第22頁的『SQL 語言環境』

第22頁的『Sybase 語言環境』

第24頁的『系統語言環境』

第25頁的『Web 登記語言環境』

第26頁的『架構 Net.Data 語言環境』

第1章 Net.Data 提供之語言環境的概觀

Net.Data 提供多種語言環境，但有些作業系統並不是支援所有的環境。表1列出 IBM 提供的語言環境。若要判定您的作業系統是否支援某語言環境，請參閱 *Net.Data* 參考手冊的作業系統參考附錄。關於您的作業系統所使用之語言環境陳述式的詳細資訊，請參閱您的 Net.Data README 檔案或「程式目錄」。

表 1. *Net.Data* 語言環境

語言環境	環境陳述式	說明
純本文檔介面	DTW_FILE	純本文檔介面 (FFI) 提供支援以文字檔當作資料來源的函數。
IMS Web	HWS_LE	IMS Web 語言環境可讓您使用 IMS Web 來提出 IMS 異動，並在 Web 瀏覽器上接收異動的輸出。
Java Applet	DTW_APPLET	Java applet 語言環境供您在您的 Net.Data 應用程式中使用 Java applet。要產生 applet 標籤，您必須提供 applet 標籤的限定元以及 applet 的參數列示。
Java 應用程式	DTW_JAVAPPS	Net.Data 透過 Java 語言環境來支援您現有的 Java 應用程式。
ODBC	DTW_ODBC	ODBC 語言環境透過 ODBC 介面來執行 SQL 陳述式，可存取多種資料庫管理系統。
Oracle	DTW_ORA	Oracle 語言環境可讓您直接存取您的 Oracle 資料。
Perl	DTW_PERL	Perl 語言環境可解譯 Net.Data 巨集 FUNCTION 區塊中指定的內部 Perl script，也可以執行儲存於個別檔案中的外部 PERL script。
REXX	DTW_REXX	REXX 語言環境可解譯 Net.Data 巨集 FUNCTION 區塊中指定的內部 REXX 程式，也可以執行儲存於個別檔案中的外部 REXX 程式。
SQL	DTW_SQL	SQL 語言環境透過 DB2 來執行 SQL 陳述式。SQL 陳述式的結果可用表格變數傳回。
Sybase	DTW_SYB	Sybase 語言環境可讓您直接存取您的 Sybase 資料。
系統	DTW_SYSTEM	系統函數可支援 FUNCTION 區塊中 EXEC 陳述式識別的外部程式呼叫。「系統」語言環境可將程式名稱及其參數傳遞給作業系統來執行，進而解譯 EXEC 陳述式。
Web 登記	DTW_WEBREG	「Web 登記」語言環境可提供函數以持續儲存應用程式相關之資料。

每一種語言環境的起始設定檔中都要有 ENVIRONMENT 陳述式，且伺服器的 /lib 或 /dll 目錄中都要有共用程式庫或 DLL 檔案。相關資訊，請參閱 *Net.Data* 管理及程式設計手冊中的架構章節。

建議：在執行變更之前，請先製作起始設定檔案的備份。

第2章 使用 Net.Data 提供的語言環境

下列區段說明 Net.Data 提供的語言環境，以及設置和使用這些語言環境的步驟。

純本文檔介面語言環境

如果您選擇使用純本文檔 (或純文字檔) 來作為您的資料來源，請使用純本文檔介面 (FFI) 及其相關函數來開啓、關閉、讀取、寫入及刪除 Web 伺服器上的檔案。在此情況下，您必須在起始設定檔中指定出 FFI_PATH 變數的路徑。

當 Web 從屬站透過瀏覽器提出要求時，檔案語言支援會使用 FFI 函數，來讀取或寫入 Web 伺服器上的檔案。FFI 在顯示檔案時，會將之視為一個記錄檔，每一筆記錄相當於 Net.Data 巨集表格變數中的一列，而記錄中的每一個值，相當於 Net.Data 巨集表格變數中的欄位值。FFI 會將檔案中的記錄讀入 Net.Data 巨集表格的橫列中，並將表格中的橫列寫到記錄中。

有關 FFI 內建函數的說明與語法，請參閱 *Net.Data 參考手冊*。

IMS Web 語言環境

IMS Web 語言環境是完整的端點對端點解決方案的一部份，可在「全球資訊網」環境中執行您的 IMS 異動。IMS Web 語言環境提供：

- 具有下列項目的 Net.Data 巨集：
 - 用來輸入異動輸入資料的 HTML
 - 呼叫 IMS Web 語言環境的 Net.Data FUNCTION 區塊
 - 顯示異動輸出的 HTML
- 由 IMS Web 語言環境所呼叫的異動 DLL 或共用程式庫

限制：只有在 Net.Data 當作 CGI 應用程式來執行時，才會支援 Net.Data 的 IMS Web 語言環境。具有 ICAPI 的 Net.Data 並不支援此語言環境。

「IMS Web 專用工具」會從異動的「訊息格式服務 (MFS)」來源，為 DLL 及巨集產生程式碼，並產生 MAK 檔案來建置 DLL 或共用程式庫。在建置 DLL 的可執行表之後，該工具會將 DLL 及巨集檔移至正在執行 Net.Data 的 Web 伺服器上。此時即可在 Web 環境中執行異動。

欲使用 IMS Web 語言環境，請：

1. 在執行 Net.Data 的 Web 伺服器上安裝 IMS Web Runtime 組件。IMS Web Runtime 組件的相關資訊，請參閱 *IMS Web User's Guide*：
<http://www.software.ibm.com/data/ims/about/imsweb/document/index.html>
2. 建立異動 DLL 檔案。

- a. 用「IMS Web 專用工具」來為您的異動產生 C++、MAK 及 Net.Data 巨集檔。
 - b. 若您執行 Net.Data 的作業系統不同於執行 IMS Web Studio 工具的作業系統，請將 DLL 原始檔案移到 IMS Web 開發機器以作為目標作業系統。例如，若您在 Windows NT 執行 IMS Web Studio 工具，而目標平台是 AIX 或 OS/390 時，請個別地移動異動 DLL 的原始檔到執行 AIX 或 OS/390 的 IMS Web 開發機器。
 - c. 用所產生的 MAK 檔案來建置異動 DLL 的可執行形式。
3. 將異動 DLL 檔案 (DTWproj.dll) 及 Net.Data 巨集檔 (DTWproj.d2w) 複製到您的 Web 伺服器。
- a. 把巨集放置在 Net.Data 取回巨集的來源目錄中。(請參閱 *Net.Data 管理及程式設計手冊* 架構章節中的 MACRO_PATH 陳述式。)
 - b. 把異動 DLL 或共用程式庫放置在 Web 伺服器取回 DLL 或共用程式庫的目錄中。
4. 使用「IMS Web 專用工具」-- DTWproj.htm 所產生之樣本檔案中的鏈結，來修改 Web 伺服器 HTML 樹狀結構中的 HTML 檔案。然後，您可以使用該鏈結來呼叫 Net.Data，並在 Web 瀏覽器上顯示異動的輸入 HTML 表格頁面。填寫異動輸入值，並選取表格上的 SUBMIT 按鈕，以執行異動，並在 Web 瀏覽器上接收其輸出。

IMS Web 使用 IMS TCP/IP Open Transaction Manager Access (OTMA) 連線，在 Web 伺服器及 IMS 環境之間通信。相關資訊，請參閱 IMS Web 首頁：

<http://www.software.ibm.com/data/ims/about/imsweb>

Java Applet 語言環境

Java applet 語言環境讓您在您的 Net.Data 應用程式中輕易地產生 Java applet 的 HTML 標籤。當您呼叫 Java applet 語言環境時，您可指定您的 applet 的名稱，並可傳遞任何該 applet 所需的參數。語言環境會處理巨集，並且會產生在 Web 瀏覽器執行 applet 所使用的 HTML applet 標籤。

此外，Net.Data 也提供一組您的 applet 可用以存取表格參數的介面。這些介面全部放在 DTW_Applet.class 中。

下面的部份將說明如何使用 Java applet 語言環境來執行您的 Java applet。

建立 Java Applet

在使用 Net.Data Java applet 語言環境之前，您必須先決定您要使用或要撰寫的 applet 有哪些。相關資訊，請參閱已建立之 applet 中的 Java 說明文件。

產生 Applet 標籤

您使用 Net.Data 函數呼叫來設定呼叫 applet 語言環境。函數呼叫不需要宣告。函數呼叫的語法如下：

@DTWA_AppletName(parm1, parm2, ..., parmN)

- DTWA_ 會識別 applet 語言環境的函數呼叫。
- AppletName 是產生標籤的 applet 之名稱。
- parm1 至 parmN 至是用來產生 PARAM 標籤的參數。

欲撰寫可產生 **applet** 標籤的巨集檔，請：

1. 在巨集檔的 DEFINE 部份中，定義 applet 所需的任何參數。這些參數包括您要用來當作 applet 輸入值的任何 applet 標籤屬性、Net.Data 變數及 Net.Data 表格參數。例如：

```
%define{
  DATABASE = "celdial"           <=Net.Data 變數：資料庫名稱
  MyGraph.codebase = "/netdata-java/" <=必要的 applet 參數
  MyGraph.height = "200"         <=必要的 applet 參數
  MyGraph.width = "400"          <=必要的 applet 參數
  MyTitle = "This is my Title"   <=Net.Data 變數：Web 網頁的名稱
  MyTable = %TABLE(all)          <=儲存查詢結果的表格
%}
```

2. 選用項目：指定查詢資料庫來產生一組結果，以作為 applet 的輸入值。當您在使用會產生圖表或表格的 applet 時，這個動作會非常有用。例如：

```
%FUNCTION(DTW_SQL) mySQL(OUT table){
  從 ibmuser.guests 選取人名，年齡
%}
```

3. 在 Net.Data 巨集中指定函數呼叫，以呼叫 Java applet 語言環境，並呼叫 applet。函數呼叫可指定您想要傳遞到語言環境之 applet 及參數的名稱。這些參數包括您要用來當作 applet 輸入值的任何 Net.Data 變數及 Net.Data 表格或直欄參數。

例如：

```
%HTML(report){                  <=HTML 區塊的開始
@mySQL(MyTable)                  <=SQL 函數呼叫
                                mySQL
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable) <=Applet 函數呼叫
%}
```

4. 呼叫 Net.Data 並執行巨集檔。欲知如何呼叫 Net.Data，請參閱 *Net.Data 管理及程式設計手冊*。

Applet 標籤屬性

您可以在 Net.Data 巨集中隨意指定 applet 標籤屬性。Net.Data 會取代所有具有 *AppletName.attribute* 形式的變數，變成 applet 標籤的屬性。在 applet 標籤中定義屬性的語法如下：

```
%define AppletName.attribute = "value"
```

對所有的 applet 而言這些屬性是必需的：

- *codebase*: applet 的位置，以 URL 來識別。

- *height*: applet 在圖點中的高度。
- *width*: applet 在圖點中的寬度。

例如，若您的 applet 稱為 MyGraph，您可將這些必需屬性定義如下：

```
%DEFINE{
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
%}
```

在DEFINE 段落中不需要實際指定值。您可以使用 DTW_ASSIGN 函數來設定值。如果您未定義 *AppletName.code* 變數，則 Net.Data 會為 applet 標籤新增一個預設的程式碼參數。程式碼參數的值為 *AppletName.class*，*AppletName* 是指您的 applet 的名稱。

Applet 標籤參數

您定義在函數呼叫中傳遞給 Java applet 語言環境的一系列參數。您可以傳遞的參數包含：

- Net.Data 變數 (包括 LIST 變數)
- Net.Data 表格
- Net.Data 表格的直欄

當您傳遞參數時，Net.Data 會用您指定給該參數的名稱及數值，在 HTML 輸出中建立 Java applet PARAM 標籤。您不能傳遞字串文字或函數呼叫的結果。

Net.Data 變數參數： 您可以將 Net.Data 變數當作參數來使用。如果您在巨集的 DEFINE 區塊中定義一個變數，並且將在 DTWA_AppletName 函數呼叫中傳遞該變數值，則 Net.Data 會產生一個具有與該變數相同之名稱及數值的 PARAM 標籤。例如，假設的巨集陳述式如下：

```
%define{
...
MyTitle = "這是我的標題"
%}

%HTML(report){
@DTWA_MyGraph( MyTitle, ...)
%}
```

Net.Data 產生下列 applet PARAM 標籤：

```
<param name = 'MyTitle' value = "這是我的標題" >
```

Net.Data 表格參數：

在每次呼叫 Java applet 語言環境時， Net.Data 就會自動產生一個名為 DTW_NUMBER_OF_TABLES 的 PARAM 標籤，以指定該函數呼叫是否已傳遞任何表格變數。該值就是 Net.Data 在函數中使用之表格變數的個數。如果函數呼叫中沒有指定任何表格變數，就會產生下列標籤：

```
<param name = "DTW_NUMBER_OF_TABLES" value = "0" >
```

您可以傳遞一或多個 Net.Data 表格變數來當作函數呼叫上的參數。如果您在 DTWA_AppletName 函數呼叫上指定 Net.Data 表格變數， Net.Data 就會產生下列 PARAM 標籤：

表格名稱參數標籤：

這個標籤指定要傳遞的表格名稱。此標籤的語法如下：

```
<param name = 'DTW_TABLE_i_NAME' value = "tname" >
```

其中 *i* 是表格數目，依函數呼叫的次序而定，而 *tname* 是表格名稱。

橫列及直欄規格參數標籤：

產生 PARAM 標籤，以指定特定表格的橫列及直欄數。此標籤的語法如下：

```
<param name = 'DTW_tname_NUMBER_OF_ROWS' value = "rows" >  
<param name = 'DTW_tname_NUMBER_OF_COLUMNS' value = "cols" >
```

其中表格的名稱為 *tname*， *rows* 是表格中的列數，而 *cols* 是表格中的直欄數。對於函數呼叫中指定的每一個唯一的表格，都將產生這一對標籤。

欄位值參數標籤：

這個 PARAM 標籤指定特定直欄的直欄名稱。此標籤的語法如下：

```
<param name = 'DTW_tname_COLUMN_NAME_j' value = "cname" >
```

其中，表格名稱為 *tname*， *j* 是直欄個數，而 *cname* 是表格中的直欄名稱。

橫列值參數標籤：

這個 PARAM 標籤指定特定橫列及直欄的值。此標籤的語法如下：

```
<param name = 'DTW_tname_cname_VALUE_k' value = "val" >
```

其中表格名稱是 *tname*， *cname* 是直欄名稱， *k* 是列號，而 *val* 是符合相對應之橫列及直欄中之數值的數值。

表格直欄參數： 您可以在函數呼叫中，將表格直欄當作參數來傳遞，以產生特定直欄的標籤。 Net.Data 僅針對所指定之直欄來產生相對應的 applet 標籤。表格直欄參數使用下列語法：

```
@DTWA_AppletName(DTW_COLUMN( x )Table)
```

其中 *x* 是該表格中之直欄的名稱或號碼。

表格直欄參數使用的 `applet` 標籤與為表格參數所定義的相同。

在不支援 Java 的瀏覽器上，Applet 標籤的替代本文

變數 `DTW_APPLET_ALTTEXT` 指定是因應在不支援 Java 或關閉 Java 支援的瀏覽器中顯示本文。例如，下列變數定義：

```
%define DTW_APPLET_ALTTEXT = "<P>對不起，您的瀏覽器不支援 Java。"
```

產生下列 HTML 標籤及文字：

```
<P>對不起，您的瀏覽器不支援 Java。<BR>
```

若未定義這個變數，則不顯示替代本文。

Java Applet 範例

下列範例將示範呼叫 Java applet 語言環境的 `Net.Data` 巨集檔，以及語言環境所產生的結果 `applet` 標籤。

`Net.Data` 巨集檔包含下列針對 `applet` 語言環境的函數呼叫：

```
%define{
DATABASE = "celdial"
DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not Java-enabled."
DTW_DEFAULT_REPORT = "無"
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "這是我的標題"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
從 ibmuser.guests 選取人名，年齡
%}
%HTML(report){
@mySQL(MyTable)
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
%}
```

`DEFINE` 部份中的 `Net.Data` 巨集行可指定 `applet` 標籤的第一行：

```
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
```

語言環境會下列限定元產生 `applet` 標籤：

```
<applet code = 'MyGraph.class'
codebase = '/netdata-java/' width = '400'
height = '200' >
```

Net.Data 傳回由輸出表 MyTable 中 Net.Data 巨集檔的 SQL 部份所產生的 SQL 查詢結果。
在 DEFINE 部份指定了該表格：

```
MyTable = %TABLE(all)
```

對巨集中 applet 的呼叫是設定在 HTML 部份：

```
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
```

根據函數呼叫中的參數，Net.Data 會產生完整的 applet 標籤，其中包含結果表格的相關資訊，例如傳回的直欄個數、橫列個數及結果橫列。Net.Data 會針對結果表中的每一個資料格產生一個參數標籤，如下列範例所示：

```
param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
```

參數名稱 *DTW_MyTable_ages_VALUE_1* 可指定表格 MyTable 中的表格資料格 (row 1, column ages)，該表格的值為 4。呼叫 applet 之函數呼叫中的關鍵字 DTW_COLUMN 可指定您是否只對結果表 MyTable 的直欄經歷時間有興趣，MyTable 如下所示：

```
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
```

下列輸出顯示 Net.Data 為該範例所產生的完整 applet 標籤：

```
<applet code = 'MyGraph.class'
codebase = '/netdata-java/' width = '400' height = '200' >
<param name = 'MyTitle' value = "這是我的標題" >
<param name = 'DTW_NUMBER_OF_TABLES' value = "1" >
<param name = 'DTW_TABLE_1_NAME' value = "MyTable" >
<param name = 'DTW_MyTable_NUMBER_OF_ROWS' value = "5" >
<param name = 'DTW_MyTable_NUMBER_OF_COLUMNS' value = "1" >
<param name = 'DTW_MyTable_COLUMN_NAME_1' value = "ages" >
<param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
<param name = 'DTW_MyTable_ages_VALUE_2' value = "32">
<param name = 'DTW_MyTable_ages_VALUE_3' value = "31" >
<param name = 'DTW_MyTable_ages_VALUE_4' value = "28" >
<param name = 'DTW_MyTable_ages_VALUE_5' value = "40" >
<P>對不起，您的瀏覽器不支援 Java。<BR>
</applet>
```

使用 Net.Data Java Applet 介面

Net.Data 在稱為 DTW_Applet.class 的類別中提供一套介面，您可以與 Java applet 一起使用，有助於處理由表格變數產生的 PARAM 標籤。您可以建立具擴充介面的 applet，以呼叫從 applet 來的常式。

Net.Data 提供以下這些介面：

- **int GetNumberOfTables()** 傳回在 applet 標籤中找到的表格數目。
- **String [] GetTableNames()** 傳回在 applet 標籤中找到的表格名稱的列示。
- **int GetNumberOfColumns(String table_name)** 傳回在表格 table_name 中的直欄數。
- **int GetNumberOfRows(String table_name)** 傳回在表格 table_name 中的橫列數。

- **String[] GetColumnNames(String table_name)** 傳回在表格 table_name 中的直欄名稱。
- **String[][] GetTable(String table_name)** 傳回含有表格之橫列與直欄值的二維陣列字串。

欲存取介面，請在您的 applet 程式碼中使用 EXTENDS 關鍵字，來將您的 applet 從 DTW_APPLET 類別中再分類，如下列範例所示：

```
import java.io.*;
import java.applet.Applet;

public class myDriver extends DTW_Applet
{
    public void init()
    {
        super.init();

        if (GetNumberOfTables() > 0)
        {
            String [] tables = GetTableNames();
            printTables(tables);
        }
    }

    private void printTables(String[] tables)
    {
        String table_name;

        for (int i = 0; i < tables.length; i++)
        {
            table_name = tables[i];
            printTable(table_name);
        }
    }

    private void printTable(String table_name)
    {
        int nrows = GetNumberOfRows(table_name);
        int ncols = GetNumberOfColumns(table_name);

        System.out.println("表格： " + table_name + " 含有 " + ncols + " 個直欄
與
                                " + nrows + " 橫列。");

        String [] col_names = GetColumnNames(table_name);

        System.out.println("-----");

        for (int i = 0; i < ncols; i++)
            System.out.print(" " + col_names[i] + " ");
        System.out.println("\n-----");

        String [][] mytable = GetTable(table_name);

        for (int j = 0; j < nrows; j++)
```



```

        {
        for (int i = 0; i < ncols; i++)
            System.out.print(" " + mytable[i][j] + " ");

        System.out.println("\n");
        }
    }
}

```

Java 應用程式語言環境

Net.Data 透過 Java 語言環境來支援您現有的 Java 應用程式。由於對 Java applet 及 Java 方法 (或應用程式) 的支援，您可以透過「Java 資料庫連接 (JDBC**) API」來存取 DB2。

JDBC 的詳細資訊，請參訪下列網站：

- IBM 軟體 (IBM Software) 備有 JDK 1.1，在您使用 JDBC 與 Net.Data 時會需要它：
<http://www.software.ibm.com/data/db2/jdbc/>
- JavaSoft 備有額外的 JDBC 驅動常式、JDBC API 文件及 JDBC 的最新更新資料：
<http://splash.javasoft.com/jdbc/>

Java 語言環境提供類似「遠端程序呼叫 (RPC)」的介面。您可以用 Net.Data 字串來作為參數，從您的 Net.Data 巨集檔來發出 Java 函數呼叫，您所呼叫的 Java 函數會傳回一個字串。當您使用 Java 語言環境時，您必須使用「Net.Data 現場連線」(「現場連線」的相關資訊，請參閱 *Net.Data 管理及程式設計手冊* 的執行效能章節)。欲使用 Java 語言環境，您必須完成下列步驟。在後續區段中會有這些步驟的詳細說明。

1. 撰寫您的 Java 函數。
2. 為您所有的 Java 函數建立 Net.Data cliette (Net.Data cliette 會啟動您的 Java 執行所在的「Java 虛擬電腦」)。
3. 在「現場連線」架構檔中定義 cliette 陳述式。
4. 啟動「連線管理程式」。
5. 執行會呼叫 Java 語言環境的 Net.Data 巨集檔。

每當您提出新的 Java 函數時，您都必須重新建立 Java cliette。

Java 語言環境檔案結構

在 Net.Data 安裝期間，Net.Data 會建立數個目錄。這些目錄包含您要建立 Java 函數、定義 cliette 及用 Java 語言環境來執行巨集時所需的檔案：

- 名為 UserFunctions.java 的樣本 Java 函數。
- 名為 makeClas 的樣本檔案。在執行時，這個檔案會為您的 Java 函數建立一個 Net.Data cliette 類別。

- 名為 launchjv 的樣本檔案，Net.Data cliette 會用它來啟動「Java 虛擬電腦」，並執行您的 Java 函數。

表2說明您作業系統上之檔案的目錄及檔名。

表 2. 用來建立 Java 函數的檔案

作業系統	檔名	目錄
OS/2	UserFunctions.java	javaapps
	launchjv.com	connect
Windows NT	UserFunctions.java	javaclas
	makeClas.bat	javaclas
	launchjv.bat	connect
UNIX	UserFunctions.java	javaapps
	launchjv	javaapps

建立 Java 函數

修改 Java 函數樣本檔案 UserFunctions.java，或以下列範例檔案 myfile.java 為模型建立一個新檔案：

```
=====myfile.java=====
import mypackage.*                                <=包含您的函數
public String myfctcall(...parameters from macro file...)
{
    return ( mypackage.mymethod(...parameters...));    <=對您函數的高階呼叫
}

public String lowlevelcall(...parameters...)
{
    string result;
    .....code using many functions of your package...
    return(result)
}
```

定義 Java 語言環境 Cliette

修改樣本檔案 makeClas.bat，或是針對您所有的 Java 函數建立一個新的 .bat 檔案來產生 Net.Data cliette 類別 dtw_samp.class。下列範例顯示批次檔 CreateServer 如何處理三個 Java 函數：

```
rem Batch file to create dtw_samp for Net.Data
java CreateServer dtw_samp.java UserFunctions.java myfile.java
javac dtw_samp.java
```

批次檔會處理下列檔案，另外也會處理 Net.Data 提供的部份檔案 Stub.java，以建立 dtw_samp.class。

- dtw_samp.java
- UserFunctions.java
- myfile.java

撰寫 JDBC 應用程式或 applet 和撰寫使用 DB2 CLI 或 ODBC 來存取資料庫的 C 應用程式非常相似。應用程式與 applet 之間的主要差別是：應用程式可能需要特定的軟體如 DB2 Client Application Enabler，才能夠和 DB2 通信。applet 則依賴可使用 Java 的 Web 瀏覽器，不需在從屬站上安裝 DB2 程式碼。

您的系統需要完成一些架構，才能夠使用 JDBC。這些注意事項在「DB2 JDBC 應用程式」及「Applet 支援網站」皆有所討論：

<http://www.software.ibm.com/data/db2/jdbc/db2java.html>

架構 Java 語言環境的 Net.Data

欲使用 Java 語言環境，您必須架構 Net.Data。請使用下列步驟來完成這些架構：

1. 建立一個批次檔來啟動 Java 應用程式，因為 Net.Data 不能直接啟動 Java 應用程式。Net.Data 使用這個檔案來啟動用來執行您的 Java 函數的「Java 虛擬電腦」。批次檔必須包括 java-classpath 陳述式，以確定可以找到必要的 Java 套裝軟體 (標準及特定應用程式套裝軟體)。例如，批次檔 launchjv.bat 包含下列 java-classpath：

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

2. 在「現場連線」架構檔 dtwcm.cnf 中，定義 cliette 來使用 Java 語言環境。針對 cliette 及具有 EXEC_NAME 架構變數的相關批次檔名稱，指定唯一埠號。在下列範例中，Java cliette 名稱被定義為 DTW_JAVAPPS，而 EXEC_NAME 架構變數被設定為批次檔 launchjv.bat 的名稱：

```
CLIETTE DTW_JAVAPPS{
MIN_PROCESS=1           <= 必要參數：此值必須是 1，
                        因為 JAVAPPS cliette 是多緒的。
MAX_PROCESS=1           <= 必要參數：此值必須是 1，
                        因為 JAVAPPS cliette 是多緒的。
START_PRIVATE_PORT=5100 <= 必須是唯一埠號
START_PUBLIC_PORT=5300  <= 必須是唯一埠號
EXEC_NAME=launchjv.bat  <= 包含 classpath 陳述式之批次檔的名稱
}
```

當您啟動「Net.Data 連線管理程式」時，Net.Data 會啟動架構檔中所指定的 Java cliette。然後您就可以使用該 cliette 來處理您的 Net.Data 巨集應用程式的 Java 語言環境要求。

3. 將每個 cliette 名稱新增至陳述式，以更新 Net.Data 起始設定檔 db2www.ini 中的 DTW_JAVAPPS ENVIRONMENT 路徑陳述式。例如：

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

建立及執行巨集檔

在您建立 Java 函數、定義 `cliette` 類別及架構 `Net.Data` 之後，您就可以執行含有 Java 函數參照的巨集檔。

1. 建立一個用來呼叫您的 Java 函數的巨集檔。例如，函數呼叫 `myfctcall` 會用 `cliette DTW_JAVAPPS` 來呼叫附隨在 `Net.Data` 的樣本函數。

```
%function (DTW_JAVAPPS) myfctcall( ....parameters from macro file ....)

%{ to call the sample provided with Net.Data %}
%function (DTW_JAVAPPS) reverse_line1(str);

%HTML(report){
you should see the string "Hello World" in reverse.
@reverse_line("Hello World")
You should have the result of your function call.
@myfctcall( ... ....)
%}
```

2. 啟動「連線管理程式」。「連線管理程式」的相關資訊，請參閱 *Net.Data* 管理及程式設計手冊 的執行效能章節。

3. 使用 HTML 鏈結、HTML 表格頁面或 URL 陳述式來啟動巨集並呼叫 `Net.Data`。例如，用下列 URL 陳述式來呼叫 `Net.Data` 巨集檔 `mymacro.d2w`：

```
http://myserver/cgi-bin/dt2www/mymacro.d2w/report
```

ODBC 語言環境

Open Database Connectivity (ODBC) 語言環境可透過 ODBC 介面來執行 SQL 陳述式。ODBC 的基礎是 X/Open SQL CAE 規格，可讓單一的應用程式存取多個資料庫管理系統。

要使用 ODBC 語言環境，您必須具備 ODBC 驅動常式以及驅動常式管理程式。您的 ODBC 驅動常式文件含有如何安裝以及架構 ODBC 環境的說明。

在 ODBC 環境下傳送 SQL 陳述式的方式和其它的 `Net.Data` 函數相似。下列範例是一個 `Net.Data` 巨集，可將多個 SQL 陳述式傳送給作為您 ODBC 資料來源的資料庫。使用 DATABASE 變數的作業系統，必須指定和 ODBC.INI 檔中之資料來源相同的資料庫。

```
%DEFINE {
    DATABASE="qesql"
    SHOWSQL="YES"
    table="int_null"
    LOGIN="netdata1"
    PASSWORD="ibmdb2"
}%

%function(dtw_odbc) sql() {
create table int_null (int1 int, int2 int)
%}
```

```

%function(dtw_odbc) sql2() {
insert into $(table) (int1) values (111)
%}

%function(dtw_odbc) sql3() {
insert into $(table) (int2) values (222)
%}

%function(dtw_odbc) sql4() {
select * from $(table)
%}

%function(dtw_odbc) sql5() {
drop table $(table)
%}

%HTML(REPORT) {
@sql1()
@sql2()
@sql3()
@sql4()
%}

```

Oracle 語言環境

Oracle 語言環境讓您以原本的方式存取 Oracle 資料。您可以從以 CGI、FastCGI、NSAPI、ISAPI 或 GWAPI 模式執行的 Net.Data 來存取 Oracle 表格。這個語言環境可支援 Oracle 7.2、7.3 及 8.0。

限制：

- 這個語言環境不支援儲存程序。
- DATABASE 變數不被用來存取 Oracle 資料庫。
- LOGIN 變數必須包含 Oracle 資料庫案例名稱。例如，*ora73* 即為定義在下列 LOGIN 變數中的案例名稱：

```
LOGON=admin@ora73
```

從 Net.Data 存取 Oracle

1. 驗證 Net.Data 起始設定檔中的 ENVIRONMENT 陳述式就 Oracle 語言環境而言是否正確。步驟及範例的相關資訊，請參閱 *Net.Data 管理及程式設計手冊* 中的架構章節。
2. 請確定您已安裝適當的 Oracle 組件，而且其運作如下：
 - a. 如果安裝 Net.Data 的機器上尚未安裝 SQL*Net，則請安裝。相關資訊，請參訪下列 URL：
http://www.oracle.com/products/networking/html/stdn_sqlnet.html
 - b. 驗證 Oracle *tnsping* 函數是否可用與 Web 伺服器所使用的同一個安全性權限來使用。若要驗證，請用您的 Web 伺服器的使用者 ID 登入，並鍵入：

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

其中 *oracle-instance-name* 是您的 Net.Data 巨集所存取之 Oracle 系統的名稱。

如果您的 Web 伺服器是在系統權限之下執行，您可能就無法驗證 Windows NT 上的 *tnsping* 函數。如果真是如此，則跳過這個步驟。

- c. 驗證 Oracle 表格可以用 Web 伺服器所使用的同一個安全性權限來存取。欲驗證，請用 SQL*Plus 字行指令工具來輸入 SQL SELECT 陳述式，以存取 Oracle 表格及具有您 Web 伺服器權限的 SQL SELECT 陳述式。例如：

```
SELECT * FROM tablename
```

如果您的 Web 伺服器是在系統權限之下執行，您可能就無法驗證 Windows NT 上的表格存取。如果真是如此，則跳過這個步驟。

疑難排解：如果上述步驟失敗，則請勿繼續。如果有任何步驟失敗，請檢查您的 Oracle 架構。

3. 確定 Oracle 環境變數皆已正確地設定在您的 Web 伺服器處理程序中。

- 若您使用 AIX，請將下列字行放置在 */etc/environment* 檔案中：

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

- 若您使用 Windows NT，請用「系統性質控制台」來新增下列環境變數：

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

暗示：您可能會需要額外的字行來供其他 Oracle 環境變數使用，而這些字行需視您所計畫使用的 Oracle 機能而定，例如國家語言支援及二階段確認寫入。這些環境變數的相關資訊，請參閱 Oracle 管理文件。

4. 測試從 Net.Data 至 Oracle 的連線。在您的 Net.Data 巨集檔中，在 LOGIN 及 PASSWORD 變數指定適當的數值。在存取 Oracle 資料庫時，請勿定義 DATABASE 變數。下列為巨集檔中連接陳述式的範例：

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

本機 Oracle 案例：

如果您只存取本機的 Oracle 案例，請勿指定遠端 Oracle 案例 (remote-oracle-instance) 名稱來作為登入使用者 ID 的一部份，如下列範例所示：

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

現場連線：

如果您使用「現場連線」，您就可以在「現場連線」架構檔中指定 LOGIN 及 PASSWORD，雖然基於安全性目的，我們不建議您這樣做。例如：

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

暗示：請勿指定 DATABASE 變數給 Oracle。

5. 執行 CGI shell script 來測試您的架構，以確定可以從您的 Web 伺服器存取 Oracle 案例，如下列範例所示：

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

另外，您也可以直接從 Net.Data 巨集執行 *tnsping*，如下列範例所示：

```
%DEFINE testora = %exec "tnsping oracle-instance-name"  
%HTML(report){  
< P>About to test Oracle access with tnsping.  
< hr>  
$(testora)  
< hr>  
< P>The Oracle test is complete.  
%}
```

疑難排解：

如果驗證步驟失敗，請檢查是否所有的進行步驟都已順利完成，驗證方法如下：

- 檢查您的 Oracle 架構。
- 驗證 Oracle 環境變數語法是否正確，並驗證沒有缺少任何變數。
- 檢查 Oracle 連線，確定您輸入的使用者 ID 和通行碼是正確的。

如果驗證步驟仍失敗，請聯絡「IBM 服務人員」。

範例：

在您完成存取驗證步驟之後，您可以用巨集檔中的函數來呼叫 Oracle 語言環境，如下列範例所示：

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

Perl 語言環境

Perl 語言環境可解釋您在 Net.Data 巨集的 FUNCTION 區塊中指定的插入式 (inline) Perl script，或者它也可以處理儲存在伺服器上的個別檔中的外部 Perl script。EXEC 陳述式可在 FUNCTION 區塊中識別對外部 Perl scripts 的呼叫，例如：

```
%EXEC{ perl-script-name [optional parameters] %}
```

Perl 語言環境無法直接傳遞或取回 Net.Data 變數，必須透過下列方式，Perl script 才可存取它們：

- Net.Data 將輸入參數傳遞至 Perl script，以作為環境變數。Perl script 可藉由讀取 Perl 組合陣列來取回參數。
- Perl script 藉由寫入具名管線 (其名稱是 Net.Data 在環境變數 DTWPIPE 中傳遞的名稱) 的方式，將輸出參數傳遞回語言環境。使用 DEFINE 陳述式語法來將資料寫入具名管線：

```
name = value
```

若有多個資料項目，請用換行或空白字元來分隔每個項目。

如果變數名稱與輸出參數同名，並且使用上方的語法，則新的值會置換現行值。如果變數名稱沒有與輸出參數相對應，則 Net.Data 會將其忽略。

下列範例顯示 Net.Data 如何從巨集檔傳遞變數。

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
    $date = 'date';
    chop $date;
    open(DTW, "> $ENV{DTWPIPE}") || die "Could not open: $!";
    print DTW "result = \"$date\"\n";
}%
%HTML(INPUT) {
    @today()
}%
```

如果 Perl script 位於名為 today.pr1 的外部檔中，相同的函數就可以下一個範例的方式來撰寫：

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
    %EXEC { today.pr1 %}
}%
```

Perl 語言環境程式可用 Net.Data 名稱來存取表格參數的值。表格 T 的欄位標題為 T_N_i，欄位值為 T_V_i_j。表格 T 中的列數及直欄數為 T_ROWS 及 T_COLS。

REPORT 與 MESSAGE 區塊可用於任何 FUNCTION 區段中。這些區塊是由 Net.Data 而不是由語言環境來處理。不過，Perl 程式可以將本文寫到標準輸出裝置資料流，並且直接操作輸出 HTML 表格頁面。

授權要訣： 請確定 Net.Data 執行下的使用者 ID 有權可以存取任何被這個語言環境參考的外部執行檔，包括正確的 Perl 直譯器版本。相關資訊，請參閱 *Net.Data 管理及程式設計手冊* 的架構章節中，有關指定 Web 伺服器對 Net.Data 檔案存取權的段落。

REXX 語言環境

REXX 語言環境可以解譯 Net.Data 巨集之 FUNCTION 區塊中指定的插入式 (inline) REXX 程式，也可以執行儲存在個別檔案中外部的 REXX 程式。外部 REXX 程式的呼叫是由 FUNCTION 區塊中的陳述式加以識別，例如：

```
%EXEC{ REXX-program-file-name [optional parameters] %}
```

REXX 語言環境使用 REXXStart() API 來告訴 REXX 直譯器執行指定的檔案，接著將後面跟有檔案名稱的參數傳遞給程式，就像您生前在命令行上輸入的一般。對於 REXX 程式，所有的參數都是透過 ARG[1] 來接收。

授權要訣： 請確定 Net.Data 執行下的使用者 ID 有權可以存取任何被這個語言環境參考的外部執行檔。相關資訊，請參閱 *Net.Data 管理及程式設計手冊* 的架構章節中，有關指定 Web 伺服器對 Net.Data 檔案存取權的段落。

變數替代：

只有在 FUNCTION 區塊的可執行陳述式區段中才會執行變數替代。不過，REXX 程式可存取參數，不論程式是定義在內部的 FUNCTION 區塊或外部的個別檔案。REXX 語言環境使用 REXX 語言處理器的 RexxVariablePool() 函數來和 REXX 程式共用 Net.Data 變數。這樣可容許 REXX 程式直接處理參數列示中識別的 Net.Data 變數。

REXX 程式可存取表格參數的值並當作 REXX 主變數 (stem variable)。就 REXX 程式而言，表格 T 的欄位標題為 T_N.i，欄位值為 T_V.i.j。表格 T 中的列數及直欄數為 T_ROWS 及 T_COLS。

改進 AIX 作業系統的執行效能：

若您在 AIX 系統多次呼叫 REXX 語言環境，您可考慮將 RXQUEUE_OWNER_PID 環境變數設定為 0。可多次呼叫 REXX 語言環境的巨集可以很容易地產生許多處理程序及大量的系統資源。

您可以將環境變數設定成下列三種方式的其中之一：

- 在巨集檔中使用 DTW_SETENV 建置函數：

```
@DTW_rSETENV("RXQUEUE_OWNER_PID", "0")
```

- 在 AIX 系統環境檔中插入下列陳述式：

```
/etc/environment: RXQUEUE_OWNER_PID = 0
```

這個方法會影響 REXX 對整個機器的行為。

- 在 HTTP Web 伺服器環境檔中；例如，對 Domino Go Webserver 插入下列陳述式：

```
InheritEnv RXQUEUE_OWNER_PID = 0
```

這個方法會影響 REXX 對 Web 伺服器的行為。

SQL 語言環境

SQL 語言環境用於透過 DB2 來執行 SQL 陳述式。SQL 陳述式的結果可用 `Net.Data` 預設表格或您指定的表格來傳回。

`Net.Data` 支援您授權的任何 SQL 陳述式。當您呼叫 `Net.Data` 當作 CGI 應用程式時，您可以就每一個 HTML 段落和一個資料庫連接，但您必須使用 `DATABASE` 變數指定資料庫名稱（OS/390 除外）。如果您的 DB2 資料庫和 Web 伺服器位於同一個機器上，您不必進行其它設定。否則，您可依據您所使用的作業系統，使用 `Client Application Enabler`（CAE）來存取遠端資料庫，也可以使用 `Database Connection Services`（DDCS）來取得 DB2 支援的所有異動支援。您也可以使用 `DataJoiner` 來存取其他資料庫。使用 `DataJoiner`，您可以對支援的資料庫使用二階段確認寫入（two-phase commit）。

Sybase 語言環境

`Sybase` 語言環境讓您以原本的方式存取 `Sybase` 資料。您可以從以 CGI、FastCGI、NSAPI、ISAPI 或 GWAPI 模式執行的 `Net.Data` 來存取 `Sybase` 表格。

限制：

- `Sybase` 語言環境不支援大型物件如影像或音效。只有對不具有 `SELECT` 陳述式的程序支援儲存程序。
- `Sybase` 語言環境需要「現場連線」使用 FastCGI。

欲從 `Net.Data` 存取 `Sybase`，請：

1. 驗證 `Net.Data` 起始設定檔中的 `ENVIRONMENT` 陳述式就 `Sybase` 語言環境而言是否正確。步驟及範例的相關資訊，請參閱 `Net.Data` 管理及程式設計手冊 中的架構章節。
2. 請確定您已安裝適當的 `Sybase` 組件，而且其運作如下：
 - a. 如果安裝 `Net.Data` 的機器上尚未安裝 `Sybase's Open Client`，則請安裝。相關資訊，請參閱 `Sybase Open Client` 文件。
 - b. 驗證 `Sybase ping` 函數是否可用與 Web 伺服器所使用的同一個安全性權限來使用。若要驗證，請用您的 Web 伺服器的使用者 ID 登入，並鍵入：

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

其中 `sybase-instance-name` 是您的 `Net.Data` 巨集所存取之 `Sybase` 系統的名稱。

如果您的 Web 伺服器是在系統權限之下執行，您可能就無法驗證 Windows NT 上的 *ping* 函數。如果真是如此，則跳過這個步驟。

- c. 驗證 Sybase 表格可以用 Web 伺服器所使用的同一個安全性權限來存取。欲驗證，請用 ISQL 字行指令工具來輸入 SQL SELECT 陳述式，以用您 Web 伺服器權限來存取 Sybase 表格。例如：

```
SELECT * FROM tablename
```

如果您的 Web 伺服器是在系統權限之下執行，您可能就無法驗證 Windows NT 上的表格存取。如果真是如此，則跳過這個步驟。

疑難排解：如果上述步驟失敗，則請勿繼續。如果有任何步驟失敗，請檢查您的 Sybase 架構。

3. 確定 Sybase 環境變數皆已正確地設定在您的 Web 伺服器處理程序中。

- 若您使用 AIX，請將下列字行放置在 */etc/environment* 檔案中：

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

- 若您使用 Windows NT，請用「系統性質控制台」來新增下列環境變數：

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

暗示：您可能會需要額外的字行來供其他 Sybase 環境變數使用，而這些字行需視您所計畫使用的 Sybase 機能而定，例如國家語言支援及二階段確認寫入。這些環境變數的相關資訊，請參閱 Sybase 管理文件。

4. 測試從 Net.Data 至 Sybase 的連線。在您的 Net.Data 巨集檔中，在 LOGIN、PASSWORD 及 DATABASE 變數中指定適當的數值。下列為巨集檔中連接陳述式的範例：

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

現場連線：如果您使用「現場連線」，您就可以在「現場連線」架構檔中指定 LOGIN 及 PASSWORD，雖然基於安全性目的，我們不建議您這樣做。例如：

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

5. 執行 CGI shell script 來測試您的架構，以確定可以從您的 Web 伺服器存取 Sybase 案例，如下列範例所示：

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

```
isql -u user_ID -p password << EOF
SELECT * FROM tablename
EOF
echo
```

疑難排解：

如果驗證步驟失敗，請檢查是否所有的進行步驟都已順利完成，驗證方法如下：

- 檢查您的 Sybase 架構。
- 驗證 Sybase 環境變數語法是否正確，並驗證沒有缺少任何變數。
- 檢查 Sybase 連線，確定您輸入的使用者 ID 和通行碼是正確的。

如果驗證步驟仍失敗，請聯絡「IBM 服務人員」。

範例：

在您完成存取驗證步驟之後，您可以用巨集檔中的函數來呼叫 Sybase 語言環境，如下列範例所示：

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

系統語言環境

「系統」語言環境是 Net.Data 定義的環境，可支援 FUNCTION 區塊中 EXEC 陳述式所識別的外部程式呼叫。

「系統」語言環境藉由傳遞程式名稱及參數給作業系統，以使用 C 語言 system() 函數呼叫來處理的方式，來解譯 EXEC 陳述式。這個方法不允許像 REXX 語言環境的作法，直接在外部程式及 Net.Data 之間來回傳遞變數，所以 Net.Data 會以下列方法來處理變數：

- Net.Data 將輸入參數傳遞至外部程式，以作為環境變數，且外部程式會將其取回：
 - UNIX CSHELL script 在環境變數名稱開頭加上錢號（\$），以參照環境變數，例如 \$x。
 - Perl 語言 script 參照組合陣列 ENV，以參照它們，例如 %ENV{'x'}。
 - DOS 批次檔參照以百分比符號（%）括住的變數名稱，例如 %x%。
- 大部份的作業系統會藉由寫入具名管線（其名稱是 Net.Data 在環境變數 DTWPIPE 中傳遞的名稱）的方式，將輸出參數傳遞回語言環境。（Net.Data for OS/400 使用環境變數將參數傳回給語言環境。）使用 DEFINE 陳述式語法來將資料寫入具名管線：

```
name = value
```

若有多個資料項目，請用換行或空白字元來分隔每個項目。

如果變數名稱有對應的輸出參數，新的值會置換現行值。 Net.Data 會忽略不符合任何輸出參數的變數名稱。

系統語言環境程式可透過 Net.Data 名稱來存取表格參數的值。表格 T 的欄位標題為 T_N_i，欄位值為 T_V_i_j。表格 T 中的列數及直欄數為 T_ROWS 及 T_COLS。

授權要訣： 請確定 Net.Data 執行下的使用者 ID 有權可以存取從「系統」語言環境呼叫的任何外部執行檔。相關資訊，請參閱 *Net.Data 管理及程式設計手冊* 的架構章節中，有關指定 Web 伺服器對 Net.Data 檔案存取權的段落。

Web 登記語言環境

「Net.Data Web 登記」可提供永久的儲存體給應用程式的相關資料。 Web 登記可用來儲存架構資訊，和其他在執行時可供 Web 應用程式動態存取的資料。只有當使用 Net.Data 或 Web 登記內建支援，透過 Net.Data 巨集，及特別為此目的設計的 CGI，您才能夠存取 Web 登記。作業系統的子集上有可用的 Web 登記。請參閱 *Net.Data 參考手冊* 中的 Net.Data 作業系統參考附錄。

要開發標準的 Web 網頁時，必須將 URL 直接放在該網頁的 HTML 來源中。這樣會造成變更鏈結的困難。此種固定的方式也會限制原本可輕易放在 Web 網頁上的鏈結類型。使用 Web 登記來儲存應用程式相關的資料 (如 URL)，有助於您用動態設定的鏈結來建立 HTML 頁面。

擁有登記寫入權的應用程式開發者及 Web 管理者，可將資訊儲存並維持在登記中。應用程式可在執行時，從其相關登記取回資訊。在此功能下，可讓您設計出有彈性的應用程式，同時也可允許應用程式和伺服器移動。您可以用 Net.Data 巨集來建立使用動態設定鏈結的 HTML 網頁。

資訊是以登記項目形式儲存在 Web 登記中。每一個登記項目都是由一對字串所組成：RegistryVariable 字串及相對應的 RegistryData 字串。任何可由一對字串代表的資訊，都可以儲存為登記項目。 Net.Data 使用變數字串來當作搜尋關鍵，以從登記尋找及取回特定的項目。

Web 登記的範例內容，請參閱 表3。

表 3. Web 登記範例

CompanyName	WorldConnect
Server	ftp.einet.net
JohnDoe/foreground	Green
CompanyURL/IBM Corp.	http://www.ibm.com
CompanyURL/Sun Microsystems Corp.	http://www.sun.com
CompanyURL/Digital Equipment Corp.	http://www.dec.com
JaneDoe/Home_page	http://jane.info.net

以下是您可以考慮使用 Web 登記的情況：

- 您可以使用 Web 登記來儲存伺服器與 URL 的別名，讓應用程式和伺服器的位置移動更容易。
- 應用程式的開發者可在其 Web 應用程式附上使其預先定義在登記中的資料(如LRL)。一般使用者可修改登記資料，以變更應用程式的行為。
- Web 登記可用來執行 URL 搜尋，讓其根據產品名稱、國家語言、製造商等等進行搜尋。

對 Web 登記中的索引項目來說，這些項目的 RegistryVariable 字串後面會附帶索引字串，所使用的語法如下：

RegistryVariable/Index

使用者會以一個個別的參數提供索引字串值給使用索引項目的內建函數。多個索引化的登記項目可以共用同一個 RegistryVariable 字串值，但也可以有不同的「索引」字串值以保持其唯一性。

表 4. 索引 Web 登記的樣本

Smith/Company_URL	http://www.ibmblink.ibm.com
Smith/Home_page	http://www.advantis.com

雖然上面二個索引登錄具有相同的 RegistryVariable 字串值 Smith，但是它們的索引字串是不同的。Web 登記函數會將它們視為兩個不同的項目。

架構 Net.Data 語言環境

您必須先架構 Net.Data 起始設定檔，才能使用具有 Net.Data 的語言環境，而且如果您有使用「現場連線」，則必須先架構「現場連線」架構檔。

以下是這些作業的概觀。關於如何架構 Net.Data 語言環境的詳細資訊，請參閱 *Net.Data 管理及程式設計手冊* 中的架構章節。另外，也請查看之前的語言環境部份，以取得特殊的架構指示。

- 驗證或更新 Net.Data 起始設定檔 db2www.ini 中的 ENVIRONMENT 陳述式。這些陳述式說明於 *Net.Data 管理及程式設計手冊* 中的『架構 Net.Data』章節。
- 在「現場連線」檔案 dtwcm.cnf 中，為資料庫或 Java 應用程式語言環境來定義 cliette。定義 cliette 的資訊說明於 *Net.Data 管理及程式設計手冊* 的架構章節中。

第2篇 非 IBM 語言環境

除了提供語言環境之外，Net.Data 還可讓您新增您自己的語言及資料庫環境。Net.Data 會將您的語言環境當作動態鏈結程式庫或共用程式庫來存取，以與 Net.Data 執行檔有所區隔。每一種語言環境都必須支援一組由 Net.Data 所定義的介面。下面幾章將討論如何建立新的語言環境，並說明語言環境程式設計介面及環境。

第29頁的『第3章 建立新的語言環境』

第37頁的『設計語言環境陳述式』

第41頁的『第4章 語言環境程式設計介面公用程式函數』

第3章 建立新的語言環境

Net.Data 把語言環境當作可插用的程式設計語言及資料庫介面來使用，並以 DLL 檔案或共用程式庫的形式來存取。Net.Data 提供一組語言環境，但是若這些語言環境不符合您的應用程式的需要，您可以建立新的語言環境。在您決定建立新的語言環境之前，請先判斷附隨 Net.Data 之 IBM 提供的語言環境是否符合您的要求。

一旦您決定要建立新的語言環境，您就必須完成下列步驟：

- 決定有哪些介面及函數是您必須提供給語言環境的。必須提供 `dtw_execute()` 介面，而且所提供的所有介面都必須完全符合 `dtwle.h` C 語言表頭中所定義的原型。
- 建立 `make` 檔或工作控制語言 (JCL) 來建置 DLL 或共用程式庫。
- 建立一個 DLL 或共用程式庫，其中含有您要提供之語言環境介面常式的集合。欲知如何建立 `make` 檔及建置 DLL 或共用程式庫，請參閱您作業系統的 C 或 C++ 說明文件。
- 使所有的介面都能夠在 DLL 或共用程式庫之外使用，以便讓 Net.Data 呼叫它們。
- 決定您的 ENVIRONMENT 架構陳述式，接著將它加到 Net.Data 起始設定檔中。詳細資訊，請參閱第37頁的『設計語言環境陳述式』。
- 將函數新增至會使用新語言環境的 Net.Data 巨集檔。

本章將說明如何設計語言環境。

- 『設計 DLL 或共用程式庫』
- 第31頁的『語言環境通信結構』
- 第35頁的『語言環境介面函數』
- 第37頁的『設計語言環境陳述式』

設計 DLL 或共用程式庫

當您建置語言環境時，您就會更新第69頁的『附錄A. 語言環境模版』中所提供的模版，以併入 Net.Data 所使用的環境介面函數及通信結構，進而與您的語言環境通信，並在模版及語言環境之間來回傳遞參數。

下列段落將說明函數及結構的概念及設計事項。語言環境介面中提供的公用程式說明於第41頁的『第4章 語言環境程式設計介面公用程式函數』。

- 第30頁的『我應該提供哪一種語言環境介面？』
- 第30頁的『處理輸入參數』
- 第31頁的『處理使用者要求』
- 第31頁的『處理輸出參數』

- 第31頁的『通信錯誤狀況』

我應該提供哪一種語言環境介面？

在您撰寫語言環境時，您必須決定要提供哪些介面。而您的決定必須視您想要語言環境從事哪些作業而定。例如，若語言環境要存取資料庫資料，或是要編寫語言，則兩者所使用的選項會有所不同。下面的段落將說明 Net.Data 語言環境介面。

dtw_execute()

您必須提供 dtw_execute() 介面，以傳遞來自巨集檔的輸入參數；這是每一個語言環境所必要的唯一介面。Net.Data 透過語言環境通信結構 dtw_lei，來將所有的輸入參數傳遞至 dtw_execute()。

dtw_initialize()

提供 dtw_initialize() 介面來配置或起始設定資料。在第一次對您的語言環境進行函數呼叫之前，針對每一個巨集呼叫，Net.Data 只會呼叫一次這個介面。如果沒有對您的語言環境進行任何函數呼叫，Net.Data 就不會呼叫 dtw_initialize() 介面。

dtw_cleanup()

在您提供 dtw_initialize() 介面時，也請提供 dtw_cleanup() 介面，另外，在巨集異常終止時，您會想要允許錯誤處理常式。針對每一個巨集呼叫，Net.Data 只會呼叫一次這個介面。

dtw_getNextRow()

提供 dtw_getNextRow() 介面來作為資料庫語言環境的一部份，或是作為可一次處理一列資料之語言環境的一部份。唯有 Net.Data 在 OS/400 或 OS/390 作業系統上執行時才會呼叫這個介面。

處理輸入參數

Net.Data 語言環境使用 dtw_execute() 介面來接收及處理參數。dtw_execute() 介面會使用 dtw_lei 結構，而 Net.Data 會用此結構來和語言環境通信。在撰寫您的語言環境時，請使用下列建議來處理輸入參數。

- 指定起始設定檔中任何隱含的參數。Net.Data 在將巨集作者指定的參數傳遞到所執行的 FUNCTION 區塊上後，會將這裡指定的參數傳遞至所有對語言環境的函數呼叫上。
- 接收輸入參數至 dtw_execute() 介面，以作為 dtw_lei 結構的一部份。巨集作者決定了在將參數指定在 Net.Data 巨集的 FUNCTION 區塊定義中時，Net.Data 傳遞參數的順序。

在第69頁的『附錄A. 語言環境模版』中，程式模版中的 processInputParms() 常式會顯示一個處理輸入參數的方法。

處理使用者要求

語言環境處理使用者要求的方式，需視語言環境接收該要求的方式而定。Net.Data 提供數種不同的方法，可讓您將要求傳達至語言環境：

- 透過 FUNCTION 區塊中指定的函數名稱。在每一個函數呼叫上，Net.Data 會將函數名稱以 dtw_lei 結構的 function_name 欄位傳遞至語言環境。
- 透過 FUNCTION 區塊參數列示。您可以指定讓參數列示中的參數能夠顯示使用者要求。在每一個函數呼叫上，Net.Data 會將參數以 dtw_lei 結構的 parm_data_array 欄位傳遞至語言環境。
- 透過 FUNCTION 區塊的可執行陳述式段落。在每一個函數呼叫上，Net.Data 會將指定在 FUNCTION 區塊中的任何可執行的陳述式，傳遞至 dtw_lei 結構的 exec_statement 欄位中的語言環境。

處理輸出參數

您用來處理輸出參數的方法，完全要由您的語言環境及它處理使用者要求的方法來決定。不過，一旦語言環境有了它要傳回給 Net.Data 巨集的資料，您就可以設計語言環境，以修改傳遞至 dtw_lei 結構 parm_data_array 欄位中的參數值。在第69頁的『附錄A. 語言環境模版』中，程式模版中的 processOutputParms() 常式會顯示一種處理輸出參數的可能方式，以及如何設定字串與表格參數值的範例。

通信錯誤狀況

函數呼叫的成功與失敗，可透過隱含 Net.Data 巨集參數 RETURN_CODE 來通信。這個變數會在從對 dtw_execute() 介面進行的呼叫傳回後，由 Net.Data 設定。它的值會被設定為 dtw_execute() 呼叫本身的回覆值。Net.Data 接著會使用這個值來處理 Net.Data 巨集 MESSAGE 區塊，如果曾對這個函數呼叫指定一個的話。

如果您未曾指定 MESSAGE 區塊，或在指定的 MESSAGE 區塊中，沒有用來處理 dtw_execute() 回覆碼的項目，Net.Data 將會顯示 dtw_lei 結構的 default_error_message 欄位的內容。這個欄位在 dtw_execute() 常式中任何時候都可被語言環境設定。程式模版（在第69頁的『附錄A. 語言環境模版』中）中的 setErrorMessage() 常式，會顯示如何設定 default_error_message 欄位的範例。

語言環境通信結構

Net.Data 使用二種結構來與您的語言環境通信。您的語言環境必須使用這些結構，並設定及傳遞結構中的資訊。

- dtw_lei
- dtw_parm_data

Net.Data 會傳遞一個語言環境介面結構（例如，`dtw_lei`）至它呼叫的語言環境函數。尤其是，這個結構含有一個參數資料陣列，包含了要傳遞至語言環境之函數的列示。Net.Data 所呼叫的語言環境函數會處理要求，更新參數資料陣列中的參數（如果有的話），並傳回給 Net.Data。

Net.Data 接著會處理參數資料陣列，更新它的參數副本，來反映語言環境函數所設定的新值，然後繼續處理 Net.Data 巨集。

dtw_lei 結構

每一個語言環境的介面函數會收到一個指向 `dtw_lei` 結構的指標。dtw_lei 結構的格式如下：

```
typedef struct dtw_lei {
    char *function_name;    /* 語言環境介面          */
    int  flags;             /* 函數區塊名稱          */
                                /* 語言環境介面旗號      */

    char *exec_statement;   /* 語言環境陳述式        */

    dtw_parm_data_t *parm_data_array; /* 參數陣列          */
    char *default_error_message; /* 預設訊息          */
    void *le_opaque_data; /* 語言環境專用資料      */

    void *row;              /* 用於一次一橫列處理*/

    char reserved[64];      /* 保留                */
} dtw_lei_t;
```

dtw_lei 結構中的欄位：

function_name

`function_name` 欄位含有指向包含函數區塊之名稱的字串的指標。在語言環境所顯示的錯誤訊息中要指定 `FUNCTION` 區塊名稱時，這會非常有用。

flags Net.Data 會使用旗號欄位來和語言環境通信。用下列常數來執行 OR 作業，以指定旗號欄位指標：

- Net.Data 設定 `DTW_STMT_EXEC`，以指示 `dtw_execute()` 介面函數：
`exec_statement` 欄位有包含來自 `EXEC` 陳述式的檔名及參數。
- Net.Data 會設定 `DTW_END_ABNORMAL`，以指示 `dtw_cleanup()` 介面函數：
發生了異常或非預期的狀況，而且語言環境應在 Net.Data 結束之前，執行任何必要的清除工作（意即，釋放所保留的資源）。
- 語言環境介面函數會設定 `DTW_LE_FATAL_ERROR`，以指示 Net.Data：語言環境中發生了嚴重的錯誤。如果已設定這個旗號，Net.Data 會停止處理 Net.Data 巨集，並呼叫所有目前在作用中，且旗號設定為 `DTW_END_ABNORMAL` 之語言環境的 `dtw_cleanup()` 介面函數，然後列印預設訊息，最後跳出。只有在語言環境呼叫傳回非零值時，旗號才會被檢查。

- 語言環境介面函數會設定 `DTW_LE_MSG_KEEP`，以指示 `Net.Data`：不應釋放 `default_error_message` 所指的儲存體。如果尚未設定此常數，`Net.Data` 會試圖釋放儲存體。
- `dtw_execute()` 介面函數會設定 `DTW_LE_CONTINUE`，以命令 `Net.Data` 呼叫 `dtw_getNextRow()` 介面函數。只有在已設定該旗號，且呼叫 `dtw_execute()` 介面函數的回覆值是零的狀況下，`Net.Data` 才會呼叫 `dtw_getNextRow()`。

exec_statement

`exec_statement` 欄位包含下列其中一個指標：

- 指向包含來自 `FUNCTION` 區塊之可執行陳述式（在變數替代之後）的指標
- 指向來自 `EXEC` 陳述式之檔名及參數的指標

parm_data_array

`parm_data_array` 欄位含有指向 `dtw_parm_data` 結構之陣列的指標。陣列的結尾是一個包含 0 的 `parm_data` 結構。 `dtw_parm_data` 結構供 `Net.Data` 用來傳遞參數與關聯的值給語言環境，以及取回由語言環境對變數值進行的任何變更。結構說明的相關資訊，請參閱『`dtw_parm_data` 結構』。

default_error_message

語言環境可將 `default_error_message` 欄位設定為一個描述錯誤狀況的字串。如果因呼叫語言環境介面函數而傳回的回覆值不是 0，而且該回覆值與 `MESSAGE` 區塊中的訊息值不符的話，就會顯示預設訊息。否則，`Net.Data` 會顯示從 `MESSAGE` 區塊中選取的訊息。

le_opaque_data

`le_opaque_data` 欄位是由語言環境中的任何介面函數所設定，可將參數從一個介面函數傳遞到另一個介面函數。`Net.Data` 會儲存指標，並且會將它傳遞到 `Net.Data` 呼叫的另一個介面函數。在處理 `Net.Data` 巨集之後，以及在傳回到 `Net.Data` 的呼叫者之前，`Net.Data` 會將指標定義為 `NULL`。由於欄位是依執行緒而定的，所以語言環境可以儲存特定執行緒的資料。只有在您有 `dtw_cleanup()` 介面函數的情況下，才能使用這個欄位，這樣函數才可以釋放與 `le_opaque_data` 欄位相關的儲存體。

row 在呼叫語言環境的 `dtw_getNextRow()` 介面函數之前，`Net.Data` 會將橫列欄位設定為一個橫列物件。`dtw_getNextRow()` 函數會用 `Net.Data` 橫列公用程式介面函數，在物件中插入一列表格資料橫列。然後，`Net.Data` 會處理該橫列，並持續呼叫 `dtw_getNextRow()`，直到沒有要處理的橫列為止。

保留欄位供 IBM 使用。

dtw_parm_data 結構

`Net.Data` 使用 `dtw_parm_data` 結構來傳遞參數給語言環境。參數的來源有三種：

- 指定在 `FUNCTION` 區塊定義上的明確參數
- `Net.Data` 起始設定檔中，指定在 `ENVIRONMENT` 架構陳述式上的參數

- FUNCTION 區塊定義上，指定在 RETURNS 關鍵字上的傳回變數

Net.Data 會先傳遞明確的參數，然後傳遞 ENVIRONMENT 陳述式中所指定的參數，最後是傳回變數。

dtw_parm_data 結構的格式如下：

```
typedef struct dtw_parm_data {          /* 參數資料          */
    int   parm_descriptor;              /* 參數描述子        */
    char *parm_name;                    /* 參數名稱          */
    char *parm_value;                   /* 參數值            */
    void *res1;                         /* 保留              */
    void *res2;                         /* 保留              */
} dtw_parm_data_t;
```

dtw_parm_data 結構中的欄位：

parm_descriptor

parm_descriptor 欄位說明要傳遞到語言環境之參數的類型及使用。Net.Data 使用下列常數執行 OR 運算，來設定這個欄位。

- DTW_IN 表示參數是只能輸入的參數。
- DTW_OUT 表示參數是只能輸出的參數。
- DTW_INOUT 表示參數是輸入及輸出參數。
- DTW_STRING 表示參數值是指向字串的指標。
- DTW_TABLE 表示參數值是指向表格的指標。

Net.Data 永遠會將 parm_descriptor 欄位設定為 DTW_IN、DTW_OUT 或 DTW_INOUT，並將邏輯 OR 用在 DTW_STRING 及 DTW_TABLE。

parm_name

parm_name 欄位是指向含有參數名稱之字串的指標。如果參數是文字字串，則 Net.Data 會設定這個指標 NULL。

parm_value

parm_value 欄位是指向含有參數值之物件的指標。如果參數值是尚未定義的變數，則 Net.Data 會將這個指標設定為 NULL。

res1 與 res2 是保留欄位。

parm_name 及 parm_value 兩者都指向由 Net.Data 執行堆集 所配置的物件，此堆集是 Net.Data 用來配置動態記憶體之記憶體區域。如果 parm_name 或 parm_value 被置換為其它字串，則必須釋放原始字串，並將其置換為指向由 Net.Data 堆集所配置之字串的指標。使用 dtw_malloc() 及 dtw_free() 公用程式函數來釋放原始字串。

語言環境介面函數

Net.Data 將四種介面函數用在一種語言環境上：您可提供這些函數的其中一個或數個函數。其中有三種函數是選用性的，但是每一個語言環境都必須要有 `dtw_execute()` 介面函數。如果 Net.Data 巨集參照沒有 `dtw_execute()` 介面函數的語言環境，Net.Data 將傳回錯誤訊息同時停止處理 Net.Data 巨集。

欲呼叫語言環境，您可在 Net.Data 巨集的 FUNCTION 區塊中參照它。您必須以下列次序來呼叫語言環境介面函數：

1. `dtw_initialize()`
2. `dtw_execute()`
3. `dtw_getNextRow()`
4. `dtw_cleanup()`

`dtw_execute()` 是唯一您必須提供在語言環境中的函數。

當 Net.Data 呼叫一個使用語言環境的函數時，它會用下列步驟來呼叫該語言環境：

1. 如果 Net.Data 已定義給這個語言環境，則 Net.Data 會呼叫 `dtw_initialize()`。此函數會執行語言環境所要求的任何起始設定作業，例如連接至資料庫，或是配置變數。
2. Net.Data 會呼叫 `dtw_execute()`，以處理包含必須處理語言環境之陳述式的巨集檔 FUNCTION 區塊。
3. 在成功的回覆中，如果 `dtw_execute()` 指示應呼叫 `dtw_getNextRow()`，Net.Data 就會呼叫 `dtw_getNextRow()`。
4. 當 Net.Data 巨集處理程序完成時，如果已將 `dtw_cleanup()` 定義在語言環境，而且之後有傳回 Web 伺服器的話，Net.Data 就會呼叫 `dtw_cleanup()`，以清除環境 (例如，切斷與資料庫的連線，或釋放變數)。

下列各節將為您說明介面函數：

- 『`dtw_initialize()`』
- 第36頁的 『`dtw_execute()`』
- 第36頁的 『`dtw_getNextRow()`』
- 第37頁的 『`dtw_cleanup()`』

dtw_initialize()

`dtw_initialize()` 介面函數會執行語言環境所要求的任何特殊起始設定，例如連接資料庫或配置變數。這個介面函數會被呼叫一次，而且是選用性的。

Net.Data 只會呼叫一次語言環境的 `dtw_initialize()` 介面函數，這次呼叫也是 Net.Data 第一次呼叫參照該語言環境的 FUNCTION 區塊。後續的語言環境參照會略過 `dtw_initialize()` 介面函數的呼叫。

這個介面函數不會影響訊息區塊處理。正值或零的回覆碼表示繼續處理；負值回覆碼表示不繼續處理。如果回覆碼為非零的值，而且 `default_error_message` 欄位中定義有預設的訊息，則會發出預設訊息；如果沒有預設訊息，Net.Data 將發出錯誤訊息。

dtw_execute()

`dtw_execute()` 介面函數會處理含有必須由語言環境處理之陳述式的巨集檔 FUNCTION 區塊。例如，參照資料庫語言環境的 FUNCTION 區塊，會包含語言環境用來查詢資料庫的 SQL 陳述式。

每當 Net.Data 巨集處理一個參照語言環境的 FUNCTION 區塊時，都會呼叫 `dtw_execute()` 介面函數。當 `dtw_execute()` 介面函數完成時，接下來會發生什麼事，需視語言環境是否有一次處理一系列資料而定。如果是的話，介面函數會在 `dtw_lei` 結構中設定 DTW_LE_CONTINUE 旗號，以命令 Net.Data 呼叫 `dtw_getNextRow()` 介面函數。`dtw_getNextRow()` 介面函數及其處理步驟的相關資訊，請參閱『`dtw_getNextRow()`』。

您可以令 `dtw_execute()` 介面函數執行所有產生報表區塊程序之輸入所需的處理程序，以使執行效能最佳化。例如，SQL 語言環境的 `dtw_execute` 介面函數會產生在報表區塊階段所要處理的整個表格。

dtw_getNextRow()

`dtw_getNextRow()` 介面函數會取回輸入，以進行一次一系列的 Net.Data 表格處理。每當您設定 DTW_LE_CONTINUE 旗號，表示需要為該表格處理其他列的資料時，就會呼叫這個介面函數。為資料庫語言環境使用 `dtw_getNextRow()`。

限制：只有 Net.Data 在 OS/400 或 OS/390 作業系統上執行的情況下，才會呼叫這個介面函數。

發生下列狀況時，Net.Data 會呼叫 `dtw_getNextRow()`：

- 呼叫語言環境的 `dtw_execute()` 呼叫順利完成 (傳回 0 值)
- `dtw_execute()` 介面函數已在 `dtw_lei` 結構中設定 DTW_LE_CONTINUE 旗號。

當 `dtw_execute()` 函數將 DTW_LE_CONTINUE 旗號設定為 on 時，Net.Data 會執行下列步驟：

1. 為 `dtw_execute()` 介面函數的回覆值處理訊息區塊。
2. 呼叫語言環境的 `dtw_getNextRow()` 介面函數，並開始一次一系列處理程序。
3. 處理報告區塊

4. 為 `dtw_getNextRow()` 介面函數的回覆值處理訊息區塊。
5. 判定 `dtw_getNextRow()` 是否有啟動 `DTW_LE_CONTINUE` 旗號：
 - 如果有，則在步驟 2 繼續處理 `dtw_getNextRow()` 介面函數。
 - 如果沒有，則一次一系列的處理程序結束，而 `Net.Data` 會繼續處理 `Net.Data` 巨集。

呼叫 `dtw_getNextRow()` 時，會將 `dtw_lei` 結構中的橫列欄位設定為指向橫列物件。欲處理橫列物件，請使用 `Net.Data` 公用程式函數 `dtw_row_SetCols()` 及 `dtw_row_SetV()`。 `Net.Data` 會假設在第一次呼叫 `dtw_getNextRow()` 介面函數後，橫列物件會包含該表格的直欄標題。後續的呼叫包含實際的表格資料。

只要有設定 `DTW_LE_CONTINUE` 旗號，就會繼續呼叫 `dtw_getNextRow()` 函數（除非訊息區塊處理有另外指示）。

dtw_cleanup()

如果您使用 `dtw_initialize()` 來起始設定語言環境，則使用 `dtw_cleanup()` 介面函數來清除語言環境。例如，切斷與資料庫的連線，或是釋放變數。這個介面函數是選用性的。

在處理 `Net.Data` 要求時，一旦 `Net.Data` 處理程序結束，或是發生錯誤，而使 `Net.Data` 停止處理巨集檔時，`Net.Data` 會呼叫語言環境的 `dtw_cleanup()` 介面函數。

如果清除處理發生異常的話，`Net.Data` 會將 `dtw_lei` 結構中的旗號欄位設定為 `DTW_END_ABNORMAL`。下列異常狀況提供您於何時使用 `dtw_cleanup()` 的範例：

- 語言環境介面函數利用設定 `dtw_lei` 結構中的旗號欄位的 `DTW_LE_FATAL_ERROR` 位元，來指示發生嚴重錯誤。
- `Net.Data` 發現無法回復的錯誤。
- `Net.Data` 巨集訊息區塊處理造成一個跳出。

如果語言環境的介面函數使用要在介面函數之間傳遞的參數來設定 `le_opaque_data` 欄位，請在處理程序結束時，用 `dtw_cleanup()` 來釋放欄位。

這個介面函數不會影響訊息區塊處理。如果回覆值不是 0，就會發出預設的訊息；如果沒有預設訊息的存在，巨集處理器就會發出警告訊息。

設計語言環境陳述式

每個語言環境在起始設定檔 `DB2WWW.INI` 中都有一個 `ENVIRONMENT` 陳述式，這個起始設定檔中含有該語言環境專用的資訊。當您建立新的語言環境時，您必須針對起始設定檔設計一個環境陳述式，並以文件說明使用者應如何將該陳述式加入起始設定檔。

ENVIRONMENT 陳述式指定語言環境的相關資訊，Net.Data 需要這些資訊來呼叫及載入語言環境 DLL 或 共用程式庫，例如要為每一個函數呼叫傳遞到語言環境的語言環境名稱、DLL 或共用程式庫名稱及參數列示。

在呼叫 Net.Data 時，Net.Data 會讀取架構資訊，但是在 FUNCTION 區塊識別出語言環境是從巨集檔內呼叫的之前，不會載入語言環境 DLL 或共用程式庫。DLL 會保持載入的狀態，直到 Net.Data 結束為止。

下列區段提供您可用在您的文件中的語法、參數說明及範例相關資訊。

ENVIRONMENT 陳述式語法

ENVIRONMENT 陳述式的格式如下：

```
ENVIRONMENT(type) library-name ([usage  
parameter, ...])
```

每一個 ENVIRONMENT 陳述式必須由一行構成。

下列為您必須指定給每一個語言環境的參數：

- *type*

使這個語言環境與 Net.Data 巨集中的 FUNCTION 區塊定義產生關聯的名稱。您也必須在 FUNCTION 區塊定義上指定語言環境類型，以告訴 Net.Data 要用哪一種語言環境來處理函數呼叫。該名稱不可用字首 DTW_ 來開頭。這個字首已被 Net.Data 附隨的語言環境預定使用。FUNCTION 區塊的相關資訊，請參閱 *Net.Data* 參考手冊 中的「函數區塊」部份。

- *library_name*

物件的名稱，這個物件含有由 Net.Data 呼叫的語言環境介面。在 Windows NT 及 OS/2 中，指定 DLL 名稱時，不需附加 *.dll* 副檔名。在 AIX 中，指定共用物件的名稱時，要加上 *.o* 副檔名，而在 OS/400 中，指定服務程式名稱時，要加上 *.SRVPGM* 副檔名。OS/390 的 DLL 檔沒有附檔名。請查看您作業系統之 Net.Data 附隨的起始設定檔，來取得如何指定這個名稱的資訊。請考慮使用完整的路徑名稱，來確定 Net.Data 可以找到 DLL 或共用程式庫。

- *parameter_list*

這個列示中的參數及 FUNCTION 區塊定義中指定的參數，會在每次函數呼叫時，傳遞給語言環境。它們會被傳遞到 *dtw_lei* 結構的 *parm_data_array* 欄位中，並接在 FUNCTION 區塊定義中所指定的參數之後。您必須先在 Net.Data 巨集中將這些參數定義為變數，之後才可以進行函數呼叫。如果函數修改了這些參數的值，在函數完成處理程序後，參數會保留已修改的值。

ENVIRONMENT 陳述式範例

下列範例顯示 Net.Data 提供之語言環境所使用的 ENVIRONMENT 陳述式。這些範例會以圖例說明如何指定參數。您放入 ENVIRONMENT 陳述式的變數，就是您要讓 Net.Data 巨集作者在他們巨集中定義或置換的變數。欲查看其他範例，請依您所使用的作業系統來參閱 *Net.Data* 參考手冊或您的 Net.Data README 檔案或「程式目錄」中的附錄。

下列範例顯示 OS/2、AIX 及 Windows NT 所使用的語法。

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

每一種作業系統上的 ENVIRONMENT 陳述式都有所不同；例如 OS/390 的 SQL 與 ODBC 存取稍有不同：

```
ENVIRONMENT (DTW_SQL)      DTWSQL      ( IN LOCATION, DB2SSID, DB2PLAN,  
TRANSACTION_SCOPE)  
  
ENVIRONMENT (DTW_ODBC)     DTWODBC     ( IN LOCATION, TRANSACTION_SCOPE)
```

第4章 語言環境程式設計介面公用程式函數

Net.Data 提供您一個程式設計介面，可在設計新的語言環境時使用。語言環境介面有一些公用程式函數，可存取 Net.Data 服務程式，以管理記憶體及架構變數，並提供表格及橫列操作特性。第69頁的『附錄A. 語言環境模版』提供一個模版，當您在設計您的語言環境時，可將此模版當作模型。

下列區段將解釋 Net.Data 語言環境介面公用程式函數。

語言環境公用程式函數

語言環境使用公用程式函數來存取 Net.Data 服務程式。而這些函數共分為四種範疇：

- 『用來管理記憶體的公用程式函數』
- 『管理架構變數用的公用程式函數』
- 第42頁的『表格操作的公用程式函數』
- 第43頁的『橫列操作的公用程式函數』

用來管理記憶體的公用程式函數

語言環境使用記憶體管理公用程式來配置 Net.Data 所擁有的儲存體，並且用 Net.Data 執行程式庫將它所配置的儲存體釋放出來。

下列範例說明了這些公用程式函數的需要。假設 Net.Data 是使用編譯器 A，並配合其對應的執行程式庫撰寫而成。但是程式設計師若使用編譯器 B 來撰寫新的語言環境，就會有不同的執行程式庫。由於二個執行程式庫之間所隱含的不相容性質，該語言環境無法釋放 Net.Data 所配置的儲存體，而且 Net.Data 也無法釋放該語言環境所配置的儲存體。

表 5. 記憶體公用程式函數

公用程式函數	說明
第46頁的『dtw_malloc()』	配置用 dtw_malloc() 取自 Net.Data 執行堆集的儲存體。
第44頁的『dtw_free()』	釋放用 dtw_malloc() 配置自 Net.Data 執行堆集而配置的儲存體。
第49頁的『dtw_strdup()』	配置取自 Net.Data 執行堆集的儲存體，並將所指定的字串複製到用 dtw_malloc() 配置到的儲存體。

管理架構變數用的公用程式函數

架構變數的管理公用程式函數可讓語言環境存取儲存在 Net.Data 起始設定檔中的架構資訊。若使用這些函數，所有語言環境就可以共用 Net.Data 起始設定檔，並可使用當中的資訊，以架構語言環境。

表 6. 架構公用程式函數

公用程式函數	說明
第45頁的『dtw_getvar()』	取回 Net.Data 起始設定檔中架構變數的值。

表格操作的公用程式函數

使用表格函數來操控任何已傳遞給語言環境的 Net.Data 巨集表格變數。

橫列及直欄號碼由一 (1) 開始。

表 7. 表格公用程式函數

公用程式函數	說明
第60頁的『dtw_table_New()』	建立一個表格物件
第52頁的『dtw_table_Delete()』	刪除一個表格物件。
第63頁的『dtw_table_SetCols()』	設定表格的寬度，並配置直欄表頭的儲存體。
第56頁的『dtw_table_GetV()』	取回表格數值。
第65頁的『dtw_table_SetV()』	設定表格數值。
第55頁的『dtw_table_GetN()』	取回一個表格直欄標題
第64頁的『dtw_table_SetN()』	設定表格直欄標題。
第62頁的『dtw_table_Rows()』	取回表格中的現行列數。
第51頁的『dtw_table_Cols()』	取回表格中的現行直欄號碼。
第59頁的『dtw_table_MaxRows()』	取回表格中所允許之橫列的最大數目。
第61頁的『dtw_table_QueryColnoNj()』	取回直欄的直欄數。
第50頁的『dtw_table_AppendRow()』	在表格尾端新增一列或多列。
第58頁的『dtw_table_InsertRow()』	在表格中插入一列或多列。
第54頁的『dtw_table_DeleteRow()』	從表格中刪除一列或多列。
第57頁的『dtw_table_InsertCol()』	在表格中插入一欄或多欄。
第53頁的『dtw_table_DeleteCol()』	從表格中刪除一欄或多欄。

橫列操作的公用程式函數

在一次一系列的處理程序期間，橫列公用程式函數會操作已傳遞至語言環境之 `dtw_getNextRow()` 介面函數的橫列物件。

橫列號碼由一 (1) 開始。

表 8. 橫列公用程式函數

公用程式函數	說明
第47頁的 『 <code>dtw_row_SetCols()</code> 』	設定橫列的寬度。
第48頁的『 <code>dtw_row_SetV()</code> 』	設定表格數值。

公用程式函數語法參考手冊

本段說明每一個公用程式函數、其格式、用法和參數，並且也提供簡單的範例。

dtw_getvar()

使用

從 `Net.Data` 起始設定檔中，取回 *var_name* 所指定的架構變數的值。 `Net.Data` 擁有 `dtw_getvar()` 所傳回的記憶體；請勿將其修改或釋放。

語法

```
char *dtw_getvar(char *var_name)
```

參數

var_name 將取回的架構變數的名稱。

範例

```
char *myBindFile;  
  
myBindFile = dtw_getvar("BIND_FILE");
```

dtw_malloc()

使用

傳回指向儲存體的指標，該儲存體是使用 `dtw_malloc()` 來從 `Net.Data` 執行堆集配置的儲存體。儲存體的度為 `nbytes`。如果 `Net.Data` 無法傳回所要求的儲存體，則會傳回 `NULL` 指標。

語法

```
void *dtw_malloc(long nbytes)
```

參數

nbytes 將配置的位元組數目。

範例

```
char *myBuf;
long  nbytes = 8192;

myBuf = (char *)dtw_malloc(nbytes);
```

dtw_row_SetCols()

使用

指定橫列的寬度，並為欄位標題配置儲存體。每個橫列可以使用一次 `dtw_row_SetCols()` 公用程式函數。

語法

```
int dtw_row_SetCols(void *row, int cols)
```

參數

<i>row</i>	指向尚未配置任何直欄的剛建立橫列的指標。
------------	----------------------

<i>cols</i>	將在新的橫列中配置的起始直欄數目。
-------------	-------------------

範例

```
void *myRow;  
  
rc = dtw_row_SetCols(myRow, 5);
```

dtw_row_SetV()

使用

指定表格數值。dtw_row_SetV() 公用程式函數的呼叫者會保留 *src* 所指之記憶體的所有權。
若要刪除目前的表格數值，請將數值指定為 NULL。

語法

```
int dtw_row_SetV(void *row, char *src, int col)
```

參數

<i>row</i>	將修改的橫列的指標。
<i>src</i>	含有將設定的新值的字串。
<i>col</i>	將設定的值的直欄號碼。

範例

```
void *myTable;  
char *myFieldValue = "newValue";  
  
rc = dtw_row_SetV(myRow, myFieldValue, 3);
```

dtw_strdup()

使用

配置來自 `Net.Data` 的執行堆集的儲存體，並將 *string* 所指定的字串複製到以 `dtw_malloc()` 所配置的儲存體。如果 `Net.Data` 無法傳回所要求的儲存體，則會傳回 `NULL` 指標。

語法

```
char *dtw_strdup(char *string)
```

參數

string 將複製到已配置儲存體中的字串值的指標。

範例

```
char *myString = "This string will be duplicated.";
char *myDupString;

myDupString = dtw_strdup(myString);
```

dtw_table_AppendRow()

使用

在表格尾端新增一或多行橫列。把橫列附加至表格後，用 `dtw_table_SetV()` 公用程式來指定新橫列的表格數值。

語法

```
int dtw_table_AppendRow(void *table, int rows)
```

參數

<i>table</i>	指向將被附加橫列的表格的指標。
--------------	-----------------

<i>rows</i>	將附加的橫列數目。
-------------	-----------

範例

```
void *myTable;
```

```
rc = dtw_table_AppendRow(myTable, 10);
```

dtw_table_Cols()

使用

傳回表格中現行直欄數目。

語法

```
int dtw_table_Cols(void *table)
```

參數

table 指向將傳回其現行直欄數目的表格的指標。

範例

```
void *myTable;  
int currentColumns;  
  
currentColumns = dtw_table_Cols(myTable);
```

dtw_table_Delete()

使用

刪除所有欄位標題、欄位值及表格物件。

語法

```
int dtw_table_Delete(void *table)
```

參數

table 將刪除的表格的指標。

範例

```
void *myTable;  
rc = dtw_table_Delete(myTable);
```


dtw_table_DeleteCol()

使用

刪除從 *start_col* 中所指定之欄位開始的一或多欄。欲刪除表格的所有橫列及欄位，請用公用程式函數 `dtw_table_Cols()` 來代替 *cols* 參數。

```
dtw_table_DeleteCol(table, 1, dtw_table_Cols());
```

語法

```
int dtw_table_DeleteCol(void *table, int start_col, int cols)
```

參數

<i>table</i>	指向將修改的表格的指標。
<i>start_col</i>	將刪除的第一個欄的直欄號碼。
<i>rows</i>	將刪除的列數。

範例

```
void *myTable;  
  
rc = dtw_table_DeleteCol(myTable, 1, 10);
```

dtw_table_DeleteRow()

使用

刪除從 *start_row* 中所指定之橫列開始的一列或多列。

語法

```
int dtw_table_DeleteRow(void *table, int start_row, int rows)
```

參數

<i>table</i>	指向將修改的表格的指標。
<i>start_row</i>	將刪除的第一列的橫列號碼。
<i>rows</i>	將刪除的橫列數目。

範例

```
void *myTable;  
  
rc = dtw_table_DeleteRow(myTable, 3, 10);
```

dtw_table_GetN()

使用

取回直欄標題。Net.Data 擁有由 *dest* 所指的記憶體；請勿將其修改或釋放。

語法

```
int dtw_table_GetN(void *table, char **dest, int col)
```

參數

<i>table</i>	指向將從其中取回直欄標題的表格的指標。
--------------	---------------------

<i>dest</i>	指向含有直欄標題的字串的指標。
-------------	-----------------

<i>col</i>	直欄標題的直欄號碼。
------------	------------

範例

```
void *myTable;  
char *myColumnHeading;  
  
rc = dtw_table_GetN(myTable, &myColumnHeading, 5);
```

dtw_table_GetV()

使用

從表格取回數值。Net.Data 擁有由 *dest* 所指的記憶體；請勿將其修改或釋放。

語法

```
int dtw_table_GetV(void *table, char **dest, int row, int col)
```

參數

<i>table</i>	指向將從其中取回一個值的表格的指標。
<i>dest</i>	指向將會包含該數值之字串的指標。
<i>row</i>	將取回的值的橫列號碼。
<i>col</i>	將取回的值的直欄號碼。

範例

```
void *myTable;  
char *myTableValue;  
  
rc = dtw_table_GetV(myTable, &myTableValue, 3, 5);
```

dtw_table_InsertCol()

使用

在所指定的直欄後面，插入一或多個直欄。

語法

```
int dtw_table_InsertCol(void *table, int after_col, int cols)
```

參數

<i>table</i>	指向將修改的表格的指標。
<i>after_col</i>	將在其後插入新直欄的直欄數目。欲在表格起始處插入直欄，請指定 0。
<i>cols</i>	將插入的直欄數目。

範例

```
void *myTable;  
  
rc = dtw_table_InsertCol(myTable, 3, 10);
```

dtw_table_InsertRow()

使用

在指定的橫列後，插入一系列或多列。

語法

```
int dtw_table_InsertRow(void *table, int after_row, int rows)
```

參數

<i>table</i>	指向將修改的表格的指標。
<i>after_row</i>	將在其後插入新橫列的橫列的號碼。欲在表格起始處插入橫列，請指定 0。
<i>rows</i>	將插入的橫列數目。

範例

```
void *myTable;  
  
rc = dtw_table_InsertRow(myTable, 3, 10);
```

dtw_table_MaxRows()

使用

傳回允許給 `Net.Data` 表格的最大列數，如 `dtw_table_New()` 公用程式函數的參數所定義之 *row_lim*。

語法

```
int dtw_table_MaxRows(void *table)
```

參數

table 指向傳回橫列最大數之表格的指標。

範例

```
void *myTable;  
int maximumRows;  
  
maximumRows = dtw_table_MaxRows(myTable);
```

dtw_table_New()

使用

建立 `Net.Data` 表格物件，並將所有直欄標題及欄位值起始設定為 `NULL`。呼叫者將指定橫列及直欄的起始數目，以及橫列的最大數目。如果橫列或欄位的起始數為 `0`，則在呼叫任何表格函數之前，您必須用 `dtw_table_SetCols()` 函數來指定橫列中的欄位數目。

語法

```
int dtw_table_New(void **table, int rows, int cols, int row_lim)
```

參數

<i>table</i>	新表格的名稱。
<i>rows</i>	將在新表格中配置的起始橫列數目。
<i>cols</i>	將在新表格中配置的起始直欄數目。
<i>row_lim</i>	此表格所能包含的最大列數。

範例

```
void *myTable;  
  
rc = dtw_table_New(&myTable, 20, 5, 100);
```


dtw_table_QueryColnoNj()

使用

傳回與欄位標題連結的直欄號碼。

語法

```
int dtw_table_QueryColnoNj(void *table, char *name)
```

參數

<i>table</i>	指向將查詢的表格的指標。
<i>name</i>	指定將傳回其直欄號碼的直欄標題的字串。如果表格中沒有直欄標題，將傳回 0。

範例

```
void *myTable;  
int columnNumber;  
  
columnNumber = dtw_table_QueryColnoNj(myTable, "column 1");
```

dtw_table_Rows()

使用

傳回表格中現行橫列數目。

語法

```
int dtw_table_Rows(void *table)
```

參數

table 將傳回其現行橫列數目的表格的指標。

範例

```
void *myTable;  
int currentRows;  
  
currentRows = dtw_table_Rows(myTable);
```

dtw_table_SetCols()

使用

設定表格的直欄數目，並配置儲存體，以供直欄標題使用。在建立好表格時，請指定欄位標題；否則，在使用其他表格函數之前，您必須先呼叫這個公用程式函數來指定欄位標題。一個表格只能使用一次 `dtw_table_SetCols()` 公用程式函數。之後請使用 `dtw_table_DeleteCol()` 或 `dtw_table_InsertCol()` 公用程式函數。

語法

```
int dtw_table_SetCols(void *table, int cols)
```

參數

<i>table</i>	指向沒有配置任何直欄或橫列的新表格的指標。
<i>cols</i>	將在新表格中配置的起始直欄數目。

範例

```
void *myTable;  
  
rc = dtw_table_SetCols(myTable, 5);
```

dtw_table_SetN()

使用

指定名稱給欄位標題。dtw_table_SetN() 公用程式函數的呼叫者會保留 *src* 參數所指之記憶體的所有權。欲刪除欄位標題，請將欄位標題值指定為 NULL。

語法

```
int dtw_table_SetN(void *table, char *src, int col)
```

參數

<i>table</i>	指向已指定欄位標題之表格的指標。
<i>src</i>	指定給新直欄位標題的字串。
<i>col</i>	直欄個數。

範例

```
void *myTable;  
char *myColumnHeading = "newColumnHeading";  
  
rc = dtw_table_SetN(myTable, myColumnHeading, 5);
```

dtw_table_SetV()

使用

在表格中指定數值。dtw_table_SetV() 公用程式函數的呼叫者會保留 *src* 參數所指之記憶體的所有權。若要刪除表格數值，請將數值指定為 NULL。

語法

```
int dtw_table_SetV(void *table, char *src, int row, int col)
```

參數

<i>table</i>	指向將要指定數值之表格的指標。
<i>src</i>	指定給新數值的字串。
<i>row</i>	新值的橫列號碼。
<i>col</i>	新值的直欄號碼。

範例

```
void *myTable;  
char *myTableValue = "newValue";  
  
rc = dtw_table_SetV(myTable, myTableValue, 3, 5);
```

第3篇 附錄與後記

附錄A. 語言環境模版

您可以使用這個模版，來建立自己的語言環境。

```

/*****
/*
/* File Name
/*
/* Description
/*
/* Functions
/*
/* Entry Points
/*
/* Change Activity
/*
/* Flag Reason Date Developer Description
/* -----
/*
/*
*****/

/*-----*/
/* Includes
/*-----*/
#include "dtwle.h"
```

圖 2. 語言環境模版 (1/29)

```

#ifdef __MVS__
#pragma export(dtw_initialize)
#pragma export(dtw_execute)
#pragma export(dtw_getNextRow)
#pragma export(dtw_cleanup)
#endif

#ifdef _AIX_
/*-----*/
/* Function */
/* dtw_getFp */
/* Purpose */
/* Set function pointers to all Language Environment Interface */
/* routines being provided by this Language Environment. If a */
/* routine in the structure is not being provided, set that field */
/* to NULL. */
/* Format */
/* int dtw_getFp(dtw_fp_t *func_pointer) */
/* Parameters */
/* func_pointer A pointer to a structure which will contain */
/* function pointers for all functions provided */
/* by this language environment. */
/* Returns */
/* Success ..... 0 */
/* Failure ..... -1 */
/*-----*/
int dtw_getFp(dtw_fp_t *func_pointer)
{
    func_pointer->dtw_initialize_fp = dtw_initialize;
    func_pointer->dtw_execute_fp = dtw_execute;
    func_pointer->dtw_getNextRow_fp = dtw_getNextRow;
    func_pointer->dtw_cleanup_fp = dtw_cleanup;
    return 0;
}
#endif

```

圖 2. 語言環境模版 (2/29)

```

/*-----*/
/*
/* Function
/*   dtw_initialize
/*
/* Purpose
/*
/* Format
/*   int dtw_initialize(dtw_lei_t *le_interface)
/*
/* Parameters
/*   le_interface      A pointer to a structure containing the
/*                     following fields:
/*
/*       function_name
/*       flags
/*       exec_statement
/*       parm_data_array
/*       default_error_message
/*       le_opaque_data
/*       row
/*
/* Returns
/*   Success ..... 0
/*   Failure ..... 0
/*-----*/
int dtw_initialize(dtw_lei_t *le_interface)
{
    return rc;
}

```

圖 2. 語言環境模版 (3/29)

```

/*-----*/
/*
/* Function
/*   dtw_execute
/*
/* Purpose
/*
/* Format
/*   int dtw_execute(dtw_lei_t *le_interface)
/*
/* Parameters
/*   le_interface      A pointer to a structure containing the
/*                     following fields:
/*
/*       function_name
/*       flags
/*       exec_statement
/*       parm_data_array
/*       default_error_message
/*       le_opaque_data
/*       row
/*
/* Returns
/*   Success ..... 0
/*   Failure ..... 0
/*-----*/
int dtw_execute(dtw_lei_t *le_interface)
{
    /*-----*/
    /* Determine if %exec statement was specified.
    /*-----*/
    if (le_interface->flags & DTW_STMT_EXEC) {
        /*-----*/
        /* Parse the %exec statement
        /*-----*/
        rc = processExecStmt(le_interface->exec_statement);
        if (rc)
        {
        }
    }
    else {
        /*-----*/
        /* Parse the inline data
        /*-----*/
        rc = processInlineData(le_interface->exec_statement);
        if (rc)
        {
        }
    }
}

```

圖 2. 語言環境模版 (4/29)

```

/*-----*/
/* Parse the input parameters */
/*-----*/
rc = processInputParms(le_interface->parm_data_array);
if (rc)
{
}
/*-----*/
/* Process the request */
/*-----*/
rc = processRequest();
if (rc)
{
}
/*-----*/
/* Process the output data */
/*-----*/
rc = processOutputParms(le_interface->parm_data_array);
if (rc)
{
}
/*-----*/
/* Process the return code and default error message */
/*-----*/
if (rc)
{
    setErrorMessage(rc, &(le_interface->default_error_message));
}
/*-----*/
/* Cleanup and exit program. */
/*-----*/
return rc;
}

```

圖 2. 語言環境模版 (5/29)

```

/*-----*/
/*
/* Function
/*   dtw_getNextRow
/*
/* Purpose
/*
/* Format
/*   int dtw_getNextRow(dtw_lei_t *le_interface)
/*
/* Parameters
/*   le_interface      A pointer to a structure containing the
/*                     following fields:
/*
/*       function_name
/*       flags
/*       exec_statement
/*       parm_data_array
/*       default_error_message
/*       le_opaque_data
/*       row
/*
/* Returns
/*   Success ..... 0
/*   Failure ..... 0
/*-----*/
int dtw_getNextRow(dtw_lei_t *le_interface)
{
    return rc;
}

```

圖 2. 語言環境模版 (6/29)

```

/*-----*/
/*
/* Function                                     */
/*   dtw_cleanup                               */
/*
/* Purpose                                     */
/*
/* Format                                     */
/*   int dtw_cleanup(dtw_lei_t *le_interface) */
/*
/* Parameters                                 */
/*   le_interface      A pointer to a structure containing the
/*                     following fields:
/*
/*       function_name
/*       flags
/*       exec_statement
/*       parm_data_array
/*       default_error_message
/*       le_opaque_data
/*       row
/*
/* Returns
/*   Success ..... 0
/*   Failure ..... 0
/*-----*/
int dtw_cleanup(dtw_lei_t *le_interface)
{
    /*-----*/
    /* Determine if this is normal or abnormal termination. */
    /*-----*/
    if (le_interface->flags & DTW_END_ABNORMAL) {
        /*-----*/
        /* Do abnormal termination cleanup.                */
        /*-----*/
    }
    else {
        /*-----*/
        /* Do normal termination cleanup.                    */
        /*-----*/
    }

    return rc;
}

```

圖 2. 語言環境模版 (7/29)

```

/*-----*/
/*
/* Function
/*   processInputParms
/*
/* Purpose
/*
/* Format
/*   unsigned long processInputParms(dtw_parm_data_t *parm_data)
/*
/* Parameters
/*   dtw_parm_data_t *parm_data
/*
/* Returns
/*   Success ..... 0
/*   Failure .....
/*
/*-----*/
unsigned long processInputParms(dtw_parm_data_t *parm_data)
{
    /*-----*/
    /* Loop through all the variables in the parameter data array. */
    /* The array is terminated by a NULL entry, meaning the parm_name */
    /* field is set to NULL, the parm_value field is set to NULL, and */
    /* the parm_descriptor field is set to 0. However, the only valid */
    /* check for the end of the parameter data array is to check */
    /* parm_descriptor == 0, since the parm_name field is NULL when a */
    /* literal string is passed in, and the parm_value field is set */
    /* to NULL when an undeclared variable is passed in. */
    /*-----*/
    for (; parm_data->parm_descriptor != 0; ++parm_data) {

```

圖 2. 語言環境模版 (8/29)

```

/*-----*/
/* Determine the usage of each input parameter.
/*-----*/
switch(parm_data->parm_descriptor & DTW_USAGE) {

    case(DTW_IN):
        /*-----*/
        /* Determine the type of each input parameter.
        /*-----*/
        switch (parm_data->parm_descriptor & DTW_TYPE) {
            case DTW_STRING:
                break;

```

圖 2. 語言環境模版 (9/29)


```
        case DTW_TABLE:
            break;
```

圖 2. 語言環境模版 (10/29)

```
        default:
            /*-----*/
            /* Internal error - unknown data type */
            /*-----*/
            break;
```

圖 2. 語言環境模版 (11/29)

```
        }
        break;
```

圖 2. 語言環境模版 (12/29)

```
        case(DTW_OUT):
            break;
```

圖 2. 語言環境模版 (13/29)

```
        case(DTW_INOUT):
            break;
```

圖 2. 語言環境模版 (14/29)

```

        default:
            /*-----*/
            /* Internal error - unknown usage */
            /*-----*/
        break;

```

圖 2. 語言環境模版 (15/29)

```

    }
}
return rc;
}

```

圖 2. 語言環境模版 (16/29)

```

/*-----*/
/*
/* Function */
/* processOutputParms() */
/*
/* Purpose */
/*
/* Format */
/* unsigned long processOutputParms(dtw_parm_data_t *parm_data) */
/*
/* Parameters */
/* dtw_parm_data_t *parm_data */
/*
/* Returns */
/* Success ..... 0 */
/* Failure ..... -1 */
/*
/*-----*/
unsigned long processOutputParms(dtw_parm_data_t *parm_data) {
    /*-----*/
    /* Get output data in some language environment-specific manner. */
    /* This is entirely dependent on what the language environment */
    /* is interfacing to, and how the LE chooses to interface to it. */
    /*-----*/
}

```

圖 2. 語言環境模版 (17/29)

```

/  /*-----*/
/* Loop through all the parms in the parameter data array, */
/* looking for output parameters. */
/*-----*/
for (; parm_data->parm_descriptor != 0; ++parm_data) {

    /*-----*/
    /* Determine usage of each parameter. */
    /*-----*/
    if (pd_i->parm_descriptor & DTW_OUT) {
        /*-----*/
        /* Determine the type of each input parameter. */
        /*-----*/
        switch (pd_i->parm_descriptor & DTW_TYPE) {
            case DTW_STRING:
                /*-----*/
                /* Give a string parameter a new value. If the */
                /* parameter value is not currently NULL, the */
                /* storage must be freed using an LE interface */
                /* utility function if it was allocated by */
                /* Net.Data. */
                /*-----*/
                if (parm_data->parm_value != NULL)
                    dtw_free(parm_data->parm_value);
                parm_data->parm_value = dtw_strdup(newValue);
            break;

```

圖 2. 語言環境模版 (18/29)

```

            case DTW_TABLE:
                /*-----*/
                /* Change the size of a table parameter. Use the */
                /* LE interface utility functions to modify the */
                /* table object. */
                /*-----*/
                /*-----*/
                /* First get the pointer to the table object. */
                /*-----*/
                void *myTable = (void *) parm_data->parm_value;

```

圖 2. 語言環境模版 (19/29)

```

/*-----*/
/* Next get the current size of the table.      */
/*-----*/
cols = dtw_table_Cols(myTable);
rows = dtw_table_Rows(myTable);
/*-----*/
/* Now set the new size (assumes the new size   */
/* values are valid).                           */
/*-----*/

/*-----*/
/* Set the columns first.                       */
/*-----*/
if (cols > newColValue)
{
    dtw_table_DeleteCol(myTable,
                        newColValue + 1,
                        cols - newColValue);
}
else if (cols < new_col_value)
{
    dtw_table_InsertCol(myTable,
                        cols,
                        newColValue - cols);
}

/*-----*/
/* Now set the rows.                           */
/*-----*/
if (newColValue > 0) {
    if (rows > newRowValue)
    {
        dtw_table_DeleteRow(myTable,
                            newRowValue + 1,
                            rows - newRowValue);
    }
    else if (rows < new_row_value)
    {
        dtw_table_InsertRow(myTable,
                            rows,
                            newRowValue - rows);
    }
}
}

```

圖 2. 語言環境模版 (20/29)

```

/*-----*/
/* Now get the last row/column value.          */
/*-----*/
dtw_table_GetV(myTable,
               &myValue;,
               newRowValue,
               newColValue);

/*-----*/
/* Delete the last row/column value.          */
/*-----*/
dtw_table_SetV(myTable,
               NULL,
               newRowValue,
               newColValue);

/*-----*/
/* Set the last row/column value.             */
/*-----*/
dtw_table_SetV(myTable,
               dtw_strdup(myNewValue),
               newRowValue,
               newColValue);

break;

```

圖 2. 語言環境模版 (21/29)

```

default:
/*-----*/
/* Internal error - unknown data type          */
/*-----*/
break;

```

圖 2. 語言環境模版 (22/29)

```

    }
  }
}
return 0;
}

```

圖 2. 語言環境模版 (23/29)

```

/*-----*/
/*
/* Function
/*   setErrorMessage()
/*
/* Purpose
/*
/* Format
/*   unsigned long setErrorMessage(int returnCode,
/*                                   char **defaultErrorMessage)
/*
/* Parameters
/*   int    returnCode
/*   char **defaultErrorMessage
/*
/* Returns
/*   Success ..... 0
/*   Failure ..... -1
/*
/*-----*/
unsigned long setErrorMessage(int returnCode,
                              char **defaultErrorMessage)
{
    /*-----*/
    /* Set the default error message based on the return code.
    /*-----*/
    switch(returnCode) {
        case LE_SUCCESS:
            break;

```

圖 2. 語言環境模版 (24/29)

```

        case LE_RC1:
            *defaultErrorMessage = dtw_strdup(LE_RC1_MESSAGE_TEXT);
            break;

```

圖 2. 語言環境模版 (25/29)

```

        case LE_RC2:
            *defaultErrorMessage = dtw_strdup(LE_RC2_MESSAGE_TEXT);
            break;

```

圖 2. 語言環境模版 (26/29)

```

        case LE_RC3:
            *defaultErrorMessage = dtw_strdup(LE_RC3_MESSAGE_TEXT);
            break;

```

圖 2. 語言環境模版 (27/29)

```

        case LE_RC4:
            *defaultErrorMessage = dtw_strdup(LE_RC4_MESSAGE_TEXT);
            rc = LE_RC1INTERNAL;
            break;

```

圖 2. 語言環境模版 (28/29)

```

    }
    return 0;
}

```

圖 2. 語言環境模版 (29/29)

附錄B. 注意事項

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

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

臺灣國際商業機器股份有限公司
台北市基隆路一段 206 號
法務部

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

臺灣國際商業機器股份有限公司
台北市基隆路一段 206 號
法務部

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

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

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

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

上述資訊之取得有其特殊要件，在某些情況下，必須付費方得使用。

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

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

著作權授權：

本書中有一些以原始語言列印出的範例應用程式，是用來說明各種作業平台上的程式設計技術。您可以基於研發、使用、銷售或散布符合作業平台應用程式介面的應用程式等目的，以任何形式複製、修改及散布這些範例程式而不必向 IBM 付費。這些範例應用程式並未經過徹底的測試，因此，IBM 不提供明示或默示其可靠性、有用性或符合特定效用的保證。您可以基於研發、使用、銷售或散布符合 IBM 應用程式介面的應用程式等目的，以任何形式複製、修改及散布這些範例程式而不必向 IBM 付費。

商標

下列詞彙是 IBM 公司在美國或其它國家或此兩者的商標：

AIX	Lotus
DataJoiner	MVS
DB2	Net.Data
Domino	OS/2
IBM	OS/390
IMS	OS/400

下列詞彙是以下其它公司的商標：

Java 及 HotJava 是 Sun Microsystems, Inc. 的商標。

Microsoft、Windows、Windows NT® 以及 Windows 95 標誌是 Microsoft Corporation 的註冊商標。

UNIX 是透過 X/Open Company Limited 獨家授權，在美國以及其它國家的註冊商標。

其它的公司、產品與服務名稱可能為其它公司的註冊商標或服務標記。

名詞解釋

七劃

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

八劃

空值 (null). 一個代表沒有資訊的特殊值。

十劃

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

十一劃

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

現場連線 (Live Connection). 與「連線管理程式」及 Web 伺服器 API 一起運作的 Net.Data 架構。「現場連線」可讓資料庫連線被再次使用。

通用資源位置 (uniform resource locator). 用來指出 HTTP 伺服器並可選擇性地指出目錄以及檔案名稱的位址，例如：
<http://www.software.ibm.com/data/net.data/index.html>

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

連線管理程式 (Connection Manager). Net.Data 中的可執行檔 dtwcm，支援「現場連線」時需用到這個檔案。

十二劃

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

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

十三劃

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

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

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

資料類型 (data type). 直欄與文字的屬性。

路徑 (path). 用來尋找檔案的搜尋路徑。

十四劃

語言環境 (language environment). 提供 Net.Data 巨集與外部資料來源 (如 DB2) 或程式設計語言 (如 Perl) 之存取的模組。有些語言環境是由 Net.Data (如 REXX、Perl 及 Oracle) 所提供。您也可建立您自己的語言環境。

十七劃

應用程式設計介面 (application programming interface, API). 為一功能性介面，由作業系統或由一個可分開訂購的授權程式所提供，讓您可以使用高階語言撰寫應用程式來使用作業系統或授權程式之特定資料或函數。Net.Data 可支援下列具有專利的 Web 伺服器 API，可讓您可在 CGI 處理上提高執行效能：ICAPI、GWAPI、ISAPI 及 NSAPI。

A

API. 應用程式設計介面。

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

B

BLOB. 二進位大型物件。

C

CGI. 通用閘道介面。

cliette. 一種專門處置 Web 伺服器所提要求之長期執行的處理。「連線管理程式」會排程 cliette 處理來處置這些需求。

CLOB. 字元大型物件。

D

DBMS. 資料庫管理系統。

H

HTML. 超文字標示語言。

HTTP. 超本文轉送通信協定。

I

ICAPI. Internet 連線 API (Internet Connection API)

ICS. Internet Connection Server。

ICSS. Internet Connection Secure Server。

Internet. 一個國際公用的 TCP/IP 電腦網路。

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

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

Intranet. 一個位於公司防火牆內的 TCP/IP 網路。

ISAPI. Microsoft 的 Internet 伺服器 API。

J

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

L

LOB. 大型物件。

N

NSAPI. Netscape API。

P

Perl. 一種解譯過的程式設計語言。

T

TCP/IP. 傳輸控制通信協定 / 網際網路通信協定。

U

URL. 通用資源位置。

W

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

索引

索引順序以中文字，英文字，及特殊符號之次序排列。

〔四劃〕

介面函數

處理次序 35
語言環境, 說明 35
dtw_cleanup() 37
dtw_execute() 36
dtw_getNextRow() 36
dtw_initialize() 35

公用程式函數

表格處理 42
架構變數 41
記憶體管理 41
語言環境 41
橫列操作 43
dtw_free() 44
dtw_getvar() 45
dtw_malloc() 46
dtw_row_SetCols() 47
dtw_row_SetV() 48
dtw_strdup() 49
dtw_table_AppendRow() 50
dtw_table_Cols() 51
dtw_table_DeleteCol() 53
dtw_table_DeleteRow() 54
dtw_table_Delete() 52
dtw_table_GetN() 55
dtw_table_GetV() 56
dtw_table_InsertCol() 57
dtw_table_InsertRow() 58
dtw_table_MaxRows() 59
dtw_table_New() 60
dtw_table_QueryColnoNj() 61
dtw_table_Rows() 62
dtw_table_SetCols() 63
dtw_table_SetN() 64
dtw_table_SetV() 65

〔六劃〕

名詞解釋 86

〔七劃〕

系統

傳遞變數 24
語言環境 24

〔八劃〕

呼叫 applet 6

注意事項 85

直欄

刪除 53
決定表格中的總數 51
指定表格中的個數 63
插入 57

表格

刪除 52
附加橫列 50
建立新的 60
操作公用程式函數 42

表格數值

刪除 52, 56, 65
取回 56
指定 48, 65

〔九劃〕

建立表格 60

指向儲存體 46

架構環境 37

架構變數

取回變數數值 45
管理用的公用程式函數 41

〔十劃〕

純本文檔介面

語言環境 5

記憶體管理公用程式函數 41

起始設定作業, 語言環境 35

〔十一劃〕

動態的記憶體配置 34

參數

命名 34

指定 34

傳遞 33, 34

parm_name 34

堆集, Net.Data run-time 34

執行語言環境陳述式 35, 36

清除

異常狀況的旗號 32, 37

處理程序後 35, 37

dtw_lei 旗號 32, 37

異常狀況

錯誤訊息 33

dtw_lei 旗號 32, 37

〔十二劃〕

最大列數 59

結構, 語言環境

dtw_lei 32

dtw_parm_data 33

〔十三劃〕

傳遞

參數 33

變數 33

〔十四劃〕

語言環境

介面函數 35

介面模版 69

公用程式函數 41

系統 24

建立 29

架構 37

純本文檔介面 5

起始設定 35

處理程序後清除 35, 37

陳述式, 執行 35

結構 31

摘要 3

語言環境 (繼續)

簡介 41

IMS Web 5

Java applet 6

Java 應用程式 13

ODBC 16

Oracle 17

Perl 20

REXX 21

SQL 22

Sybase 22

Web 登記 25

〔十五劃〕

模版, 語言環境 69

〔十六劃〕

橫列

刪除 53, 54

取回現行個數 62

附加 50

指定寬度 47

插入 58

傳回 33, 35, 36

傳回所允許的最大值 59

dtw_getNextRow() 介面函數 33

橫列操作公用程式函數 43

錯誤狀況 31

錯誤狀況訊息 33

〔十七劃〕

儲存體

配置 46, 47, 49, 63

釋放 32, 34, 44

dtw_lei 旗號 32

〔二十劃〕

嚴重錯誤, dtw_lei 旗號 32, 37

〔二十一劃〕

欄位標題

刪除 52, 55, 64

欄位標題 (繼續)

取回 55

指定名稱 64

配置儲存體 47, 63

傳回直欄個數 61

〔二十三劃〕

變數

傳遞 33

釋放 37

D

DB2, SQL 語言環境 22

dtw_ 介面函數 35

dtw_ 公用程式 41

dtw_ 結構 31

DTW_APPLET 6

DTW_FFI 5

DTW_IMS 5

DTW_JAVAPPS 13

dtw_lei

結構 32

欄位

函數名稱 32

旗號 32

default_error_messages 33

exec_statement 33

le_opaque_data 33

parm_data_array 33

row 33

DTW_LE_CONTINUE 36

DTW_ODBC 16

DTW_ORA 17

dtw_parm_data

結構 33

欄位

parm_descriptor 34

parm_name 34

parm_value 34

DTW_PERL 20

DTW_REXX 21

DTW_SQL 22

DTW_SYB 22

DTW_SYSTEM 24

DTW_WEBREG 25

E

ENVIRONMENT 陳述式

針對新的語言環境 37

語法 37

範例 39

exec 陳述式, dtw_lei 旗號 32

F

FUNCTION 區塊

名稱 32

執行陳述式 35, 36

I

IMS Web

專用工具 5

語言環境 5

J

Java applet

呼叫 6

建立 6

產生標籤 6

語言環境 6

範例 10

類別 11

Java 應用程式

呼叫 16

建立 cliettes 14

建立函數 14

特殊架構 15

語言環境 13

O

ODBC

巨集檔中的 SQL 陳述式 16

語言環境 16

Oracle

存取 17

特殊架構 17

Oracle (繼續)

- 語言環境 17
- 儲存程序 17

P

parm_data_array 結構, 指定名稱 33

Perl

- 語言環境 20
- script 中的 Net.Data 變數 20

R

REXX

- 語言環境 21
- 變數替代 21

row-at-a-time 處理程序

- dtw_getNextRow() 35, 36
- dtw_lei 旗號 33
- DTW_LE_CONTINUE 33

S

SQL

- 使用 DataJoiner 22
- 受支援的陳述式 22
- 語言環境 22

Sybase

- 大型物件 22
- 存取 22
- 特殊架構 22
- 語言環境 22

W

Web 登記

- 語言環境 25
- 說明 25

折疊線

台北市敦化南路一段二號十二樓

臺灣國際商業機器股份有限公司
中文支援中心 啟

廣告回信
台灣地區郵政管理局 登記
北台字第 0587 號

(免貼郵票)

寄件人 姓名：
地址：

寄

折疊線

讀者意見表

爲使本書盡善盡美，本公司極需您寶貴的意見；懇請您使用過後，撥冗填寫下表，惠予指教。

請於下表適當空格內，填入記號（✓）；我們會在下一版中，作適當修訂，謝謝您的合作！

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

其他：（篇幅不夠時，請另紙說明。）

[illegible]

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。



Printed in Singapore