Net.Data

語言環境介面參考手冊

Net.Data

語言環境介面參考手冊

注意事項

請務必先詳讀第61頁的『附錄D. 注意事項』中的資訊，再使用此資訊及其所支援的產品。

第六版 (1999 年 5 月)

目錄

序文

感謝您選購 IBM® 開發工具 -Net.Data®- 來建立動態網頁！有了 Net.Data 您就可以快速開發具有動態內容的網頁，方法是納入來自各種資料來源的資料以及使用您熟悉的程式設計語言功能。

關於 Net.Data

有了 IBM 的 Net.Data 產品，您就可以使用關聯式及非關聯式資料庫管理系統 (DBMS) 的資料建立動態網頁 (包括 DB2、IMS、具有 ODBC 功能的資料庫以及可透過 DRDA 存取的資料庫)，以及使用以程式設計語言撰寫的應用程式如 Java、JavaScript、Perl、C、C++ 和 REXX。 Net.Data 產品系列對執行下列作業系統的機器提供類似功能： Windows NT、AIX、OS/2、OS/390、OS/400、HP-UX、Sun Solaris、Santa Cruz 作業系統 (SCO) 和 Linux 作業系統。

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

使用格式 URL (一致資源定址器) 的要求從瀏覽器 (如 Netscape Navigator 或 Internet Explorer) 傳送到 Web 伺服器，該伺服器將要求轉送至 Net.Data 以供執行。 Net.Data 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：

- 在使用但不限於 C、C++、RPG、COBOL、JAVA、Perl 或 REXX 程式設計語言撰寫的應用程式內封裝企業邏輯
- 存取資料庫，例如 DB2
- 存取其它資料原始檔如純本文檔

Net.Data 將此 Web 網頁傳給 Web 伺服器，該伺服器會透過網路轉送此網頁以便顯示在瀏覽器上。

Net.Data 可用於一些架構來使用介面如「超本文轉送通信協定」(HTTP) 和「通用閘道介面」(CGI) 的伺服器環境。 HTTP 是瀏覽器與 Web 伺服器彼此交談的業界標準介面， CGI 是閘道應用程式如 Net.Data 的 Web 伺服器呼叫的業界標準介面。這些介面可讓您選取最喜歡的瀏覽器或 Web 伺服器來配合 Net.Data 使用。

爲了增進效能，Net.Data 支援多種 Web 伺服器「應用程式設計介面」(API)。此外，可以把 Net.Data 當作 Java servlet 來啓動。

關於這本書

本書說明 Net.Data 的「語言環境介面」(LEI)，您可使用該介面開發自己的自訂語言環境供 Net.Data 使用。

本書可能會提及一些已發表，但還未上市的產品或特性。

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

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

誰應閱讀本書

想要擴充 Net.Data 功能以滿足自己企業需求的人可使用本書撰寫 Net.Data 的自己的語言環境。

若要瞭解本書討論的概念，您應熟悉以下資訊：

- C 程式設計語言
- *Net.Data 管理和程式設計指南*和 *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。 藉由使用函數呼叫來呼叫語言環境，您可以使用這些語言環境提供以便與商用應用程式一起使用的功能。 例如，您可直接存取 ODBC 資料庫，使用 Perl 語言環境執行 Perl script，或呼叫 Java Applet 語言環境以執行 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viii頁的圖1示範 Net.Data 與語言環境互動的情況。

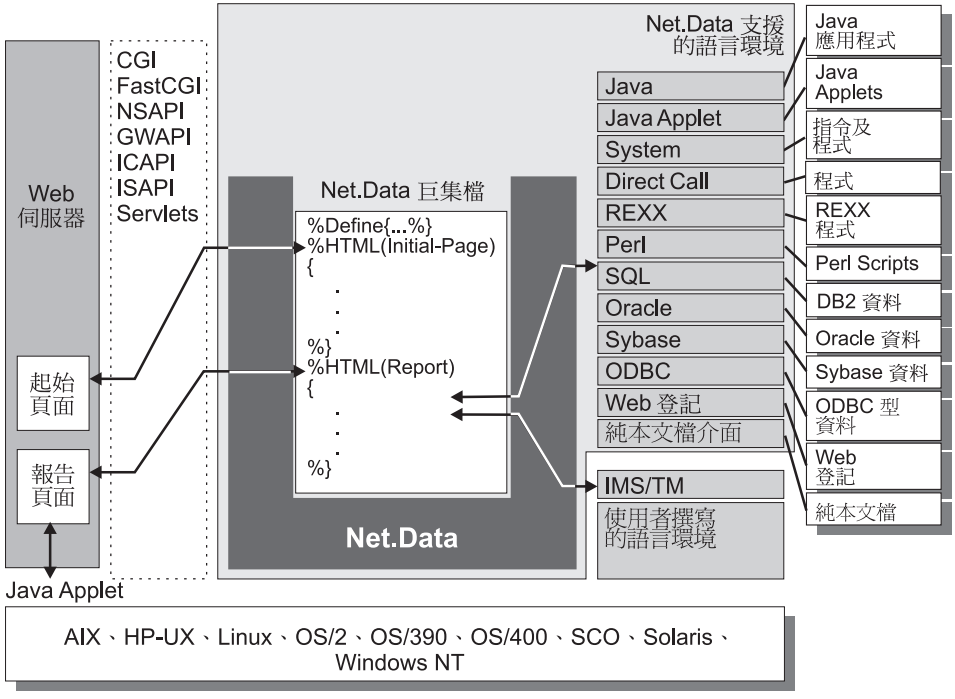

圖 1. Net.Data 及其語言環境

本書說明用來建立新語言環境的 Net.Data 語言環境介面。 Net.Data 提供的語言環境說明於您作業系統的 *Net.Data 管理和程式設計指南*的語言環境這一章。

第1章 建立新的語言環境

Net.Data 使用語言環境作爲可插入的程式設計語言和資料庫介面，根據作業系統環境來存取成 DLL 檔、共用程式庫或服務程式。在本文件中一般參照這些類型檔案時會使用詞彙*共用程式庫*。 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需求，您可使用 Net.Data 語言環境介面建立自己的語言環境。

建立新的語言環境涉及下列步驟：

- 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。
- 建置一個共用程式庫，該程式庫實施您要提供的語言環境介面常式集。請參閱編譯器的技術文件以瞭解如何建置共用程式庫。
- 使您可在共用程式庫以外使用全部介面以便 Net.Data 能夠呼叫它們。
- 決定您的 ENVIRONMENT 架構陳述式，接著將它加到 Net.Data 起始設定檔中。
- 將函數新增至使用新語言環境的 Net.Data 巨集。

決定建立新的語言環境之前，請判斷 Net.Data 提供的語言環境是否滿足您的需求。

本章將說明如何設計語言環境。

- 『建立共用程式庫』
- 第3頁的『語言環境通信結構』
- 第7頁的『語言環境介面函數』
- 第9頁的『設計語言環境陳述式』

若要瞭解語言環境程式設計介面，請參閱第13頁的『第2章 語言環境程式設計介面公用程式函數』。

建立共用程式庫

建置某語言環境時，您可使用第41頁的『附錄B. 語言環境模版』所提供的模版，該模版提供 Net.Data 使用的環境介面函數和通信結構與您的語言環境通信，並且在語言環境之間來回傳送參數。

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13頁的『第2章 語言環境程式設計介面公用程式函數』。

- 第2頁的『我應該提供哪一種語言環境介面？』
- 第2頁的『處理輸入參數』
- 第3頁的『處理使用者要求』

- 第3頁的『處理輸出參數』
- 第3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() 介面以便從巨集傳送輸入參數；它是每一個語言環境的唯一必要介面。 Net.Data 透過語言環境通信結構 dtw_lei_t 將全部輸入參數傳給 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()

提供 dtw_initialize() 介面而且您想要釋放任何資源時，提供 dtw_cleanup() 介面。

dtw_getNextRow()

提供 dtw_getNextRow() 介面來作爲資料庫語言環境的一部份，或是作爲可一次處理一列資料之語言環境的一部份。若在 OS/400® 或 OS/390 作業系統中執行 Net.Data 會呼叫此介面。

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() 介面使用 dtw_lei_t structure，Net.Data 使用它與語言環境通信。在撰寫您的語言環境時，請使用下列建議來處理輸入參數。

- 在 Net.Data 起始設定檔中，在語言環境的 ENVIRONMENT 陳述式中指定任何隱含參數。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。
- 將輸入參數接收到 dtw_execute() 介面作爲 dtw_lei_t 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41頁的『附錄B. 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 將函數呼叫傳給 dtw_lei_t 結構的 function_name 欄位中的語言環境。
- 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 將參數傳給 dtw_lei_t 結構的 parm_data_array 欄位中的語言環境。
- 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 將在 FUNCTION 區塊中指定的任何可執行的陳述式傳給 dtw_lei_t 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，一旦語言環境擁有必須傳回到 Net.Data 巨集的資料時，您可以指定語言環境在 dtw_lei_t 結構的 parm_data_array 欄位中修改傳送的參數值。在第41頁的『附錄B. 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。它的值會被設定爲 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。

若未指定 MESSAGE 區塊，或在指定的 MESSAGE 區塊中沒有任何項目來處理 dtw_execute() 傳回的回覆碼， Net.Data 會顯示 dtw_lei_t 結構的 default_error_message 欄位內容。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41頁的『附錄B. 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_t
- dtw_parm_data_t

Net.Data 將語言環境介面結構 (例如，dtw_lei_t) 傳給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_t 結構

每一個語言環境的介面函數都接收到指向 dtw_lei_t 結構的指標。dtw_lei_t 結構有下列格式：

```
typedef struct dtw_lei_t {              /* 語言環境介面                  */
    char *function_name;               /* 函數區塊名稱                  */
    int   flags;                       /* 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_t 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，且旗號設定爲 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 dtw_parm_data_t 結構陣列的指標。陣列的結尾是一個包含 0 的 parm_data 結構。 Net.Data 使用 dtw_parm_data_t 結構將變數和關聯值傳給某語言環境，然後取回可能是由該語言環境對變數值所做的任何變更。結構說明的相關資訊，請參閱『dtw_parm_data_t 結構』。

default_error_message

語言環境可將 default_error_message 欄位設定爲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 會將指標定義爲 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 會將橫列欄位設定爲一個橫列物件。 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爲止。

`reserved` 欄位僅供 IBM 人員使用。

dtw_parm_data_t 結構

Net.Data 使用 dtw_parm_data_t 結構將參數傳給語言環境。參數的來源有三種：

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

- Net.Data 起始設定檔中，指定在 ENVIRONMENT 架構陳述式上的參數

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_t 結構有下列格式：

```
typedef struct dtw_parm_data_t {        /* 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_t 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 欄位設定爲 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 會將這個指標設定爲 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的記憶體區域。如果 parm_name 或 parm_value 被置換爲其它字串，則必須釋放原始字串，並將其置換爲指向由 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 呼叫 dtw_execute() 以處理巨集 FUNCTION 區塊，該區塊含有語言環境必須處理的陳述式。
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)。

下列各節將爲您說明介面函數：

- 『dtw_initialize()』
- 第8頁的『dtw_execute()』
- 第8頁的『dtw_getNextRow()』
- 第9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。

就每一個巨集而言，Net.Data 只呼叫一次某語言環境的 dtw_initialize() 介面函數，也就是 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。如果回覆碼爲非零的值，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() 介面函數處理巨集 FUNCTION 區塊，這些區塊包含必須由此語言環境處理的陳述式。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列資料而定。如果是的話，介面函數會在 dtw_lei_t 結構中設定 DTW_LE_CONTINUE 旗號，以通知 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，dtw_execute 介面函數可產生要在報表區塊階段處理的整個表格。

dtw_getNextRow()

dtw_getNextRow() 介面函數會取回輸入，以進行一次一列的 Net.Data 表格處理。每當您設定 DTW_LE_CONTINUE 旗號，表示需要爲該表格處理其他列的資料時，就會呼叫這個介面函數。爲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_t 結構中設定 DTW_LE_CONTINUE 旗號。

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

1. 爲 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，並開始一次一列處理程序。
3. 處理報告區塊

4. 爲 dtw_getNextRow() 介面函數的回覆值處理訊息區塊。
5. 判定 dtw_getNextRow() 是否有啓動 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，則一次一列的處理程序結束，而 Net.Data 會繼續處理 Net.Data 巨集。

呼叫 dtw_getNextRow() 後，會設定 dtw_lei_t 結構中的橫列欄位以指向橫列物件。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。針對像切斷資料庫的連線或釋放變數的作業使用此函數。這個介面函數是選用性的。

處理 Net.Data 要求時，Net.Data 會在 Net.Data 處理結束時或某錯誤停止 Net.Data 處理巨集時，呼叫語言環境的 dtw_cleanup() 介面函數。

Net.Data 在 dtw_lei_t 結構中設定旗號欄位成爲 DTW_END_ABNORMAL (若清除處理發生異常)。下列異常狀況提供您於何時使用 dtw_cleanup() 的範例：

- 語言環境介面函數指示發生嚴重錯誤，方法是在 dtw_lei_t 結構中的旗號欄位中設定 DTW_LE_FATAL_ERROR 位元。
- 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

在 Net.Data 起始設定檔中每一個語言環境都有一個 ENVIRONMENT 陳述式，該檔案含有專屬於該語言環境的資訊。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，例如要爲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。

呼叫 Net.Data 時它會讀取架構資訊，但要等到 FUNCTION 區塊確認從巨集內呼叫該語言環境之後才會載入語言環境 DLL 或共用程式庫。 DLL 會保持載入的狀態，直到 Net.Data 結束爲止。

下列區段提供您可用在您文件中的語法、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  ([specification parameter_list, ...)
```

每一個 ENVIRONMENT 陳述式必須由一行構成。

下列爲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。 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。

  重要事項：名稱開頭不可以是字首 DTW。這個字首已被 Net.Data 附隨的語言環境預定使用。 若使用 DTW 字首，Net.Data 無法載入語言環境 DLL。

- *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。就每一個作業系統而言副檔名各不相同：

  – 在 AIX® 中，使用 *.o* 副檔名指定共用程式庫名稱。
  – 在 HP/UX 中，使用 *.sl* 副檔名指定共用程式庫名稱。
  – 在 OS/2® 和 Windows NT 中，使用 *.dll* 副檔名指定 DLL 名稱。
  – 在 OS/390® 中，不使用 *.dll* 副檔名指定 DLL 名稱。
  – 在 OS/400® 中，使用 *.SRVPGM* 副檔名指定服務程式名稱。
  在 SUN、SCO 和 LINUX 中，使用 *.so* 副檔名指定共用程式庫名稱。

  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。

- *specification*

  參數傳送規格指示 Net.Data 是否使用輸入、輸出或輸入及輸出的參數。可能值：

  IN　　用於輸入的參數

OUT 用於輸出的參數

INOUT 用於輸入和輸出的參數

- *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。 它們都傳到 *dtw_lei_t* 結構的 *parm_data_array* 欄位，位於在 FUNCTION 區塊定義中指定的參數後面。您必須先在 Net.Data 巨集中將這些參數定義爲變數，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。

下例使用 OS/2、AIX 和 Windows NT 的語法顯示 Net.Data 提供的語言環境的 ENVIRONMENT 陳述式。

```
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)   () CLIETTE "DTW_JAVAPPS"
ENVIRONMENT (DTW_PERL)     DTWPERL   ()
ENVIRONMENT (DTW_REXX)     DTWREXX   ()
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ()
ENVIRONMENT (HWS_LE)       DTWHWS    ()
```

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2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41頁的『附錄B. 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。而這些函數共分爲四種範疇：

- 『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14頁的『表格操作的公用程式函數』
- 第15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 1. 記憶體公用程式函數

公用程式函數	說明
第18頁的『dtw_malloc()』	配置用 dtw_malloc() 取自 Net.Data 執行堆集的儲存體。
第16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21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 2. 架構公用程式函數

公用程式函數	說明
第17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 3. 表格公用程式函數

公用程式函數	說明
第32頁的『dtw_table_New()』	建立一個表格物件
第24頁的『dtw_table_Delete()』	刪除一個表格物件。
第35頁的『dtw_table_SetCols()』	設定表格的寬度，並配置直欄表頭的儲存體。
第28頁的『dtw_table_GetV()』	取回表格數值。
第37頁的『dtw_table_SetV()』	設定表格數值。
第27頁的『dtw_table_GetN()』	取回一個表格直欄標題
第36頁的『dtw_table_SetN()』	設定表格直欄標題。
第34頁的『dtw_table_Rows()』	取回表格中的現行列數。
第23頁的『dtw_table_Cols()』	取回表格中的現行直欄號碼。
第31頁的『dtw_table_MaxRows()』	取回表格中所允許之橫列的最大數目。
第33頁的『dtw_table_QueryColnoNj()』	取回直欄的直欄數。
第22頁的『dtw_table_AppendRow()』	在表格尾端新增一列或多列。
第30頁的『dtw_table_InsertRow()』	在表格中插入一列或多列。
第26頁的『dtw_table_DeleteRow()』	從表格中刪除一列或多列。
第29頁的『dtw_table_InsertCol()』	在表格中插入一欄或多欄。
第25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 4. 橫列公用程式函數

公用程式函數	說明
第19頁的『dtw_row_SetCols()』	設定橫列的寬度。
第20頁的『dtw_row_SetV()』	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_free()

使用

釋放用 dtw_malloc() 配置自 Net.Data 執行堆集而配置的儲存體。緩衝區會指向將釋放的已配置儲存體。

語法

```
void dtw_free(void *buffer)
```

參數

緩衝區 指標指向將釋放的配置記憶體。

範例

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

myBuf = (char *)dtw_malloc(nbytes);

dtw_free((void *)myBuf);
```

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。儲存體的度爲 *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，並爲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

row	指向尚未配置任何直欄之剛建立橫列的指標。
cols	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，請將數值指定爲 NULL。

語法

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

參數

row	將修改的橫列的指標。
src	含有將設定之新值的字串。
col	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

table	指向將被附加橫列的表格的指標。
rows	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

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

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

table	指向將從其中取回直欄標題的表格的指標。
dest	指向含有直欄標題的字串的指標。
col	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

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

table	指向將修改的表格的指標。
after_col	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。
cols	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，插入一列或多列。

語法

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

參數

table	指向將修改的表格的指標。
after_row	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。
rows	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爲 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爲 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

table	新表格的名稱。
rows	將在新表格中配置的起始橫列數目。
cols	將在新表格中配置的起始直欄數目。
row_lim	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

table	指向將查詢的表格的指標。
name	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

table	指向沒有配置任何直欄或橫列的新表格的指標。
cols	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爲 NULL。

語法

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

參數

table	指向已指定欄位標題之表格的指標。
src	指定給新直欄位標題的字串。
col	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，請將數值指定爲 NULL。

語法

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

參數

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

範例

```
void *myTable;
char *myTableValue = "newValue";

rc = dtw_table_SetV(myTable, myTableValue, 3, 5);
```

附錄A. Net.Data 技術圖書庫

可以從 Net.Data 網站存取「Net.Data 技術圖書庫」，網址：

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

文件	說明
• *Net.Data Administration and Programming Guide for OS/390* • *OS/2 版、Windows NT 版和 UNIX 版 Net.Data 管理及程式設計指南* • *OS/400 版 Net.Data 管理及程式設計指南*	包含關於安裝、架構和呼叫 Net.Data 的概念和作業資訊。也說明如何撰寫 Net.Data 巨集，使用 Net.Data 效能技術，使用 Net.Data 語言環境，管理連線以及使用 Net.Data 記錄和追蹤故障檢修及效能調整。
Net.Data 參考手冊	說明 Net.Data 巨集語言、變數和內建函數。
Net.Data 語言環境介面參考手冊	說明 Net.Data 語言環境介面。
Net.Data 訊息與訊息碼參考手冊	列出 Net.Data 錯誤訊息和回覆碼。

附錄B. 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C. 建置檔案範例

提供下列範例作爲在不同作業系統上建置 LE (語言環境) 的參考。因爲您的特定作業系統和環境有自己的獨特性質，所以這些範例應該僅供參考。建置共用程式庫時，請參閱專屬於您正在作業系統上使用的編譯器的文件。

OS/390 JCL 範例

```
//BLDUSER  JOB ,'BLDUSER ',TIME=1,
// MSGCLASS=H,CLASS=A,
// USER=IBMUSER,MSGLEVEL=(1,1)
//*********************************************************************
//*  COMPILE STEP: C++ DLL     (USER-WRITTEN LE)                      *
//*                C/C++ FOR MVS/ESA(R) COMPILER V3 R2.0              *
//*                AD/CYCLE LE/370 V1 R7.0                            *
//*********************************************************************
//COMPILE EXEC PGM=CBC320PP,REGION=32M,
//          PARM=('/CXX SO,OPT,EXP,SE(''CEEV1R70.SCEEH.+'')',
//          'DEF(_XOPEN_SOURCE_EXTENDED)')
//STEPLIB   DD DSN=CEEV1R70.SCEERUN,DISP=SHR
//          DD DSN=CBCV3R20.SCBC3CMP,DISP=SHR
//SYSMSGS   DD DUMMY,DSN=CBCV3R20.SCBC3MSG(EDCMSGE),DISP=SHR
//SYSXMSGS  DD DUMMY,DSN=CBCV3R20.SCBC3MSG(CBCMSGE),DISP=SHR
//SYSIN     DD DSN=IBMUSER.NETDATA.USERLANG.C(USERLANG),DISP=SHR
//SYSLIB    DD DSN=CBCV3R20.SCLB3H.H,DISP=SHR
//USERLIB   DD DSN=IBMUSER.NETDATA.H,DISP=SHR
//SYSLIN    DD DSN=IBMUSER.NETDATA.USERLANG.OBJ(USERLANG),DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSCPRT   DD SYSOUT=*
//SYSTERM   DD DUMMY
//SYSUT1    DD DSN=&&SYSUT1;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT2    DD DSN=&&SYSUT2;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT3    DD DSN=&&SYSUT3;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT4    DD DSN=&&SYSUT4;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
```

```
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5    DD DSN=&&SYSUT5;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT6    DD DSN=&&SYSUT6;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7    DD DSN=&&SYSUT7;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8    DD DSN=&&SYSUT8;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9    DD DSN=&&SYSUT9;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10   DD SYSOUT=*
//SYSUT14   DD DSN=&&SYSUT14;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT15  DD  SYSOUT=*
//*
//*********************************************************************
//*  PRELINK STEP: C++ DLL     (USER-WRITTEN LE)                      *
//*                C/C++ FOR MVS/ESA COMPILER V3 R1.0                 *
//*                AD/CYCLE LE/370 V1 R7.0                            *
//*********************************************************************
//PLKED EXEC PGM=EDCPRLK,REGION=32M,COND=(0,NE,COMPILE),
//          PARM='MAP,UPCASE,MEMORY,DLLNAME(userdll)'
//STEPLIB   DD DSN=CEEV1R70.SCEERUN,DISP=SHR
//SYSMSGS   DD DSN=CEEV1R70.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB    DD DSN=CEEV1R70.SCEECPP,DISP=SHR
//SYSMOD    DD DSN=IBMUSER.NETDATA.USERLANG.DLLP(PRLKUSER),
//             DISP=SHR
//OBJECT    DD DSN=IBMUSER.NETDATA.USERLANG.OBJ,DISP=SHR
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSDEFSD  DD DSN=IBMUSER.NETDATA.USERLANG.DEFSD(USEREXP),
//             DISP=SHR
//SYSIN     DD DSN=IBMUSER.NETDATA.DEFSD(DTWLESHR),DISP=SHR
//          DD DSN=CBCV3R20.SCLB3SID(COMPLEX),DISP=SHR
//          DD DSN=CBCV3R20.SCLB3SID(APPSUPP),DISP=SHR
//          DD DSN=CBCV3R20.SCLB3SID(COLLECT),DISP=SHR
```

```
//          DD *
    INCLUDE OBJECT(USERLANG)
//*
//*********************************************************************
//*  LINK STEP: C++ DLL     (FFI LANGUAGE ENVIRONMENT)                 *
//*             C/C++ FOR MVS/ESA COMPILER V3 R2.0                     *
//*             AD/CYCLE LE/370 V1 R7.0                                *
//*********************************************************************
//LKED EXEC PGM=HEWL,REGION=2048K,COND=(4,LT,PLKED),
//          PARM='MAP,LIST,XREF,RENT,REUS,COMPAT=PM2'
//SYSLIB    DD DSN=CEEV1R70.SCEELKED,DISP=SHR
//SYSLIN    DD DSN=IBMUSER.NETDATA.USERLANG.DLLP(PRLKUSER),
//             DISP=SHR
//          DD *
            NAME USERDLL(R)
//SYSLMOD   DD DSN=IBMUSER.NETDATA.USERLANG.DLL,DISP=SHR
//SYSUT1    DD DSN=&&SYSUT1;,UNIT=SYSALLDA,DISP=(NEW,PASS),
//          SPACE=(32000,(30,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT  DD SYSOUT=*
//*
//*********************************************************************
//*  COPY STEP:  C++ DLL     (FFI LANGUAGE ENVIRONMENT)                *
//*              C/C++ FOR MVS/ESA COMPILER V3 R2.0                    *
//*              AD/CYCLE LE/370 V1 R7.0                               *
//*********************************************************************
//COPY EXEC PGM=IEWBLINK,REGION=500K,COND=(0,NE,LKED),
//          PARM='LIST,REUS,RENT,NCAL,LET,MAP,CASE=MIXED,COMPAT=PM2'
//SYSPRINT  DD SYSOUT=*
//INLIB     DD DSN=IBMUSER.NETDATA.USERLANG.DLL,DISP=SHR
//*
//SYSLMOD   DD PATH='/usr/lpp/netdata/cgi-bin',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=(SIRWXO,SIRWXG,SIRWXU)
//*
//SYSLIN    DD *
            INCLUDE INLIB(USERDLL)
            ENTRY CEESTART
            NAME userdll(R)
//*
```

make 檔範例 (OS/390 專用)

```
### Target
TARGET          = userlang
### C Compiler
CC              = c89
CFLAGS          = -D_XOPEN_SOURCE_EXTENDED=1 -O -I/u/USER01/netdata/include
### C++ Compiler
CCC             = c++
CCFLAGS         = -D_XOPEN_SOURCE_EXTENDED=1 -O -I/u/USER01/netdata/include
### Linker/Loader
LD              = c++
LDFLAGS         = $(CCFLAGS) -W l,dll
### Sources Headers and Objects
OBJS            = userlang.o
HDRS            = /u/USER01/netdata/include/dtwle.h userlang.h
#######################
### Program Libraries #
#######################
LIBS = /u/USER01/netdata/defsd/dtwleshr.x
#######################
### Additional Targets #
#######################
all:        $(TARGET)
$(TARGET):  $(OBJS) $(LIBS)
            echo "Linking $(TARGET) ..."
            $(LD) $(LDFLAGS) -o $(TARGET) $(OBJS) $(LIBS)
            echo "done"
userlang.o: $(HDRS) userlang.c
            $(CCC) $(CCFLAGS) -c userlang.c
clean:
            rm -f $(OBJS) $(TARGET)
```

OS/400 CL 範例

假設下列狀況，請使用後續步驟在 AS/400 建置「語言環境」：

- SRC 是原始檔案 (以 C 撰寫)。
- MYLE 包含可匯出的程序 `dtw_execute`。
- 檔案 QSRVSRC 成員 MYLEEXP 含有匯出程序 `dtw_execute` 的規格。

1. 建立模組：

   ```
   CRTCMOD MODULE(MYLIB/MYLE) SRCFILE(MYLIB/SRC)
   ```

2. 建立服務程式：

   ```
   CRTSRVPGM SRVPGM(MYLIB/MYLE) MODULE(MYLIB/MYLE)
                                SRCFILE(MYLIB/QSRVSRC) SRCMBR(MYLEEXP)
                                BNDSRVPGM(QTCP/QTMHLE)
   ```

QTCP/QTMHLE 是服務程式，它含有全部 Net.Data 可連結的 API。在 V4R3 和後續版次上，雖然可使用 QTCP/QTMHLE，但您應使用 QHTTPSVR/QTMJLE。

附錄D. 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，不表示或暗示只能使用 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交易上並不接受明示或默示保證的放棄聲明，因此此項聲明不見得適用於您。

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

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：

臺灣國際商業機器股份有限公司
台北市基隆路 1 段 206 號
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 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。

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。

商標

下列專有名詞是 IBM 公司在美國或 (及) 其它國家的商標：

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

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。

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`。

七劃

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

八劃

空值 (null). 一個代表沒有資訊的特殊值。

十劃

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

十一劃

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

通用閘道介面 (Common Gateway Interface 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

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

超本文轉送通信協定 (hypertext transfer protocol - HTTP). 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。

十七劃

應用程式設計介面 (application programming interface, API). 爲一功能性介面，由作業系統或由一個可分開訂購的授權程式所提供，讓您可以使用高階語言撰寫應用程式來使用作業系統或授權程式之特定資料或函數。 Net.Data 支援下列專屬 Web 伺服器 API 以便透過 CGI 處理來增進效能：ICAPI、GWAPI、ISAPI 和 NSAPI。

A

API. 應用程式設計介面。 Net.Data 支援 3 個 Web 伺服器 API 以便透過 CGI 處理來增進效能。

applet. 一種包含在 HTML 頁面中的 Java 程式。 Applet 使用具有 Java 功能的瀏覽器 (如 Netscape Navigator) 而且在處理 HTML 頁面時被載入。

C

CGI. 通用閘道介面。

D

DBMS. 資料庫管理系統。

Domino *Go* Web 伺服器. Lotus 和 IBM 提供的 Web 伺服器，該伺服器提供一般和安全連線。 ICAPI 和 GWAPI 是隨此伺服器提供的介面。

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 連線 API (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。

P

Perl. 一種解譯式程式設計語言。

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). 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〕

〔六劃〕

〔八劃〕

〔九劃〕

〔十劃〕

〔十一劃〕

〔十二劃〕

〔十三劃〕

〔十四劃〕

〔十五劃〕

〔十六劃〕

〔十七劃〕

〔二十劃〕

〔二十一劃〕

〔二十三劃〕

D

E

F

P

R

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：

中華民國文化支援中心

台北市敦化南路一段二號十二樓

啟

寄

讀者意見表

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

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

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

其他：（篇幅不夠時，請另紙說明。）

上述改正意見，一經採用，本公司有合法之使用及發佈權利，特此聲明。

Printed in Singapore