



Net.Data 程式設計手冊

目錄

注意事項	v
商標	v
第1章 簡介	1
本書的適用對象	1
本書中的範例	1
取得更多的 Net.Data 資訊	1
第2章 瞭解 Net.Data	3
解譯語言	3
自由格式	3
無任何類型的變數	3
Net.Data 的函數	4
錯誤的處置方式	4
第3章 架構 Net.Data	5
路徑陳述式	5
MACRO_PATH	6
EXEC_PATH	6
INCLUDE_PATH	7
FFI_PATH	7
HTML_PATH	7
環境陳述式	8
變數陳述式	8
維持您資產的安全	8
第4章 呼叫 Net.Data	11
錨點參照	12
HTML 表格頁面	12
第5章 開發 Net.Data 巨集	15
解析 Net.Data 巨集檔	15
定義區塊	16
HTML 區塊	16
函數定義區塊	18
Net.Data 巨集的變數	18
定義變數	19
參照變數	20
條件變數	21
環境變數	21
執行變數	21
隱藏變數	22
列示變數	23
表格變數	24
隱含變數	24
Net.Data 巨集的函數	24
定義函數	24
呼叫函數	26
呼叫儲存程序	27
訊息區塊	27

在巨集中產生 HTML	30
HTML 區塊.	30
報表區塊.	32
使用大型物件	33
第6章 使用內建函數	35
一般目的函數	35
算術函數.	35
字串函數.	35
字組函數.	35
表格函數.	35
純本文檔介面	36
何謂 FFI ?	36
安全注意事項	36
其他注意事項	36
Web 登記	38
第7章 使用語言環境	41
第8章 提高執行效能	43
現場連線.	43
現場連線的優點	43
我應該使用現場連線嗎?	44
啟動連線管理程式	44
架構現場連線	44
使用 ICAPI	46
使用 ISAPI	47
使用 NSAPI.	47
附錄A. 動態查詢範例	49
附錄B. 數值資料類型樣本	51
名詞解釋	55
索引	57

注意事項

在本書內提及各種 IBM 產品、程式或服務項目時，並不表示 IBM 有意於所有其營業的國家內，提供這些產品、程式或服務項目。凡提及 IBM 產品、程式或服務項目時，亦不表示只可用 IBM 的產品、程式或服務項目。在 IBM 有效智慧財產權或其他合法保護權之下，任何功能性相等的產品、程式或服務可用來代替 IBM 產品、程式或服務。但是，在使用這些並非由 IBM 專門設計的產品時，一切與產品有關的作業評估及驗證，均由使用者自行負責。在本文件中包含著 IBM 所擁有之專利或暫准專利。而提供本文件，並不表示允許您使用這些權利。您可以書面方式提出特許權限之相關問題，並郵寄至：

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

獲得本程式的授權者，如需其相關資訊作為下列用途：(1) 交換獨立創作的程式與其它程式 (包括本程式在內) 的相關資訊，(2) 互相使用彼此交換的資訊請洽：

IBM Corporation
555 Bailey Avenue, W92/H3
P.O. Box 49023
San Jose, CA 95161-9023

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。

商標

下列詞彙為 IBM 公司在美國或其它國家的商標或註冊商標：

DB2	OS/390
IBM	OS/400
Net.Data	

下列詞彙是其他公司的商標：

UNIX 為經由 X/Open Company Limited 授權在美國以及其它國家專用的註冊商標。

Microsoft、Windows、Windows NT 和 Windows 95 商標，是 Microsoft Corporation 的商標或註冊商標。

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

第1章 簡介

本書的適用對象

本書是綜合討論 (即不特別針對哪個平台) 如何使用 *Net.Data* 來設計程式。本書適用於想規劃與撰寫 *Net.Data* 應用程式的人員。有關各平台的相異處、*Net.Data* 訊息、與其他資訊，則詳述於 *Net.Data* 參考手冊。

爲了瞭解本書中所提及的各種概念，您必須熟悉 Web 伺服器的工作方式，清楚簡單的 SQL 陳述式與 HTML 標籤 (包括 HTML 表格頁面標籤)。

本書可能會參照一些已發佈但尚未可用的產品或特性。

本書中的範例

本書中會舉簡單的範例，爲您圖解說明某些特定的概念，且這些範例並不涵蓋各個 *Net.Data* 建構的每一種方式。有些範例僅舉片段，要能運作時還需用到其他的程式碼。

取得更多的 *Net.Data* 資訊

在 *Net.Data* 參考手冊 與 *Net.Data* 語言環境手冊 中會詳細說明 *Net.Data* 的特性。除此您也可以自 World Wide Web 中取得其他資訊：

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

在上述網址中，您可以取得 *Net.Data* 的樣本巨集、示範程式、本書的最新版、常見問題列示、並且在論壇中提出問題。

第2章 瞭解 Net.Data

當您單獨使用 HTML 時，您僅能建立靜態的 Web 網頁，也就是說，除非您加以編輯否則這些網頁不會變更。而如果要在 Web 中放置一些動態資料與應用程式，則通常是透過撰寫 CGI 程式，以動態建置一些 Web 網頁，像是目前的銷售統計等。然而撰寫各類型的程式並不容易。

Net.Data 可簡化互動式 Web 應用程式的撰寫，亦即，其會透過巨集將邏輯、變數、程式呼叫、與報表新增 HTML 中。所謂巨集，乃一種文字檔，其中包含了 Net.Data 巨集語言、HTML、以及操作資料所需的陳述式 (如 SQL 或 PERL)。這些巨集中乃將基本的 HTML 能力與 Web 伺服器程式的動態機能合併在一起，使您能輕鬆地從區域或遠端的資料庫、純本文檔、應用程式、與系統服務程式中，將動態資料加進靜態的 Web 網頁內。

當 Web 伺服器收到一個參照到 Net.Data 巨集的 URL 時，其會將 Net.Data 當成 DLL 或共用程式庫來呼叫，藉此將 Net.Data 視為「一般開道介面 (CGI)」處理或 Web 伺服器的「應用程式設計介面 (API)」緒，來啟動之。URL 中含有 Net.Data 的相關資訊，其中包括要處理哪個巨集檔。當 Net.Data 完成巨集檔的處理後，其會傳送最後結果的 HTML 給 Web 伺服器，然後 Web 伺服器再將之傳遞到顯示瀏覽器所在的 Web 從屬站上。

Net.Data 極適合用來建立動態的 Web 網頁，這是因為巨集語言比 Web 伺服器程式設計較為簡單，且其可讓您納入一些您所熟悉的語言，像是：HTML, SQL, PERL, 與 JavaScript。

另一個重要的好處是，Net.Data 可支援多種資料庫格式，也因此，您可以在大部份的平台上操作各種資料來源 (如：DB2, Oracle, 與 Sybase 等資料庫) 中的資料。有關詳細的說明，請參閱“提高執行效能”。*Net.Data 語言環境手冊* 中的附錄。

解譯語言

Net.Data 巨集語言是一種經過解譯的語言。當呼叫 Net.Data 來處理巨集時，Net.Data 會從檔案頂端起按順序直接解譯檔案中的每一條語言陳述式。而未經解譯的語言，則必須先編譯進程式物件中，如此才能執行。您只要指出處理該巨集的 URL，即可馬上察看任何您對巨集所做的變更，而不需經過重新編譯。

自由格式

Net.Data 在程式設計格式上的規定不多。因此，在程式格式上有頗大的自由與彈性。單一指示可跨多行，而多條指示也可全寫在同一行上。指示可起自任何直欄上。而空格或整行也可以跳過。

無任何類型的變數

Net.Data 是將所有資料全視為字串。只要字串是代表一個有效的數字 (包括採用指數格式的數字)，Net.Data 便可透過 Net.Data 函數，在該字串上執行算術運算。有關巨集語言變數的討論，請參閱 第18頁的『Net.Data 巨集的變數』。

Net.Data 的函數

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

錯誤的處置方式

當 Net.Data 巨集中有錯時，會傳回一些帶有說明的訊息給從屬站。您可以採任何的 HTML，自行設定您的錯誤訊息。有關詳細的說明，請參閱 *Net.Data 參考手冊*。

第3章 架構 Net.Data

Net.Data 使用起始設定檔來自行設定一些設定值、架構語言環境、以及搜尋路徑。有關檔案的位置及其內容與語法，會視您所用的平台與 API 而定。有關詳細資訊，請參考您平台中的相關說明（在 *Net.Data* 參考手冊。

起始設定檔中的基本資訊，都是跨所有平台的一般資訊，並且是以下列一種公認的架構陳述式來設定的。這些陳述式類型是：

- 路徑陳述式
- 環境陳述式
- 變數陳述式

有些系統可藉由額外的架構來啟用現場連線（會維護永久性的資料庫連線），並使用 Web 伺服器應用程式設計介面（API）來提高本身的執行效能。Net.Data 會根據您的平台，提供以下的 API 支援：

- Internet Connection Server API (ICAPI)
- Microsoft Internet Server API (ISAPI)
- Netscape API (NSAPI)

路徑陳述式

Net.Data 會辨識這些路徑陳述式，藉此找到處理 Net.Data 巨集時所需的檔案。

- MACRO_PATH
- EXEC_PATH
- FFI_PATH
- INCLUDE_PATH
- HTML_PATH

這些路徑陳述式（HTML_PATH 除外），乃代表一組目錄，可讓 Net.Data 藉此找到巨集檔、可執行檔、純本文檔、與併入檔。每一個指定目錄都是以分號 (;) 區隔開來。斜線 (/) 與反斜線 (\) 則視為一樣。此種多路徑特性，可讓您組織您的檔案，並藉由將您的 Web 應用程式放在其個別的目錄中加以區隔開來。

其會搜尋您所指定的目錄，但不會搜尋次目錄。例如，當 Net.Data 巨集存在於這些目錄中時，您必須在路徑陳述式中指出每一個巨集所在：

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

您的 MACRO_PATH 陳述式可能會像：

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

下列段落中有關 Net.Data 架構上的範例，其中會提及路徑上的一些規格，不過這些規格可能會與您的平台與架構有所出入。

MACRO_PATH

MACRO_PATH 陳述式的語法，與其他的路徑陳述式有些類似：

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

等號 (=) 是選用性的，在指出時會加上方括弧。MACRO_PATH 陳述式，是用來指出一或多個要搜尋的目錄 (按您所指定的順序搜尋)，以找出 Net.Data 巨集檔。例如，傳送下列的 URL 要求，要求取得路徑與檔名為 /WWW/macro/sqlm.mac 的 Net.Data 巨集：

```
http://hostname/cgi-bin/db2www/WWW/macro/sqlm.mac/report
```

Net.Data 會將路徑 /WWW/macro 附加到 MACRO_PATH 架構陳述式的路徑中 (從左到右)，直到找到 Net.Data 巨集或搜尋過所有路徑為止。有關呼叫 Net.Data 巨集的詳細說明，請參閱 第11頁的『第4章 呼叫 Net.Data』。

範例：

Net.Data 起始設定檔：

```
MACRO_PATH = ../macro;product1/macro
```

HTML 錨點參照：

```
<A HREF="cgi-bin/db2www/query.mac/input">提出另一個查詢。</A>
```

若 *query.mac* 是在 ../macro 目錄中找到，則 Net.Data 巨集的完整名稱即為 /WWW/macro/query.mac。

EXEC_PATH

EXEC_PATH 的格式乃和 MACRO_PATH 類似：

```
EXEC_PATH [=] path1;path2;...;path  
n
```

EXEC_PATH 陳述式是用來指出一或多個要搜尋的目錄 (按您所指定的順序搜尋)，以找出 EXEC 陳述式所呼叫的外部程式。一旦找到該程式，則會將外部程式的名稱附加到具完整檔名的路徑設定中，以便傳遞給語言環境準備執行。

範例 1：

Net.Data 起始設定檔：

```
EXEC_PATH = ../pgms;product1/pgms;product2/pgms
```

Net.Data 巨集：

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

若 Net.Data 是自目錄 /www/cgi-bin 來執行，且檔案 *myfunction.cmd* 是在 product1/pgms 目錄中找到，則程式的完整名稱即為 /www/cgi-bin/product1/pgms/myfunction.cmd。

有關區域訊息區塊的另一範例說明，請參閱 *Net.Data 參考手冊*。

INCLUDE_PATH

INCLUDE_PATH 的格式乃和 MACRO_PATH 類似：

```
INCLUDE_PATH [=] path1;path2;...;path  
n
```

INCLUDE_PATH 是用來指出一或多個要搜尋的目錄 (按您所指定的順序)，以便找出 Net.Data 巨集中 INCLUDE 陳述式所指的檔案。一旦找到該檔，會將此併入檔的名稱附加在路徑設定中，以產生完整的併入檔名稱。

範例 1：

Net.Data 起始設定檔：

```
INCLUDE_PATH = ../macro;product1/macro
```

Net.Data 巨集：

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

若 Net.Data 是自目錄 /WWW/cgi-bin 來執行，且 *myinclude.txt* 是在 ../macro 目錄中找到，則此併入檔的完整名稱即為 /WWW/macro/myinclude.txt。

範例 2：

Net.Data 起始設定檔：

```
INCLUDE_PATH = ../INCLUDES
```

Net.Data 巨集：

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

若 Net.Data 是自目錄 /www/cgi-bin 來執行，而併入檔是在 /WWW/INCLUDES/OE 目錄中找到，則併入檔的完整名稱即為 /WWW/INCLUDES/OE/oeheader.inc。

FFI_PATH

FFI_PATH 的格式乃和 MACRO_PATH 類似：

```
FFI_PATH [=] path1;path2;...;path  
n
```

FFI_PATH 陳述式是用來指出一或多個要搜尋的目錄 (按您所指定的順序)，以便找出搭配純本文檔介面 (FFI) 函數使用的純本文檔。

範例：

Net.Data 起始設定檔：

```
FFI_PATH = pub/ffi;pub/ffi/data
```

當呼叫 FFI 語言環境時，其會尋找此處所指的路徑。

HTML_PATH

HTML_PATH 的格式乃和 MACRO_PATH 類似：

```
HTML_PATH [=] path1;path2;...;path  
n
```

範例：

Net.Data 起始設定檔：

```
HTML_PATH = pub/www/htm
```

當查詢傳回一個大型物件 (LOB) 時，Net.Data 會將它儲存在 HTML_PATH 架構變數指定的目錄中。因為 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮系統限制問題。有關區域訊息區塊的另一範例說明，請參閱 第33頁的『使用大型物件』。

環境陳述式

ENVIRONMENT 陳述式是用來架構語言環境。所謂語言環境，為一種可插入的模組，而 Net.Data 會透過其來存取資料來源 (如：DB2 資料庫)。語言環境乃被當成 DLL 或共用程式庫來存取的。Net.Data 中提供了多種語言環境，其中包括和資料庫溝通的介面。您可以按 *Net.Data 語言環境手冊*。為了讓 Net.Data 能和語言環境建立起溝通介面，在 Net.Data 起始設定檔中必須含有 ENVIRONMENT 陳述式。此陳述式的語法如下：

```
ENVIRONMENT(type) library_name ([usage argument, ...] [CLIETTE "LE_type:  
dbname"])
```

有關區域訊息區塊的另一範例說明，請參閱 *Net.Data 語言環境手冊*。

變數陳述式

變數陳述式為帶有如下語法的字串：

```
NAME [=] VALUE_STRING
```

變數陳述式具有各種用途。在大部份的情況下，變數陳述式是用來設定一個語言環境所需的值，以便使其正常運作，或者用於替代模式上。Net.Data 所用的路徑陳述式，是一種特殊的變數陳述式範例。Net.Data 會提供您自語言環境中來存取變數陳述式的介面。有關區域訊息區塊的另一範例說明，請參閱 *Net.Data 語言環境手冊*。

維持您資產的安全

雖然 Net.Data 不直接提供任何類型的安全措施，但您可以延用過去保護系統和資料時所使用的現有措施，來保護您的資產。您必須決定出適用於資產的安全層次。

- 身份驗證

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

- 大多數的 Web 伺服器可讓您設定要保護的伺服器目錄。您也可以讓系統要求使用者先輸入使用者 ID 及通行碼，才能存取到您所指定之目錄中的檔案。有關您 Web 伺服器如何決定系統功能的詳細說明，請參閱 *管理者手冊*。

- DB2 有一個有關資料庫存取的身份驗證系統，這個系統可限制只有某些使用者才可存取表格及直欄。您可以使用 Net.Data 的特殊變數，LOGIN 和 PASSWORD，鏈結到 DB2 身份驗證常式。OS/390 與 OS/400 所用的安全設計不同，詳細說明請參閱 *Net.Data* 參考手冊。

- 加密

在您使用 Web 伺服器時，如果有搭配使用「安全 Socket 層 (Secured Sockets Layer, SSL)」或「安全超本文轉送通信協定 (Secured Hypertext Transfer Protocol, SHTTP)」，就可以對從屬站系統和 Web 伺服器間往來的所有資料加密。這些安全措施會對下列的資料進行加密：登入 ID、通行碼，以及透過 HTML 表格頁面，自從屬站輸入的資料及所有自 Web 伺服器傳送的資料。

- 防火牆

Net.Data 可搭配 IBM Firewall 及其他大多數的防火牆產品一起使用，以防止外部的窺視或侵犯到 Net.Data 伺服器和網路。請根據您組織的安全策略，仔細考量這些架構問題。

- 架構 A

在此種架構中，會建立一個其種僅含 Net.Data 與 Web 伺服器的子網路。此種是最安全的架構方式，因為 Net.Data 與 DB2 都是放在防火牆內。此種架構方式需用到以下的安全措施：

- 在封包過濾架構檔中，允許埠 80 (標準 HTTP 埠) 上之任何主電腦處傳來的 TCP 封包，存取 Web 伺服器。同時允許從 Web 伺服器送出的 TCP 認可封包，傳至任何主電腦上。
- 在防火牆內安裝 socks 伺服器。在 socks 伺服器架構檔中，加進了一個項目，以允許 ftp 與 telnet 能透過 Intranet 存取到 Web 伺服器。對於使用 Intranet 而希望 ftp 或 telnet 子網路的使用者來說，通常得編輯 /etc/socks.conf 檔，才能透過 socks 伺服器導引他們的要求。
- 為了讓 Net.Data 能在 Intranet 內存取到 DB2，請將「從屬站應用程式啓用器 (Client Application Enabler, CAE)」安裝在 Web 伺服器上 (若為 OS/390 與 OS/400 則不需要)，並新增一條封包過濾規定，以允許 DB2 從屬站向 Net.Data 提出要求，及認可從 DB2 送自 Net.Data 上的封包。OS/390 與 OS/400 不需用到 CAE。

- 架構 B

另一種架構方式是將 Net.Data 放在工作站平台之防火牆外頭，將 DB2 放在防火牆內，而兩者間以 DDCS 進行通信。此種架構方式較「架構 A」來得簡單，然仍提供資料庫保護特性。

此種架構方式顯示 DB2 是位於防火牆內。Web 伺服器上必須裝有 CEA。除此，防火牆必須具備一條封包過濾規定，允許 DB2 從屬站向 Net.Data 提出要求，並認可由 DB2 送封包至 Net.Data 上。

- 架構 C

在此種架構方式下，防火牆不需用到任何封包過濾規定，不過 Net.Data 與 DB2 皆未受到保護，而有遭外來侵犯之虞。

另一種保護您資產的方法是，使用 Net.Data 建立您自己的保護體系。例如，您可以要求使用者透過 HTML 表格頁面輸入驗證資訊，並使用資料庫中的資料或者透過外部程式 (由 Net.Data 巨集來呼叫)，來驗證之。

您也可以允許人員傳送至資料庫的 SQL 陳述式，來保護您的資產。例如，將 SELECT 陳述式限制在兩個表格上。至於保護資產的詳細資訊，請參閱常見問題 (FAQ) 中有關 Internet 安全列示的部份，其 Web 網址如下：

<http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html>

至於其他的安全問題，您可以考慮使用隱藏變數，隱藏您資料庫的內部結構，以防止他人用 Web 瀏覽器檢視您的 HTML 原始資料。有關區域訊息區塊的另一範例說明，請參閱 第22頁的『隱藏變數』。

第4章 呼叫 Net.Data

對大部份的平台而言，您可以將 Net.Data 設定成當做「一般開道介面 (CGI)」來執行，也可以用 Web 伺服器 API，如：Lotus Go Webserver, Internet Connection Server (ICAPI), Netscape Server (NSAPI), 與 Microsoft Internet Server (ISAPI) 來執行。呼叫 Net.Data 所用的語法，是視 Net.Data 的架構方式而定。有關區域訊息區塊的另一範例說明，請參閱 第43頁的『第8章 提高執行效能』。

不管您採用哪一種方式，Net.Data 是透過 HTML 錨點參照、HTML 表格頁面、或直接當成 URL，從 Web 瀏覽器來呼叫的。當 Web 伺服器收到會呼叫 CGI 應用程式或 API 處理的參照、表格頁面、或 URL 後，Net.Data 即會啟動。Web 伺服器會將 Net 巨集名稱、Net.Data 巨集中的 HTML 區塊名稱、以及任何的輸入變數，傳遞給 Net.Data。

如果您的 Net.Data 巨集要花上很長的時間才能完成 (例如，若它們要連接遠端資料庫)，請考慮變更您 script 的逾時值。一般而言，預設的逾時時間為 5 分鐘，此值對大多數的巨集來說已夠用。請參閱您的 Web 伺服器文件以取得特定的說明。

在下列的範例中，是說明如何以各種方式將 Net.Data 當成 CGI 應用程式來呼叫。字串 `cgi-bin/db2www`

乃代表呼叫 Net.Data 的字串，且對大部份的系統而言為預設值。您可按您 Web 伺服器中的文件說明，將字串定義在您的 Web 伺服器架構檔中。

- 錨點參照：

```
<A HREF="http://server/cgi-bin/db2www/  
filename.ext/block/  
[?name=val&...]">任何文字</A>
```

- HTML 表格頁面：

```
<FORM METHOD=method ACTION="http://  
server/cgi-bin/db2www/  
filename.ext/block/[?name=val&...]">任何文字</FORM>
```

- URL：

```
http://server/cgi-bin/db2www/filename.ext/block/[?name=val&...]
```

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

filename.ext

Net.Data 巨集檔的名稱與副檔名。

block 設定在 Net.Data 巨集檔中之 HTML 區塊的名稱。

method

與表格頁面一起使用的 HTML 方法。建議值為 POST。

?name=val&...

要傳遞給 Net.Data 之一或多個選用性參數。例如，您可以傳遞使用者 ID，如此一來，應用程式的使用者僅需輸入一次使用者 ID，或者您也可以傳遞一個 Net.Data 巨集名稱，以另選一個應用程式。

ICAPI 會採用和 CGI 相同的語法，且透過 Web 伺服器的架構檔，httpd.conf 來管理。若您是使用 ISAPI 或 NSAPI，請按“使用ISAPI”與“使用 NSAPI”中的說明來呼叫 Net.Data。第47頁的『使用 ISAPI』與 第47頁的『使用 NSAPI』。

典型的 URL 加上參數乃類似如下：

```
http://www.ibm.com/cgi-bin/db2www/queryA.mac/input?field1=custno&field2=custname
```

以標準輸入方式傳送 HTML 表格頁面時，自錨點參照傳送給 Net.Data 的參數，是以 QUERY_STRING 介面傳送的。

錨點參照

錨點參照通常稱為超鏈結或簡稱為鏈結。使用鏈結時，您可以設定 Web 巨集和 HTML 區塊名稱來呼叫 Net.Data。您可以在鏈結中放進文字或圖形。只要使用者按一下鏈結的文字或圖形，即會呼叫 Net 巨集。以下的範例，是說明如何利用鏈結，自 Web 網頁中呼叫一個(canned query)。(所謂(canned query)，乃是一種不需要輸入的 SQL 查詢。)

```
<a href="http://www.ibm.com/cgi-bin/db2www/listA.mac/report">
列出所有的監視器</a>
```

鏈結會呼叫下列的巨集：

```
%DEFINE DATABASE="MNS97"

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

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

此項查詢會傳回一個列有所有監視器的表格，其中包括了型號、成本、及每一部監視器的說明。本範例會產生一個預設的報表。有關使用 REPORT 區塊，自行設定您報表的說明，請參閱 第32頁的『報表區塊』。

一般而言，Net.Data 巨集中每個區塊的開頭為 %block_name{ {且結尾為}。%}。有關語法的詳細說明，請參考 *Net.Data 參考手冊*。

HTML 表格頁面

HTML 表格頁面可讓您自行設定 Net.Data 巨集作業，並且對大部份的應用程式來說將有所助益。本範例與上述的監視器列示範例類似，不過本範例可讓應用程式使用者選取要查看的產品。您可以透過以下的鏈結，來呼叫巨集：

```
<a href="http://www.ibm.com/cgi-bin/db2www/equip1st.mac/input">
顯示硬體列示</a>
```

以下是鏈結所呼叫的巨集：

```
%DEFINE DATABASE="MNS97"

%HTML(input){
<H1>硬體查詢表格頁面</H1>
<HR>
```

```

<FORM METHOD=POST ACTION="cgi-bin/db2www/equip1st.mac/report">
<P>您想查看何種硬體？
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="MON" checked> 監視器
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PNT"> 指標裝置
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PRT"> 印表機
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="SCN"> 掃描器
</MENU>

<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
%}
%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='$(hdware)'
%REPORT{
<H3>此為您所要的列示</H3>
%ROW{
<HR>
$(N1): $(V1), $(N2): $(V2)
<P>$(N3): $(V3)
%}
%}
%}

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

```

在應用程式使用者做出選擇並按下「提出」按鈕後，Web 伺服器會處理 FORM 標籤的 ACTION 參數，以便讓 Net.Data 去呼叫 HTML 報表區塊。有關用於 ROW 區塊中之變數的詳細說明，請參閱 *Net.Data* 參考手冊。

第5章 開發 Net.Data 巨集

本章是解釋組成 Net.Data 巨集檔的各個部份，並說明這些部份如何一起運作在應用程式中。

解析 Net.Data 巨集檔

在本節中，我們會舉一個簡單的 Net.Data 巨集為例，來說明巨集語言的元素。這個範例巨集會呈現一個表格頁面，讓您輸入資訊，以傳給 REXX 程式。此巨集將資訊傳給一個稱為 OMPSAMP.CMD 的外部 REXX 程式（該程式會回應使用者輸入的資料）。之後會將結果顯示在第二個 HTML 頁面中。

首先請先看看整個巨集，然後再看看每個區塊中的詳細說明：

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

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

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

%{ ***** HTML 區塊：輸入 *****%}
%HTML (INPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>輸入表格頁面</h1>
Today is @today()

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

<hr>
<p>[<a href="/">Home page]
</body></html>
}%

%{ ***** HTML 區塊：輸出 *****%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>輸出頁面</h1>
<p>@rexx1(input_data)
<p><hr>
```

```
<p>[<a href="/">首頁</a> |  
<a href="input">前一頁</a>]  
</body></html>  
%}
```

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

這二個 HTML 區塊中所含的 HTML 標籤十分類似，這是爲了方便您撰寫 Web 巨集。如果您已經十分熟悉 HTML，在簡單建置一個巨集上，可加進一些要在伺服器上動態處理的巨集陳述式與要傳給資料庫的 SQL。

雖然巨集看起來很像 HTML 文件，但 Web 伺服器會透過 Net.Data，將它當成一個 CGI 程式或 Web 伺服器 API 來存取。Net.Data 需用到兩個參數：要處理的巨集名稱，以及在該巨集中要顯示的 HTML 區段。

當呼叫巨集檔時，Net.Data 會從頭開始處理之。下一節我們會告訴您 Net.Data 在處理檔案時會發生什麼。

定義區塊

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

第一行是一個備註。所謂備註，是指位於 `%{` 與 `%}`。備註不能位於其他巨集區塊之內。下一個陳述式會開始 DEFINE 區段。您可以在一個定義區塊中定義多個變數。本範例中只定義一個變數，`page_title`。定義好後，您可以使用語法 `$(page_title)`，在巨集的任何地方參照此參數。透過變數的使用，可讓您以後在對您的巨集進行整體的變更時較爲簡單。本區塊的最後一行上的 `%` 表示，`%}`，是代表 DEFINE 區塊的結尾。

下個區塊是一函數定義區塊，不過在本範例中，您會先看到稱爲 INPUT 的 HTML 區塊。

HTML 區塊

```
%{ ***** HTML 區塊：輸入 *****%}  
%HTML (INPUT) { <--- 代表此 HTML 區塊的名稱。  
<html>  
<head>  
<title>$(page_title)</title> <--- 請注意變數的替代。  
</head><body>  
<h1>輸入表格頁面</h1>  
Today is @today() <--- 本行包含一個對函數的呼叫。  
  
<FORM METHOD="post" ACTION="output"> <--- 當  
提出此表格頁面時，  
會呼叫 "output" HTML 區塊。  
請輸入一些要傳給 REXX 程式的資料：  
<INPUT NAME="input_data" <---  
"input_data" 會在提出此表格頁面時定義，  
TYPE="text" SIZE="30"> 並可參照到此巨集的任一處。  
其被起始設定成使用者在輸入欄位中所鍵入的任  
何項目。  
  
<p>  
<INPUT TYPE="submit" VALUE="Enter">  
  
<hr>
```

```

<p>
[
<a href="/">首頁</a>]
</body><html>
%}

```

<--- 結束 HTML 區塊。

此區塊中的 HTML 表格頁面十分簡單，只有一個輸入欄位。整個區塊都用 HTML 區塊 ID，%HTML (INPUT) { ... %} 圍住。*INPUT* 會指出此區塊的名稱。您可以為其指定任何名稱。HTML <title> 標籤中有一個巨集替代的範例。變數 *page_title* 的內容，會替代成表格頁面的標題。

此區塊中還含有一個函數呼叫的範例。表示式 @today() 是用來呼叫函數 *today*。此函數是定義在 FUNCTION 區塊中 (以後會提到)。而 *today* 函數的結果 (現行日期)，會插到和 @today() 表示式位置相同之 HTML 文字中。

FORM 陳述式中的 ACTION 參數，即是 HTML 區塊之間或巨集之間的導引範例。若是參照 ACTION 參數中的另一個區塊名稱，則當提出表格頁面時，即會存取該區塊。任何來自 HTML 表格頁面的輸入資料，都會被當成隱含變數來傳給區塊。定義在此表格頁面中的單一輸入欄位，亦可適用。當提出格式後，即會將輸入在此欄位中的資料傳遞給變數 *input_data* 中的 HTML 輸出區塊。

只要巨集檔是位在同一 Web 伺服器上，您便可以使用相關參照方式，來存取其他巨集檔中的 HTML 區塊。例如，ACTION="../othermacro.mac/main" 可以使您存取巨集檔 othermacro.mac 中稱為 main 的 HTML 區塊。同時，任何您在此表格頁面中所鍵入的資料，都會在變數 *input_data*。

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

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

接收輸入資料時，不必再像使用大部份的 CGI 程式般，得處理環境變數。Net.Data 會代您處理這些。您只需參照變數名稱即可。

本例中的下一個 HTML 區塊為 OUTPUT 區塊。

```

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

</head><body>
<h1>輸出頁面</h1>
<p>@rex1(input_data) <--- 本行中含有一個對函數 rex1 的呼叫
                        此函數會傳遞引數 "input_data"。

<p>
<hr>
<p>
[
<a href="/">首頁</a> |
<a href="input">前一頁</a>]
%}

```

類似 INPUT 區段，本區塊是一個標準的 HTML，亦即其中含有用來替代變數的巨集陳述式以及一個函數呼叫。同樣地，*page_title* 變數會替代成 title 陳述式。一如前述，本區塊中含有一個函數呼叫。在本例中，其會呼叫函數 *rex1* 並將自 INPUT 表格頁面中所收到之變數 *input_data* 的內容傳給它。您可以和函數間互傳多個變數。函數定義會決定變數的傳遞數量及類型。

有關函數的詳細說明，請參閱 第17頁的『函數定義區塊。』。

函數定義區塊。

```
%{ ***** 函數定義區塊 *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- 此函數會接受
                                                         一個參數，並傳回一個
                                                         替代了相關函數
                                                         呼叫的結果
    %EXEC{ompsamp.cmd %} <-- 此函數會執行一個外部 REXX 程式，
                           "ompsamp.cmd"
%}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
    result = date() <-- 此函數的單一原始陳述式是以
                       列入(inline) 方式存在的。
%}
```

此函數定義區塊中含有兩個函數宣告。第一個 *rexx1*，是一個 REXX 函數宣告，執行外部 REXX 程式 *ompsamp.cmd*。此函數會接受輸入變數，*input*，並將之自動傳給外部 REXX 指令。REXX 指令也會傳回一個稱為 *result*。REXX 指令中的 *result* 變數內容，會取代 OUTPUT 區塊中的 @*rexx1*() 函數呼叫。變數 *input* 與 *result* 可讓 REXX 程式直接存取，相關說明，可參考 *ompsamp.cmd* 的原始格式。：

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

此函數中的程式碼會對傳給它的資料做出回應。您可以按您的意思來製作結果文字的格式，只要將所要求之 @*rexx1*() 函數呼叫以一般 HTML 樣式標籤 (如： 或)。REXX 程式不使用 *result* 變數，只須用 REXX SAY 標籤，即可將寫好的 HTML 陳述式以標準格式輸出。

第二個函數宣告 *today*，也參照 REXX 程式。不過，本例中的整個 REXX 程式 (一整行) 都包含在其本身的函數宣告中。不需要一個外部程式。REXX 和 PERL 函數都可接受列入 (inline) 程式，這是因為它們皆為解譯過的語言，可動態解析及執行之。列入程式不需用到另一個程式檔來管理，因此具有簡便上的好處。第一個 REXX 函數也可以用列入方式處理。

Net.Data 巨集的變數

Net.Data 可讓您在 Net 巨集內定義及參照變數。另外，您可以將巨集中的這些變數傳遞給語言環境後再傳回。Net.Data 記號 (token)，如：變數名稱與引號內的字串，最多可有 64KB 的資料。而在 OS/400 方面，其最大大小則由系統決定。

巨集變數可以分成三大類型：

- 使用 DEFINE 陳述式明確地定義的變數。
- 事先定義好的變數，這些變數可讓 Net.Data 使用，並且設有一值。
- 隱含的定義變數，有下列四種類型：
 - 此類的變數雖非明確地定義，不過在首次參照時可被個案化。
 - 參數變數，為 FUNCTION 區塊定義的一部份，且僅能在 FUNCTION 區塊中。
 - 此類變數可讓 Net.Data 個案化，且會對應到表格頁面資料或 URL 資料名稱-值配對上。

- 此類變數乃和 Net.Data 表格相連結，且僅能在 ROW 區塊或 REPORT 區塊中被加以參照。

當宣告或個案化一個識別字(通常是一個變數或函數)時，該識別字會變成可見狀態(亦即可被參照到)。而變成可見狀態之識別字所在的區域，即稱為其範圍。範圍的類型有下列五種：

- 廣域

若您可在巨集檔的任一處參照到此識別字，則該識別字即具有廣域範圍。具有廣域範圍的識別字如下：

- Net.Data 的內建函數
- 表格頁面資料
- URL 資料
- 可自 HTML 區塊中被個案化的變數

- 巨集檔

當識別字的宣告是出現在任一區塊的外頭時，則該 ID 的範圍即屬此種。區塊是以一個 "{" 做為開頭，而以 "%}" 做結尾。(請注意：本定義中不包括 DEFINE 區塊，且 DEFINE 區塊乃被視為獨立的 DEFINE 陳述式。)帶有巨集檔範圍的識別字，從其被宣告處一直到巨集檔尾端都是在可見的狀態。

- 函數區塊

當識別字的宣告動作或個案化動作是位在 FUNCTION 區塊(如 OUT 變數)時，則該識別字的範圍即屬此種。(此項亦適用於：在函數中之 REPORT 或 ROW 區塊內被參照，但卻未明確地定義在巨集檔中之變數，或者由 Net.Data 隱含定義的變數)。

- 報表區塊

當識別字僅能參照到 REPORT 區塊中被參照(例如：表格直欄名稱 N1, N2, ..., Nn) 則該識別字範圍即屬此種。...n)。只有 Net.Data 隱含定義成其表格處理之一部份的變數，才能具有報表區塊範圍。而其它任何立即可用的變數，則屬於函數區塊範圍。

- 橫列區塊

一項對識別字的參照時，會使此識別字被換成識別字的值。若對變數的參照找不到其相關的值，或者對該函數呼叫沒有回覆值，則該參照會被置換成一個空字串。

下節是說明如何定義及參照變數，並說明如何使用各種不同的變數類型其使用方式。

定義變數

定義 Net 巨集中變數的方法有下列 3 種：

- DEFINE 陳述式或區塊

在 Net 巨集中定義所用變數之最簡單方式就是使用 DEFINE 陳述式。此為 Net.Data 中的特殊語法：

```
%DEFINE variable_name="variable value"

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

variable_name 是您指定給變數的名稱。變數名稱必須以一個字母或底線符號開始，並且可以包括任何的英數字字元或是底線符號。所有的變數名稱有大小寫之區分，但 N_columnName 與 V_columnName 除外(這兩種屬於表格變數)。

請使用連續兩個雙引號來併入字串中的引號。只鍵入兩個連續的引號等於空字串。

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

當顯示時，變數 HI 會讀取 *say "hello"*，變數reply會讀取 *hello*，而變數 empty 則為空值 (null)。

要以一個DEFINE陳述式定義數個變數時，請使用一個 DEFINE 區塊：

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

- HTML 表格頁面之 SELECT 與 INPUT 標籤

此範例使用標準 HTML 表格頁面標籤以定義一個變數：

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

或：

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

此處的 *variable_name* 是指定您給變數的名稱，而變數之值視表格頁面中所收到的輸入而定。有關如何在 Net.Data 巨集中使用此種變數定義類型的範例說明，請參閱 第12頁的『HTML 表格頁面』。

自 INPUT 或 SELECT 陳述式中接收而來的變數值，會蓋掉 Net 巨集中 DEFINE 所設定的變數值。

- URL 資料

您可以將 Net.Data 巨集當成 URL 來呼叫，並且在 URL 中併入變數來傳給 Net.Data，像是：使用者 ID。例如：

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

在本例中，Net.Data 是以接收表格頁面資料般的方式來接收此額外的資料，*field=custno*，並以同一方式進行處理。

參照變數

在 Net.Data 巨集中，變數的參照方式是以在 \$(與) 內指定變數名稱的方式來進行的 (除了僅使用變數名稱的 IF 區塊以外)。例如：

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

當 Net.Data 發現一項變數參照，該變數參照會被替換成相對應之值。不接受循環參照。例如，不可以使用下列的 DEFINE 陳述式，否則在參照變數及計算最終的值時，會出現錯誤：

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

您可以將變數視為您 HTML 的一部份。例如，當您將變數 homeURL 如下般定義時：

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

則會將首頁參照為 `$(homeURL)`，並以下列方式建立錨點參照：

```
<A href="$(homeURL)">首頁</A>
```

您可以在 Net 巨集的任何一部份中參照變數。當變數在被參照時若尚未定義好，則 Net.Data 會定義該變數，並先給予一個起始值 `null`。在解析 Net 巨集時若有發現變數參照，則加以計算，並再線內將其置換成該變數的現行值。

條件變數

條件變數是用來確定是否有一個不是為空值得變數存在。若是它存在，則將第一個值指定給此變數；否則，則將第二個值指定給它。條件變數的語法為：

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

若有定義 `varB`，則 `varA="value_1"`，否則，`varA="" value_2"`。這與等於下列使用 IF 區塊的範例相同：

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

有關與列示變數一起使用條件變數的範例說明，請參閱 第23頁的『列示變數』。

環境變數

您可以參照執行 Net.Data 之處理內的 Net.Data 環境變數，例如：

```
The client is @DTW_rGETENV("SERVER_NAME")
```

其輸出結果如下：

```
The HTTPD server is IBM Internet Connection Server/4.1
```

有關 `@DTW_GETENV` 與 `@DTW_rGETENV` 函數的進一步說明，請參閱 *Net.Data 參考手冊*。

執行變數

透過執行變數特性，可讓您從一個變數參照呼叫其他程式。Net.Data 巨集是以下列方式來定義執行變數：

```
%DEFINE runit=%exec "testProg"
```

Net.Data 會根據 Net.Data 起始設定檔中的 `EXEC_PATH`，來尋找可執行程式。有關區域訊息區塊的另一範例說明，請參閱 第6頁的『EXEC_PATH』。

在 Net.Data 巨集中對變數 `runit` 發出有效的變數參照時，即會執行程式 `testProg`。在下列的簡例中，是從另一個變數定義來參照一個可執行變數：

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

不論 `$(dateRpt)` 是出現在 Net.Data 巨集的哪一處，Net.Data 都會傳回：

```
Today is Tue 11-07-1995
```

執行變數永遠不會被設定其所呼叫之執行程式的輸出值。以上述範例來說，日期的值為空值。若您在 DTW_ASSIGN 函數呼叫中使用它，以將它的值給指定另一個變數，則在做了這項指定之後，新變數的值也是為空值。執行變數的唯一目的是用來呼叫其所定義的程式。您也可以藉由在變數定義中指定參數及程式名稱，然後將參數傳給要執行的程式。在本例中，是將 distance 和 time 的值，傳給程式 calcMPH。

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

下面範例是將系統日期當成 HTML 報表的一部份傳回：

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

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

%HTML(report){
<H1>報表製作: $(tstamp) </H1>
@myQuery()
%}
```

每一個報表都有日期，以便於追蹤。在本例中，亦會替另一個 Net.Data 巨集，在錨點參照中加入客戶編號和名稱。您只要按一下報表中的任何一個客戶，即可呼叫 exmp.mac Net.Data 巨集，並將編號和名稱傳給 Net.Data 巨集。

隱藏變數

您可以使用隱藏變數，避免他人在用 Web 瀏覽器檢視您的 HTML 原始檔案時看到實際的變數名稱。

- 定義每一個要隱藏之字串的變數。
- 在參照變數的 HTML 區塊中以雙貨幣符號取代單貨幣符號，即可參照變數。例如，用 \$\$ (X) 取代 \$(X)。

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

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

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

·
·
·

當 Web 瀏覽器顯示 HTML 表格頁面時，分別用 `$(name)` 與 `$(addr)` 來取代 `$(name)` 與 `$(addr)`，讓實際的表格和直欄名稱不顯示在 HTML 表格頁面上，以便使人看不出還有真正的變數名稱。當客戶提出表格頁面時，即會呼叫 `HTML(REPORT)` 區塊。當 `@mySelect()` 呼叫 `FUNCTION` 區塊時，SQL 陳述式中的 `$(Field)` 會被替換成 SQL 查詢中的 `customer.name` 或 `customer.addr`。

列示變數

列示變數可讓您建立一串以分隔字元隔開的值，幫助您建立多項目的 SQL 查詢，就像某些 `WHERE` 或 `HAVING` 子句一般。列示變數語法如下：

```
%LIST " value_separator " variable_name
```

空白是具有其意義的。在大部份的情況下，我們建議在值分隔字元的前後各加上一個空格。大部份的查詢會使用布林或算術運算子 (例如，`AND`、`OR` 或 `>`) 做為值分隔字元。這個範例舉例說明如何使用條件、隱藏、以及列示變數：

```
%HTML(INPUT){  
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/example2.mac/report">  
<H2>選取一個或多個城市：</H2>  
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond1)">Sao Paolo<BR>  
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond2)">Seattle<BR>  
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond3)">Shanghai<BR>  
<INPUT TYPE="submit" VALUE="Submit Query">  
</FORM>  
%}  
  
%DEFINE{  
  DATABASE="custcity"  
  %LIST " OR " conditions  
  cond1="Sao Paolo"  
  cond2="Seattle"  
  cond3="Shanghai"  
  whereClause=Conditions ? "WHERE $(conditions)" : ""  
%}  
  
%FUNCTION(DTW_SQL) mySelect(){  
  SELECT name, city FROM citylist  
  $(whereClause)  
%}  
  
%HTML(REPORT){  
  @mySelect()  
%}
```

在 HTML 表格頁面中，若未勾選出任何勾選框，則 `conditions` 為空值，也因此查詢中，`whereClause` 也為空值。否則，`whereClause` 會有一些選定的值，且值與值之間以 `OR` 隔開。例如，若三個城市全選，則 SQL 查詢會是：

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

在此圖中，乃選取了 `Seattle`，而在此 SQL 查詢中產生的結果如下：

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

表格變數

表格變數可以定義一套相關資料。它裡面含有相同的記錄或橫列的陣列，以及說明每一橫列中各欄位之直欄名稱的陣列。在 `Net.Data` 巨集中可用如下列陳述式的方式定義表格：

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

`TABLE` 後面的數字，是這個表格可以包含的列數。如果您想指定一個沒有列數限制的表格，您可以如下列範例所示，使用預設值或指定 `ALL`：

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

您可以參照表格變數名稱，在函數之間傳遞表格。表格中的各個元素可在函數中的 `REPORT` 區塊中加以參照。表格變數通常用於 `SQL` 函數輸出的以及至報表的輸入，但也可以把它們當作是 `IN`、`OUT`、`INOUT` 參數，傳給任何非 `SQL` 函數。表格只能當作 `OUT` 參數傳給 `SQL` 函數。

表格中的直欄名稱和欄位值都被當作為陣列元素，而且其以 1 開始，不同於標準 `C` 和 `C++` 語言慣例以 0 開始的陣列。

隱含變數

定義表格變數可讓 `Net.Data` 隱含定義二組變數，供您參照表格的直欄名稱和欄位內容。這兩組隱含變數中的一組，會在 `Net.Data` 巨集中 `FUNCTION` 區塊的 `REPORT` 區塊內被加以參照，另一組則在自語言環境所呼叫的程式中被參照。在 `Net.Data` 巨集的其他區塊中，無法參照這些變數。

表 1. 報表變數

<code>N1, N2, ..., Nj</code>	為 j^{th} 直欄的名稱。
<code>N_columnName</code>	為 <code>columnName</code> 之值。
<code>NLIST</code>	為 <code>N1</code> 到 <code>Nj</code> 所有連續直欄名稱。
<code>V1, V2, ..., Vj</code>	包含現行橫列中 j^{th} 直欄之值。
<code>V_columnName</code> 之值	包含現行橫列中 j^{th} 直欄之值。
<code>VLIST</code>	為現行橫列中 <code>V1</code> 到 <code>Vj</code> 所有連續欄位值 j 的結合。
<code>ROW_NUM</code>	包含現行橫列之列號。
<code>NUM_COLUMNS</code>	包含表格的直欄數。
<code>TOTAL_ROWS</code>	包含表格的橫列數。

Net.Data 巨集的函數

`Net.Data` 中含有許多函數，而這些函數對您的 `Web` 應用程式很有幫助。除此您也可以輕鬆地撰寫您自己的函數。

定義函數

您可以定義自己的函數，或者使用 `Net.Data` 的函數程式庫。如果您在 `Net.Data` 程式庫中找不到所要的函數，可使用 `FUNCTION` 區塊，將該函數定義到 `Net.Data` 巨集中。詳細說明請參閱 *Net.Data 語言環境手冊*。語法如下：


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

type

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

Net.Data 會提供幾個預設的語言環境。

function-name

此為 FUNCTION 區塊的名稱。在 FUNCTION 區塊的執行上，是藉由在 Net.Data 巨集中參照位於 at (@) 符號後面的此名稱來執行的。執行一個 FUNCTION 區塊可以說是一個函數呼叫。詳細說明請參閱「呼叫函數」。

可有多個 FUNCTION 區塊共用同一名稱。但參數列示必須相同。在呼叫函數時，即按照定義在 Net.Data 巨集中的順序，來執行所有同名的 FUNCTION 區塊。

usage 有 IN, OUT, 或 INOUT。這表示要將參數傳送到 FUNCTION 區塊，或從 FUNCTION 處回收，或兩者皆可。選出的用法類型，會套用在參數列示其後所有的參數上，一直到變成另一種用法類型為止。預設類型為 IN。

parameter

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

您也可以在您指定類型的函數呼叫上，傳遞隱含參數。不過，必須在起始設定檔的 ENVIRONMENT 陳述式中定義這些參數。

return-var

在 RETURNS 關鍵字後指定此參數。其代表某個特殊的 OUT 參數。在 Net.Data 巨集的處理上，指派給函數呼叫之 return 變數的值，可以替代函數呼叫。不指定 RETURNS 子句時，如果呼叫語言環境的回覆碼為 0，則函數呼叫值為空字串，否則就是回覆碼值。

executable-statements

在替代完變數並處理函數後，會將這些陳述式傳給您所指定的語言環境供其執行。每一個語言環境以不同的方式處理陳述式。有關如何指定可執行的陳述式、如何呼叫可執行程式的詳細說明，請參閱「執行變數」。第21頁的『執行變數』。

report-block

請參閱「報表區塊」。第32頁的『報表區塊』。

message-block

請參閱「訊息區塊」。第27頁的『訊息區塊』。

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

呼叫函數

在 at (@) 字元後面加上 FUNCTION 區塊名稱，即可從 Net.Data 巨集呼叫函數：

@function_name([argument,...])

function_name

此為要呼叫之 FUNCTION 區塊的名稱。除非該函數為內建函數，否則此函數必須已定義在 Net.Data 巨集中。

argument

為一個已定義的變數名稱或一個文字字串。函數呼叫中的引數必須搭配 FUNCTION 區塊中的參數，每一個參數在 FUNCTION 區塊期間是被指定為相對應的引數值。引數的號碼和類型，必須與相對應參數的相同。

呼叫 FUNCTION 函數後，Net.Data 的處理方式如下：

1. Net.Data 讓函數呼叫上的所有引數，與 FUNCTION 區塊中的參數相對應。如果變數的號碼或使用類型不符，會出現錯誤。
2. 此組變數是建自 FUNCTION 區塊中所設的全部函數參數，與起始設定檔中之 ENVIRONMENT 陳述式中所設的引數。在此步驟中，此處有兩組特殊的變數：一個是廣域組，是由目前已定義的全部變數組成；一組是區域組，僅建給函數使用。有關廣域變數的討論，請參閱 第18頁的『Net.Data 巨集的變數』。
3. 將 FUNCTION 區塊執行檔文字中的變數參照換成變數實際值時，是先尋找區域變數組中的變數，然後再到廣域變數組中尋找。如果二組都有這個變數 (例如，指定為函數參數以及在前一個 DEFINE 中)，則以區域組優先。
4. 會處理 FUNCTION 區塊中之執行檔文字的函數呼叫。執行檔文字中的函數呼叫的內容與 Net.Data 無關。例如，如果在執行檔文字中，您的函數呼叫擁有條件性邏輯，則即使 Net.Data 並未遇到您所設的條件情況，其依然會處理函數呼叫。
5. 區域組的變數與可執行的陳述式文字一併傳送至語言環境。語言環境負責將 IN 和 INOUT 變數傳送至語言處理器、解譯可執行的陳述式或呼叫語言處理器執行陳述式、以及在程式完成時取回語言處理器傳回的任何 OUT 或 INOUT。
6. 語言環境返回 Net.Data 後，立即取得參數列示中的任何 OUT 或 INOUT 參數，用它們的值來替代區域及廣域變數組中的相對應引數值。如果有 RETURNS 子句，則會將傳回的變數值與 FUNCTION 區塊資訊一併儲存，以便可以用它來取代 Web 巨集表示式中的函數呼叫。

在函數呼叫 FUNCTION 區塊中使用以及修改變數的主要規則如下：

- 所有的變數 (包括廣域定義的變數與函數參數) 會在呼叫函數前，被用來在可執行的陳述式上執行變數替代工作。有關廣域變數的討論，請參閱 "Net.Data巨集的變數"。第18頁的『Net.Data 巨集的變數』。
- 只有 IN 或 INOUT 參數才會傳送至語言環境。
- 函數只能修改 OUT 或 INOUT 參數。
- 所有的變數 (包括廣義變數與函數參數) 會在呼叫函數後，被用來在 FUNCTION 區塊之報表區塊上執行變數替代工作。

當 MESSAGE 區塊和 REPORT 區塊處理完後，會用函數呼叫值來替代 Net.Data 巨集中的函數呼叫。

呼叫儲存程序

儲存程序的呼叫與 SQL 函數相同。儲存程序會將編譯過的 SQL 陳述式和資料庫伺服器，因此在執行效能和整合性上較高。

儲存程序可在大部份的平台上使用這些資料類型：

表 2. 儲存程序的資料類型

BLOB	DOUBLEPRECISION	SMALLINT
CHAR	FLOAT	TIME
CLOB	INTEGER	TIMESTAMP
DATE	GRAPHIC	VARCHAR
DBCLOB	LONGVARCHAR	VARGRAPHIC
DOUBLE	LONGVARGRAPHIC	

有關這些資料類型的說明，請參閱您的資料庫文件。Net.Data 未必能支援您資料庫所支援的各個資料類型。詳細說明請參閱 *Net.Data 參考手冊* 中的附錄。

以下是舉例說明，如何以函數名稱 `stored_proc1` 來呼叫儲存程序: `stored_proc1`:

```
%FUNCTION(DTW_SQL) stored_proc1 ( IN float(7,2) arg1,
                                   INOUT SMALLINT arg2,
                                   OUT VARCHAR(9) retval)
                                   RETURNS (RESULT) {

CALL statsrpt
%}

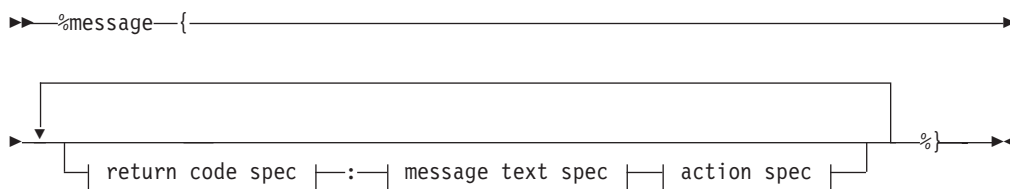
%HTML(REPORT) {
@stored_proc1(arg1, arg2, retval)
.
.
.
%}
```

在上例中，儲存程序的名稱為 `statsrpt`，其是以 `CALL` 陳述式來呼叫的。函數 `stored_proc1` 則為用來呼叫儲存程序的函數名稱。

訊息區塊

MESSAGE 區塊可讓您決定在順利完成 (或未順利完成) 函數呼叫後，要如何繼續處理，並且可讓您顯示相關資訊給函數呼叫者。

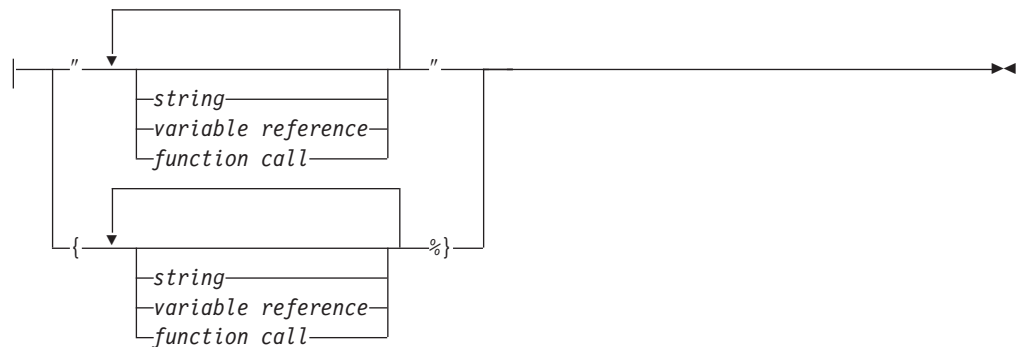
Net.Data 會為每一個函數呼叫，設定一個隱含變數 `RETURN_CODE`。`RETURN_CODE` 乃為呼叫函數所呼叫之語言環境的回覆碼。當函數呼叫完成後，MESSAGE 區塊會透過 `RETURN_CODE` 值來判斷該如何繼續處理。MESSAGE 區塊是由一系列的訊息陳述式所組成，每個陳述式都設有回覆碼值、訊息文字與要執行的動作。MESSAGE 區塊的語法如下：



return code spec



message text spec



action space



%message

爲這個區段的關鍵字，其中定義了一組回覆碼、相關的訊息、以及當傳回函數呼叫時 `Net.Data` 所採取的動作。

return_code spec:

一個正整數或負整數。若 `Net.Data RETURN_CODE` 變數之值，符合 *return_code spec* 之值，則會利用訊息陳述式中剩下的資訊，來處理函數呼叫。您也可以針對未特別輸入進 `$MESSAGE` 區塊中的回覆碼，指定訊息。

DEFAULT

此關鍵字是表示要使用預設的訊息碼。若 `RETURN_CODE` 不爲 0，並且未指定完全相符時，則此訊息陳述式中的資訊將用來處理此函數呼叫。

+DEFAULT

此關鍵字是表示要使用預設的負訊息碼。若 `RETURN_CODE` 小於 0，且未指定要完全相符時，則會使用此訊息陳述式中的資訊來處理函數呼叫。

-DEFAULT

此關鍵字是表示要使用預設的正訊息碼。若 `RETURN_CODE` 大於 0，且未指定要完全相符，且未指定 `+DEFAULT` (`RETURN_CODE` 大於 0) 或 `-DEFAULT` (`RETURN_CODE` 小於 0) 之值時，則會使用此訊息陳述式中的資訊，來處理函數呼叫。

number

指出可在處理期間發生之錯誤或警告的訊息碼。爲一串從 0 到 9 的數字。

message text spec

要傳送給 Web 瀏覽器的字串。當 `RETURN_CODE` 符合此訊息之訊息陳述式中的 *return_code* 值時，即會傳送此字串。

string

此字串中可使用任何英文字母、數字、與標點符號，但不能使用定位字元 (tabulator)、換行字元、或空格。

variable reference

會傳回先前定義的變數值，並且會在參數上加入 \$ 與 ()。例如：若 VAR = 'abc'，則 \$(VAR) 會傳回值 'abc'。

function call

以所指定的引數來呼叫一或多個先前所定義的 FUNCTION 區塊。

action spec:

指出當 RETURN_CODE 符合此訊息之訊息陳述式中的 *return_code* 值時，Net.Data 要採取什麼動作。

EXIT

此關鍵字是指出當發生對應到所指訊息碼的錯誤或警告時，即馬上跳出巨集。

CONTINUE

此關鍵字是指出當發生對應到所指訊息碼的錯誤或警告時，仍繼續處理。

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

Net.Data 會採用下列的規則，來處理從某函數呼叫所傳回的 RETURN_CODE：

1. 檢查區域 MESSAGE 區塊是否有完全相符的；然後根據所指定的，看是要跳出或繼續。
2. 若 RETURN_CODE 不為 0，則檢查區域 MESSAGE 區塊是否為 +default 或 -default (視 RETURN_CODE 的符號而定)，然後根據所指定的，看是要跳出或繼續。
3. 若 RETURN_CODE 不為 0，則檢查區域 MESSAGE 區塊為 default，然後根據所指定的，看是跳出或繼續。
4. 檢查廣域 MESSAGE 區塊是否有完全相符的；然後根據所指定的，看是要跳出或繼續。
5. 若 RETURN_CODE 不為 0，則檢查廣域 MESSAGE 區塊是否為 +default 或 -default (視 RETURN_CODE 的符號而定)，然後根據所指定的，看是要跳出或繼續。
6. 若 RETURN_CODE 不為 0，則檢查廣域 MESSAGE 區塊是否為 default，然後根據所指定的，看是跳出或繼續。
7. 若 RETURN_CODE 不為 0，則發出 Net.Data 內部預設的訊息並跳出。

在下列範例中，為部份 Net.Data 巨集，其中帶有一個廣域的 MESSAGE 區塊，與一個函數的 MESSAGE 區塊。

```
%{ global message block %}
%MESSAGE {
    -100      : "回覆碼 -100 的訊息"      : exit
    100       : "回覆碼 100 的訊息"       : continue
    +default : {
此為超出一行的長訊息。
您可以在此訊息中使用包括錨點和表格頁面的 HTML 標籤。%} : continue
```

```

%}

%{ 區域訊息區塊位於 FUNCTION 區塊中 %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE {
    -100      : "回覆碼 -100 的訊息"      : exit
    100       : "回覆碼 100 的訊息"       : continue
    -default : {
此為超出一行的長訊息。
您可以在此訊息中使用包括錨點和表格頁面的
HTML 標籤。%}      : exit
  %}

```

當 *my_function()* 傳回一個 RETURN_CODE 設為 50 的回覆碼時，Net.Data 會以下列順序來處理錯誤：

1. 檢查區域 MESSAGE 區塊中是否有完全相符的。(此處無。)
2. 檢查區域 MESSAGE 區塊中是否有 +default。(此處無。)
3. 檢查區域 MESSAGE 區塊中是否有 default。(此處無。)
4. 檢查廣域 MESSAGE 區塊中是否有完全相符的。(此處無。)
5. 檢查廣域 MESSAGE 區塊中是否有 +default。(此處有一個。)

此時 Net.Data 乃找到一個相符項，其會將訊息文字送給 Web 瀏覽器，並檢查所要求的動作。由於所設的是 continue (繼續)，因此 Net.Data 會在印出訊息文字後繼續處理 Net.Data 巨集。

例如，當巨集呼叫五次 *my_functions()*，且在處理範例中之 MESSAGE 區塊時發現了錯誤 100，則該程式的輸出乃類似如下：

```

.
.
.
11 May 1997                $245.45
13 May 1997                $623.23
19 May 1997                $ 83.02
return code 100 message
22 May 1997                $ 42.67

Total:                     $994.37

```

有關區域訊息區塊的其他範例，請參閱附錄A."動態查詢樣本"。第49頁的『附錄A. 動態查詢範例』。

在巨集中產生 HTML

Net.Data 可讓您輕易地將標準 HTML，呈現在應用程式使用者的瀏覽器上。下面的區塊中將介紹如何在 Net.Data 巨集中製作 HTML 的格式。

HTML 區塊

HTML 區塊與 HTML 區塊所呼叫的函數，都是 Net.Data 巨集的區塊，其可產生的 HTML 輸出至瀏覽器上。不論在何時呼叫 Net.Data 時，都必須指定 HTML 區塊。此區塊中的內容，會控制 Net.Data 的其餘呼叫作業。

任何有效的 HTML 都可出現在一個 HTML 區塊中。另外，INCLUDE 陳述式、函數呼叫、與變數參照，都可放在 HTML 區塊中。在下列的 Net.Data 巨集範例中，是說明 HTML 區塊在 Net.Data 巨集中的一般用法：

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT){
<H1>硬體查詢表格頁面</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/equiplst.mac/report">
<dl>
<dt>您要列出什麼硬體?
<dd><input type="radio" name="hardware" value="MON" checked> 監視器
<dd><input type="radio" name="hardware" value="PNT">指標裝置
<dd><input type="radio" name="hardware" value="PRT">印表機
<dd><input type="radio" name="hardware" value="SCN">掃描器
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}

%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hardware)
%REPORT{
<B>此為您要的列示:</B><BR>
%ROW{
<HR>
$(N1): $(V1)      $(N2): $(V2)
<P>
$(V3)
%}
%}
%}

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

您可以從類似下面的錨點參照中來呼叫 Net.Data 巨集：

```
<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.mac/input">硬體的列示</a>
```

當應用程式使用者按一下此參照時，即可呼叫 Net.Data 來解析 Net.Data 巨集檔。進入呼叫時指定的 HTML 區塊中後，在本例中為 HTML(INPUT) 區塊，就會開始處理區塊中的文字。當 Net.Data 無法從任何資料中認出 Net.Data 巨集語言結構時，會將之視為 HTML，而送往瀏覽器顯示出。

當您做好選擇並按「提出」按鈕後，會開始執行 HTML FORM 元素的 ACTION 部份，其是指去呼叫 Net.Data 巨集的 HTML(REPORT) 區塊。之後，HTML(REPORT) 區塊會以 HTML(INPUT) 區塊的方式處理。queryHardware() 函數呼叫之前的所有資料，都會當成 HTML 輸出至瀏覽器上。

然後處理 queryHardware() 函數呼叫，它會傳回呼叫 SQL FUNCTION 區塊。在 SQL 陳述式中的 \$(hardware) 變數參照，換成輸入格式中傳回的值後，即開始查詢。此時，Net.Data 會再度將 HTML 傳給瀏覽器，並根據 REPORT 區塊中指定的 HTML 來顯示查詢結果。

在 REPORT 區塊處理完後，會再回到 HTML(REPORT) 區塊上，將 @queryHardware() 函數呼叫後面指定的其餘 HTML 傳送出去，以完成處理。

每一個 Net.Data 呼叫，只處理一個 HTML 區塊。不過，應用程式的使用者如果使用 HTML 錨點參照和表格頁面，即可輕易地在另一個 HTML 區塊上，呼叫另一個 Net.Data，且完全由使用者控制。

報表區塊

REPORT 區塊可用來顯示 FUNCTION 區塊所輸出的資料並加以格式化。雖然您所指定的可能是 HTML 標籤、巨集變數參照與函數呼叫等任何有效的組合，但一般來說，此種輸出類型通常是表格資料。您可以選擇在 REPORT 區塊中指定表格名稱（不一定得如此做）。如果您未指定表格名稱，則所用的表格資料會是此 FUNCTION 區塊之參數列示中表格的資料。若 FUNCTION 區塊中未指定任何表格，則會採用預設的表格資料。

REPORT 區塊是由三個可選用的部份所組成：

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

如果您不想顯示 ROW 區塊所輸出的任何表格，僅需留白即可。

Net.Data 在處理 FUNCTION 區塊時，會呼叫語言環境並傳回資料。之後，Net.Data 會處理 REPORT 區塊。

在 REPORT 區塊中，Net.Data 會提供數個隱含的定義變數，讓您可存取到 Net.Data 巨集結果表格中的資料。有關這些變數的說明，請參閱表 1。第 24 頁的表 1。而進一步的詳細資訊，請參考參考手冊中的「報表變數」一節。

表頭和註腳資訊不像 REPORT 區塊般是以明示的方式來指定。Net.Data 乃簡單地將在 ROW 區塊之前找到的一切都當成表頭資料，之後找到的視為註腳資訊。在使用 HTML 區塊上，Net.Data 巨集處理器會將表頭、ROW、與註腳區塊中的一切，都視為 HTML，並將資料傳給瀏覽器。您也可以加進函數與變數。

為了避免產生報表，請省略 REPORT 區塊，並將 DTW_DEFAULT_REPORT 設為 NO。若您將之設為 YES，則會以預設格式顯示報表，例如：

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

當您將 DTW_HTML_TABLE 設為 YES 時，會採用 HTML 表格標籤，而不採用替預設報表事先格式化的文字。

下面的範例是說明如何使用特殊變數及 HTML 標籤，來自行設定報表格式。其會自表格 CustomerTbl 中，顯示名稱、電話號碼、以及傳真號碼：

```
%FUNCTION(DTW_SQL) custlist() {  
    SELECT Name, Phone, Fax FROM CustomerTbl  
    %REPORT{  
<I>電話查詢結果:</I>  
<BR>  
=====
```

Name: \$(V1)
Phone: \$(V2)
Fax: \$(V3)

```
-----  
}
```

```

%}
取回的總記錄數：$(ROW_NUM)
%}
%}

```

查詢報表在瀏覽器中看來如下：

電話查詢結果：

=====

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

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

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

取回的總記錄數：3

Net.Data 會以下列方式來產生報表：

1. 在報表起頭處列印一次電話查詢結果：。
2. 在取回每一列時，分別替名稱、電話及傳真，給予 V1, V2, 與 V3 的變數值。
3. 在取回每一列之後，就繪製一條直線，以便閱讀。
4. 在報表尾端，印出字串取回的總記錄數：與 ROW_NUM 的值一次。

使用大型物件

DB2 已可在越來越多的平台上支援大型物件 (LOB)。若要知道其是否支援 LOB，請參考您的 DB2 文件。DB2 可支援下列三種類型的LOB：

- 二進位大型物件 (BLOB)
- 字元大型物件 (CLOB)
- 二位元組大型物件 (DBLOB)

當查詢傳回一個 LOB 時，Net.Data 會將之儲存到 HTML_PATH 架構變數指定的目錄中。由於 LOB 會快速消耗資源，所以在使用 LOB 時，需考慮到系統限制問題。某些 LOB，如音效檔，需用到特殊的硬體與軟體。

LOB 的前幾個位元組中通常有一個檔案簽名，指出檔案所含的資訊類型為何。若 Net.Data 認得 LOB，就會將副檔名新增至暫存檔和代表其名稱的 Net.Data 巨集變數中。若沒有 REPORT 區塊，CLOB 會為其加上 .txt 副檔名。以下是 Net.Data 所認得的 LOB 格式：

- 位元圖形 (.bmp)
- 圖形壓縮檔格式 (.gif)
- 標籤壓縮檔格式 (.tif)
- Postscript (.ps) 這些必須儲存為 BLOB 而非 CLOB。

而其他檔案類型 Net.Data 都不認得，也不支援。任何大型物件都不支援 UPDATE 和 INSERT SQL 陳述式。在下面的第一個範例中，是直接顯示圖片。在第二個範例中，應用程式使用者必須按一下檔名來呼叫檢查器。


```
<IMG SRC="/tmplobs/filename">
<A HREF="/tmplobs/filename">filename</A>
```

此範例是說明如何在應用程式中使用 .WAV 檔。 Net.Data 不認得此種檔案類型，因此會使用 EXEC 變數，將副檔名附加至檔案中。

```
%DEFINE{
docroot="/usr/lpp/internet/server_root/html"
rename=%EXEC "rename $(docroot)$(V3) $(docroot)$(V3).wav"
%}

%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
%REPORT{
<P>此為您所選的壓縮檔：<P>
%ROW{
$(rename)
$(V1) Voice sample <IMG SRC="$(V2)">
<A HREF="$(V3.wav)">聲音樣本</A><P>
%}
%}
%}

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

此 queryData 函數會傳回下列的 HTML：

```
<P>此為您所選的壓縮檔：<P>
Kinson Yamamoto
<IMG SRC="/tmplobs/p2345n1.gif">
<A HREF="/tmplobs/p2345n2.wav">聲音樣本</A><P>
Merilee Lau
<IMG SRC="/tmplobs/p2345n3.gif">
<A HREF="/tmplobs/p2345n4.wav">聲音樣本</A><P>
```

此 REPORT 區塊會使用隱含的表格變數 V1、V2、與 V3。

- V1 代表個人的名稱，為純文字。
- V2 代表 .GIF 格式的個人照片檔。此圖片會以列入方式顯示。 Net.Data 會自動併入 LOB 暫時目錄和 .GIF 副檔名。
- V3 是 .WAV 格式的個人聲音樣本檔。當 Net.Data 遇到不認得的檔案格式時，(如：.WAV 檔)，其會將檔案寫入 LOB 暫時目錄中，而不加上副檔名。此範例是說明在處理此種檔案類型時，該如何使用 EXEC 變數來新增副檔名。當解析出變數 \$(V3) 後，其會在檔案名稱前，加進路徑 /tmplobs/。例如，/tmplobs/sound2a。處理此類檔案的方法之一，是寫入 REXX 程式把斜線改成反斜線後，並且變更檔案的名稱。當應用程式的使用者按一下「聲音樣本」時，就會播放聲音樣本。

並非所有的 Web 瀏覽程式都支援圖形及聲音。您可能需用到特殊的軟體及硬體，例如音效卡與驅動常式，才能支援此處所說的功能。

第6章 使用內建函數

Net.Data 乃提供一大組內建函數，可幫助您簡化 Web 頁面的開發。這些函數已經過 Net.Data 的定義，而您不必再用 FUNCTION 區塊來定義。您只要在巨集中可呼叫使用者定義函數的任何地方，呼叫這些函數即可。

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

- **DTW_**, **DTWF_**, 與 **DTWR_**: 以輸出參數傳回呼叫結果，或者未傳回任何結果。
DTWF_ 與 **DTWR_** 分別是純本文檔與 Web 登記函數。
- **DTW_r**, **DTWF_r**, 與 **DTWR_r**: 將結果傳給函數呼叫值。
- **DTW_m**: 在傳給函數的參數中傳回多個結果。

有些內建函數沒有所屬類型。有關詳細資訊，請參閱 *Net.Data* 參考手冊。

一般目的函數

此組函數可讓您改變資料或存取系統服務程式，藉此協助您開發 Web 網頁。您可以使用此組函數，來查詢與設定環境變數，來使用 HTML escape 碼，以及從系統中取得其他有用的資訊。

算術函數

這些函數會執行算術運算，協助您計算或改變數字性資料。除了標準的算數運算外，您也可以執行模數的除法，指定結果的精確度，以及使用科學記數法。

字串函數

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

字組函數

這些函數可讓您處理字串中的字組。這些函數中的絕大部份，乃與字串函數的作用相同，只不過其是以整個字組運作。這些函數可讓您：計算字串中的字組數，除去字組，從字串中找出某個字組，以及指定一些字組。

表格函數

這些函數可讓您用來操作 Net.Data 表格變數。在表格變數中，乃含有一個陣列的值，以及其相關的直欄名稱。這些函數方便您將整群的值傳給函數。

純本文檔介面

當您選擇以純本文檔做為您的資料來源，則您可以使用純本文檔介面 (FFI) 與其相關的 `Net.Data` 函數，來開啓、關閉、讀取、寫入、與刪除 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。

何謂 FFI ？

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。

安全注意事項

您可以指出 FFI 函數可用 `Net.Data` 起始設定檔中 `FFI_PATH` 陳述式，來存取的哪些檔案。FFI 只會搜尋陳述式中所列出的路徑，因此其他目錄中的檔案是安全的。以下為陳述式的範例說明：

```
FFI_PATH      C:\public;.\;E:\WWW;E:\guest;A:
```

會從頭到尾搜尋 `FFI_PATH` 中所列出的所有路徑。會採用第一個找到的拷貝。若起始設定檔中沒有 `FFI_PATH`，則 FFI 會試著在現行目錄或指定目錄中尋找檔案(例如：`../reports/nov96.txt`)。有關路徑搜尋的詳細說明，請參考「進階明細」。 `Net.Data` 起始設定檔中在出廠時沒有指定 `FFI_PATH`。

在您設定 FFI 之前，請先詳細規劃。請注意下列幾點：

- 選擇適用於純本文檔作業的目錄。這些目錄需新增至 `FFI_PATH` 中，以便限制只搜尋這些目錄。
- 當您讓他人巨集中執行 `DTWF_REMOVE` 或其他匯出作業時，必須特別注意，以免他人除去或變更現行目錄中副檔名為 `.dll` 與 `.cmd` 的檔案。
- 合理控制系統上新增的巨集，用適當的步驟來保護系統上的檔案。
- 不要在 `FFI_PATH` 中指定可讓匿名 FTP 使用者寫入的路徑。否則，任何人都可以在系統上放入一個 `Net.Data` 巨集，而執行了一些先前所不能接受的動作。
- 建議您最好不要在 `FFI_PATH` 中，加進 `Net.Data` 起始設定檔的路徑。

其他注意事項

一般注意事項

- 雖然您可以匯入任何的純文字檔，不過 `Net.Data` 巨集的語法是交由 `Net.Data` 來解譯的，而可能存在於文字中的 HTML 標籤，會被瀏覽器用來製作文字的格式。
- 只有在作業系統有區分大小寫時，FFI 參數才有大小寫之區分。

現行目錄

- `Net.Data` 的現行目錄通常是 `\www\cgi-bin`，不過此視您 Web 伺服器的架構而定。若伺服器預設的要求路徑或資源對映已被變更，則現行目錄可能也會跟著改變。

- 建議您可以用一個句點加上一條斜線，如：`.\` 來參照 `FFI_PATH` 中的現行目錄。當您在 `FILENAME` 參數中的檔名前面，加上句點和斜線時，則表示讓 `FFI` 只尋找現行目錄。就讀取作業來說，此表示不搜尋 `FFI_PATH` 中的其他路徑。

DELIMITER 參數

- 當 `FFI` 根據所要求的轉換，將檔案分成部份（像是將一橫列的直欄）時，其會使用旗號或分隔字元作為分隔字元。
- 就讀取作業來說，分隔字元會將檔案的內容分成表格的橫列與直欄。而在寫入作業方面，會將分隔字元放在檔案中，指出表格之橫列與直欄值的結尾。分隔字元是以 `Net.Data` 巨集字串形式傳給 `FFI`。除非在定界字元參數中明確列出，否則空字元不會放在字元的結尾。也就是說，如果您要在分隔字元中使用空字元，定界字元參數必須以 `"\0"`（一個反斜線和一個零，外面再括以雙引號）指定，而不是空字串 `""`（以兩個雙引號代表）。如果所指定的是 `ASCIITEXT` 轉換方式，則必須在分隔字元中使用換行字元，而不採用其他要求的定界字元。
- 如果寫入作業使用的定界字元與讀取作業先前所使用的不同，則檔案可能會產生您不希望看到的變更。如果寫入所用的分隔字元與讀取作業先前所用的不同，則該檔會採用新的分隔字元寫入。
- 定界字元的長度上限是 256 個字元。

FFI_PATH

- `FFI_PATH` 中的路徑必須包含有效的可列印字元。`FFI` 不容許路徑中含有 `?`（問號）或 `"`（雙引號）。
- 除非在 `FFI_PATH` 中有明確指定，否則 `FFI` 不會去搜尋 `FFI_PATH` 中所列路徑的次目錄。例如，`FFI` 不會將 `.\test`（位於 `FFI_PATH` 中）當做完全符合的 `test` 目錄（可能位於 `FILENAME` 參數的開頭）。因此，如果您要在 `test` 這個次目錄找到 `myfile.txt` 檔，路徑 `.\test` 必須放在 `FFI_PATH` 中，且 `FILENAME` 參數也得是 `.\test\myfile.txt` 或 `myfile.txt`。這和要求 `FILENAME` 參數中的 `test\myfile.txt`，並在 `FFI_PATH` 中加入現行目錄不同，此種做法將無法運作。
- 如果 `FFI_PATH` 中沒有明確指定現行目錄，則系統會先搜尋 `FFI_PATH` 中明確指出的路徑，最後才搜尋現行目錄。

DTWF_SEARCH 函數

- 針對 `DTWF_SEARCH` 所傳回的表格有三個直欄。頭兩個直欄是相符字元所在的欄號與列號，最後一直欄則是含有 `SearchFor` 參數中所指定之字元的直欄值。例如，如果檔案的第 4 列含有第 3 欄的相符字元，則傳回的表格，其中一列的第一欄為號碼 `"4"`，則表示其是出自於該檔的第 4 列；其第 2 欄為號碼 `"3"`，表示該檔的第 3 欄中含有相符的字元，而第 3 欄則是完整的直欄值。
- 在 `SearchFor` 參數中，不能含有分隔字元參數的內容。

STARTROW 和 ROWS 參數

- 對 `DTWF_DELETE`、`DTWF_INSERT`、`DTWF_UPDATE` 與 `DTWF_WRITE` 來說，若指定的 `StartRow` 值，比所指定的最後一列值還大，則 `StartRow` 會變更成代表最後一列，並傳回錯誤。
- 對 `DTWF_READ` 與 `DTWF_SEARCH` 來說，`Rows` 值在傳回時，乃代表表格中的列數。

TABLE 參數

- `FFI` 表格中一列的字元數上限為 16383 個。在此限制中，亦將 `Net.Data` 巨集表格中每個直欄的空字元計算在內。

TRANSFORM 參數

- 此參數是用來指出如何根據 Net.Data 巨集表格的欄列數，來分割檔案。例如，「ASCITEXT 轉換」是表示檔案的每一行都對應到 Net.Data 巨集表格的一列，而該 Net.Data 巨集表格只有一欄。「DELIMITED 轉換」是表示會對列中的字元做檢查以找到 DELIMITER,而 DELIMITER 之後為下一籃的內容。
- 檔案中的換行字元，會在 ASCITEXT 和 DELIMITED 轉換時指出 Net.Data 巨集表格之列的尾端。

檔案鎖定

- 除非您使用 DTWF_OPEN 來開啓檔案，否則不會鎖定檔案。當檔案未被鎖定时，該檔案可在讀取與更新期間變更。這可能會失去先前的變更。而當您使用 DTWF_OPEN 時，則會在執行巨集期間使用檔案系統的鎖定機制來開啓檔案。

DTWF_APPEND

- 檔案的現行內容，會對使用 DTWF_APPEND 後的結果造成影響，特別是最後一列之最後一欄的內容。如果該檔最後一列的最後一欄值後面是換行字元，則附加的資料會放到新的一列中。否則，附加的資料便會成為該檔最後一列的一部份。

Web 登記

「Net.Data Web 登記」適用於部份的平台上，其可提供永久的儲存空間給應用程式的相關資料存放。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。

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。這樣一來便很難更改到超鏈結。此種固定的方式，亦會限制了原本可輕易放在 Web 網頁的超鏈結類型。使用 Web 登記來儲存應用程式的相關資料 (如 URL)，有助您動態地設定超鏈結以建立 HTML 頁面。

資訊可由應用程式開發者和 web 管理者儲存在登記中，並加以維護，這些軟體開發者和 Web 管理者對於登記都具有寫入權。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，同時也可讓應用程式和伺服器移動。您可以利用動態設定的超鏈結，透過 Net.Data 巨集來建立 HTML 頁面。

資訊是以登記項目形式儲存在 Web 登記中。每一個註冊項目是由一對字串所組成：RegistryVariable 字串與對應的 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。該變數字串可作為搜尋關鍵字，來尋找和取回登記中的特定項目。

有關 Web 登記的範例內容可參閱表3。 第38頁的表 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

表 3. Web 登記範例 (繼續)

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URL)。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 字串後面會附帶加上索引字串，例如，"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.ibm.link.ibm.com
Smith/Home_page	http://www.advantis.com

雖說上述兩個索引化項目的 RegistryVariable 字串值皆為 "Smith"，但這兩個案例中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。

第7章 使用語言環境

Net.Data 會提供數種語言環境讓您在資料來源之間互傳資訊。例如：SQL 語言環境可讓您傳遞本地的 SQL 查詢給資料庫；REXX 語言環境可讓您呼叫 REXX 程式。

設計 Net.Data 的目的，是讓您能以「即插」的方式，來新增新的語言和資料庫介面。這些語言環境乃被當成動態鏈結程式庫 (DLL) 或共用程式庫 (獨立於 Net.Data 主可執行檔之外) 來存取。DLL 的名稱乃放在 Net.Data 的起始設定檔中，且各有一個相關的語言環境名稱。每一種語言環境都必須支援 Net.Data 所定義的標準介面組。Net.Data 會在第一次呼叫指定該語言環境的 FUNCTION 區塊時，載入起始設定檔中所指定的 DLL。

有關語言環境的完整說明，請參閱 *Net.Data 語言環境手冊* 。

第8章 提高執行效能

您可以下列幾種方式來提高執行效能：

- 儲存程序
- 審慎編寫程式
- 使用 API
- 使用現場連線 (live connection)

通常用來增加 *Net.Data* 執行效能的方法有二：使用現場連線(live connection)，以及使用 Web 伺服器應用程式設計介面 (API)，如：ICAPI、ISAPI、與 NSAPI。如果您想知道您平台所支援的為何，請參閱 *Net.Data* 參考手冊。

現場連線

某些伺服器可藉由現場連線，維持永久的資料庫連線來增強執行效能。有些 *Net.Data* 動作所需要的啟動時間很長。例如，處理必須先讓 DBMS 能認出自己，並連上資料庫後，才能發出資料庫查詢。這對要存取資料庫的 *Net.Data* 巨集來說，會花上很長的處理時間。另外，像 Java 虛擬機器的啟動費用也是很貴的，執行 Java 應用程式 (但非 Java applet)時需要使用它。由於 CGI 程式的作業方式所致，每次您提出要求至 Web 伺服器時，光是這些程式的啟動，就得花掉您一些成本。

現場連線可幫您省下這些額外的啟動時間，大大提升執行效能。省錢的原因是因為會有一項或多項專門執行啟動函數的處理持續進行著。這些處理然後會等著來服務的要求。只要您將 *Net.Data* 當成 CGI 程式，便可執行現場連線。若您所用的是 Web 伺服器 API，則您必須使用現場連線。

現場連線是由一個連線管理程式和一些 *cliette* 所組成的。*Cliette* 是單一緒的處理，當伺服器執行時，連線管理程式會啟動並維持在作用中狀態。*cliette* 會處理資料，並利用關鍵字 *CLIETTE*，和起始設定檔中所設的 *Net.Data* 語言環境通信。每種 *cliette* 類型都負責一種特定的後端處理函數，以 *DB2 cliette* 為例，其會連接資料庫，並且會設定一些作業，以便在 *Net.Data* 處理任何 *Net.Data* 巨集前，先執行 SQL 呼叫。此可執行檔是在架構檔 *dtwcm.cnf* 中加以指定。

現場連線的優點

使用現場連線的主要優點有三：

- **提高執行效能**

重複使用連線乃比建立新連線來得有效率。您必須判斷一下，和所有的處理時間相較，您花在連線時間上的費用是多少。一般說來，如果您所要求的 SQL 陳述式不大 (例如，列數在 100,000 以內的簡單資料庫查詢)，或者您的連線較困難 (例如，遠程伺服器)，則連線的時間將會較長。

- **多個資料庫存取**

現場連線可讓您在同一時間內，讓一個 *Net.Data* 巨集連接至多個資料庫。因為每個資料庫都有一個獨一無二的 *cliette*，因此，*Net.Data* 就可與多個 *cliette* 進行通信。

- **循序顯示**

連線管理程式具有處理 SELECT 陳述式和捨棄前 N 列的能力。如此一來，可讓您有能力使用有提供「下一個」與「前一個」選項 (用以使用者大型列示) 的應用程式。藉由設定變數 START_ROW_NUM 為 N，您可以去查詢以顯示可管理的一塊細向資訊，並讓應用程式的使用者按一下 NEXT 的按鈕來取回下一組的橫列。

我應該使用現場連線嗎？

若您選取使用 API (而不是使用 CGI) 來與您的資料庫通信，您就必須使用現場連線。若您的應用程式有下列需求，則使用現場連線可能較為有利：

- 您的應用程式需用到多個資料庫中的資料。
- 與 API 搭配使用的現場連線，可為不少系統提高執行效能 (視這些系統的載入與架構而定)，這是因為當從一個 URL 向 Web 伺服器提出要求時，不必建立新處理。然而，若是使用連線管理程式，則需用到額外的系統管理。您必須多次嘗試，才能決定出最適合您系統的架構。

有許多應用程式可在不須使用現場連線的情況下，增進執行效能，亦即，它們可透過 ACTIVATE DATABASE 或 START DATABASE 命令，來節省建立資料庫連線的時間。有關您資料庫所用命令的詳細說明，請參閱您的資料庫文件。同時，請翻閱您的平台文件，以瞭解是否有其他步驟可幫助執行效能的提升。

啟動連線管理程式

連線管理程式是一個獨立於 Net.Data 之外的可執行檔，其名稱為 dtwcm.exe。在您啟動 Web 伺服器時，您就應啟動它。當您啟動連線管理程式時，它會讀取架構檔，並且會建立一組負責處理資料的處理。在每個處理中，連線管理程式會開始執行特定的 cliette。

一般而言，只要資料庫、Web 伺服器和連線管理程式，都經過架構並處在執行中，一旦啓用了現場連線，Net.Data 處理便會執行下列這些步驟：

- Web 伺服器被呼叫並啟動一個 CGI 處理或一個 API 緒來執行 Net.Data。
- Net.Data 會開始處理 Net.Data 巨集。
- 當 Net.Data 遇到有函數呼叫使用現場連線時，其會從起始設定檔來判斷需要哪種 cliette 類型。如果是 DB2，cliette 類型通常是根據 DB2 資料庫名稱來命名。在本例中，cliette 名稱是 DTW_SQL:CELDIAL。Net.Data 會向連線管理程式要求此類型的 cliette。而連線管理程式即會去尋找這種類型的 cliette 是否有可用的。如果找不到，連線管理程式會將此項要求放進等待佇列中，等到有適當而可用的 cliette 類型時再行處理。如果有可用的 cliette，則連線管理程式會告知 Net.Data 如何與 cliette 通信。
- Net.Data 要求 cliette 去處理函數。
- 從步驟 3 開始重複執行此項處理，直到 Net.Data 巨集處理完成。
- 釋出所有的 cliette。

若起始設定檔中設有 cliette，而連線管理程式不在執行狀態，則 Net.Data 會載入 DLL 並處理巨集。若您使用了 API，則您可能會收到錯誤，而需啟動連線管理程式。

架構現場連線

現場連線會使用架構檔 dtwcm.cnf，來判斷需啟動哪些 cliette。本例中會舉一個架構檔來說明，在此架構檔中乃含有下列的區塊：

- 管理資訊

- DB2 連線的 SQL clette 資訊
- Java 應用程式 clette 資訊

命令行會附上號碼，以便參照：

```

1 CONNECTION MANAGER{
2   MAIN_PORT=7100
3   ADMIN_PORT1=7101
4   ADMIN_PORT2=7102
5 }
6
7 CLIETTE DTW_SQL:CELDIAL{
8   MIN_PROCESS=1
9   MAX_PROCESS=5
10  START_PRIVATE_PORT=7200
11  START_PUBLIC_PORT=7210
12  EXEC_NAME=./c1tdb2
13  DATABASE=CELDIAL
14  BINDFILE=/usr/... ./d2wsq1.bnd
15  LOGIN=marshall
16  PASSWORD=stlpwd
17 }
18
19 CLIETTE DTW_APPLET{
20   MIN_PROCESS=1
21   MAX_PROCESS=5
22   START_PRIVATE_PORT=7300
23   START_PUBLIC_PORT=7310
24   EXEC_NAME=./javaapp
25 }

```

1 - 5 行是架構檔所需之行。7 - 12 和 17 行是所有 SQL clette 所需之行。您可以加入其他資訊，例如，如果連接 DB2 資料庫，則可加入使用者 ID 和通行碼。這些額外之值會出現在 13 - 16 行。如果您使用 Java applet，則 19 - 25 行全都需要。

在 LOGIN 與 PASSWORD 變數方面，您可以指定使用預設值，也就是說，讓 Net.Data 使用同一使用者 ID，來啟動連線管理程式，以連接 DB2 資料庫。此舉可讓您避免將此資訊放進架構檔中。例如，將第 15 與 16 行換成：

```

LOGIN=*USE_DEFAULT
PASSWORD=*USE_DEFAULT

```

您必須決定您系統要使用的埠號。同時，您也必須指出 MIN_PROCESS 和 MAX_PROCESS 的值。當連線管理程式啟動時，即會啟動 MIN_PROCESS 所指的處理數。之後，當同時存在的處理抵達時，連線管理程式會啟動更多的 clette (若有需要的話)，直到到達 MAX 所指的上限值為止。您所使用的值，可能會影響執行效能，但是您稍後可以再做變更。

要使用現場連線時，您必須在 Net.Data 的起始設定檔中，於 DTW_SQL (可能亦含 DTW_ODBC) 一項納入 ENVIRONMENT 陳述式。詳細說明請參閱 *Net.Data 語言環境手冊*。

在您修改架構檔之前，請注意下列幾點：

- 連線管理程式所用的 Clette 名稱，必須是能夠獨一無二地識別出一組 clette。
- 如果是資料庫 clette，則對於您要存取之每一個資料庫來說，都必須有一組以指名名的 clette。至於很少存取的資料庫，您可將 clette 的 MIN 和 MAX 數設定為 1。或者，您也可將 MIN 設為 0，此表示直到 Net.Data 提出 clette 要求時，才建立處理。

- cliette 的 NAME 必須和起始設定檔中的名稱相符。此名稱是位在 DTW_SQL 環境陳述式的行尾。此名稱的格式為 CLIETTE "xxx"，此處的 xxx 代表這個名稱內建在 Net.Data 巨集檔中的方式。其中可包含變數。以 DB2 cliette 為例，則應該包含變數 \$(DATABASE)。預設值是 DTW_SQL:\$(DATABASE)。當遇到 SQL 區段時，\$(DATABASE) 會換成 DATABASE 的現行值。也因此您可以存取多個資料庫。如果您想在 Net.Data 巨集中存取三個資料庫 (例如，D1、D2 和 D3)，且您的起始設定檔具有標準的 CLIETTE "DTW_SQL:\$(DATABASE)" 行，則您的架構檔中必須有下列三個區段：

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

- 處理已啟動，但無法停止。如果您將處理的 MAX 數字設定為 M，不管任何時候都會同時進行 M 個處理，它們會維持在作用狀態，直到您將連線管理程式關閉為止。所以您大概不會想將 MAX 數字設得太高，以免將用來啟動一些較少使用之處理的系統資源用完。建議您嘗試不同的 MIN 和 MAX 值，看看何值最適用於您的系統。若連線管理程式收到的要求，比所設定的 MAX 值還多，則會將最後的放在等待佇列中，直到 cliette 完成處理後再拿出來處理。當 cliette 可用時，則會處理要求。此點對應用程式的使用者來說是顯而易見的。
- 在使用埠號碼時要小心，不要有衝突。每個 cliette 會用到 2 個埠。當您指定一組埠時，您必須指定要使用的埠。前兩個值為 START_PUBLIC_PORT 和 START_PRIVATE_PORT。另一個是 cliette 的 MAX 值。下列範例會顯示所用的埠。

```
START_PUB_PORT=1000
START_PRIV_PORT=1010
MAX_NUM_PROC=5
```

本例所用的埠有：

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

常見的錯誤是：讓兩組 cliettes 彼此所用的埠重疊。請向您的系統管理者查詢，以確保您要使用的埠為可用的。您平台上的 Readme 檔中，會有一般性的指引，您可以從中瞭解您作業系統可使用哪些埠。

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

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

使用 ICAP1

由於「Internet Connection API」不會透過 CGI 處理來進行，因此可提高執行效能。有關「Internet Connection」的詳細說明，請參閱 *Web Programming Guide*：

<http://www.ics.raleigh.ibm.com/pub/icswpg.htm>

如果您想瞭解哪些系統可使用 ICAPI，及其進一步的資訊，可察看 README 檔。以下的步驟是教您如何架構 Net.Data，以便在您的伺服器上搭配 ICAPI 一起執行：

- DLL 或共用程式庫必須放在伺服器的 CGI-BIN 目錄中。例如：

```
WWW\CGI-BIN\
```

Net.Data 之其他的 DLL 或共用程式庫，則必須放在伺服器的 DLL 或共用程式庫的目錄中。

- 您必須在您 Web 伺服器的架構檔 (httpd.conf 或 httpd.cnf) 中，加進一條 service 陳述式，以呼叫 API。例如：

```
Service /cgi-bin/db2www* c:\www\cgi-bin\dtwicapi:dtw_icipi*
```

- ICAPI 具備完整的相容性，而可支援現有的應用程式。在 URL、表格頁面、或錨點的呼叫上，乃與呼叫 CGI 的方式一樣。

使用 ISAPI

如果您想瞭解哪些系統可使用 ISAPI，及其進一步的資訊，可察看 README 檔。以下的步驟是教您如何架構 Net.Data，以便在您的伺服器上搭配 ISAPI 一起執行：

爲了能使用 ISAPI，Net.Data 出場時附有一個 DLL。此 DLL 是放在伺服器的次目錄中。例如：

```
/inetsrv/scripts/dtwisapi.dll
```

ISAPI 會略過 CGI 處理，因此，您必須變更一些會讓 CGI 呼叫 Net.Data 的 Web 網頁與 Web 巨集。除去表格頁面和錨點參照中cgi-bin/db2www/ 部份，並將之換成 dtwisapi.dll。例如，以下的 URL 會將 Net.Data 當成 CGI 程式來呼叫：

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

以下的 URL 會將 Net.Data 視爲 ISAPI 應用程式：

```
http://server1.stl.ibm.com/scripts/dtwisapi.dll/test1.mac/report
```

若您將您的 Net.Data 巨集放在好幾個目錄中，則在 DLL 名稱之後會附上目錄名稱。例如，以下的 URL 會呼叫一個位於 /orders/ 目錄中的 Net.Data 巨集：

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

更新過的 URL 會保留目錄名稱，而變成：

```
http://server1.stl.ibm.com/scripts/dtwisapi.dll/orders/test1.mac/report
```

使用 NSAPI

要針對所有的語言環境使用 NSAPI 時，則您的 Web 伺服器必須使用現場連線。由於 NSAPI 不會透過 CGI 處理來進行，因此可提高執行效能。有關 Netscape Server API 網頁中的詳細資訊，請進入：

```
http://home.netscape.com/newsref/std/server_api.html
```

如果您想瞭解哪些系統可使用 NSAPI，及其進一步的資訊，可察看 README 檔。以下的步驟是教您如何架構 Net.Data，以便在您的伺服器上搭配 NSAPI 一起執行：

- 爲了能使用 NSAPI，Net.Data 在出廠附有一個 DLL。此 DLL 是放在伺服器的目錄中。例如：

/netscape/server/bin/httpd/dtwnsapi.dll

- 在使用 NSAPI 前，請先變更您的伺服器架構。需要變更的檔案有：

obj.conf	在檔案頂端加進下式：
	Init fn="load-modules" shlib="dtwnsapi.dll" funcs=dtw_nsapi
obj.conf	在 Services 部份加進下式：
	Service fn="dtw_nsapi" method=(GET HEAD POST) type="magnus-internal/d2w"
mime.types	加入下式：
	type=magnus-internal/dtw exts=mac

- 從 netdata/macro 中將 Net.Data 巨集檔移到伺服器的主文件目錄中：
- /netscape/server/docs/
- 在起始設定檔中，將 MACRO_PATH 改成主文件目錄，以指向您所複製的 Net.Data 巨集。此舉只是告訴 Net.Data 何處可找到這些檔案。
 - NSAPI 會略過 CGI 處理，因此，您必須變更一些會讓 CGI 呼叫 Net.Data 的 Web 網頁與 Web 巨集。請除去表格頁面及錨點參照中 URL 的 cgi-bin/db2www/ 部份。伺服器會根據 MAC 字尾，識出這些檔案即爲 Net.Data 巨集，這是因爲當您變更 Netscape 架構檔時，曾按此定義過它。例如，以下的 URL 會將 Net.Data 當成 CGI 程式來呼叫：

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

以下的 URL 會將 Net.Data 視爲 NSAPI 應用程式：

http://server1.stl.ibm.com/test1.mac/report

若您將您的 Net.Data 巨集放在好幾個目錄中，則某些步驟可能稍有不同：

- 將目錄連同其中的 Net.Data 巨集，移到伺服器的文件目錄中。
- 更新起始設定檔中的 MACRO_PATH 變數。
- 修改指向這些 Net.Data 巨集的錨點參照與表格頁面，並保留其目錄名稱。例如，以下的 URL 會呼叫一個位於 /orders/ 目錄的 Net.Data 巨集：

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

更新過的 URL 較短，但仍保留目錄名稱，例如：

http://server1.stl.ibm.com/orders/test1.mac/report

附錄A. 動態查詢範例

在此範例應用程式中，會顯示一個列有員工的列示。應用程式的使用者可任選一個員工名稱，檢視其詳細資訊。此應用程式是使用 SQL 語言環境，並透過 Net.Data 的變數 LOGIN 與 PASSWORD，對要存取資料庫的使用者進行身份驗證。

```
%{***** 動態查詢範例 *****}
*   FileName = sqlsaml.mac                                     *
*   說明：                                                    *
*       此 Net.Data 巨集檔 ...                                  *
*       - 會從資料庫來建立一個動態選擇列示                  *
*       - 使用預設的表格以顯示 SQL 的結果                    *
*****}
%{*****}
*   併入檔 - *
*****}
%INCLUDE "sqlsaml.hti"
%{*****}
*   函數：queryDB          語言環境：SQL                      *
*   說明：查詢資料庫，並從結果中建立一個選擇列示。          *
*       資料庫名稱與表格名稱，為定義在上述併入檔中的        *
*       變數。                                                    *
*****}
%FUNCTION(DTW_SQL) queryDB() {
    SELECT * FROM $(myTable)
    %MESSAGE {
        -204: {<p><b>錯誤 -204: 找不到表格 $(myTable)。</b></p>
        <p>請確定所用的併入檔無誤。</p>
        %} : exit
        +default: "警告 $(RETURN_CODE)" : continue
        -default: "非預期的錯誤 $(RETURN_CODE)" : exit
    }
}

%REPORT {
<select name=emp_name>
%ROW{
<option>$(V2)
%}
</select>
%}
%}

%{*****}
*   函數區塊                                                    *
*   說明：傳送 SQL 查詢給資料庫，以查詢所選的名字，          *
*       並產生一個預設表格                                          *
*****}
%FUNCTION(DTW_SQL) fname(){
    SELECT FIRSTNME, MIDINIT, LASTNAME, PHONENO, JOB FROM $(myTable) WHERE FIRSTNME='$(emp_name)'
    %MESSAGE {
        -204: "錯誤 -204: 找不到表格 "
        -104: "錯誤 -104: 語法錯誤"
        100: "警告 100: 無記錄" : continue
        +default: "警告 $(RETURN_CODE)" : continue
        -default: "意外的 SQL 錯誤" : exit
    }
}
%}
%}
```

```

%{ *****
*   HTML 區塊：INPUT           標題：動態查詢選擇           *
*                               *
*   說明：查詢資料庫，以建立員工           *
*   的選擇列示           *
*****%}
%HTML(INPUT){
<html><head><title>動態查詢選擇</title></head><body><h3>$(exampleTitle)</h3>
<p>本例會查詢資料庫，並動態地使用結果，以使用
<em>%REPORT</em> 區塊建立一個選擇列示。
<hr>
<p>以下的選擇列示是藉由察看資料庫動態建置而成的
<form method="post" action="report">
@queryDB()<input type="submit" value="Select Employee">
</form>
<hr>
</body>
</html>
%}

%{ *****
*   HTML 區塊：    REPORT      標題：動態查詢選擇           *
*   說明：結果頁面 - 呼叫 SQL 函數，以顯示曾在前一頁中           *
*   選出之員工的相關資訊           *
*****%}
%HTML(REPORT){
<html>
<head>
<title>動態查詢選擇</title>
</head>
<body>
<h3>您所選的員工名稱 = $(emp_name)</h3>
<p>以下為該員工的相關資訊：
<PRE>
----- 資料庫查詢結果 -----
@fname()
</PRE>
<hr><a href="input">回到前一頁</a>
</body>
</html>
%}

%{      End of Net.Data macro 1 %}
=====
%{ ***** 併入檔 *****
*   FileName = sqlsamp1.hti           *
*   說明：           *
*   此併入檔會提供廣域的 DEFINE 給           *
*   sqlsamp1.mac Net.Data 巨集。           *
*           *
*           *
*****%}
%define {
    DATABASE  ="SAMPLE"
    LOGIN     ="USERID"
    PASSWORD  ="PASSWORD"
    emp_name  ="a name"
    exampleTitle = "SQL Example"
    myTable   = "EMPLOYEE"
%}

%{      End of include file %}

```

Net.Data 巨集建立包含所有數值類型而名為 mytable 的表格。

%{***** 數值資料類型範例 *****}

FileName = **<sqlsamp2.mac>**

說明：

在 mydb 資料庫中建立具有所有數值類型的表格 mytable。

在表格中插入正負值和最小值與

最大值。使用預設表格和不同的報表格式，

來查詢表格。捨棄此表格。

%}

```
%define{
```

```

DATABASE="sample"

```

```
DB CASE="lower"
```

```

DB_CASE Power
LOGIN="userid"

```

```
PASSWORD="password"
```

TRANSACTION SCOPE="MULTIPLE"

```
db2www="db2www"
```

```
RPT MAX ROWS="100"
```

%}

```
%FUNCTION(DTW SQL) Create() {
```

```
create table mytable (intcol int, sintcol smallint, deccol decimal(31,5),
```

```
numcol numeric(31,5), sfloatcol float,
```

```
dfloatcol double precision )
```

%}

```
%FUNCTION(DTW SQL) Insert () {
```

```
insert into mytable values (-1, -1, -1.1, -1.1, -1.1, -1.1)
```

%}

```
%FUNCTION(DTW SQL) Insert () {
```

```
insert into mytable values (+1, +1, +1.1, +1.1, +1.1, +1.1)
```

%}

```
%FUNCTION(DTW SQL) Insert () {
```

```
insert into mytable values (-2147483648, -32768,
```

```
-99999999999999999999999999999999.99999,
```

-99999999999999999999999999999999.99999,

-1.7976900000000000E+30, +2.224999999999999E-030)

%}

```
%FUNCTION(DTW SQL) Insert () {
```

```
insert into mytable values (+2147483647, +32767,
```

[illegible]

+99999999999999999999999999999999.99999,

-2.2249999999999999E-030,

```
-2.123456789012345+019 )
```

%}

```

%FUNCTION(DTW_SQL) Query( ) {
select * from mytable
%REPORT {
<pre>
<P> 選了 $(ROW_NUM) 列
<P> 回覆碼為 $(RETURN_CODE)
<A NAME="exampleNlist"><P> $(NLIST)</A>
<OL>
<A NAME="exampleVlist">%ROW {<LI> $(VLIST)%}</A>
</OL>
</pre>
%}
%}

%FUNCTION(DTW_SQL) Query1( ) {
select * from mytable
%REPORT {
<pre>
<P> 選了 $(ROW_NUM) 列
<A NAME="exampleNn"><P> $(N1) $(N2) $(N3) $(N4) $(N5) $(N6)</A>
<A NAME="exampleN ColName">
<P> ...且第 1 欄之 N_column-name 值為 $(N_INTCOL)</A>
<OL>
%ROW {
<LI>
<UL>
<A NAME="exampleV_ColName"><LI> $(V_INTCOL)</A>
<LI> $(V_SINTCOL)
<LI> $(V_DECCOL)
<LI> $(V_NUMCOL)
<LI> $(V_SFLOATCOL)
<LI> $(V_DFLOATCOL)
</UL>
%}
</OL>
</pre>
%}
%}

%FUNCTION(DTW_SQL) Query2( ) {
select * from mytable
%}

```


[illegible][illegible]

最後回覆碼為 0

名詞解釋

API. 應用程式設計介面 (Application programming interface)

applet. 一種包含在 HTML 頁面中的 Java 程式。Applet 會使用可使用 Java 的瀏覽器 (如：Netscape)，且會在載入 HTML 頁面時一起載入。

應用程式設計介面 (API). 一種功能性介面，由作業系統或個別訂購的授權程式所提供。此種介面可讓以高階語言寫成的應用程式，能使用作業系統或授權程式中的特定資料或功能。Net.Data 可支援三種具有專利的 Web 伺服器 API，讓您可在 CGI 處理上提高執行效能。

BLOB. 二進位大型物件 (Binary large object)

CGI. 通用閘道介面 (Common Gateway Interface)

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 處理的執行時間來處置這些需求。

CLOB. 字元大型物件 (Character large object)

通用閘道介面. 一種標準的路徑，可讓 Web 伺服器會依循此路徑，將控制傳給應用程式並收回資料。

連線管理程式. 為 Net.Data 中的可執行檔，其名稱為 dtwcm。要支援現場連線時，需用到此檔。

資料庫. 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 (DBMS). 一種軟體系統，用以控制資料庫的建立、組織、與修改作業，以及控制對其中所存資料的存取。

資料類型. 直欄與文字的屬性。

DBMS. 資料庫管理系統 (Database management system)

防火牆. 一個帶有防火牆軟體的電腦，其負責保護內部的網路不讓外來者侵犯。

純本文檔介面. 一種 Net.Data I 語言環境，可讓您讀取與寫入純文字檔中的資料。

html. 超本文標記語言 (hypertext markup language)

http. 超本文轉送通信協定 (hypertext transfer protocol)

超本文標記語言. 一種用來撰寫 Web 文件的標籤語言。

超本文轉送通信協定. 用於 Web 伺服器與瀏覽器間的通信協定。

ICAPI. Internet 連線 API (Internet Connection API)

ICS. Internet 連線伺服器 (Internet Connection Server)

ICSS. Internet 連線安全伺服器 (Internet Connection Secure Server)

Internet. 國際公用 TCP/IP 電腦網路。

Internet 連線伺服器. IBM 之無安全特性的 Web 伺服器。

Internet 連線安全伺服器. IBM 之具安全特性的 Web 伺服器。

intranet. 公司防火牆內的 TCP/IP 網路。

ISAPI. Microsoft 的 Internet 伺服器 API。

Java. 一種不依附在平台下之物件導向型程式設計語言，特別適用於 Internet 的應用程式。

語言環境.

現場連線. 一種 Net.Data 架構，乃搭配連線管理程式與 Web 伺服器 API 一起運作。在現場連線下，可讓您重複使用資料庫連線。

LOB. 大型物件 (Large object)

NSAPI. Netscape API。

空值. 一個代表沒有資訊的特殊值。

路徑. 用來找到檔案的搜尋途徑。

PERL. 一種解譯過的程式設計語言。

埠. 一種 16 位元號碼，用來讓 TCP 能和高階通信協定 (或應用程式) 間通信。

TCP/IP. 傳輸控制通信協定/Internet 通信協定 (Transmission Control Protocol/Internet Protocol)

傳輸控制通信協定/Internet 通信協定. 為一組通信協定，可為區域與廣域網路提供對等 (peer-to-peer) 連線功能的支援。

URL. term (Uniform resource locator)

通用資源位置(URL). 一種位址，其中會指定出 http 伺服器名稱，除此，亦可附上目錄與檔名，例如，
<http://www.software.ibm.com/data/net.data/index.html>。

Web 伺服器. 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。

〔三劃〕

大型物件 33

〔四劃〕

內建函數 34

〔五劃〕

加密 8

〔六劃〕

列示變數 23

安全 8

〔七劃〕

身份驗證 8

防火牆 9

〔八劃〕

函數

內建 34

呼叫 25

定義 24

純本文檔介面 35

儲存程序 26

Web 登記 38

呼叫 Net.Data 10

表格變數 24

〔九劃〕

架構

路徑 5

語言環境 8

架構 Net.Data 4

〔十劃〕

純本文檔介面 35, 36

起始設定檔 4

〔十一劃〕

執行效能 41

執行變數 21

條件變數 21

現場連線 43

〔十二劃〕

提高執行效能 41

〔十三劃〕

路徑陳述式 5

〔十四劃〕

語言環境 39

〔十五劃〕

樣本巨集 49

樣本巨集 B 51

〔十七劃〕

儲存程序 26

環境變數 21

隱含變數 24

隱藏變數 22

〔二十三劃〕

變數

列示 23

定義 19

表格 24

參照 20

執行 21

條件 21

範圍 19

環境 21

隱含 24

隱藏 22

類型 18

變數陳述式 8

C

cliette 43

D

DEFINE 區塊 16

DTW_FFI 35

DTW_WEBREG 38

E

ENVIRONMENT 陳述式 8

EXEC_PATH 6

F

FFI_PATH 7

FUNCTION 區塊 18

H

HTML 區塊 16, 30

HTML_PATH 7

I

ICAPI 46

INCLUDE_PATH 6

ISAPI 47

L

LOBS 33

M

MACRO_PATH 5

MESSAGE 區塊 27

N

Net.Data 巨集

說明 3

NSAPI 47

R

REPORT 區塊 32

W

Web 登記 38