



Net.Data 参考



Net.Data 参考

注意

在使用本信息及其所支持的产品之前，务必请阅读第243页的『附录C. 注意事项』中的信息。

目录

前言	ix
关于 Net.Data	ix
关于本书	ix
谁应当阅读本书	ix
有关本书中的例子	x
如何阅读语法图	x
第1章 Net.Data 宏语言结构	1
Net.Data 宏文件语法	1
公用的语法成份	4
变量名	4
变量引用	4
字符串	4
宏语言结构	5
注解块	7
DEFINE 块或语句	9
ENVVAR 语句	12
EXEC 块或语句	13
FUNCTION 块	15
函数调用 (@)	21
HTML 块	23
IF 块	25
INCLUDE 语句	30
INCLUDE_URL 语句	32
LIST 语句	34
MACRO_FUNCTION 块	36
MESSAGE 块	39
REPORT 块	43
ROW 块	45
TABLE 语句	47
WHILE 块	48
第2章 变量	51
用户定义的变量	52
条件变量	52
环境变量	53
可执行变量	53
隐藏变量	54
列表变量	55
表格变量	56
Net.Data 表格处理变量	57
Nn	58
NLIST	59
NUM_COLUMNS	60
NUM_ROWS	61
ROW_NUM	62
TOTAL_ROWS	63
V_columnName	64
VLIST	65

Vn	66
Net.Data 报表变量	67
ALIGN	68
DTW_DEFAULT_REPORT	69
DTW_HTML_TABLE	70
RPT_MAX_ROWS	71
START_ROW_NUM	72
Net.Data 语言环境变量	74
DATABASE.	75
DB_CASE	76
DB2PLAN	77
DB2SSID.	78
DTW_APPLET_ALTTEXT.	79
DTW_EDIT_CODES.	80
DTW_MBMODE	81
DTW_SAVE_TABLE_IN	82
DTW_SET_TOTAL_ROWS	83
LOCATION	84
LOGIN	85
NULL_RPT_FIELD	86
PASSWORD.	87
SHOWSQL	88
SQL_STATE	89
TRANSACTION_SCOPE	90
Net.Data 杂项变量	91
DTW_CURRENT_FILENAME	92
DTW_CURRENT_LAST_MODIFIED	93
DTW_DEFAULT_MESSAGE.	94
DTW_LOG_LEVEL	95
DTW_MACRO_FILENAME	96
DTW_MACRO_LAST_MODIFIED.	97
DTW_MP_PATH	98
DTW_MP_VERSION	99
DTW_PRINT_HEADER.	100
DTW_REMOVE_WS.	101
RETURN_CODE	102
第3章 Net.Data 内部函数	103
函数名.	103
输入和输出参数	103
函数结果格式	103
函数参数规则	104
一般函数.	104
DTW_ADDQUOTE	106
DTW_CACHE_PAGE	108
DTW_DATE.	111
DTW_EXIT	112
DTW_GETCOOKIE	113
DTW_GETENV	115
DTW_GETINIDATA.	116
DTW_HTMLLENCODE	117
DTW_QHTMLLENCODE	119

DTW_SENDMAIL	120
DTW_SETCOOKIE	123
DTW_SETENV	126
DTW_TIME	127
DTW URLESCSEQ	129
数学函数	131
DTW_ADD	132
DTW_DIVIDE	133
DTW_DIVREM	134
DTW_FORMAT	135
DTW_INTDIV	138
DTW_MULTIPLY	139
DTW_POWER	140
DTW_SUBTRACT	141
字符串函数	142
DTW_ASSIGN	143
DTW_CONCAT	144
DTW_DELSTR	145
DTW_INSERT	146
DTW_LASTPOS	148
DTW_LENGTH	149
DTW_LOWERCASE	150
DTW_POS	151
DTW_REVERSE	152
DTW_STRIP	153
DTW_SUBSTR	154
DTW_TRANSLATE	155
DTW_UPPERCASE	157
字处理函数	158
DTW_DELWORD	159
DTW_SUBWORD	160
DTW_WORD	161
DTW_WORDINDEX	162
DTW_WORDLENGTH	163
DTW_WORDPOS	164
DTW_WORDS	165
表格函数	166
DTW_TB_APPENDROW	167
DTW_TB_COLS	168
DTW_TB_DELETEROW	169
DTW_TB_DLIST	170
DTW_TB_DUMPH	172
DTW_TB_DUMPV	173
DTW_TB_GETN	174
DTW_TB_GETV	175
DTW_TB_HTMLENCODE	176
DTW_TB_INPUT_CHECKBOX	177
DTW_TB_INPUT_RADIO	178
DTW_TB_INPUT_TEXT	179
DTW_TB_INSERTCOL	180
DTW_TB_INSERTROW	181
DTW_TB_LIST	182

DTW_TB_MAXROWS	183
DTW_TB_QUERYCOLNONJ.	184
DTW_TB_ROWS	185
DTW_TB_SELECT	186
DTW_TB_SETCOLS.	187
DTW_TB_SETN	188
DTW_TB_SETV	189
DTW_TB_TABLE.	190
DTW_TB_TEXTAREA	192
平面文件接口函数	193
访问平面文件数据源.	193
平面文件接口定界符.	195
锁定文件.	196
DTWF_APPEND	197
DTWF_CLOSE.	199
DTWF_DELETE	200
DTWF_INSERT	202
DTWF_OPEN	204
DTWF_READ	205
DTWF_REMOVE.	207
DTWF_SEARCH	208
DTWF_UPDATE	210
DTWF_WRITE.	212
Web 注册表函数	214
DTWR_ADDENTRY.	215
DTWR_CLEARREG	216
DTWR_CLOSEREG	217
DTWR_CREATEREG	218
DTWR_DELENTY	219
DTWR_DELREG	220
DTWR_LISTREG	221
DTWR_LISTSUB	222
DTWR_OPENREG	223
DTWR_RTVENTRY.	224
DTWR_UPDATEENTRY	225
持久性宏函数	226
DTW_ACCEPT.	227
DTW_COMMIT	228
DTW_ROLLBACK	229
DTW_RTVHANDLE.	230
DTW_STATIC	231
DTW_TERMINATE	232
附录A. DB2 WWW Connection	235
EXEC_SQL	235
HTML_INPUT	235
HTML_REPORT	235
SQL	235
SQL_MESSAGE	236
SQL_REPORT	236
SQL_CODE	236

附录B. Net.Data 操作系统参考	237
附录C. 注意事项	243
商标	244
词汇表.	245
索引	247

前言

感谢您选择 Net.Data 版本 2 - IBM 的开发工具来创建动态 Web 页面! 使用 Net.Data 之后, 您就可以迅速地开发具有动态内容的 Web 页面, 这只要通过结合来自广泛种类数据源的数据并使用您已知的程序设计语言的功能即可实现。

Net.Data 版本 2 提供了显著改进的性能和一些新的功能, 这些新功能将赋予您构建与采纳自己的 Internet 商业方案的能力。

关于 Net.Data

采用 IBM 的 Net.Data 产品之后, 您就可以使用来自关系型或非关系型数据库管理系统 (DBMS, 包括 DB2、IMS 和 允许使用 ODBC 的数据库) 的数据来创建动态的 Web 页面; 还可以使用各种编程语言 (例如 Java、JavaScript、Perl、C、C++ 和 REXX) 所编写的应用程序。

您可以将 Net.Data 看作是一个宏处理器, 在 Web 服务器上作为中件执行。您可以编写 Net.Data 应用程序 (称之为宏), Net.Data 将对它进行解释以便使用根据用户输入、数据库当前状态、现有商业逻辑以及您在宏中所设计的其它因素而定制的内容来创建动态的 Web 页面。

一个 URL (统一资源定位器) 形式的请求, 从浏览器 (例如 Netscape 或 Internet Explorer) 流动到将请求转发给 Net.Data 进行执行的 Web 服务器, Net.Data 找出这个宏加以执行, 并构建一个根据您所编写的函数定制的 Web 页面。这些函数能够:

- 在 Perl 脚本、C 与 C++ 应用程序或 REXX 程序中封装商业逻辑
- 访问诸如 DB2 等数据库

Net.Data 支持诸如超文本传输协议 (HTTP) 和公共网关接口 (CGI) 等工业标准接口。HTTP 用在浏览器和 Web 服务器之间, 而 CGI 用在 Web 服务器和 Net.Data 之间。这样您就可以选择您所喜爱的浏览器或 Web 服务器与 Net.Data 一起使用。Net.Data 还支持多个操作系统上的 FastCGI 和主要的 Web 服务器 API。

关于本书

本书在总体上解释了 Net.Data 语言结构、变量和函数的语法和用法。

本书可能引用已经发布、但现在还未进入实用阶段的某些产品或功能。

示例 Net.Data 宏、演示程序以及本书最新拷贝的更多信息, 可以从以下 World Wide Web 站点获得:

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

谁应当阅读本书

涉及计划和编写 Net.Data 应用程序的人员可以使用本书中的信息, 了解 Net.Data 提供什么样的语言结构、变量和函数。

要了解本书中讨论的概念，您应当熟悉 Web 服务器、简单的 SQL 语句和 HTML (包括使用 HTML 表)，以及 *Net.Data* 管理和程序设计指南中的信息。

有关本书中的例子

本书中出现的例子相对较为简单，目的是演示特定的概念，而不说明 *Net.Data* 构造元的所有用法。某些例子只是一些片段，不能单独运行。

如何阅读语法图

以下规则适用于本书中使用的语法图：

- 从左到右、从上到下、沿着直线所指的路径阅读语法图。

▶▶—— 符号表示一条语句的开始。

——▶ 符号表示语句语法要接续到下一行。

▶—— 符号表示此语句是接续上一行的。

——▶▶ 符号表示一语句的结束。

语法单位的图例不同于其他完整的语句，它以 ▶—— 符号开头，以 ——▶ 符号结束。

- 必需的项目出现在水平的直线(主路径)上。

▶▶—— 必需项目出现在主路径的下面。▶▶——

如果必需项目出现在主路径的上面，说明那个项目对本语句的执行没有影响，只用于增加可读性。

- 如果某项目有两种或更多种选择，则它们出现在同一位置，以垂直方式排列。▶▶——

如果某项目必须被选择，则在主路径中出现一个单项目的堆栈。

▶▶—— 必需项目是可选的，必需选项堆栈出现在主路径下面。▶▶——

如果某项目是缺省的，则堆栈出现在主路径上面，并且在其下面显示剩余的选项。▶▶——

- 一个箭头通过主路线上方返回左边，表示一个选项可以被重复。▶▶——

如果重复项中包含标点符号，则必须使用这个指定的标点符号来分隔重复的项目。▶▶——

栈上面需重复箭头表示可以重复栈中的项目。▶▶——

- 关键字以大写形式出现(例如，FROM)。在 *Net.Data* 中，关键字可以以大写或小写形式出现。关键字以外的项目可以以小写字母形式出现 (例如，*column-name*)。这表示用户提供的名称或值。

- 如果出现标点符号、括号、算术运算符或其它此类符号，则必须将其作为语法的一部分输入。

第1章 Net.Data 宏语言结构

本章描述 Net.Data 宏文件中使用的 Net.Data 宏语法和语言结构。这些语言结构由 Net.Data 宏中的关键字和语句或块组成，指定了不同的变量类型、并执行其它特殊任务(如包含文件)。

本章描述:

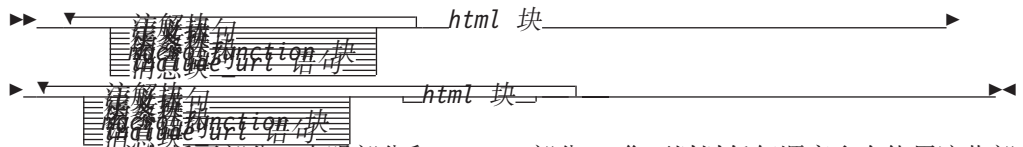
- 『Net.Data 宏文件语法』
- 第4页的『公用的语法成份』
- 第5页的『宏语言结构』

Net.Data 宏文件语法

Net.Data 宏是一个普通文本文件，它由一系列 Net.Data 宏语言结构组成:

- 指定 Web 页的版面设置
- 定义变量和函数
- 调用在宏文件中定义的函数，或是 Net.Data 传送给语言环境进行处理的函数
- 格式化 HTML 中的处理输出，并把它返回给 Web 浏览器

每个语句由一个或多个语言结构组成，而语言结构由关键字、特殊字符、字符串、名称和变量所组成。以下图例描述了一个在语法上有效的 Net.Data 宏的总体结构。请参阅『第1章 Net.Data 宏语言结构』，以了解总体结构中每个元素的详细语法。



Net.Data 宏包括两部分: 声明部分和 HTML 部分。您可以以任何顺序多次使用这些部分。

- 声明部分包含宏文件中变量和函数的定义。
- HTML 部分包含 HTML 块，其中含指定 Web 页的版面设置的 HTML 语句。这一部分包含 report 部分。

第2页的图1显示宏文件的声明和 HTML 部分。



图 1. 宏文件结构

声明或者 HTML 部分中使用的变量和函数必须在变量引用或函数调用之前先定义。

第3页的图2演示一个宏的各个部分。声明部分包含 DEFINE 和 FUNCTION 定义块。HTML 块作为输入和输出块。

```

%{ *****                                定义块                                *****%}
%DEFINE{
    page_title="Net.Data macro Template"
}%

%{ *****                                函数定义块                                *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
    { %EXEC{ompsamp.cmd %}
}%

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

%{ *****                                HTML 块: 输入                                *****%}
%HTML(INPUT) {
    <html>
    <head>
    <title>$(page_title)<title>
    </head><body>
    <h1>Input Form</h1>
    Today is @today()

    <FORM METHOD="post" ACTION="output">
    输入一些数据传送给一个 REXX 程序:
    <INPUT NAME="input_data" TYPE="text" SIZE="30">
    <p>
    <INPUT TYPE="submit" VALUE="Enter">

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

%{ *****                                HTML 块: 输出                                *****%}
%HTML(OUTPUT) {
    <html>
    <head>
    <title>$(page_title)</title>
    </head><body>
    <h1>Output Page</h1>
    <p>@rexx1(input_data)
    <p><hr>
    <p>[<a href="/">Home page</a> |
    <a href="input">Previous page</a>]
    </body></html>
}%

```

图 2. 宏文件模板格式

Net.Data 宏语言是一种自由形式的语言，允许您灵活地编写自己的宏。除非特别说明，否则将忽略额外的空格字符。每个 Net.Data 宏语言结构都将在以下章节中描述，还包括用于定义这些结构的其它成分。Net.Data 宏语言支持 DB2 WWW Connection 语言成分，以提供向后兼容能力。尽管第235页的『附录A. DB2 WWW Connection』中描述了这些语言成份，但仍建议使用 Net.Data 语言结构。

这些例子描述了在宏文件中使用语言结构、变量、函数和其它成分的几种方法。可以从 Net.Data Web 页中下载示例和演示程序，获取更有扩展性的例子：

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

公用的语法成份

语言结构说明中频繁使用以下语法成份:

- 『变量名』
- 『变量引用』
- 『字符串』

变量名

目的:

标识一个或多个名称; 每个后继名称都用一个句点 (.) 并置。一个名称是一个字母或数字字符串, 以字母字符或下划线开头, 可以包含字母、数字或下划线字符的任意组合。

引号(『』)中的字符串可以包含除新建行字符外的任何字符。如果此字符串在方括号- ({%})-内, 则它包含任何字符, 包括新建行字符。

变量名必须以一个字母或下划线(_)开头, 可以包含任何字母数字字符或下划线。 除 *N_columnName* 和 *V_columnName* 外, 所有的变量名都是区别大小写的(请参阅第57页的『Net.Data 表格处理变量』, 获取有关这两个例外的更多信息。).

语法:

▶▶ `名称` ▶▶

变量引用

目的:

返回一个先前定义的变量的值, 用 \$ 和 () 来指定。 例如: 如果 `VAR = 'abc'`, `$(VAR)` 返回 'abc' 值。变量引用是在运行时求值的。若某个变量是为 EXEC 语句或块定义的, 则 Net.Data 将在读取此变量引用时运行指定的操作。

被引用的变量必须在引用之前先在 Net.Data 宏中定义。如果此变量没有定义, 则返回一个空字符串。

语法:

▶▶ `$ (变量名)` ▶▶

字符串

字母、数字字符和标点符号的任意序列。如果此字符串出现在双引号内, 则换行字符是不允许的。现在, 变量引用被字面解释为字符串的一部分, 而不被解释为变量引用或函数调用, 除了另外注明的以外。参见每个语言结构中的字符串参数说明, 了解它和语言结构一起使用时的限制。

要在一个引用字符串中指定双引号, 就必须使用两对双引号。作为函数变量的字符串, 或者作为比较表达式中某一项的字符串, 都可以包含双引号。例如, 如果把一个字符串值定义成:

```
%DEFINE result = " ""Hello world!"" "
```


result 的值是:

```
"Hello world!"
```

一条 HTML 语句是一个字符串。

仅针对 OS/400 用户: 用作函数变量、项和变量值的字符串可以包含变量引用和函数调用。在下面的例子中, 函数调用 `myfunc2` 中有一个包含变量引用和函数调用的字符串参数。

```
%html(report) {  
    @myfunc2("abc$(var1)@myfunc()")  
%}
```

`Net.Data` 在将字符串传递给函数 `myfunc2` 之前, 分辨出变量引用 `$(var1)` 和函数调用 `@myfunc()`, 而不是将它们字面解释为字符串的一部分。

宏语言结构

本章节描述 `Net.Data` 宏文件中使用的 `Net.Data` 宏语言结构。

每个语言结构说明可以包含以下信息:

目的 定义为什么在 `Net.Data` 宏中使用语言结构。

语法 提供语言结构的逻辑结构的图例。

参数 定义语法图例中的所有成份, 并提供与其它语言结构的语法和例子的交叉引用。

上下文 说明可以在 `Net.Data` 宏结构中的哪些地方使用此语言结构。

限制 定义它可以包含的语言成份, 并指定任何用法上的限制。

例子 提供关于在 `Net.Data` 宏中如何使用关键字语句或块的一些简单例子和说明。

宏中使用以下结构; 请参阅每个结构说明的语法和例子。

- 第7页的『注解块』
- 第9页的『DEFINE 块或语句』
- 第12页的『ENVVAR 语句』
- 第13页的『EXEC 块或语句』
- 第15页的『FUNCTION 块』
- 第21页的『函数调用 (@)』
- 第23页的『HTML 块』
- 第25页的『IF 块』
- 第30页的『INCLUDE 语句』
- 第32页的『INCLUDE_URL 语句』
- 第34页的『LIST 语句』
- 第36页的『MACRO_FUNCTION 块』

- 第39页的『MESSAGE 块』
- 第43页的『REPORT 块』
- 第45页的『ROW 块』
- 第47页的『TABLE 语句』
- 第48页的『WHILE 块』

注解块

目的

把 Net.Data 宏的函数文档化。因为可以在宏文件的任何地方使用 COMMENT (注解) 块，所以它不在其它语法图中再作说明。

语法

►► _%{__文本__}% _____ ◄◄
值

文本 一行或多行中的任何字符串。Net.Data 将忽略所有注解的内容。

上下文

可以在一个 Net.Data 宏的两个 Net.Data 语言结构之间的任何位置放上注解。

限制

允许任何文本或字符；但是，注解块不可以嵌套。

例

例 1: 一个基本的注解块

```
%{  
This is a comment block. It can contain any number of lines  
and contain any characters. Its contents are ignored by Net.Data.  
%}
```

例 2: 一个 FUNCTION 块中的注解

```
%function(DTW_REXX) getAddress(IN name,   %{ customer name %}  
                                IN phone,  %{ customer phone number %}  
                                OUT address %{ customer address %}  
                                )  
{  
    ....  
%}
```

例 3: 一个 HTML 块内的注解

```
%html(report) {  
  
    %{ run the query and save results in a table %}  
    @myQuery(resultTable)  
  
    %{ build a form to display a page of data %}  
    <form method="POST" action="report">  
  
    %{ send the table to a REXX function to send the data output %}  
    @displayRows(START_ROW_NUM, submit, resultTable, RPT_MAX_ROWS)  
  
    %{ pass START_ROW_NUM as a hidden variable to the next invocation %}  
    <input name="START_ROW_NUM" type="hidden" value="${START_ROW_NUM}">  
  
    %{ build the next and previous buttons %}  
    %if (submit == "both" || submit == "next_only")  
        <input name="submit" type="submit" value="next">  
    %endif
```

```

%if (submit == "both" || submit == "prev_only")
  <input name="submit" type="submit" value="previous">
%endif
</form>
%}

```

例 4 一个 DEFINE 块中的注解

```

%define {
  START_ROW_NUM = "1"           %{ starting row number for output table %}
  RPT_MAX_ROWS = "25"          %{ maximum number of rows in the table %}
  resultTable = %table          %{ table to hold query results           %}
%}

```

DEFINE 块或语句

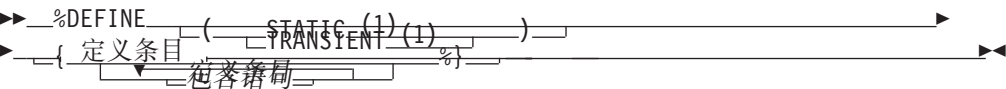
目的

DEFINE 段定义宏的声明部分中的变量名，它或可以是一个语句，或是一个块。

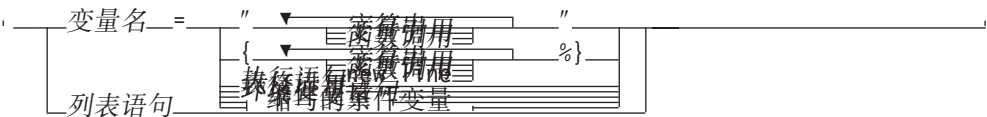
- 使用这些语句一次定义一个变量
- 使用块定义几个变量

用双引号(""), 变量定义可以在单行中; 使用方括号或百分号({ %}), 则可以跨多行。在定义变量后, 您可以在宏的任何地方引用它。

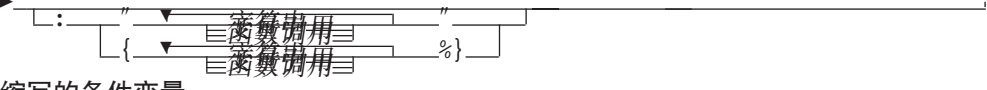
语法



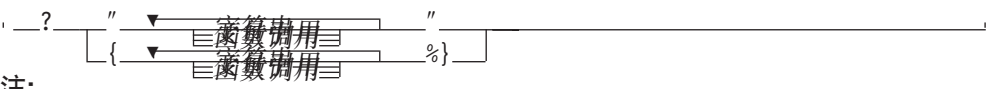
定义条目



条件变量



缩写的条件变量



注:

1. STATIC 和 TRANSIENT 是持久性宏的关键字, 目前仅能在 OS/400 操作系统上使用。

值

%DEFINE

一个用于定义变量的关键字。

STATIC

指定指明变量在一个持久性事务的各宏调用之间保留其值的关键字。这是持久性宏的缺省值。

TRANSIENT

指定指明变量在各宏调用之间不保留其值的关键字。这是非持久性宏的缺省值。

定义项:

变量名

一个或多个名称, 每个附加的名称用一个句点(.)相连。请参阅第4页的『变量名』中的语法信息。

字符串

字母、数字字符和标点符号的任意序列。如果此字符串出现在双引号内, 则换行字符是不允许的。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。 例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

执行语句

EXEC 语句。一个外部程序的名称，在引用一个变量或调用一个函数时执行这个程序。请参阅第13页的『EXEC 块或语句』中的语法和例子。

表格语句

TABLE 语句。定义了一组相关数据，其中包含一组相同的记录(或行)、一组用于描述每行中各字段的列名。请参阅第47页的『TABLE 语句』中的语法和例子。

环境变量语句

ENVVAR 语句。引用环境变量。请参阅第12页的『ENVVAR 语句』中的语法和例子。

条件变量

根据另一个变量或字符串的值来设置一个变量的值。

缩写的条件变量

根据另一个变量或字符串的值来设置一个变量的值。条件变量的一种短格式。

列表语句

LIST 语句。定义了一组变量，它们用于构建一个定界的值列表。请参阅第34页的『LIST 语句』中的语法和例子。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

上下文

DEFINE 块或语句必须在 IF 块内，或在 Net.Data 宏声明部分的所有其它块之外。

限制

- 可以包含以下成份：
 - 注解块
 - 条件变量
 - LIST 语句
 - TABLE 语句
 - 变量引用
 - INCLUDE 语句
 - EXEC 语句
 - 函数调用
 - ENVVAR 语句

- 不能在变量自己的定义中使用它。例如，不允许下列的变量定义：

```
%DEFINE var = "The value is $(var)."
```

例

例 1: 简单变量定义

```
%DEFINE var1 = "orders"
%DEFINE var2 = "$(var1).html"
```

在运行期间，变量引用 `$(var2)` 的结果为 `orders.html`。

例 2: 字符串内的引号

```
%DEFINE hi = "say ""hello"""
%DEFINE empty = ""
```

在显示时，变量 `hi` 的值为 `say "hello"`。变量 `empty` 为空。

例 3: 多变量的定义

```
%DEFINE{ DATABASE = "testdb"
          home = "http://www.software.ibm.com"
          SHOWSQL = "YES"
          PI = "3.14150"
%}
```

例 4: 一个变量的多行定义

```
%DEFINE text = {This variable definition
                spans two lines
%}
```

例 5: 这个条件变量例子演示，如果结果值不包含任何 NULL 值，则变量 `var` 如何获取在值 (『』) 内部的结果值。

```
%DEFINE var = ? "Hello! $(V)@MyFunc()"
%}
```

ENVVAR 语句

目的

在 DEFINE 块中把变量定义为环境变量。当引用 ENVVAR 变量时，Net.Data返回同名的环境变量的当前值。使用这种方法来引用环境变量比使用 DTW_GETENV 更为高效。关于 DTW_GETENV 的更多信息，请参阅第115页的『DTW_GETENV』。

语法

►► %ENVVAR _____ ◄◄

上下文

ENVVAR 语句可以在 DEFINE 块或语句中。

值

%ENVVAR

在 DEFINE 块中把变量定义为环境变量的关键字。此变量获取宏文件中任何地方的环境变量的值。

限制

ENVVAR 语句不可以包含其它成分。

例

例 1: 在此例中，ENVVAR 定义一个变量，当它被引用时，返回环境变量 SERVER_SOFTWARE 的当前值，即 Web 服务器的名称。

```
%DEFINE SERVER_SOFTWARE = %ENVVAR
```

```
%HTML(REPORT) {  
The server is $(SERVER_SOFTWARE).  
%}
```


EXEC 块或语句

目的

当引用变量或调用某函数时，指定要执行的外部程序。

当 Net.Data 在宏文件中遇到可执行变量时，它将使用下列方法寻找被引用的可执行程序：

- 1. 在 Net.Data 初始化文件中搜索 EXEC_PATH。参阅 *Net.Data 管理与程序设计指南* 的配置一章，进一步获取有关 EXEC_PATH 的信息。
- 2. 如果 Net.Data 没有找到该程序，它将搜索系统定义的目录。如果找到了此可执行程序，则 Net.Data 运行它。

权限提示： 确保此 Web 服务器具有对 EXEC 语句或块引用的任何文件的访问权。请参阅 *Net.Data 管理和程序设计指南* 的配置一章中关于指定 Web 服务器对 Net.Data 文件的访问权限这一部分，以获取更多信息。

EXEC 语句和块在两种上下文中使用，根据使用的不同场合，它们可以有不同的语法。在 DEFINE 块中使用 EXEC 语句，在 FUNCTION 块中使用 EXEC 块。

语法

在 DEFINE 块中使用的 EXEC 语句语法：

在 FUNCTION 块中使用的 EXEC 块语法：

值

%EXEC

此关键字指定了在引用一个变量或调用一个函数时要执行的外部程序的名称。当 Net.Data 遇到在 EXEC 语句中定义的变量引用时，它就处理 EXEC 语句为此变量而声明的操作。

字符串

字母、数字字符和标点符号的任意序列。如果此字符串出现在双引号内，则换行字符是不允许的。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

上下文

可以在这些上下文中找到 EXEC 块或语句：

- DEFINE 块
- FUNCTION 块

限制

EXEC 块或语句可以包含这些元素:

- 注解块
- 字符串
- 变量引用
- 函数调用

例

例 1: 由一个变量引用的可执行文件

```
%DEFINE mycall = %EXEC "MYEXEC.EXE $(empno)"
```

```
%HTML (report){  
<P>Here is the report you requested:  
<HR>$(mycall)  
%}
```

这个例子在每个对此变量 (mycall) 的引用上执行 MYEXEC.EXE 。

例 2: 由一个函数引用的可执行文件

```
%FUNCTION(DTW_REXX) my_rexx_pgm(INOUT a, b, IN c, INOUT d){  
  %EXEC{ mypgm.cmd this is a test %}  
%}
```

这个例子在调用函数 my_rexx_pgm 时即执行 mypgm.cmd 。

FUNCTION 块

目的

定义 Net.Data 从宏文件中调用的一个子例行程序。FUNCTION 块中的可执行语句可以是直接由语言环境解释的内联语句，或可以是对一个外部程序的调用。

Function 块中的 EXEC 块：如果在 FUNCTION 块内使用 EXEC 块，则它必须是 FUNCTION 块中唯一的可执行语句。在把此可执行语句传递给语言环境之前，Net.Data 把 EXEC 块中此程序的文件名追加到由初始化文件中的 EXEC_PATH 配置语句所确定的路径名。结果字符串被传递到语言环境去执行。

语言环境用于处理 EXEC 块的方法依赖于特定的语言环境。只有 REXX、System 和 Perl Net.Data 提供的语言环境支持 EXEC 块。

在语言语句中使用特殊字符：当匹配 Net.Data 语言结构语法的字符在函数块的语言结构节中作为一部分语法上有效的嵌入程序码(例如 REXX 或 Perl)使用时，它们可能被作为 Net.Data 语言结构而被误解，因而导致错误或宏中不可预测的结果。

例如，Perl 函数可能使用 COMMENT 块定界符 %{。运行宏时，%{ 被作为 COMMENT 块的开头来解释。然后 Net.Data 查找 COMMENT 块的结尾，当它读到函数块结尾时就认为是找到了。Net.Data 然后继续查找函数块的结尾，但当找不到时，就发出一个错误。

使用以下方法之一将 Net.Data 特殊字符用作嵌入程序代码的一部分，而不让它们被 Net.Data 作为特殊字符来解释：

- 使用 EXEC 语句来调用程序代码，而不是将代码放在内部。
- 使用一个变量引用来指定特殊字符。

例如，以下 Perl 函数包含表示一个 COMMENT 块定界符 %{ 的字符作为 Perl 语言语句的一部分：

```
%function(DTW_PERL) func() {  
    ...  
    for $num_words (sort bynumber keys %{ $Rtitles{$num} }) {  
        &make_links($Rtitles{$num}){$num_words});  
    }  
    ...  
    %}
```

要保证 Net.Data 将 %{ 字符作为 Perl 源码而不是作为 Net.Data COMMENT 块定界符，可以用以下方式之一重写函数：

- 使用 %EXEC 语句：

```
%function(DTW_PERL) func() {  
    %EXEC{ func.pr1 %}  
    %}
```

- 使用一个变量引用来指定 %{ 字符：

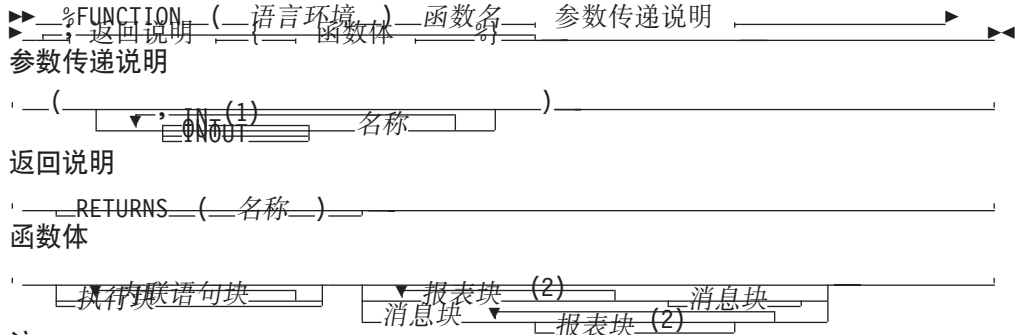
```
%define percent_openbrace = "%{"  
  
%function(DTW_PERL) func() {  
    ...  
    for $num_words (sort bynumber keys ${percent_openbrace} $Rtitles{$num} }) {
```

```

        &make_links($Rtitles{$num}{$num_words});
    }
    ...
%}

```

语法



注:

1. 如果参数列表的开始没有指定参数类型，则应用缺省的参数类型 **IN**。没有参数类型的参数使用参数列表中最近指定的类型，如果还未指定过，则使用 **IN** 类型。例如，参数列表 (*parm1*、**INOUT** *parm2*、*parm3*、**OUT** *parm4*、*parm5*)，其中参数 *parm1*、*parm3* 和 *parm5* 没有参数类型。参数 *parm1* 的类型是 **IN**，因为还没有指定过初始参数类型。参数 *parm3* 的类型是 **INOUT**，因为它是最近指定的参数类型。类似地，参数 *parm5* 的类型是 **OUT**，因为它是参数列表中最近指定的类型。
2. 重复的报表块适用于:
 - 在处理返回多个结果集的存储过程时的数据库语言环境(对于 OS/2、Windows NT 和 UNIX 操作系统)。
 - 调用任何 OS/400 操作系统的语言环境的函数。

值

%FUNCTION

指定 `Net.Data` 从宏文件中调用的一个子例行程序的关键字。

lang_env

处理函数体的语言环境。参阅 `Net.Data` 语言环境参考获取更多的信息。

function_name

定义函数的名称可以是一个字母或数字字符串，它以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

名称

一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

参数传递说明:

IN 指定 `Net.Data` 把输入数据传送至语言环境。 **IN** 是缺省。

OUT

指定语言环境把输出数据返回至 `Net.Data`。

INOUT

指定 `Net.Data` 把输入数据传送至语言环境，语言环境再把输出数据返回给 `Net.Data`。

返回说明:

RETURNS

声明在函数完成后, 包含语言环境指定的函数值的变量。

函数体:

内联语句块

来自函数定义中指定的语言环境(例如: REXX、SQL 或 Perl)、语法上有效的语句。 参阅 *Net.Data* 语言环境参考中对使用的语言环境的说明。参见程序设计语言中对语法和用法的程序设计参考。代表内联语句块的字符串可以包含 Net.Data 变量引用和函数调用, 这可以在执行内联语句块之前先进行求值。**限制:** 不带任何 Net.Data 变量引用或函数调用的最长的连续内联语句块限制在以下长度:

- 对于 OS/2 和 NT: 64KB
- 对于 AIX: 256KB
- 对于 OS/390: 256KB
- 对于 OS/400: 256KB

执行块

EXEC 块。一个外部程序的名称, 在引用一个变量或调用一个函数时执行这个程序。 请参阅第13页的『EXEC 块或语句』中的语法和例子。

报表块

REPORT 块。用于格式化函数调用的输出结果的一系列命令。您还可以在报表中使用标题和脚注。 请参阅第43页的『REPORT 块』中的语法和例子。

消息块

MESSAGE 块。当从一个函数调用返回时使用的一组返回码、相关消息和 Net.Data 采用的操作。 请参阅第39页的『MESSAGE 块』中的语法和例子。

上下文

可以在这些上下文中找到 FUNCTION 块:

- IF 块
- 位于 Net.Data 宏声明部分的任何块或者语句之外。

限制

FUNCTION 块可以包含这些成份:

- 注解块
- EXEC 块
- MESSAGE 块
- REPORT 块
- Inline 语句块

只有 REXX、System 和 Perl Net.Data 提供的语言环境支持 EXEC 语句。

例

以下例子是一般例子，没有覆盖所有的语言环境。参阅 *Net.Data* 语言环境参考获取更多使用具有特定语言环境的 FUNCTION 块的信息。

例 1: 一个 REXX 子串函数

```
%DEFINE lstring = "longstring"
%FUNCTION(DTW_REXX) substrng(IN x, y, z) RETURNS(s) {
    s = substr("$x)", $(y), $(z));
}%
%DEFINE a = {@substrng(lstring, "1", "4")%} %{ assigns "long" to a %}
```

在对 *a* 求值时，发现 @substrng 函数调用，并执行子串 FUNCTION 块。变量在 FUNCTION 块中的可执行语句中被替换，然后文本字符串 *s* = substr("longstring", 1, 4) 传送给 REXX 解释器执行。因为指定 RETURNS 子句，在 *a* 的求值过程中 @substrng 函数调用的值用 『long』进行替换，其值为 *s*。

例 2: 调用一个外部的 REXX 程序

- Net.Data 宏:

```
%FUNCTION(DTW_REXX) my_rexx_pgm(INOUT a, b, IN c, OUT d) {
    %EXEC{ mypgm.cmd this is a test %}
}%
%HTML(INPUT) {
    <P> Original variable values: $(w) $(x) $(z)
    <P> @my_rexx_pgm(w, x, y, z)
    <P> Modified variable values: $(w) $(x) $(z)
}%
```

变量 *w* 和 *x* 对应于函数中的 INOUT 参数 *a* 和 *b*。对应于 IN 参数 *c*，它们的值和 *y* 的值应当已经从 HTML 格式输入或 DEFINE 语句中定义。当参数 *a* 和 *b* 返回值时，变量 *a* 和 *b* 被赋予新值。当 OUT 参数 *d* 返回值时变量 *z* 接受定义。

- REXX 程序 mypgm.cmd:

```
/* Sample REXX Program for Example 2 */
/* Test arguments */
num_args = arg();
say 'There are' num_args 'arguments';
do i = 1 to num_args;
    say 'arg' i 'is "'arg(i)'"
end;
/* Set variables passed from Net.Data */
d = a || b || c; /* concatenate a, b, and c forming d */
a = ''; /* reset a to null string */
b = ''; /* reset b to null string */
return;
```

- mypgm.cmd 的输出:

```
There are 1 arguments
arg 1 is "this is a test"
```

EXEC 语句通知 REXX 语言环境，进而告诉 REXX 解释程序，让它执行外部的 REXX 程序 mypgm.cmd。因为 REXX 语言环境可以直接与 REXX 程序共享 Net.Data 变量，所以在执行 mypgm.cmd 之前，将 REXX 变量 *a*、*b* 和 *c* 的值赋成 Net.Data 变量 *w*、*x* 和 *y* 的值。mypgm.cmd 可以直接使用 REXX 语句中的 *a*、*b* 和 *c* 变量。当程序终止时，REXX 变量 *a*、*b* 和 *d* 在 REXX 程序中接受检索，其值赋予 Net.Data 变量

w、x 和 z。因为 my_rexx_pgm FUNCTION 块的定义中未使用 RETURNS 子句，所以如果返回代码是 0，则 @my_rexx_pgm 函数调用的值是空字符串『』；如果返回代码非 0，则是此 REXX 程序返回代码的值。

例 3: 一个 SQL 查询和报表

```
%FUNCTION(DTW_SQL) query_1(IN x, IN y) {
    SELECT customer.num, order.num, part.num, status
    FROM customer, order, shippingpart
    WHERE customer.num = '$(x)'
      AND customer.ordernumber = order.num
      AND order.num = '$(y)'
      AND order.partnumber = part.num
%REPORT{
    <P>Here is the status of your order:
    <P>$(NLIST)
<UL>
%ROW{
    <LI>$(V1) $(V2) $(V3) $(V4)
    %}
</UL>
    %}
%}
%DEFINE customer_name="IBM"
%DEFINE customer_order="12345"
%HTML(REPORT) {
    @query_1(customer_name, customer_order)
%}
```

@query_1 函数调用用 SELECT 语句中的值把 IBM 替换为 \$(x)，12345 替换为 \$(y)。因为 SQL 函数 query_1 的定义没有指明输出表格变量，因此将使用缺省的表格(请参阅 TABLE 变量块中的细节)。在 REPORT 块中引用的 NLIST 和 Vi 变量是由缺省的表格定义所定义的。由 REPORT 块产生的报表放置在输出的 HTML 中，在其中调用 query_1 函数。

例 4: 执行 Perl 脚本的系统调用

- Net.Data 宏:

```
%FUNCTION(DTW_SYSTEM) today() RETURNS(result) {
    %exec{ perl "today.prl" %}
    %}
%HTML(INPUT) {
    @today()
    %}
```

- Perl 程序 today.prl:

```
$date = 'date';
chop $date;
open(DTW, "> $ENV{DTWPIPE}") || die "Could not open: $!";
print DTW "result = \"\$date\"\n";
```

System 语言环境解释 FUNCTION 块中的可执行语句的方法是：通过 C 语言中的 system() 函数调用将它们传递给操作系统。这种方法不允许将 Net.Data 变量直接传递到可执行语句，也不允许可执行程序直接检索这些变量，就象在 REXX 语言环境中一样，因此，System 语言环境按以下方式传递和检索变量：

- 输入参数象系统环境变量那样被传递到 putenv() 函数，并可以被可执行程序所检索。不同的语言引用变量的方法是不同的。UNIX cshell 脚本通过在环境变量名前面添加 '\$' 来引用此环境变量，例如 \$x。而 Perl 语言环境通过引用关联数组 %ENV 来引用环境变量，例如 %ENV{'x'}。DOS 批处理文件(.BAT)引用括在百分号中的变量名，例如 %x%。

- 将输出参数传回语言环境的方法是：将结果写入在环境变量 DTWPIPE 中传递的管道名称中，但 OS/400 平台除外，其输出参数作为系统环境变量传回语言环境。写入已命名管道的数据其格式是 name="value"，和 DEFINE 语句相同。如果一个与输出参数对应的变量名是以这种方式写入的，则新的值将替换当前值。如果写入的变量名不与一个输出参数相对应，则被忽略。

遇到 @today 函数调用时，Net.Data 在可执行语句上执行变量替换。对于此例，可执行语句中没有 Net.Data 变量，因此没有执行变量替换。可执行语句和参数被传递给 System 语言环境，它将创建一个命名管道，并将环境变量 DTWPIPE 设置为管道的名称。

然后通过 C 语言的 system() 函数调用来调用外部程序。外部程序以只读方式打开管道，并象对一个标准流式文件一样，把输出参数的值写入管道。外部程序通过写至 STDOUT，生成 HTML 输出。在本例子中，系统日期程序的输出被赋给结果变量，即由 FUNCTION 块的 RETURNS 子句所标识的变量。结果变量的值替换 HTML 块中的 @today() 函数调用。

例 5: Perl 语言环境

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

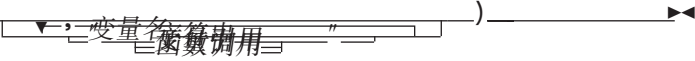
把这个例子和例 4 相比较，看如何使用 EXEC 块。例 4 中，System (系统)语言环境并不了解如何解释 Perl 程序，但是此语言环境知道如何调用外部程序。EXEC 块告诉它调用称为 perl 的一个外部程序。实际的 Perl 语言语句是由外部 Perl 程序解释的。例 5 没有 EXEC 块，因为 Perl 语言环境能够直接解释 Perl 语言语句。

函数调用 (@)

目的

调用先前已定义的 FUNCTION 块、 MACRO_FUNCTION 块或以指定的参数调用内部函数。如果函数不是内部函数，则必须在指定函数调用之前，先在 Net.Data 宏中定义它。

语法

» @function_name ()

值

@function_name

任何现有函数的名称。一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

变量名

一个或多个名称，每个附加的名称用一个句点(.)相连。请参阅第4页的『变量名』中的语法信息。

字符串

字母、数字字符和标点符号的任意序列，换行字符除外。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。

上下文

可以在这些上下文中找到函数调用:

- HTML 块
- REPORT 块
- ROW 块
- DEFINE 块
- IF 块
- MACRO_FUNCTION 块
- MESSAGE 块
- WHILE 块
- 函数调用语句
- 位于 Net.Data 宏声明部分的任何块之外

限制

- 函数调用可以包含这些成份:
 - 注解块

- 字符串
- 函数调用
- 变量引用
- 函数调用不能包含任何定义给函数定义中的 OUT 或 INOUT 参数的变量引用或函数调用。

例

例 1: 对 SQL 函数 formQuery 的一个调用

```
%FUNCTION(DTW_SQL) formQuery(){
SELECT $(queryVal) from $(tableName)
%}

%HTML (input){
<P>Which columns of $(tableName) do you want to see?
<FORM METHOD="POST" ACTION="report">
<INPUT NAME="queryVal" TYPE="CHECKBOX" VALUE="NAME">Name
<INPUT NAME="queryVal" TYPE="CHECKBOX" VALUE="MAIL">E-mail
<INPUT NAME="queryVal" TYPE="CHECKBOX" VALUE="FAX">FAX
<INPUT TYPE="SUBMIT" VALUE="Submit request">
%}

%HTML (report){
<P>Here are the columns you selected:
<HR>@formQuery()
%}
```

例 2: 对具有输入和输出参数的 REXX 函数的调用

```
%FUNCTION(DTW_REXX) my_rexx_pgm(INOUT a, b, IN c, OUT d) {
%EXEC{ mypgm.cmd this is a test %}
%}
%HTML(INPUT) {
<P> Original variable values: $(w) $(x) $(z)
<P> @my_rexx_pgm(w, x, y, z)
<P> Modified variable values: $(w) $(x) $(z)
%}
```

例 3: 对使用变量引用和函数调用的 REXX 函数的一个调用，但没有参数

```
%FUNCTION(DTW_REXX) my_rexx_pgm(IN a, b, c, d, OUT e) {
...
%}
%HTML(INPUT) {
<p> @my_rexx_pgm$(myA), @getB(), @retrieveC(), $(myD), myE)
%}
```

HTML 块

目的

包含任何将由客户的 Web 浏览器或任何可以理解 HTML 的工具进行处理的 HTML 标记或文本。 HTML 块还包含大部分 Net.Data 宏语言语句，这些语句在运行时求值和执行。Net.Data 查找 Net.Data 宏语句并执行之。 Net.Data 假定所有其它的文本都是 HTML 的，并把它发送给 Web 服务器。

语法



值

%HTML

此关键字用于指定包含了要在客户浏览器中显示的 HTML 标记和文本的块。

名称

一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合，包括句点(除了 OS/390)。

exec_sql 语句

DB2WWW 发行版 1 语言成份，它保证了兼容性。请参阅第235页的『附录A. DB2 WWW Connection』或 DB2 World Wide Web Release 1 文档。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有单个的前导加号(+)或者减号(-)。参阅第25页的『IF 块』中的语法和例子。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

include_url 语句

INCLUDE_URL 语句。读取并将其它文件包含到指定此语句的 Net.Data Web 宏中。指定的文件可以存在于本地或者远程服务器上。请参阅第32页的『INCLUDE_URL 语句』中的语法和例子。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

上下文

可以在这些上下文中找到 HTML 块:

- IF 块
- 位于 Net.Data 宏声明部分的任何块之外

限制

HTML 块可以包含这些成份:

- 注解块
- EXEC_SQL 语句
- IF 块
- HTML 语句
- INCLUDE 语句
- INCLUDE_URL 语句
- WHILE 块
- 变量引用
- 函数调用

例

例 1: 带页眉和页脚的包含文件的 HTML 块

```
%HTML(example1){
%INCLUDE"header.html"
<P>You can put <EM>any</EM> HTML in an HTML block.
An SQL function call is made like this:
@xmpl()
%INCLUDE"footer.html"
%}
```

例 2: 名称中包含句点的 HTML 块

```
%HTML(my.report){
%INCLUDE"header.html"
<P>You can put <EM>any</EM> HTML in an HTML block.
An SQL function call is made like this:
@xmpl()
%INCLUDE"footer.html"
%}
```

IF 块

目的

可以执行有条件的字符串处理。 IF 块提供测试一个或多个条件的能力，然后基于条件测试的结果执行一个语句块。您可以在一个 Net.Data 宏的声明部分、HTML 块、MACRO_FUNCTION 块、REPORT 块、 WHILE 块、 ROW 块中使用 IF 块，也可嵌套在另一个 IF 块内部。

如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。 它们可以有一个单个的前导加号(+)或者减号(-)。

限制: Net.Data 不支持非整型数的数值比较。例如浮点数。

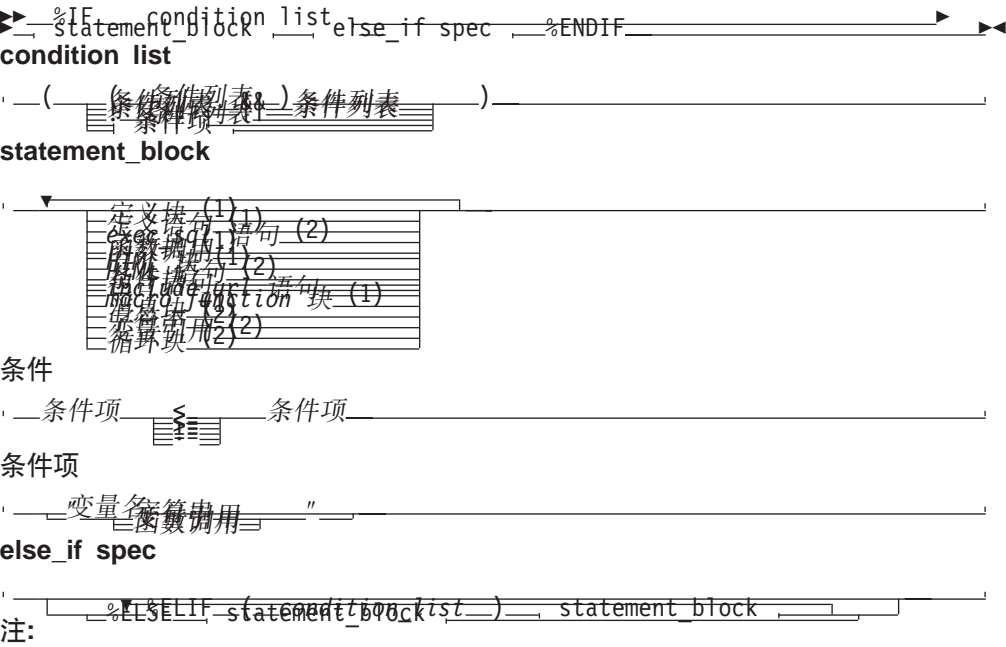
嵌套的 IF 块: IF 块的语法规则是由块在宏文件中的位置确定的。如果 IF 块嵌套在一个 IF 块内，而后者在声明部分中任何其它块的外部，则它可以使用外部块可以使用的任何元素。如果 IF 块嵌套在另一个嵌套在某个 IF 块的块中，则它遵循它所处的那个模块的语法规则。

在下面的例子中，嵌套的 IF 块必须遵循在它处于一个 HTML 块中时使用的规则。

```
%IF 块
...
%HTML 块
...
    %IF 块
```

参阅本章节后面列出的限制。

语法



- 1. 如果 IF 块位于宏的声明部分中任何其它块的外面，则此语言结构有效。
- 2. 当 IF 块位于 HTML 块、MACRO_FUNCTION 块、REPORT 块、ROW 块或 WHILE 块中时，此语言结构有效。

值

%IF

用于指定有条件的字符串处理的关键字。

条件列表

比较条件和条件项的值。条件列表可以使用布尔运算符进行连接。一个条件列表可以嵌套在另一个条件列表中。

statement_block

下列有效的 Net.Data 宏结构。请参阅图例注解和限制确定上下文，使Plea其中的宏结构有效。

定义语句

DEFINE 块或语句。用于定义变量和设置配置变量。变量名必须以一个字母或下划线(_)开头，可以包含任何字母数字字符或下划线。请参阅第9页的『DEFINE 块或语句』中的语法和例子。

exec_sql 语句

DB2WWW 发行版 1 语言成份，它保证了兼容性。请参阅第235页的『附录A. DB2 WWW Connection』或 DB2 World Wide Web Release 1 文档。

函数块

指定可以从 Net.Data 宏中进行调用的子程序的关键字。FUNCTION 块中的可执行语句可以包含直接由语言环境解释的语言语句，或可以指示对一个外部程序的调用。请参阅第15页的『FUNCTION 块』中的语法和例子。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 块

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有一个单个的前导加号(+)或者减号(-)。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

include_url 语句

INCLUDE_URL 语句。读取并将其它文件包含到指定此语句的 Net.Data Web 宏中。指定的文件可以存在于本地或者远程服务器上。请参阅第32页的『INCLUDE_URL 语句』中的语法和例子。

macro_function 块

指定可以从 Net.Data 宏中进行调用的子程序的关键字。MACRO_FUNCTION 块中的可执行语句可以包含 Net.Data 宏语言源语句。参阅第36页的『MACRO_FUNCTION 块』中的语法和例子。

消息块

MESSAGE 块。当从一个函数调用返回时使用的一组返回码、相关消息和 Net.Data 采用的操作。请参阅第39页的『MESSAGE 块』中的语法和例子。

字符串

字母、数字字符和标点符号的任意序列。如果此字符串在条件列表的条件项中，则它可以包含任何字符，新建行字符除外。如果此字符串在代码的可执行块中，则它可以包含任何字符，包括新建行字符。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

条件

两个条件项之间使用比较运算符的一个比较表达式。如果下面两个条件都为真，则 IF 条件看成是一个数值比较：

- 条件操作符是下列操作符之一：<,<=,>,>=,==,!=
- 两个条件项都是代表整数的字符串，其中有效整数是一个数字字符串，可选地，前面可以有前导加号(+)或减号(-)，并且没有空格。

如果两者有一个不为真，则执行一个一般的字符串比较。

条件项

变量名、字符串、变量引用或函数调用。

%ELIF

启动可选处理路径的关键字，可以包含条件列表和大多数 Net.Data 宏语句。

%ENDIF

一个用于结束 %IF 块的关键字。

%ELSE

用于在所有其它的条件列表都不满足时执行相关语句的一个关键字。

上下文

可以在这些上下文中找到 IF 块：

- 位于 Net.Data 宏声明部分中的任何其它块之外
- HTML 块
- IF 块
- MACRO_FUNCTION 块
- REPORT 块
- ROW 块
- WHILE 块

限制

当 IF 块位于 Net.Data 宏声明部分中的任何其它块之外时，它可以包含这些成份：

- 注解块
- DEFINE 块
- DEFINE 语句
- FUNCTION 块
- 函数调用
- HTML 块
- IF 块
- INCLUDE 语句
- INCLUDE_URL 语句
- MACRO_FUNCTION 块
- MESSAGE 块
- 变量引用

如果 IF 块位于 Net.Data 宏的 HTML 块、MACRO_FUNCTION 块、REPORT 块、ROW 块或 WHILE 块中，则它可以包含这些成份：

- 注解块
- EXEC_SQL 语句
- 函数调用
- IF 块
- INCLUDE 语句
- INCLUDE_URL 语句
- HTML 语句
- 字符串
- 变量引用
- WHILE 块

例

例 1: 在一个 Net.Data 宏的声明部分中的 IF 块

```
%DEFINE a = "1"
%DEFINE b = "2"
...
%IF ($(DTW_HTML_TABLE) == "YES")
%define OUT_FORMAT = "HTML"
%ELSE
%define OUT_FORMAT = "CHARACTER"
%ENDIF

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

例 2: 在一个 HTML 块内部的 IF 块

```
%HTML(REPORT){
@myFunctionCall()
%IF ($RETURN_CODE) == $(failure_rc))
  <P> The function call failed with failure code $(RETURN_CODE).
%ELIF ($RETURN_CODE) == $(warning_rc))
```



```

    <P> The function call succeeded with warning code $(RETURN_CODE).
%ELIF ($(RETURN_CODE) == $(success_rc))
    <P>The function call was successful.
%ELSE
    P>The function call returned with unknown return code $(RETURN_CODE).
%ENDIF
%}

```

例 3: 一个数值型比较

```

%IF (ROW_NUM < "100")
    <p>The table is not full yet...
%ELIF (ROW_NUM == "100")
    <p>The table is now full...
%ELSE
    <p>The table has overflowed...
%ENDIF

```

因为隐式表格变量 `ROW_NUM` 总是返回一个整数值，并且正在比较的值也是一个整数，所以执行数值比较。

例 4: 嵌套的 IF 块

```

%IF (MONTH == "January")
    %IF (DATE = "1")
        HAPPY NEW YEAR!
    %ELSE
        Ho hum, just another day.
    %ENDIF
%ENDIF

```

INCLUDE 语句

目的

读取并将一个文件结合到其中指定这条语句的 `Net.Data` 宏中。

`Net.Data` 将搜索在初始化文件的 `INCLUDE_PATH` 语句中所指定的目录，以查找此包含文件。

可以按照与大部分高级语言所使用的相同方式使用包含文件。它们可以用于插入公用标题和脚注、定义公用的变量集合，或将 `FUNCTION` 模块定义的公用子程序库包含到 `Net.Data` 宏中。

`Net.Data` 在处理宏时只执行 `INCLUDE` 语句一次，并把包含进来的文件的内容插入宏文件中 `INCLUDE` 语句的位置。在包含文件的名称中的任何变量引用都在第一次执行 `INCLUDE` 语句、而不是在要执行包含文件的内容时进行分析的。

当 `INCLUDE` 语句在一个 `ROW` 或 `WHILE` 块中时，`Net.Data` 不重复执行 `INCLUDE` 语句。`Net.Data` 在它第一次执行 `ROW` 或 `WHILE` 块时执行 `INCLUDE` 语句，并把包含文件的内容结合到这个块中，然后用包含文件的内容重复执行 `ROW` 或 `WHILE` 块。

权限技巧：请确保执行 `Net.Data` 的用户 ID 对 `INCLUDE` 语句引用的任何文件都具有访问权限。请参阅 *Net.Data 管理和程序设计指南* 的配置一章中关于指定 Web 服务器对 `Net.Data` 文件的访问权限这一部分，以获取更多信息。

提示：如果想要包含本地 Web 服务器中的某个 HTML 文件，则如例 3 中所示的 `INCLUDE_URL` 使用 `INCLUDE_URL` 结构。通过使用所演示的语法，您不必更新 `Net.Data` 初始化文件中的 `INCLUDE_PATH` 来指定其实 Web 服务器已经知道的目录。

语法

►► `%INCLUDE_` " 变量引用 " ◀◀
值

`%INCLUDE`

指示要被读取并结合到 `Net.Data` 宏中的某个文件的关键字。

名称

一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

字符串

字母、数字字符和标点符号的任意序列，换行字符除外。

变量引用

返回一个先前定义的变量的值，用 `$` 和 `()` 来指定。例如：如果 `VAR='abc'`，则 `$(VAR)` 返回值 `'abc'`。请参阅第4页的『变量引用』中的语法信息。

上下文

可以在这些上下文中找到 `INCLUDE` 语句：

- `DEFINE` 块
- `HTML` 块

- REPORT 块
- ROW 块
- IF 块
- MESSAGE 块
- MACRO_FUNCTION 块
- WHILE 块
- 位于 Net.Data 宏声明部分的任何块之外

限制

INCLUDE 语句可以包含这些成份:

- 注解块
- 字符串
- 变量引用

字符串中的函数调用是不允许的。

例

例 1: 在一个 HTML 块中的 INCLUDE 语句

```
%HTML(start){
%INCLUDE "header.hti"
...
%}
```

例 2: 在一个 REPORT 块中的 INCLUDE 语句

```
%REPORT {
%INCLUDE "report_header.txt"
%ROW {
%INCLUDE "row_include.txt"
%}
%INCLUDE "report_footer.txt"
%}
```

例 3: 在一个 INCLUDE 语句中的变量引用

```
%define library = "/qsys.lib/mylib.lib/"
%define filename = "macros.file/incfile.mbr"

%include "${library}${filename}"
```

INCLUDE_URL 语句

目的

读取并将另一个文件结合到其中指定这条语句的 Net.Data 生成的输出中。指定的文件可以存在于本地或者远程服务器上。

使用 INCLUDE_URL 语句，可以从另一个宏中调用一个宏，而不用要求应用程序用户选择“递交”按钮。

Net.Data 在处理宏时只执行 INCLUDE_URL 语句一次，并把包含进来的文件的内容插入宏文件中 INCLUDE_URL 语句的位置。在包含文件的名称中的任何变量引用都在第一次执行 INCLUDE_URL 语句、而不是在要执行包含文件的内容时进行分析的。

当 INCLUDE_URL 语句在一个 ROW 或 WHILE 块中时，Net.Data 不重复执行 INCLUDE_URL 语句。Net.Data 在它第一次执行 ROW 或 WHILE 块时执行 INCLUDE_URL 语句，并把包含文件的内容结合到这个块中，然后用包含文件的内容重复执行 ROW 或 WHILE 块。

语法

►► %INCLUDE_URL " ~~变量引用~~ " ◀◀
值

%INCLUDE_URL

指示要被读取并结合到本地或远程服务器上的 Net.Data 宏中的某个文件的关键字。

字符串

字母、数字字符和标点符号的任意序列，换行字符除外。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

上下文

可以在这些上下文中找到 INCLUDE_URL 语句：

- HTML 块
- REPORT 块
- ROW 块
- WHILE 块
- MACRO_FUNCTION 块
- 位于 Net.Data 宏声明部分的任何块之外

限制

INCLUDE_URL 语句可以包含这些成份：

- 注解块
- 字符串
- 变量引用

INCLUDE_URL 文件具有以下文件尺寸限制:

- OS/2 和 Windows NT: 64 KB
- AIX: 256 KB
- OS/390: 256 KB

INCLUDE_URL 在 OS/400 环境中是不受支持的。

例

例 1: 包含一个在另一台服务器上的 HTML 文件

```
%include_url "http://www.ibm.com/path/myfile.html"
```

例 2: 通过调用一个远程服务器的名称, 包含一个在此远程服务器上的 HTML 文件

```
%include_url "myserver/path/myfile.html"
```

其中 myserver 是服务器名称。

例 3: 包含一个在本地 Web 服务器上的 HTML 文件

```
%include_url "/path/myfile.html"
```

提示: 使用这种方法, 您不必更新 Net.Data 配置文件中的 INCLUDE_URL 路径来指定 Web 服务器其实已经知道的目录。如果此字符串 不是以斜杠开头的, 则 Net.Data 假定此字符串是一个服务器名称, 并以相应的名称试图在服务器上检索此文件。

例 4: 包含在一台远程服务器上的其它 Net.Data 宏

```
%REPORT{  
<P>Current hot pick as of @DTW_rTIME():  
%include_url "http://www.ibm.com/cgi-bin/db2www/hotpic.mac/report?custno=$(custno)"
```

在此例子中, 调用宏文件 hotpic.mac, 并把 custno 作为变量传递。如果此字符串是以斜杠开头的, 则 Net.Data 在本地 Web 服务器上检索此 INCLUDE 文件。

LIST 语句

目的

构建一张定界的值的列表。在构建带多个条件项(例如在某些 WHERE 或 HAVING 子句中可找到的项目)的 SQL 查询时, 可以使用 LIST 语句。

语法

►► %LIST " " 函数调用 " 变量名 ◀◀

值

%LIST

此关键字指定了要用于构建一个带定界符的值列表的变量。

字符串

字母、数字字符和标点符号的任意序列, 换行字符除外。

变量引用

返回一个先前定义的变量的值, 用 \$ 和 () 来指定。例如: 如果 VAR='abc', 则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块, 或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

变量名

一个或多个名称, 每个附加的名称用一个句点(.)相连。请参阅第4页的『变量名』中的语法信息。

上下文

可以在这些上下文中找到 LIST 语句:

- DEFINE 语句

限制

LIST 语句可以包含这些成份:

- 注解块
- 变量引用
- 函数调用
- 字符串

例

例 1: 一张变量列表

```
%DEFINE{
DATABASE="custcity"
%LIST " OR " conditions
conditions="cond1='Sao Paolo'"
conditions="cond2='Seattle'"
}
```

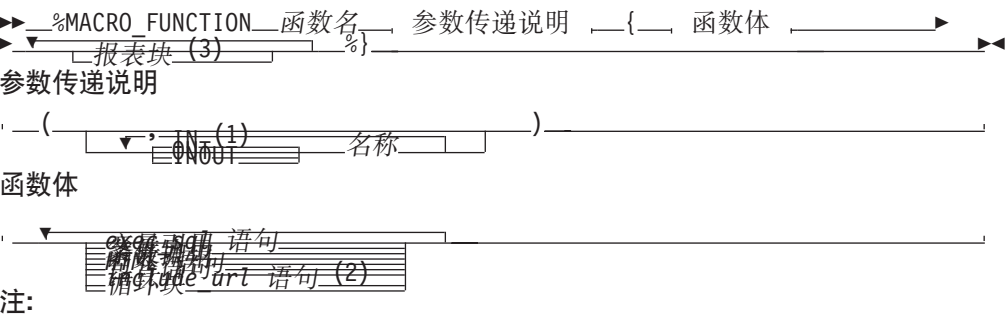
```
| conditions="cond3='Shanghai'"  
| whereClause=conditions ? "WHERE $(conditions)" : ""  
| %}
```

MACRO_FUNCTION 块

目的

定义一个可以从 Net.Data 宏中进行调用的子程序。MACRO_FUNCTION 块中的可执行语句必须是 Net.Data 宏语言源语句。

语法



1. 如果参数列表的开始没有指定参数类型，则应用缺省的参数类型 IN。没有参数类型的参数使用参数列表中最近指定的类型，如果还未指定过，则使用 IN 类型。例如，参数列表 (parm1、INOUT parm2、parm3、OUT parm4、parm5)，其中参数 parm1、parm3 和 parm5 没有参数类型。参数 parm1 的类型是 IN，因为还没有指定过初始参数类型。参数 parm3 的类型是 INOUT，因为它是最近指定的参数类型。类似地，参数 parm5 的类型是 OUT，因为它是参数列表中最近指定的类型。
2. OS/400 不支持 INCLUDE_URL 语句。
3. 只有 OS/400 的 MACRO_FUNCTION 块中才支持 REPORT 块。重复的报表块适用于调用任何语言环境的函数。

值

%MACRO_FUNCTION

指定可以从 Net.Data 宏中进行调用的子程序的关键字。MACRO_FUNCTION 块中的可执行语句必须包含 Net.Data 直接解释的语言语句。

function_name

要定义的函数名称。一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

参数传递说明:

IN 指定 Net.Data 把输入数据传送至语言环境。IN 是缺省。

OUT

指定语言环境把输出数据返回至 Net.Data。

INOUT

指定 Net.Data 把输入数据传送至语言环境，语言环境再把输出数据返回给 Net.Data。

名称

一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。名称可以表示一个 Net.Data 表或一个结果集。

函数体:

exec_sql

DB2WWW 发行版 1 语言成份, 它保证了兼容性。请参阅第235页的『附录A. DB2 WWW Connection』或 DB2 World Wide Web Release 1 文档。

变量引用

返回一个先前定义的变量的值, 用 \$ 和 () 来指定。例如: 如果 VAR='abc', 则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字符串, 并且没有前导或尾随空格, 则作为数值型进行比较。它们可以有一个前导加号(+)或者减号(-)。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块, 或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

include_url 语句

INCLUDE_URL 语句。读取并将另一个文件结合到其中指定这条语句的 Net.Data 宏中。指定的文件可以存在于本地或者远程服务器上。请参阅第32页的『INCLUDE_URL 语句』中的语法和例子。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

报表块

REPORT 块。用于格式化函数调用的输出结果的一系列命令。您还可以在报表中使用标题和脚注。请参阅第43页的『REPORT 块』中的语法和例子。

上下文

可以在这些上下文中找到 MACRO_FUNCTION 块:

- IF 块
- 位于 Net.Data 宏声明部分的任何块之外

限制

此结构在 OS/390 操作系统中不可用。

MACRO_FUNCTION 块可以包含这些成份:

- 注解块
- EXEC_SQL 语句
- HTML 语句
- IF 块

- INCLUDE 语句
- INCLUDE_URL 语句
不支持 OS/400
- REPORT 块
只支持 OS/400
- WHILE 块
- 变量引用
- 函数调用

例

例 1: 指定消息处理的一个宏函数

```
%MACRO FUNCTION setMessage(IN rc, OUT message) {
%IF (rc == "0")
    @dtw_assign(message, "Function call was successful.")
%ELIF (rc == "-1")
    @dtw_assign(message, "Function failed, out of memory.")
%ELIF (rc == "-2")
    @dtw_assign(message, "Function failed, invalid parameter.")
%ENDIF
%}
```

例 2: 指定标题信息的一个宏函数

```
%MACRO FUNCTION setup(IN browserType) {
%{ call this function at the top of each HTML block in the macro %}
%INCLUDE "header_info.html"
@dtw_rdate()
%IF (browserType == "IBM")
    @setupIBM()
%ELIF (browserType == "MS")
    @setupMS()
%ELIF (browserType == "NS")
    @setupNS()
%ELSE
    @setupDefault()
%ENDIF
%}
```

MESSAGE 块

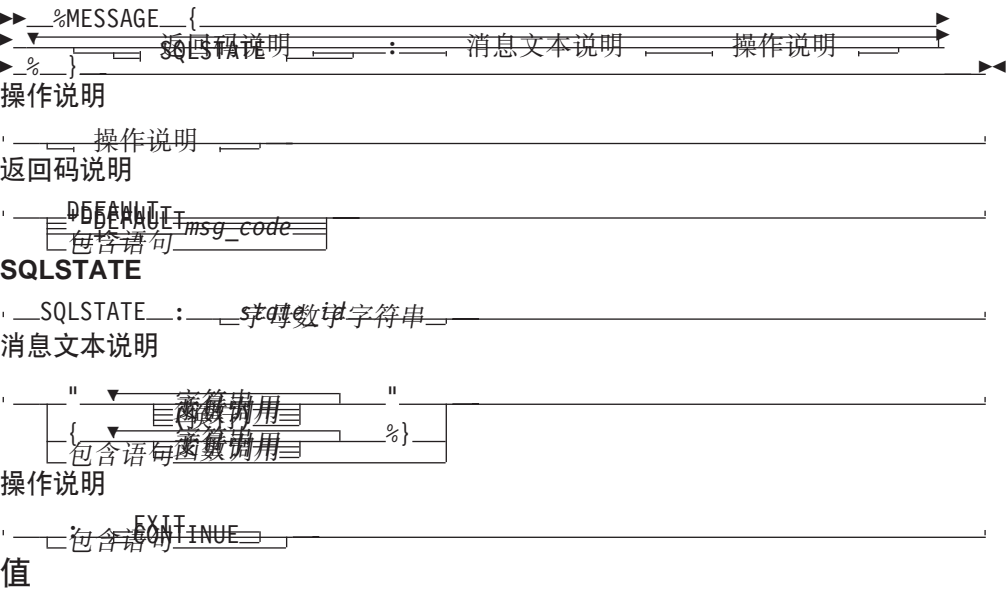
目的

指定了要显示的消息，并根据来自函数的返回码要执行的操作。

在 MESSAGE 块中定义返回码的集合，及其相应的消息和操作。当完成一个函数调用时，Net.Data 把其返回码和 MESSAGE 块中定义的返回码进行比较。如果函数的返回码与 MESSAGE 模块中的某个返回码相匹配，则 Net.Data 显示相应的消息并对操作进行求值，确定是继续处理还是退出 Net.Data 宏。

MESSAGE 块在作用域上可以是全局的、或局部于一个 FUNCTION 块。如果 MESSAGE 块是在最外层定义的，则认为它的作用域是全局的。当定义多个全局 MESSAGE 块时，只有最后一个被处理的模块是活动的块。如果 MESSAGE 块是在 FUNCTION 块中定义的，则此块的作用域局限于定义它的 FUNCTION 块。请参阅 Net.Data 管理和程序设计指南 中的 MESSAGE 块，获取有关返回码的处理规则。

语法



%MESSAGE

使用此关键字的块定义了一组返回码、相应的消息以及在函数调用返回时 Net.Data 所要采用的操作。

返回码说明

一个正整数或负整数。如果 Net.Data RETURN_CODE 变量的值与返回码说明值相匹配，则使用消息语句中的剩余信息来处理这个函数调用。您也可以为返回码指定消息，而不需要在 MESSAGE 块中特别输入。

+DEFAULT

一个用于指定缺省的正消息代码的关键字。如果 RETURN_CODE 大于 0 并且未指定精确匹配，则 Net.Data 使用这条消息语句中的信息来处理函数调用。

-DEFAULT

指定缺省的负消息代码的关键字。如果 RETURN_CODE 小于 0 并且未指定精确匹配，则 Net.Data 使用这条消息语句中的信息来处理函数调用。

DEFAULT

指定缺省消息代码的关键字。如果满足下面所有的条件，则 `Net.Data` 使用这条消息语句中的信息来处理函数调用：

- 如果 `RETURN_CODE` 大于或小于零，但不为零
- 如果未指定返回码的精确匹配
- 如果 `RETURN_CODE` 大于或小于零时，并未指定 `+DEFAULT` 或 `-DEFAULT` 值

msg_code

用于指定在处理期间出现的错误和警告的消息代码。数字是 0 到 9 的数字字符串。

SQLSTATE

提供应用程序给针对普遍错误情况的公用代码的关键字。`SQLSTATE` 值基于 `SQL` 标准中所含的 `SQLSTATE` 说明，并且编码机制和 `SQL` 的所有 `IBM` 实现都相同。

限制：OS/400 平台不支持。

state_id

一个字母或数字字符串，以字母字符或下划线开头，可以包含字母、数字或下划线字符的任意组合。

字母数字字符串

一个字母或数字顺序的字符串，它包含字母或数字字符的任意组合。它不含标点符号。

消息文本说明

当 `RETURN_CODE` 与当前消息语句中的 `return_code` 值相匹配时，发送给 Web 浏览器的一个字符串。

字符串

字母、数字字符和标点符号的任意序列。如果此字符串出现在双引号内，则换行字符是不允许的。

变量引用

返回一个先前定义的变量的值，用 `$` 和 `()` 来指定。例如：如果 `VAR='abc'`，则 `$(VAR)` 返回值 `'abc'`。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 `FUNCTION` 或 `MACRO_FUNCTION` 块，或带指定参数调用 `Net.Data` 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

操作说明

当 `RETURN_CODE` 与当前消息语句中的 `return_code` 值相匹配时，`Net.Data` 所执行的操作。

EXIT

这个关键字指定了：当发生与指定的消息返回码相应的错误与警告时，立即退出这个宏。此值是缺省值。

CONTINUE

这个关键字指定了：当发生与指定的消息返回码相应的错误与警告时，继续进行处理。

包含语句

INCLUDE 语句。 读取并将文件包含到 Net.Data 宏中。 INCLUDE 语句可以出现在 MESSAGE 中的任何地方。请参阅第30页的『INCLUDE 语句』中的语法和例子。

上下文

可以在这些上下文中找到 MESSAGE 块:

- FUNCTION 块
- IF 块
- 位于 Net.Data 宏声明部分中的所有块或语句之外

限制

MESSAGE 块可以包含这些成份:

- 注解块
- 函数调用
- 变量引用
- HTML 语句
- 字符串
- INCLUDE 语句

OS/400 平台不支持 SQLSTATE。

例

例 1: 一个本地 MESSAGE 块

```
%{ local message block inside a FUNCTION block %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE{
-601: {<H3>The table has already been created, please go back and enter your name.</H3>
<P><a href="input">Return</a>
  %}
  default: "<H3>Can't continue because of error ${RETURN_CODE}</H3>"%}      : exit
  %}
```

例 2: 一个全局 MESSAGE 块

```
%{ 全局消息块 %}
%MESSAGE {
  -100      : "Return code -100 message"      : exit
  100      : "Return code 100 message"       : continue
  +default : {
This is a long message that spans more
than one line. You can use HTML tags, including
links and forms, in this message. %} : continue
  %}

%{ local message block inside a FUNCTION block %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE {
    -100      : "Return code -100 message"      : exit
    100      : "Return code 100 message"       : continue
    -default : {
```

```
This is a long message that spans more
than one line. You can use HTML tags, including
links and forms, in this message. %}    : exit
%}
```

例 3: 一个包含 INCLUDE 语句的 MESSAGE 块。

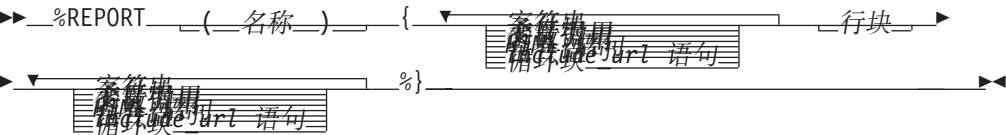
```
%message {
    %include "rc1000.msg"
    %include "rc2000.msg"
    %include "defaults.msg"
%}
```

REPORT 块

目的

格式化来自一个函数调用的输出。可以输入一个表格名称参数，指定此报表将使用已命名的表格中的数据。否则，此报表由函数参数列表中的找到的第一个输出表格来生成，或者如果在此列表中找到表格名称，则用缺省的表格数据来生成。

语法



值

%REPORT

这个关键字指定了用于格式化函数调用的输出结果的一系列命令。您还可以在报表中使用标题和脚注。

名称

这个值代表一个 Net.Data 表或结果集。请参阅 *Net.Data 管理与程序设计指南* 中的 Report 块一节，以获取更多信息。

字符串

字母、数字字符和标点符号的任意序列。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有一个前导加号(+)或者减号(-)。参阅第25页的『IF 块』中的语法和例子。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用 Net.Data 内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

include_url 语句

INCLUDE_URL 语句。读取并将另一个文件结合到其中指定这条语句的 Net.Data 宏中。指定的文件可以存在于本地或者远程服务器上。请参阅第32页的『INCLUDE_URL 语句』中的语法和例子。

行块

ROW 块。针对每一行返回自某个函数调用的数据，显示一次 HTML 格式化数据。请参阅第45页的『ROW 块』中的语法和例子。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

上下文

可以在这些上下文中找到 REPORT 块:

- FUNCTION 语句或块
- MACRO_FUNCTION 块

限制

REPORT 块可以包含这些成份:

- 注解块
- IF 块
- INCLUDE 语句
- INCLUDE_URL 语句
- ROW 块
- WHILE 块
- 函数调用

对于 OS/390 平台: 不能在 SQL 函数内部调用 SQL 函数。

- HTML 语句
- 字符串
- 变量引用

例

例 1: 一张显示名称和位置列表的双列 HTML 表格

```
%FUNCTION(DTW_SQL) mytable() {
%REPORT{
<H2>Query Results</H2>
<P>Select a name for details.
<TABLE BORDER=1>
<TR><TD>Name</TD><TD>Location</TD>
%ROW{
<TR>
<TD>
<a href="/cgi-bin/db2www/name.mac/details?name=$(V1)&location=$(V2)">$(V1)</a></TD>
<TD>$(V2)</TD>
%}
</TABLE>
%}
```

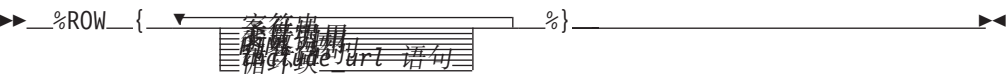
若在表格中选择一个名称, 则调用 Net.Data 宏 *name.mac* 中详细的 HTML 块, 并把这两个值作为 URL 的一部分发送给它。 对于此例子, 可以在 *name.mac* 中使用这些值来查找有关名称的其它细节。

ROW 块

目的

处理返回自一个函数调用的每个表格行。Net.Data 为表格中的每一行处理一次 ROW 块中的语句。

语法



值

%ROW

此关键字指定了要显示的、经过 HTML 格式化的数据，对从函数调用返回的每行数据，都要处理一遍。

字符串

字母、数字字符和标点符号的任意序列。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有一个单个的前导加号(+)或者减号(-)。参阅第25页的『IF 块』中的语法和例子。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

包含语句

INCLUDE 语句。读取并将文件包含到 Net.Data 宏中。请参阅第30页的『INCLUDE 语句』中的语法和例子。

include_url 语句

INCLUDE_URL 语句。读取并将另一个文件结合到其中指定这条语句的 Net.Data 宏中。指定的文件可以存在于本地或者远程服务器上。请参阅第32页的『INCLUDE_URL 语句』中的语法和例子。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

上下文

可以在这些上下文中找到 ROW 块：

- REPORT 块

限制

ROW 块可以包含这些成份:

- 注解块
- IF 块
- INCLUDE 语句
- INCLUDE_URL 语句
- WHILE 块
- 函数调用

对于 **OS/390** 平台: 不能在 SQL 函数内部调用 SQL 函数。

- 变量引用
- HTML 语句
- 字符串

例

例 1: 一张显示名称和位置列表的双列 HTML 表格

```
%REPORT{
<H2>Query Results</H2>
<P>Select a name for details.
<TABLE BORDER=1>
<TR><TD>Name</TD><TD>Location</TD>

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

</TABLE>
%}
```

若在表格中选择一个名称, 则调用 Net.Data 宏 *name.mac* 中详细的 HTML 块, 并把这两个值作为 URL 的一部分发送给它。 对于此例子, 可以在 *name.mac* 中使用这些值来查找有关名称的其它细节。

TABLE 语句

目的

定义一个变量，它是一个相关数据的集合。它包含一个相同记录或列的数组，和一个在每一行描述字段的列名的数组。表格语句只能在 `DEFINE` 语句或模块中。

语法

`▶▶ %TABLE 上限`
上限
`' (数值)`
值

%TABLE

此关键字定义了一组相关数据，其中包含一组相同的记录(或行)、一组用于描述每行中各字段的列名。

上限

表格中可以包含的行数。如果没有指定上限值，则该表中可以包含无限行。

数值

数值是 0 到 9 的数字字符串。0 值允许表格中行数不限。

ALL

此关键字允许在表格中包含任意多表格行。

上下文

可以在这些上下文中找到 `TABLE` 语句:

- `DEFINE` 语句

限制

`TABLE` 语句可以包含这些成份:

- 注解块
- 数值

例

例 1: 上限为 30 行的 `Net.Data` 表格

```
%DEFINE myTable1=%TABLE(30)
```

例 2: 使用缺省为所有行的 `Net.Data` 表格

```
%DEFINE myTable2=%TABLE
```

例 3: 指定所有行的 `Net.Data` 表格

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

WHILE 块

目的

提供一个基于条件字符串处理的循环结构。可以在 HTML 块、REPORT 块、ROW 块和 MACRO_FUNCTION 块中使用 WHILE 块。如果条件列表中的字符串值是代表整数的字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有一个单个的前导加号(+)或者减号(-)。

语法

```
►► %WHILE condition list { 语句 } ►►
```

condition list

```
( condition list )
```

条件

```
condition
```

条件项

```
variable name | string | function call
```

值

%WHILE

指定循环处理的关键字。

条件列表

比较条件和条件项的值。条件列表可以使用布尔运算符进行连接。一个条件列表可以嵌套在另一个条件列表中。

条件

两个条件项之间使用比较运算符的一个比较表达式。如果下面两个条件都为真，则 IF 条件看成是一个数值比较：

- 条件操作符是下列操作符之一：<,<=,>,>=,==,!=
- 两个条件项都是代表整数的字符串，其中有效整数是一个数字字符串，可选地，前面可以有前导加号(+)或减号(-)，并且没有空格。

如果两者有一个不为真，则执行一个一般的字符串比较。

条件项

变量名、字符串、变量引用或函数调用。

exec_sql 语句

DB2WWW 发行版 1 语言成份，它保证了兼容性。请参阅第235页的『附录A. DB2 WWW Connection』或 DB2 World Wide Web Release 1 文档。

函数调用

调用一个或多个先前已定义过的 FUNCTION 或 MACRO_FUNCTION 块，或带指定参数调用内部函数。请参阅第21页的『函数调用 (@)』中的语法和例子。

HTML 语句

包含任何要在客户浏览器中显示的字母或数字字符、以及 HTML 标记。

条件块

IF 块。可以执行有条件的字符串处理。如果条件列表中的字符串值是代表整数的字

字符串，并且没有前导或尾随空格，则作为数值型进行比较。它们可以有一个前导加号(+)或者减号(-)。参阅第25页的『IF 块』中的语法和例子。

循环块

WHILE 块。执行具有条件字符串处理的循序。参阅第48页的『WHILE 块』中的语法和例子。

变量引用

返回一个先前定义的变量的值，用 \$ 和 () 来指定。例如：如果 VAR='abc'，则 \$(VAR) 返回值 'abc'。请参阅第4页的『变量引用』中的语法信息。

字符串

字母、数字字符和标点符号的任意序列。用在条件列表中的此字符串可以包含任何字符，新建行字符除外。

变量名

一个或多个名称，每个附加的名称用一个句点(.)相连。请参阅第4页的『变量名』中的语法信息。

上下文

可以在这些上下文中找到 **WHILE 块**：

- HTML 块
- REPORT 块
- ROW 块
- MACRO_FUNCTION 块
- IF 块
- WHILE 块

限制

WHILE 块可以包含这些成份：

- 注解块
- EXEC_SQL 语句
- IF 块
- WHILE 块
- 字符串
- HTML 语句
- 函数调用
- 变量引用
- INCLUDE 语句

例

例 1：一个生成表格中若干行的 **WHILE 块**

```
%DEFINE loopCounter = "1"

%HTML(build_table) {
%WHILE (loopCounter <= "100") {
```

```

        %{ generate table tag and column headings %}
        %IF (loopCounter == "1")
        <TABLE BORDER>
        <TR>
            <TH>Item #
            <TH>Description
        </TR>
        %ENDIF

        %{ generate individual rows %}
        <TR>
        <TD>
            <TD>$(loopCounter)
            <TD>@getDescription(loopCounter)
        </TR>

        %{ generate end table tag %}
        %IF (loopCounter == "100")
        </TABLE>
        %ENDIF

        %{ increment loop counter %}
        @dtw_add(loopCounter, "1", loopCounter)
    %}
    %}

```

第2章 变量

Net.Data 提供两种类型的变量：用户定义的变量和 Net.Data 变量。

第52页的『用户定义的变量』

为您的应用程序定义的变量。可以定义执行以下任务的变量：

- 第52页的『条件变量』

基于另一个变量或字符串的值分配一个变量值。

- 第53页的『环境变量』

使用 ENVVAR 语言结构引用环境变量。

- 第53页的『可执行变量』

使用 EXEC 语言结构从变量引用中调用其它程序。

- 第54页的『隐藏变量』

隐藏来自 HTML 源的变量引用。

- 第55页的『列表变量』

使用 LIST 语言结构构建一个定界的值字符串。

- 第56页的『表格变量』

和一个函数来往传送值数组。可用于报表输出。

Net.Data 变量

供多种处理和文件操作、表格处理、报表格式化饱和语言环境使用的变量。

一些变量具有可以定义或者修改的值，另一些则由 Net.Data 来定义。变量的说明指定您是否定义了值。参阅变量的说明，确定如何定义其值。

Net.Data 提供下列变量类型：

- 第57页的『Net.Data 表格处理变量』

由 Net.Data 定义，以便让您处理 Net.Data 表。使用这些变量从 SQL 查询和函数调用访问数据。除非另加说明，否则只有在 REPORT 块中可识别它们。

- 第67页的『Net.Data 报表变量』

帮助您从函数定制报表。您必须在引用之前定义这些变量。可以在任何 Net.Data 宏块中定义或引用报表变量。

- 第74页的『Net.Data 语言环境变量』

帮助定制使用语言环境处理 FUNCTION 块的方式。

- 第91页的『Net.Data 杂项变量』

由 Net.Data 定义，影响 Net.Data 处理，查找函数调用的状态，并获取有关数据库查询结果集合的信息。一些杂项变量由 Net.Data 设置，且不能更改。

许多 Net.Data 变量的输出依赖于它所运行的操作系统而变化。

Net.Data 宏中，常量可以最长达 256KB。这样，在宏文件中不能把初始化一个变量或设置一个缺省值的长度大于 256 KB。

本章中指定了对支持每个变量的操作系统。下面的列表定义操作系统缩写:

HP-UX	Hewlett Packard UNIX 操作系统
SCO	Santa Cruz UNIX 操作系统
SUN	Sun Solaris UNIX 操作系统
Win NT	Microsoft 的 Windows NT 操作系统

用户定义的变量

本章节描述用户定义的变量。在宏文件内部定义这些变量。

条件变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

条件变量的值是根据另一个变量或字符串的值来有条件地设置的。这也称为三元操作。

条件变量的语法是:

`test ? trueValue : falseValue`

其中:

test 要测试的条件。

trueValue

当 `test` 为真时使用的值。

falseValue

当 `test` 为假时使用的值。

例 1: 用两个可能的值定义的条件变量

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

如果 `varB` 存在, `varA=value_1`, 否则 `varA=value_2` 。

例 2: 用一个变量引用定义的条件变量

`varname = ? "${value_1}"`

在此情况下, 如果 `value_1` 为空, 则 `varname` 为空, 否则 `varname` 设置成 `value_1` 。

例 3: 使用一个 `LIST` 语句和 `WHERE` 子句的条件变量

```
%DEFINE{
%list " AND " where_list
where_list    = ? "custid = $(cust_inp)"
where_list    = ? "product_name LIKE '$(prod_inp)%'"
where_clause  = ? "WHERE $(where_list)"
%}
```



```
%FUNCTION(DTW_SQL) mySelect() {  
    SELECT * FROM prodtbale $(where_clause)  
}%}
```

条件变量和列表变量一起使用时最有效。上面的例子显示了如何在 **DEFINE** 块中建立一个 **WHERE** 子句。变量 *cust_inp* 和 *prod_inp* 都是从 Web 浏览器(通常是 HTML 表)传递过来的 HTML 输入变量。变量 *where_list* 是一个 **LIST** 变量, 由两个条件语句组成, 每个语句包含一个来自 Web 浏览器的变量。

如果 Web 浏览器对变量 *cust_inp* 和 *prod_inp* 同时返回值, 例如 **IBM** 和 **755C**, 则 *where_clause* 是:

```
WHERE custid = IBM AND product_name LIKE '755C%'
```

如果变量 *ccust_inp* 或 *cprod_inp* 中有一个为空或未定义, 则更改 **WHERE** 语句, 省略其中的空值。 例如, 如果 *prod_inp* 为空, 则 **WHERE** 子句是:

```
WHERE custid = IBM
```

如果两个值都为空或未定义, 则变量 *cwhere_clause* 为空, 并且包含有 *\$(where_clause)* 的 **SQL** 查询中没有 **WHERE** 子句。

环境变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

环境变量允许使用 **Net.Data ENVVAR** 语言结构来引用存在于运行 **Net.Data** 的进程中的环境变量。

例 1: 给一个变量分配一个环境变量的值

```
%define SERVER_NAME=%ENVVAR  
  
...  
  
服务器是 $(SERVER_NAME)
```

环境变量 *SERVER_NAME* 是当前服务器名称的值, 在此例中, 是 **www.software.ibm.com**。

服务器是 **www.software.ibm.com**

参阅第12页的『**ENVVAR** 语句』获取更多有关 **ENFVAR** 语句的信息。

可执行变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

可执行变量允许您从一个使用可执行变量特征的变量引用调用其它的程序。可执行变量在 **Net.Data** 宏中用 **EXEC** 语言成份来定义。有关 **EXEC** 语言成份的更多信息, 请参阅 第13页的『**EXEC** 块或语句』。

当 Net.Data 在宏文件中遇到可执行变量时，它将使用下列方法寻找被引用的可执行程序：

1. 在 Net.Data 初始化文件中搜索 EXEC_PATH。参阅 *Net.Data 管理与程序设计指南* 的配置一章，进一步获取有关 EXEC_PATH 的信息。
2. 如果 Net.Data 没有找到该程序，它将搜索系统定义的目录。如果找到了此可执行程序，则 Net.Data 运行它。

例 1： 一个可执行变量定义

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

定义变量 *runit* 来执行可执行程序 *testProg*；*runit* 成为一个可执行变量。

Net.Data 在 Net.Data 宏中遇到一个可执行变量时运行可执行程序。例如，当 Net.Data 宏中有一个可执行变量引用建立成变量 *runit* 时，即执行 *testProg* 程序。

一种简单的方法是从另一个变量定义中引用一个可执行变量。例 2 演示了这个方法。变量 *date* 定义成一个可执行变量，*dateRpt* 定义成一个变量引用，它包含这个可执行变量。

例 2： 作为一个变量引用的可执行变量

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

当 Net.Data 分辨出变量引用 *\$(dateRpt)*，Net.Data 搜索此可执行日期，运行该程序，并返回：

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

可执行变量永远不设置成它调用的可执行程序的输出值。使用先前的例子，日期的值是空的。如果在 DTW_ASSIGN 函数调用中使用它来把它的值分配给另一个变量，则赋值后新变量的值也是空。可执行变量的唯一目的是去调用它定义的程序。

也可以给要执行的程序，通过在变量定义上指定此程序名，将参数传送给它。

例 3： 带参数的可执行变量

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

distance 和 *time* 的值传送给程序 *calcMPH*。

隐藏变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

使用隐藏变量，可以引用变量，并将实际的变量值隐藏在 HTML 源文件中。要使用隐藏变量：

1. 为每个需要隐藏的字符串定义一个变量。
2. 在引用此变量的 HTML 块中，使用两个美元符号代替一个美元符号来引用变量。例如，用 *\$(X)* 代替 *\$(X)*。

例 1： HTML 格式的隐藏变量

```
%HTML(INPUT) {
<FORM ...>
<P>Select fields to view:
<SELECT NAME="Field">
<OPTION VALUE="$$ (name)"> Name
<OPTION VALUE="$$ (addr)"> Address
.
.
.
</FORM>
%}

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

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

当此 HTML 显示在 Web 浏览器上时，`$(name)` 和 `$(addr)` 分别被 `$(name)` 和 `$(addr)` 替换，所以实际的表格和列名肯定不出现在 HTML 表上，也没有人可以知道实际的变量名被隐藏掉了。当客户提交这个表时，调用 `HTML(REPORT)` 块。当 `@mySelect()` 调用 `FUNCTION` 块时，`$(Field)` 在 SQL 语句中用 SQL 查询的 `customer.name` 或 `customer.addr` 替换。

列表变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

可以列出列表变量来构建一个定界的值字符串。当要构建一个具有多个项目的 SQL 查询时（象某些 `WHERE` 或 `HAVING` 语句一样），它们特别有用。

其中的空格是必须的。通常，值的两边都有一个空格。大部分查询使用布尔或数学运算符（例如，`AND`、`OR` 或 `>`）。请参阅第34页的『`LIST` 语句』中的语法和其它信息。

例 1：条件、隐藏和列表变量的使用

```
%HTML(INPUT){
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/example2.max/report">
Select one or more cities:<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond1)">Sao Paulo<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond2)">Seattle<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond3)">Shanghai<BR>
<INPUT TYPE="submit" VALUE="Submit Query">
</FORM>
%}

%DEFINE{
DATABASE="custcity"
%LIST " OR " conditions
cond1="cond1='Sao Paolo'"
cond2="cond2='Seattle'"
cond3="cond3='Shanghai'"
whereClause= ? "WHERE $(conditions)" : ""
%}
}
```

```
%FUNCTION(DTW_SQL) mySelect(){
SELECT name, city FROM citylist
$(whereClause)
%}

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

在 HTML 表中，如果没有选择任何框，则 *conditions* 为空，因此查询中的 *whereClause* 也为空。 否则，*whereClause* 具有选定值，值之间用布尔运算符 OR 分隔。例如，如果选择了所有这三个城市，则 SQL 查询为:

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

例 2: 值分隔符

```
%DEFINE %LIST " | " VLIST
%REPORT{
%ROW{
<EM>$(ROW_NUM):</EM> $(VLIST)
%}
%}
```

表处理变量 VLIST 使用两个引号和一个 OR 栏 (|) 作为本例中的值分隔符。值的字符串之间用引号中的值来分隔。

表格变量

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

表格变量中包含了一个值数组以及相关的列名。 数组中每个元素是一个行。可以使用表格变量将一组值传递给一个函数。 可以在一个函数的 REPORT 块中引用表格中的各个元素(行)。表格变量通常用于一个 SQL 函数的输出，或用于一个报表的输入，但也可以将它们作为 IN、OUT 或 INOUT 参数传递给任何非 SQL 函数。表格只能作为 OUT 参数传递给 SQL 函数。 请参阅第47页的『TABLE 语句』中的语法和其它信息。

例 1: 传送给一个 REXX 程序的 SQL 结果设置

```
%DEFINE{
DATABASE = "iddata"
MyTable = %TABLE(ALL)
DTW_DEFAULT_REPORT = "NO"
%}

%FUNCTION(DTW_SQL) Query(OUT table) {
select * from survey
%}

%FUNCTION(DTW_REXX) showTable(INOUT table) {
Say 'Number of Rows: 'table_ROWS
Say 'Number of Columns: 'table_COLS
do j=1 to table_COLS
Say "Here are all of the values for column " table_N.j ":"
do i = 1 to table_ROWS
Say "<B>"i"</B>: " table_V.i.j
end
end
%}
```

```
%HTML (report){  
<HTML>  
<PRE>  
@Query(MyTable)  
<p>  
@showTable(MyTable)  
</PRE>  
</HTML>  
%}
```

HTML REPORT 块调用一个 SQL 查询，将结果保存在一个表格变量中，然后将此变量传递给 REXX 函数。

Net.Data 表格处理变量

除非另有说明，否则 Net.Data 定义这些变量供 REPORT 和 ROW 块使用。使用这些变量引用您的查询所返回的值。

限制：不要在 DEFINE 段定义这些变量的值。

- 第58页的『*Nn*』
- 第59页的『*NLIST*』
- 第60页的『*NUM_COLUMNS*』
- 第61页的『*NUM_ROWS*』
- 第62页的『*ROW_NUM*』
- 第63页的『*TOTAL_ROWS*』
- 第64页的『*V_columnName*』
- 第66页的『*Vn*』
- 第65页的『*VLIST*』

Nn

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

由对列 n 的函数调用或查询返回的列名。

您可以在 REPORT 块和 ROW 块中引用 Nn。

这些变量是预定义的变量，其值不能修改。指定您想要引用的列号，就可以将这些变量用作变量引用。

例

例 1： 一个列名的变量引用

The name of column 2 is \$(N2)。

例 2： 用 DTW_ASSIGN， 保存一个列名的值， 供 REPORT 块外使用

```
%define coll=""
...
%function (DTW_SQL) myfunc() {
  select * from atable
  %report {
    @dtw_assign(coll, N1)
    %row{ %}
  %}

  %html(report) {
    @myfunc()
    The column name for the first column is $(coll)
  %}
```

这个例子显示了如何使用 DTW_ASSIGN 在 REPORT 块之外使用此变量。有关的更多信息， 请参阅第143页的『DTW_ASSIGN』。

例 3： Nn 在一个 HTML 表格内部来定义列名

```
%REPORT{
<H2>Product directory</H2>
<TABLE BORDER=1 CELLPADDING=3>
<TR><TD>$(N1)</TD><TD>$(N2)</TD><TD>$(N3)</TD>
%ROW{
<TR><TD>$(V1)</TD><TD>$(V2)</TD><TD>$(V3)</TD>
%}
</TABLE>
%}
```

NLIST

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

- | 包含一张来自函数调用或者查询的结果的所有列名的列表。缺省分隔符是一个空格。
- | 您可以在 REPORT 块和 ROW 块中引用 NLIST。
- | 此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1: 带 ALIGN 设置的列名列表

```
%DEFINE ALIGN="YES"
...
%FUNCTION (DTW_SQL) myfunc() {
select * from MyTable
  %report{
Your query was on these columns: $(NLIST).
%row {
...
%}
%}
%}
```

此列名列表在列名之间使用一个空格(ALIGN 设置成 YES)。

例 2: 把分隔符改成 " | " 的 %LIST 变量

```
%DEFINE %LIST " | " NLIST
...
%FUNCTION (DTW_SQL) myfunc() {
select * from MyTable
  %report{
Your query was on these columns: $(NLIST).
%row {
...
%}
%}
%}
```

NUM_COLUMNS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

Net.Data 在报表块中正在处理的表格列数；这些列由一个函数调用或查询来返回。

您可以在 REPORT 块和 ROW 块中引用 NUM_COLUMNS。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1: NUM_COLUMNS 用作一个变量引用，NLIST

```
%REPORT{
Your query result has $(NUM_COLUMNS) columns: $(NLIST).
...
%}
```


NUM_ROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

Net.Data 正在 REPORT 块中处理的表格中的行数。行的数目受上限参数值的影响，该参数给 Net.Data 表格定义来存放数据。例如，如果上限设置成 30，但是 SELECT 语句返回 1000 行，则 NUM_ROWS 的值是 30。另外，如果上限设置成 30，而 SELECT 语句返回 20 行，则 NUM_ROWS 等于 20。参阅 第47页的『TABLE 语句』获取 TABLE 语句和上限参数的更多信息。

只要 START_ROW_NUM 不传送给语言环境，则 NUM_ROWS 不受 START_ROW_NUM 的值的影响。例如，如果 START_ROW_NUM 设置成 5（指定显示在 Web 页面上的表格应当在第 5 行开始处理），而 SELECT 语句返回 25 行，则 NUM_ROWS 设置成 25，而不是 21。前面的四行从表格中被舍弃，但是仍包含在 NUM_ROWS 值中。但是，如果 START_ROW_NUM 传送给语言环境，那么 NUM_ROWS 将只包含从 START_ROW_NUM 指定的行开始算的行数。在上面的例子中，NUM_ROWS 将设置成 21。

您可以在 REPORT 块和 ROW 块中引用 NUM_ROWS。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1：显示 REPORT 块中正在处理的名字个数

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"

%REPORT{
<H2>E-mail directory</H2>
<UL>
%ROW{
<LI>Name: <a href="mailto:$(V1)">$(V2)</a><BR>
Location: $(V3)
%}
</UL>
Names displayed: $(NUM_ROWS)<BR>
Names found: $(TOTAL_ROWS)
%}
```

ROW_NUM

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

| 一个表格变量，其值在 Net.Data 表格中接受处理，Net.Data 每次增加一行。此变量作
| 为一个计数器，它的值是当前正在处理的行号。

RPT_MAX_ROWS 可以影响 ROW_NUM 的值。例如，如果表格中有 100 行，并且您
已把 RPT_MAX_ROWS 设置为 20，则 ROW_NUM 的最终值是 20，因为第 20 行是
要处理的最后一行。

您只能从 ROW 块中引用 ROW_NUM。

| 此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1: 通过使用 ROW_NUM 为表格中每一行标号，处理 HTML 输出中的列

```
%REPORT{
<TABLE BORDER=1>
<TR><TD> Row Number </TD> <TD> Customer </TD>
%ROW{
<TR><TD> $(ROW_NUM) </TD> <TD> $(V_custname) </TD>
%}
</TABLE>
%}
```

REPORT 块产生一张象下面显示的表格。

Row Number	Customer
1	Jane Smith
2	Jon Chiu
3	Frank Nguyen
4	Mary Nichols

TOTAL_ROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

一次查询返回的总行数，它与 TABLE 语言结构的upper_limit 值无关。例如，如果 RPT_MAX_ROWS 设置为最多显示 20 行，但查询返回 100 行，则在完成 ROW 处理后，此变量设置为 100。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

操作系统区别:

- 在 OS/400 操作系统上，此变量可以在 REPORT 或 ROW 块中的任何地方引用。
- 在 OS/2、Windows NT 和 UNIX 操作系统上，此变量只可以在 REPORT 脚注上引用。

必需: 要使用此变量，必须把 DTW_SET_TOTAL_ROWS 设置成 YES。请参阅第83页的『DTW_SET_TOTAL_ROWS』以获取更多的信息。

例

例 1: 显示找到的总的名字个数

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"

%REPORT{
<H2>E-mail directory</H2>
<UL>
%ROW{
<LI>Name: <a href="mailto:$(V1)">$(V2)</a><BR>
Location: $(V3)
%}
</UL>
Names displayed: $(NUM_ROWS)<BR>
Names found: $(TOTAL_ROWS)
%}
```

V_columnName

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

当前行的指定列名的值。对于未定义的列名，不设置此变量。如果一个查询中包含两个列名，而它们的名称相同，则可能产生不可预测的结果。考虑在您的SQL中使用一个AS子句来重新命名重复的列名。

您只能在ROW块中引用V_columnName。

这些变量是预定义的变量，其值不能修改。指定您想要引用的列名，就可以将这些变量用作变量引用。

值

V_columnName

表 1. V_columnName 值

值	说明
columnName	数据库表格的当前行中的列名。

例

例 1: V_columnName 用作一个变量引用

```
%FUNCTION(DTW_SQL) myQuery() {
  SELECT NAME, ADDRESS from $(qtable)
%REPORT{

%ROW{

  Value of NAME column in row $(ROW_NUM) is $(V_NAME).<BR>
%}
%}
%}
```

VLIST

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

一张针对一个 ROW 块中正在处理的当前行的字段值的列表。

您只能从 ROW 块中引用 VLIST。缺省分隔符是一个空格。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1: 使用列表标记显示查询结果

```
%DEFINE ALIGN="YES"

%REPORT{
Here are the results of your query:
<OL>
%ROW{
<LI>$(VLIST)
%}
</OL>
%}
```

例 2: 使用列表变量把分隔符更改至 <P>

```
%DEFINE %LIST "<P>" VLIST

%REPORT{
Here are the results of your query:
%ROW{
<HR>$(VLIST)
%}
%}
```

V_n

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

当前行中为列号 *n* 指定的值。

您只能从 ROW 块中引用 V_n。

Net.Data 为表格中的每一字段都分配变量；在变量引用中使用此变量，来指定您想要引用的字段号。要在块外使用此变量，则把 V_n 的值分配给一个先前定义的全局变量或一个 OUT 或 INOUT 函数参数变量。

例

例 1: 显示一个 HTML 表格的报表

```
%REPORT{
<H2>E-mail directory</H2>
<TABLE BORDER=1 CELLPADDING=3>
<TR><TD>Name</TD><TD>E-mail address</TD><TD>Location</TD>
%ROW{
<TR><TD>$(V1)</TD>
<TD><a href="mailto:$(V2)">$(V2)</a></TD>
<TD>$(V3)</TD>
%}
</TABLE>
Found $(NUM_ROWS) models matching your description.
%}
```

第二列显示了 e-mail 地址。通过单击一个人的链接，可以给这个人发送一条消息。

Net.Data 报表变量

这些变量可以帮助您定制报表。您必须在引用之前定义这些变量:

- 第68页的『ALIGN』
- 第69页的『DTW_DEFAULT_REPORT』
- 第70页的『DTW_HTML_TABLE』
- 第71页的『RPT_MAX_ROWS』
- 第72页的『START_ROW_NUM』

ALIGN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

控制和表格处理变量 `NLIST` 和 `VLIST` 一起使用的前导和尾随空格。

性能技巧: 仅在必要是时候才使用 `ALIGN`，因为它需要 `Net.Data` 确定表中所有列的最大列长，从而计算出填充时所需的大小。此处理可能影响性能。

如果设置成 `YES`，`ALIGN` 填补空白，以对齐要显示的表格处理变量。 如果想要在 `HTML` 链路或表格操作中嵌入查询结果，则使用缺省值 `NO`，阻止 `Net.Data` 在报表变量周围加前导或尾随空格。

使用 `DEFINE` 语句或 `@DTW_ASSIGN()` 函数指定这个变量的值。

值

`ALIGN="YES"|"NO"`

表 2. `ALIGN` 值

值	说明
<code>YES</code>	<code>Net.Data</code> 给报表变量添加前导和尾随空格，以对齐显示。
<code>NO</code>	<code>Net.Data</code> 不添加前导或者尾随空格。 <code>NO</code> 是缺省。

例

例 1: 使用 `ALIGN` 变量用空格分隔每一列

```
%DEFINE ALIGN="YES"
<P>Your query was on these columns: $(NLIST)
```


DTW_DEFAULT_REPORT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

| 确定 Net.Data 是否为没有 REPORT 块的函数生成缺省报表。如果此变量设置为 YES，
| 则 Net.Data 生成缺省的报表。如果设置为 NO，则 Net.Data 关闭缺省的报表生成操作。
| 例如，如果把函数调用的结果接收到一个表格变量中，并且想把结果传送给另一个函
| 数来处理，则关闭缺省报表是有用的。

| 使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DTW_DEFAULT_REPORT="YES"|"NO"

表 3. DTW_DEFAULT_REPORT 值

值	说明
YES	Net.Data 为没有 REPORT 块的函数生成缺省报表，并把结果显示在浏览器中。YES 是缺省。
NO	Net.Data 对没有 REPORT 块的函数舍弃生成缺省报表。

例

例 1：覆盖由 Net.Data 生成的缺省报表

%DEFINE DTW_DEFAULT_REPORT="NO"

DTW_HTML_TABLE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

把结果以 HTML 表格形式显示，而不把表格作为文本类型来显示(即，使用 TABLE 标签，而不使用 PRE 标签)。

生成的 TABLE 标记中包括了边界和单元填充说明：

```
<TABLE BORDER CELLPADDING=2>
```

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

```
DTW_HTML_TABLE="YES"|"NO"
```

表 4. DTW_HTML_TABLE 值

值	说明
YES	使用 HTML 表格标记显示表格数据。
NO	使用 PRE 标记，以文本格式显示表格数据。NO 是缺省。

例

例 1：用 HTML 标记显示来自一个 SQL 函数的结果

```
%DEFINE DTW_HTML_TABLE="YES"

%FUNCTION(DTW_SQL){
SELECT NAME, ADDRESS FROM $(qTable)
%}
```

RPT_MAX_ROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

指定行数，它们显示在由函数 `REPORT` 块生成的表格中。

使用 `DEFINE` 语句或 `@DTW_ASSIGN()` 函数指定此变量的值。

OS/400、Windows NT、OS/2 和 UNIX 的性能技巧：将 `RPT_MAX_ROWS` 传递给数据库语言环境有助于限制从语言环境返回的行数，从而提高性能。要想将 `RPT_MAX_ROWS` 传递给语言环境(例如，`DTW_SQL`)，可以在初始化文件中将这个变量作为 `ENVIRONMENT` 语句中的 `IN` 变量(对于您正在使用的数据库语言环境)。参阅 *Net.Data 管理与程序设计指南* 的配置一章，进一步学习数据库语言环境语句。

值

`RPT_MAX_ROWS="ALL"|"0"|"number"`

表 5. `RPT_MAX_ROWS` 值

值	说明
ALL	表示对由函数调用生成的表格中显示的行数没有限制。将显示所有行。
0	指定表格中的所有行都将显示出来。这个值和指定 <code>ALL</code> 是相同的。
number	一个正整数，表示在由函数调用生成的表格中显示的最大行数。 如果 <code>FUNCTION</code> 块包含一个 <code>REPORT</code> 和 <code>ROW</code> 块，则这个数字指定了执行 <code>ROW</code> 块的次数。

例

例 1：在 `DEFINE` 语句中定义 `RPT_MAX_ROWS`

```
%DEFINE RPT_MAX_ROWS="20"
```

上述方法显示任何函数返回的行数为 20 行。

例 2：使用 `HTML` 输入定义 `HTML` 格式的变量

```
Maximum rows to return (0 for no limit):  
<INPUT TYPE="text" NAME="RPT_MAX_ROWS" SIZE=3>
```

上例中各行可以放在一个 `FORM` 标记中，以使应用程序用户设置他们希望查询返回的行数。

START_ROW_NUM

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

指定在报表中开始显示一张 Net.Data 表格的结果的行号。将这个变量与 RPT_MAX_ROWS 一起使用，可以将具有庞大结果集合的查询分成几个小表格，然后使用“下一页”按钮来游历生成的表格。

OS/400、OS/2、Windows NT 和 UNIX 的性能技巧：请确保 START_ROW_NUM 对于数据库 IE 是作为 ENVIRONMENT 语句上的 IN 变量指定的。否则，数据库 LE 将从第一行开始返回结果集。如果没有在 ENVIRONMENT 语句上指定，则 Net.Data 必须舍弃指定行前的所有行，而不是 LE 没有在前面的位置返回行。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

START_ROW_NUM="number"

表 6. START_ROW_NUM 值

值	说明
number	一个正整数，表示开始显示一张报表的行号。 如果初始化文件的数据库语言环境的环境语句中指定 START_ROW_NUM，则此数字指定了数据库语言环境所处理的结果集合的行号。 如果 START_ROW_NUM 没有传送给语言环境，则这个数字指定用于显示一张报表的 Net.Data 表格的行号。

例

例 1: 用 HTML 格式“下一页”和“前一页”按钮滚动

```
%define {
    DTW_HTML_TABLE      = "YES"
    START_ROW_NUM       = "1"
    RPT_MAX_ROWS        = "10"
    totalSize           = ""
    includeNext         = "YES"
    includePrev         = "YES"
    includeLast         = "YES"
    includeFirst        = "YES"
}%

%function(DTW_SQL) myQuery(){
    select * from NETDATADEV.CUSTOMER
}%

%function(DTW_SQL) count(OUT size){
    select count(*) from NETDATADEV.CUSTOMER
    %report{
        %row{
            @DTW_ASSIGN(size,V1)
        }
    }
}%
%}
```

```

%html(report) {
    %{ get the total number of records if we haven't already %}
    %{ if (totalSize == "")
        @count(totalSize)
    %endif

    %{ set START_ROW_NUM based on the button user clicked %}
    %{ if (totalSize <= RPT_MAX_ROWS)
        %{ there's only one page of data %}
        @DTW_ASSIGN(START_ROW_NUM, "1")
        @DTW_ASSIGN(includeFirst, "NO")
        @DTW_ASSIGN(includeLast, "NO")
        @DTW_ASSIGN(includeNext, "NO")
        @DTW_ASSIGN(includePrev, "NO")
    %elif (submit == "First Page" || submit == "")
        %{ first time through or user selected "First Page" button %}
        @DTW_ASSIGN(START_ROW_NUM, "1")
        @DTW_ASSIGN(includePrev, "NO")
        @DTW_ASSIGN(includeFirst, "NO")
    %elif (submit == "Last Page")
        %{ user selected "Last Page" button %}
        @DTW_SUBTRACT(totalSize, RPT_MAX_ROWS, START_ROW_NUM)
        @DTW_ADD(START_ROW_NUM, "1", START_ROW_NUM)
        @DTW_ASSIGN(includeLast, "NO")
        @DTW_ASSIGN(includeNext, "NO")
    %elif (submit == "Next")
        %{ user selected "Next" button %}
        @DTW_ADD(START_ROW_NUM, RPT_MAX_ROWS, START_ROW_NUM)
        %{ if (@DTW_rADD(START_ROW_NUM, RPT_MAX_ROWS) > totalSize)
            @DTW_ASSIGN(includeNext, "NO")
            @DTW_ASSIGN(includeLast, "NO")
        %endif
    %endif
    %elif (submit == "Previous")
        %{ user selected "Previous" button %}
        @DTW_SUBTRACT(START_ROW_NUM, RPT_MAX_ROWS, START_ROW_NUM)
        %{ if (START_ROW_NUM <= "1" )
            @DTW_ASSIGN(START_ROW_NUM, "1")
            @DTW_ASSIGN(includePrev, "NO")
            @DTW_ASSIGN(includeFirst, "NO")
        %endif
    %endif
    %endif

    %{ run the query to get the data %}
    @myQuery()

    %{ output the correct buttons at the bottom of the report %}
    <center>
    <form method="POST" action="report">
    <input name="START_ROW_NUM" type="hidden" value="$(START_ROW_NUM)">
    <input name="totalSize" type="hidden" value="$(totalSize)">
    %{ if (includeFirst == "YES" )
        <input name="submit" type="submit" value="First Page">
    %endif
    %{ if (includePrev == "YES" )
        <input name="submit" type="submit" value="Previous">
    %endif
    %{ if (includeNext == "YES" )
        <input name="submit" type="submit" value="Next">
    %endif
    %{ if (includeLast == "YES" )
        <input name="submit" type="submit" value="Last Page">
    %endif
    %endif
    </form>
    </center>
    %}

```

Net.Data 语言环境变量

和函数一起使用这些变量，帮助定制使用语言环境处理 FUNCTION 块的方式。在引用这些变量之前先要定义之。

- 第75页的『 DATABASE 』
- 第76页的『 DB_CASE 』
- 第77页的『 DB2PLAN 』
- 第78页的『 DB2SSID 』
- 第79页的『 DTW_APPLET_ALTTEXT 』
- 第80页的『 DTW_EDIT_CODES 』
- 第81页的『 DTW_MBMODE 』
- 第82页的『 DTW_SAVE_TABLE_IN 』
- 第83页的『 DTW_SET_TOTAL_ROWS 』
- 第84页的『 LOCATION 』
- 第85页的『 LOGIN 』
- 第86页的『 NULL_RPT_FIELD 』
- 第87页的『 PASSWORD 』
- 第88页的『 SHOWSQL 』
- 第89页的『 SQL_STATE 』
- 第90页的『 TRANSACTION_SCOPE 』

DATABASE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

指定在调用一个数据库函数时要访问的数据库或 ODBC 数据源。此变量可以在一个宏内更改多次，以访问多数据库或多 ODBC 数据源。

OS/400 操作系统：此变量是可选的。缺省地，Net.Data 指定 DATABASE=『*LOCAL』；而 DTW_SQL 语言环境使用本地关系数据库目录项。

Windows NT、OS/2 和 UNIX 操作系统：在调用任何数据库函数之前先定义此变量，使用 DTW_ORA (Oracle) 语言环境除外。另外，在同一个 HTML 块中，通过同一语言环境访问多个数据库时，必须使用“现场连接”。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DATABASE="dbname"

表 7. DATABASE 值

值	说明
dbname	Net.Data 要连接至的数据库的名称。

例

例 1：指定连接到 CDLDIAL 数据库进行 SQL 操作

```
%DEFINE DATABASE="CEDDIAL"

%FUNCTION (DTW_SQL) getRpt() {
SELECT * FROM customer
%}

%HTML (report){
%INCLUDE "rpthead.htm"
@getRpt()
%INCLUDE "rptfoot.htm"
%}
```

调用函数 getRpt 时访问数据库 CEDDIAL 。

例 2：以 DTW_ASSIGN 覆盖前一 DATABASE 定义

```
%DEFINE DATABASE="DB2C1"
...
%HTML(monthRpt){
@DTW_ASSIGN(DATABASE,"DB2D1")
%INCLUDE "rpthead.htm"
@getRpt()
%INCLUDE "rptfoot.htm"
%}
```

不管 DATABASE 先前的值是什么，此 HTML 块将查询数据库 DB2D1。

DB_CASE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

指定将哪种情况用于 SQL 命令，并把所有字符转换成大写或小写的。如果此变量未定义，则缺省动作是不转换 SQL 命令字符。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DB_CASE="UPPER"|"LOWER"

表 8. DB_CASE Values

值	说明
UPPER	把所有 SQL 命令字符转换为大写。
LOWER	把所有 SQL 命令字符转换为小写。

例

例 1: 指定将所有 SQL 命令用大写

%DEFINE DB_CASE="UPPER"

DB2PLAN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
			X				

目的

为和一个本地 DB2 子系统的连接分配一个计划。此变量指定在 Net.Data 将访问的本地 DB2 子系统上针对 Net.Data SQL 语言环境的计划名。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

要求：此变量必须在 Net.Data 初始化文件中指定在 DTW_SQL ENVIRONMENT 语句上，宏文件中可选指定。如果在 Net.Data for OS/390 的初始化文件中没有指定这个变量，或者在一个宏内部而不是在初始化文件中未定义，则当这个宏企图执行一个 SQL 函数时将发生错误。

值

DB2PLAN=*plan_name*

表 9. DB2PLAN 值

值	说明
<i>plan_name</i>	DB2 计划的名称。此名称可以是 8 个字符或更少些。

例

例 1：在 DEFINE 语句中指定计划

%DEFINE DB2PLAN="DTWGAV21"

DB2SSID

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
			X				

目的

建立一个和本地 DB2 子系统的连接。这些变量指定了 Net.Data 将要访问的本地 DB2 子系统的子系统标识符。 每个宏只允许有一个本地数据库连接。

要求：此变量必须指定在 Net.Data 初始化文件中，宏文件中可选指定。 如果没有在 Net.Data for OS/390 初始化文件中指定这个变量，也没有在宏当中定义，则当这个宏企图执行一个 SQL 函数时将发生一个错误。

值

DB2PLAN="*subsytem_id*"

表 10. DB2SSID 值

值	说明
<i>subsystem_id</i>	DB2 子系统的名称。此名称可以是 8 个字符或更少些。

例

例 1：在 DEFINE 语句中指定一个子系统 ID

%DEFINE DB2SSID="DBNC"

DTW_APPLET_ALTTEXT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X		X	X	X

目的

| 将 HTML 标记和文本在不能识别 APPLET 标记的浏览器中显示出来，并和 Applet 语
| 言环境一起使用。

| 使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DTW_APPLET_ALTTEXT="HTML_text_and_tags"

表 11. DTW_APPLET_ALTTEXT 值

值	说明
HTML_text_and_tags	针对不能识别 APPLET 标记的浏览器的 HTML 标记和文本。

例

例 1: 指示一个 Web 浏览器限制的替换文本

%DEFINE DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not java-enabled."

DTW_EDIT_CODES

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

转换 NUMERIC、DECIMAL、INTEGER 和 SMALLINT 数据类型，它们作为针对 DTW_SQL 语言环境的一个 SQL 操作的结果而返回。变量 DTW_EDIT_CODES 是一个字符型字符串，对应于 DTW_SQL LE 将构建的表格的结果列；例如，DTW_EDIT_CODES 中的第五个字符将应用于结果集合的第五列(如果该列是支持的类型之一)。这个单一的字符可以是支持系统提供的任何编辑代码，这在数据描述说明参考中有定义。

例如，一个 DECIMAL(6,0) 字段通常将显示为字符型字符串 ‘112698’。通过在变量 DTW_EDIT_CODES 中为该列指定一个编辑码 ‘Y’，结果表中相应的列将作为一个字符型字符串显示，代表日期 ‘11/26/98’。

提示： 把一个用户提供的编辑码应用到某一列，而该列会导致一个具有非数字字符(例如逗号或货币符号)的字符型字符串，则当此字符串送还给服务器以继续处理 Net.Data 宏中的后继操作时，将会引起语法错误。例如，非数值列值可能用于后继的 DTW_SQL 函数调用中的数值型比较，则引起语法错误。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DTW_EDIT_CODES="edit_code"

表 12. DTW_EDIT_CODES 值

值	说明
edit_code	指定一个字符型字符串，它相应于 SQL 语言环境构建的表格的结果列。

例

例 1:

@DTW_ASSIGN(DTW_EDIT_CODES "JJLJJ*****Y")

DTW_MBMODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X		X	X	X

目的

提供多字节字符集 (MBCS)，支持由“缺省”语言环境使用的字符串和字处理函数。可以在 Net.Data 初始化文件中设置这个变量，但是您可以在宏文件中使用它来设置或覆盖当前设置。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

OS/400 用户： Net.Data for OS/400 自动允许 MBCS 支持功能，它不需要这个变量。Net.Data for OS/400 忽略被移植到 OS/400 操作系统上的宏文件中的这个变量。

值

DTW_MBMODE="YES"|"NO"

表 13. DTW_MBMODE 值

值	说明
YES	指定 MBCS 支持字符串和字处理函数。
NO	指定字符串和字处理函数没有 MBCS 支持。NO 是缺省。

例

例 1: 覆盖 INI 文件中的值

INI 文件:

```
DTW_MBMODE "NO"
...
ENVIRONMENT (DTW_DEFAULT) [d11] (IN DTW_MBMODE, OUT
RETURN_CODE )
```

宏文件:

```
%DEFINE DTW_MBMODE = "YES"
```

DTW_SAVE_TABLE_IN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

标识一个表格变量，SQL 语言环境用它来存储来自一个查询的表格数据。然后可以使用此表格，例如，用在分析表格数据的 REXX 程序中。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DTW_SAVE_TABLE_IN="table_name_var"

表 14. DTW_SAVE_TABLE_IN 值

值	说明
table_name_var	针对 SQL 语言环境的一张表格的名称，用来存储来自查询的表格数据。

例

例 1: 在一个 REXX 调用中使用的先前定义的表格变量

```
%DEFINE theTable = %TABLE(2)
%DEFINE DTW_SAVE_TABLE_IN = "theTable"

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

%FUNCTION(DTW_REXX) analyze_table(myTable) {
  %EXEC{ anzTbl.cmd %}
}%

%HTML(doTable) {
  @doQuery()
  @analyze_table(theTable)
}%
```

REXX FUNCTION 块调用 REXX 程序 anzTbl.cmd，这使用表格变量 theTable 来分析表格中的数据。变量 theTable 是从前一 SQL 函数调用中返回的。

DTW_SET_TOTAL_ROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

指定一个数据库语言环境，针对一个查询的结果集合中的总行数应当指定为 TOTAL_ROWS。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

要自动传送此变量，则把它作为一个 IN 变量包含在初始化文件的数据库语言环境 ENVIRONMENT 语句中。参阅*Net.Data 管理与程序设计指南*的配置一章，进一步学习数据库语言环境语句。

值

DTW_SET_TOTAL_ROWS="YES"|"NO"

表 15. DTW_SET_TOTAL_ROWS 值

值	说明
YES	把总行数的值指定为 TOTAL_ROWS 变量。 重要： 如果您想要引用变量 TOTAL_ROWS 来确定从查询返回的行数，则必须设置这个值。
NO	Net.Data 不设置 TOTAL_ROWS 变量，TOTAL_ROWS 在宏文件中不能引用。NO 是缺省。

性能提示： 如果将 DTW_SET_TOTAL_ROWS 设置为 YES，这将影响其性能，因为为了确定总行数，数据库语言环境需要检索所有行。

例

例 1： 定义 DTW_SET_TOTAL_ROWS 使用 TOTAL_ROWS

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"

...

%FUNCTION (DTW_SQL) myfunc() {
select * from MyTable
%report {
...
%row
...
%}
<P>$(NUM_ROWS) returned. Your query is limited to $(TOTAL_ROWS) rows.
%}
%}
```

LOCATION

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
			X				

目的

建立一个和远程数据库服务器的连接。此变量指定了本地 DB2 子系统中用于标识远程服务器的名称。LOCATION 的值必须在 Communications Database (CDB) 中的 SYSIBM.SYSLOCATIONS 表格中定义。如果未在宏中定义过这个变量，那么由宏发出的任何 SQL 请求都将在本地 DB2 子系统中执行。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

LOCATION="remote_dbase_name"

表 16. LOCATION 值

值	说明
remote_dbase_name	一个有效远程数据库服务器的名称，它在 CDB 的 SYSIBM.SYSLOCATIONS 表格中定义。此名称可以是 8 个字符或更少些。

例

例 1: 在 DEFINE 语句中定义远程数据库的位置

```
%DEFINE LOCATION="QMFDJ00"
```


LOGIN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

通过向数据库语言环境传送一个用户标识符，提供对保护数据的访问。这个变量与 PASSWORD 一起使用 DB2 的安全性算法。

OS/400 用户: 如果 DATABASE 变量没有定义或者该变量设置为 `"*LOCAL"`，则 OS/400 将忽略 LOGIN 和 PASSWORD。数据库的访问是通过 Net.Data 在其下运行的用户概要来获得了。

安全性提示: 在可以将这个值编码在 Net.Data 宏中的同时，也建议让应用程序用户在 HTML 表中输入用户标识符。另外，使用 Web 服务器 ID 的缺省值提供的访问级别可能不满足您的安全性需要。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

LOGIN=`database_user_id`

表 17. LOGIN 值

值	说明
<code>database_user_id</code>	一个有效的数据库用户标识符。缺省时，使用用于启动 Web 服务器的用户 ID。

例

例 1: 限制访问用户标识符，DB2USER

```
%DEFINE LOGIN="DB2USER"
```

例 2: 使用 HTML 格式输入行

```
USERID&#58; <INPUT TYPE="text" NAME="LOGIN" SIZE=6>
```

这个例子中显示可以作为 HTML 一部分的一行，应用程序用户可以用它来输入其用户标识符。

NULL_RPT_FIELD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

指定一个字符串，用户可以把它提供给 DTW_SQL 语言环境，代表返回在一个 SQL 结果集中的空(NULL)值。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

NULL_RPT_FIELD="null_char"

表 18. NULL_RPT_FIELD 值

值	说明
<i>null_char</i>	指定一个字符串来代表 SQL 结果集中返回的空值。缺省是一个空字符串。

例

例 1: 指定一个代表 SQL 语言环境中空(NULL)值的字符串

```
%DEFINE NULL_RPT_FIELD = "++++"
```

PASSWORD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

通过向数据库语言环境传送一个口令，提供对保护数据的访问。这个变量与 LOGIN 一起使用 DB2 的安全性算法。

OS/400 用户：如果 DATABASE 变量没有定义或者该变量设置为 “*LOCAL”，则 OS/400 将忽略 LOGIN 和 PASSWORD。数据库的访问是通过 Net.Data 在其下运行的用户概要来获得了。

安全性提示：在可以将这个值编码在 Net.Data 宏中的同时，也建议让应用程序用户在 HTML 格式下输入口令。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

PASSWORD=*password*

表 19. PASSWORD 值

值	说明
<i>password</i>	指定一个有效的口令，提供自动访问数据库语言环境。

例

例 1：用口令 NETDATA 限制应用程序用户访问

```
%DEFINE PASSWORD="NETDATA"
```

例 2：HTML 格式输入行

```
PASSWORD&#58; <INPUT TYPE="password" NAME="PASSWORD" SIZE=8>
```

这个例子中显示可以作为 HTML 一部分的一行，应用程序用户可以用它来输入自己的口令。

SHOWSQL

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

隐藏或显示用在 Web 浏览器上的查询 SQL。在测试期间显示 SQL，在调试 Net.Data 宏时特别有帮助。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

SHOWSQL="YES"|"NO"

表 20. SHOW_SQL 值

值	说明
YES	显示发送给数据库的查询的 SQL。
NO	隐藏发送给数据库的查询的 SQL。 NO 是缺省。

例

例 1: 显示所有 SQL 查询

%DEFINE SHOWSQL="YES"

例 2: 使用 HTML 格式输入指定是否显示 SQL

SHOWSQL: <INPUT TYPE="radio" NAME="SHOWSQL" VALUE="YES"> Yes
 <INPUT TYPE="radio" NAME="SHOWSQL" VALUE="" CHECKED> No

SQL_STATE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

访问或者显示从数据库返回的 SQL 状态值。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

例 1: 在 REPORT 块中显示 SQL 状态

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

TRANSACTION_SCOPE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

指定 SQL 命令的事务处理范围，确定 Net.Data 是否在每条 SQL 命令或在所有 SQL 命令在 HTML 块中成功完成后发出 COMMIT。如果指定在提交之前必须先成功完成所有 SQL 命令，则不成功的 SQL 命令会使模块中先前对同一个数据库执行的所有 SQL 命令都被撤消。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

一致性考虑：在不是 OS/400 和 OS/390 的操作系统中，如果所有以下条件都成立，则当提交对由同一 HTML 模块访问的其它数据库的更新时，对这个数据库进行的更新（接收到不成功的响应）将被撤消：

- 指定 TRANSACTION_SCOPE = "MULTIPLE"
- 在同一个 HTML 块中访问多个数据库(当使用“现场连接”时这是可能的)
- 从 SQL 请求中返回不成功的应答

如果您在 OS/400 上从 Net.Data 访问多个数据库，或使用 IBM 的 DataJoiner 访问多个数据库，那么在从 Net.Data 更新时，这多个数据库将可以协调一致地得以更新。

在 OS/400 和 OS/390 中，TRANSACTION_SCOPE = "MULTIPLE" 将使得从一个 HTML 模块发出的所有 IBM 数据库更新操作被一起提交或回滚。

在不是 OS/400 的操作系统中，REXX、Perl 和 Java 语言环境是在它们自己的操作系统进程中运行的。因此，您从这些语言环境发出的数据库更新操作，与从 Net.Data 宏文件中发出的数据库更新操作是相互独立的，可以分别被提交或回滚，而与 Net.Data TRANSACTION_SCOPE 值的设置无关。

值

TRANSACTION_SCOPE="SINGLE"|"MULTIPLE"

表 21. TRANSACTION_SCOPE 值

值	说明
SINGLE	在 HTML 块中的每条 SQL 命令成功完成后，Net.Data 发出一个 COMMIT。
MULTIPLE	指定只有在 HTML 块中的所有 SQL 命令都成功完成后，Net.Data 才发出 COMMIT。MULTIPLE 是缺省。

例

例 1：指定在每个事务处理后发出一条 COMMIT

```
%DEFINE TRANSACTION_SCOPE="SINGLE"
```

Net.Data 杂项变量

这些是 Net.Data 定义的变量，可用于影响 Net.Data 处理，查找函数调用的状态，并获取有关数据库查询结果集合的信息，并确定文件位置和日期的信息。在您编写的函数中这些变量非常有用，您还可以用这些变量测试 Net.Data 宏。

- 第92页的『DTW_CURRENT_FILENAME』
- 第93页的『DTW_CURRENT_LAST_MODIFIED』
- 第94页的『DTW_DEFAULT_MESSAGE』
- 第95页的『DTW_LOG_LEVEL』
- 第96页的『DTW_MACRO_FILENAME』
- 第97页的『DTW_MACRO_LAST_MODIFIED』
- 第98页的『DTW_MP_PATH』
- 第99页的『DTW_MP_VERSION』
- 第100页的『DTW_PRINT_HEADER』
- 第101页的『DTW_REMOVE_WS』
- 第102页的『RETURN_CODE』

DTW_CURRENT_FILENAME

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

当前输入文件的名称和扩展名。输入文件或是一个 Net.Data 宏，或是 INCLUDE 语句中指定的一个文件。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

```
<P>This file is <I>$(DTW_CURRENT_FILENAME)</I>,  
and was updated on $(DTW_CURRENT_LAST_MODIFIED).
```


DTW_CURRENT_LAST_MODIFIED

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

上次修改当前文件时的日期和时间。当前文件可以是一个 **Net.Data** 宏文件或在 **INCLUDE** 语句中指定的一个文件。其输出格式是由 **Net.Data** 运行的系统所确定的。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

```
<P>This file is <I>$(DTW_CURRENT_FILENAME)</I>,  
and was updated on $(DTW_CURRENT_LAST_MODIFIED).
```

DTW_DEFAULT_MESSAGE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

当调用内部函数或语言环境发生错误时，从调用返回的消息文本。

可以在 Net.Data 宏文件的任何部分使用 DTW_DEFAULT_MESSAGE 变量。

此变量是一个预定义变量，不建议修改它的值。把此变量用作一个变量引用。

例

例 1： 一条表示函数是否完成成功的消息

```
@function1()
%IF ("$(RETURN_CODE)" == "0")
  The function completed successfully.
%ELSE
  The function failed with the return code $(RETURN_CODE). The error message
  returned is "$(DTW_DEFAULT_MESSAGE)".
%ENDIF
```

例 2： 针对函数返回非零返回码时的缺省文本

```
%MESSAGE{
default: {<h2>Net.Data received return code: $(RETURN_CODE).
Error message is $(DTW_DEFAULT_MESSAGE)</h2> %} : continue
%}
```

如果函数返回了不是 0 的返回码，则用户见到缺省的错误信息。

DTW_LOG_LEVEL

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X			X	X	X

目的

Net.Data 写入记录文件的消息等级。

可以使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定此变量的值。

要求: 在 Net.Data 初始化文件中定义 DTW_LOG_DIR 以初始化记录; 否则, 当您在宏文件中指定 DTW_LOG_LEVEL 变量时 Net.Data 不记录消息。

值

DTW_LOG_LEVEL="OFF|ERROR|WARNING"

表 22. DTW_LOG_LEVEL 值

值	说明
OFF	Net.Data 不记录错误。OFF 是缺省。
ERROR	Net.Data 记录错误信息。
WARNING	Net.Data 记录警告和错误信息。

例

%DEFINE DTW_LOG_LEVEL="ERROR"

DTW_MACRO_FILENAME

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

当前 Net.Data 宏文件的名称和扩展名。

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

<P>This Net.Data macro is <I>\$(DTW_MACRO_FILENAME)</I>,
and was updated on \$(DTW_MACRO_LAST_MODIFIED).

DTW_MACRO_LAST_MODIFIED

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

上次修改 Net.Data 宏时的日期和时间。其输出格式是由 Net.Data 运行的系统所确定的。

| 此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

<P>This Net.Data macro is <I>\$(DTW_MACRO_FILENAME)</I>,
and was updated on \$(DTW_MACRO_LAST_MODIFIED).

DTW_MP_PATH

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

Net.Data 可执行文件的路径和名称。根据您的系统，输出看上去如下面例示出的路径和名称：

```
| /usr/lpp/internet/server_root/cgi-bin/db2www
```

| 此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

The Net.Data executable file is \$(DTW_MP_PATH).

DTW_MP_VERSION

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

运行在服务器上的 Net.Data 的版本和发行号。输出为如下的格式:

Net.Data 版本 2.1

此变量是一个预定义的变量，它的值不能修改。将此变量用作为变量引用。

例

This Web application uses \$(DTW_MP_VERSION).

DTW_PRINT_HEADER

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

指定用于 HTTP 首部的文本。

必须在 Net.Data 处理任何发送到 Web 浏览器的任何文本之前先设置此变量，因为 Net.Data 是在输出文本之前一次读取此变量的，以后将不再读。在 Net.Data 把文本发送到浏览器之后，对 DTW_PRINT_HEADER 变量的任何更改都将被忽略。

如果使用 DTW_PRINT_HEADER 来生成自己的标题 (DTW_PRINT_HEADER="NO"), 则必须设置 DTW_REMOVE_WS="NO"。

使用 DEFINE 语句或 @DTW_ASSIGN() 函数指定这个变量的值。

值

DTW_PRINT_HEADER="YES"|"NO"

表 23. DTW_PRINT_HEADER 值

值	说明
YES	Net.Data 针对 HTTP 标题打印出文本 Content-type: text/html。YES 是缺省。
NO	Net.Data 不打印 HTTP 标题。可以生成定制 HTTP 标题信息。

例

这个变量最常用于启用 Net.Data 宏，使它发送 cookie。要设置 cookie，则 DTW_PRINT_HEADER 变量必须设置为 NO，并且前三行必须为 Content-type 标题、Set-Cookie 语句和一个空行。

例 1: 允许 Net.Data 发送 cookie

```
%DEFINE DTW_PRINT_HEADER="NO"

%HTML(cookie1) {
Content-type: text/html
Set-Cookie: UsrId=56, expires=Friday, 12-Dec-99, 12:00:00 GMT; path=/

<P>
Any text
%}
```


DTW_REMOVE_WS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

通过压缩由制表程序、空格和换行字符引起的多余空格，减少自动生成的 Web 页面的大小。

在 DEFINE 块中指定这个变量的值。

使用 <PRE></PRE> 标记: 把此变量定义为 YES 将影响打印出的空格数和空格类型。如果此变量设置成 YES，使用 <PRE></PRE> 标记的 HTML 页面部分可能不会如期望的那样显示。

如果您正在使用 DTW_PRINT_HEADER 生成自己的标题(DTW_PRINT_HEADER="NO")，则必须设置 DTW_REMOVE_WS="NO"。

OS/390 用户: 在 Net.Data 初始化文件中设置这个变量，以便为所有的宏指定一个值。可以通过宏文件中的定义覆盖值。如果没有在宏文件中定义 DTW_REMOVE_WS，则它使用初始化文件中的值。

值

DTW_REMOVE_WS="YES" | "NO"

表 24. DTW_REMOVE_WS 值

值	说明
YES	Net.Data 把两个或者更多个空格序列压缩为一个换行字符，从而生成较短的 HTML 结果页面。
NO	Net.Data 不压缩空格。 NO 是缺省。

例

例 1: 压缩空格

DTW_REMOVE_WS="YES"

RETURN_CODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

调用内部函数或语言环境所返回的返回码。Net.Data 使用这个值来处理 MESSAGE 块。您可以使用这个值来确定一个函数调用是成功了还是失败了。 值 0 表示已成功地完成一个函数调用。

可以在 Net.Data 宏文件的任何部分引用 RETURN_CODE 变量。

这个值是预定义的；不建议修改它。把它用作一个变量引用。

例

例 1 一条表示函数是否完成成功的消息

```
@function1()
%IF ("$(RETURN_CODE)" == "0")
  The function completed successfully.
%ELSE
  The function failed with the return code $(RETURN_CODE).
%ENDIF
```

例 2: 返回码不是 0 时的缺省消息

```
%MESSAGE{
default: "<h2>Net.Data received return code: $(RETURN_CODE)</h2>" : continue
%}
```

如果函数返回不是 0 的返回码，则显示缺省消息。

第3章 Net.Data 内部函数

Net.Data 提供了大量不同的函数，您可以不用创建自己的 FUNCTION 块即可使用它们。Net.Data 内部函数分成以下类别：

- **一般目的函数**可以帮助您使用 Net.Data 来开发 Web 页，这些函数不包括在其它类别中。参阅第104页的『一般函数』。
- **数学函数**用于执行数学运算。请参阅第131页的『数学函数』。
- **字符串处理函数**修改字符串和字符。请参阅第142页的『字符串函数』。
- **字处理处理函数**修改单词或单词集合。请参阅第158页的『字处理函数』。
- **表格处理函数**帮助您从表格数据中生成表格和报表。请参阅第166页的『表格函数』。
- **平面文件接口函数**执行文件的输入和输出。请参阅第193页的『平面文件接口函数』。
- **Web 注册函数**执行 Web 注册表上的操作。请参阅 第214页的『Web 注册表函数』。
- **持久性的宏函数**支持 Net.Data 中的事务处理。请参阅第226页的『持久性宏函数』。

在下面的描述中，所说明的函数参数是 *string*、*integer*、*float* 和 *table* 类型。所有 Net.Data 变量都是字符串类型的，但仍使用术语“整数”、“浮点数”来分别表示整数、浮点数值的一个字符串。

函数名

Net.Data 内部函数以 DTW 开头，它是预定义的前缀。用户定义的函数不应使用此前缀。

对不是 Net.Data 内部函数的函数使用 DTW 前缀可能会导致无法预计的结果。

内部函数名是不区别大小写的。

输入和输出参数

函数可以有参数传递说明，它确定 Net.Data 是否使用参数来输入、输出或者既输入又输出。这些参数传递说明是由下列关键字来指定的：

IN 指定参数把输入数据从 Net.Data 传送至语言环境。

OUT 指定参数把输出数据从语言环境返回至 Net.Data。

INOUT

指定参数把输入数据从 Net.Data 传送至语言环境，再把输出数据从语言环境返回至 Net.Data。

函数结果格式

许多函数具有一个或多个的下列格式：

- 以 DTW_r、DTWF_r 和 DTWR_r 开头的函数将结果返回到函数调用，因此不需要指定输出参数。此例子显示了服务器的时间：
当前的本地时间是 @DTW_rTIME()。
- 以 DTW_m 开头的函数需要带多个参数。每个参数都同时作为输入参数和输出参数使用。函数按参数执行，结果则返回到函数中。本例将三个输入参数全部转换成大写字母，显示时外观保持一致：
@DTW_mUPPERCASE(model, style, shipNo)
Shipment \$(shipNo) contains \$(quantity) of model \$(model) \$(style).
- 其它以 DTW_、DTWF_ 和 DTWR_ 开头的函数将结果返回在一个输出参数中。因此必须指定输出参数。此例子显示了服务器的时间：
@DTW_TIME(nowTime)
当前的本地时间是 \$(nowTime)。

函数参数规则

按正确的顺序放置函数参数。可以在指定最后一个输入参数之前指定所有输入参数，或者指定一个空(『 』)来接受缺省值。例如，可以如下例所示调用 DTW_TB_INPUT_TEXT:

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2", "", "", "32")
```

在上面的例子中，第四和第五个参数使用缺省值。把它们作为空值包含进来，指出『 32』 是生成的 HTML 中 MAXLENGTH 的值。最后一个参数未指定，因此使用缺省值。 如果选择接受 MAXLENGTH 和前两个参数的缺省值，则将其省略，如下所示:

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2")
```

当后续参数中存在非空的输入参数时，必须指定输入参数列表中中间参数的空值。在指定最后的输出参数之前，不必要指定中间的空输入参数。

一般函数

一般函数帮助您使用 Net.Data 来开发 Web 页，这些函数不适合在其它类别中。下列函数是一般目的的函数:

- 第106页的『 DTW_ADDQUOTE 』
- 第108页的『 DTW_CACHE_PAGE 』
- 第111页的『 DTW_DATE 』
- 第112页的『 DTW_EXIT 』
- 第113页的『 DTW_GETCOOKIE 』
- 第115页的『 DTW_GETENV 』
- 第116页的『 DTW_GETINIDATA 』
- 第117页的『 DTW_HTMLLENCODE 』
- 第119页的『 DTW_QHTMLLENCODE 』
- 第120页的『 DTW_SENDMAIL 』
- 第123页的『 DTW_SETCOOKIE 』

- 第126页的『DTW_SETENV』
- 第127页的『DTW_TIME』
- 第129页的『DTW_URLESCSEQ』

DTW_ADDQUOTE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将一个输入字符串中的单引号替换成双引号。需要置换，使得当一个字符串包含单引号时能够正确处理 SQL 语句。

考虑对所有 SQL INPUT 语句使用此函数。例如，如果输入 O'Brien 作为姓名，则如下例所示，其中的单引号会导致一个错误：

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('O'Brien', 'Patrick')
```

使用 DTW_ADDQUOTE 函数更改 SQL 语句并防止错误：

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('O''Brien', 'Patrick')
```

格式

```
@DTW_ADDQUOTE(stringIn, stringOut)
@DTW_rADDQUOTE(stringIn)
@DTW_mADDQUOTE(stringMult, stringMult2, ..., stringMultn)
```

值

表 25. DTW_ADDQUOTE 参数

数据类型	参数	用法	说明
字符串	stringIn	IN	一个变量或文字串。 DTW_mADDQUOTE 可以有多个输入字符串。
字符串	stringOut	OUT	包含 stringIn 的修改格式的变量。
字符串	stringMult	INOUT	<ul style="list-style-type: none">输入： 一个包含字符串的变量。输出： 一个包含输入字符串的变量，其中每个单引号(')字符都被替换成两个单引号。

例

例 1： 在 OUT 参数上添加一个额外的单引号

```
@DTW_ADDQUOTE(string1,string2)
• 输入: string1="John's Web page"
• 返回: string2="John''s Web page"
```

例 2： 在函数调用的返回值上添加一个额外的单引号

```
@DTW_rADDQUOTE("The title of the article is 'Once upon a time'")
• 返回: "The title of the article is ''Once upon a time''"
```

例 3： 在函数的每个 INOUT 参数上添加额外的单引号

```
@DTW_mADDQUOTE(string1,string2)
```

- 输入: string1="Joe's bag", string2="'to be or not to be'"
- 返回: string1="Joe''s bag", string2="''to be or not to be''"

例 4: 在插入 DB2 表格的数据中插入额外的单引号

```
%FUNCTION(DTW_SQL) insertName(){
INSERT INTO USER1.CUSTABLE (LNAME,FNAME)
VALUES ('@DTW_rADDQUOTE(lastname)', '@DTW_rADDQUOTE(firstname)')
%}
```

- 输入: lastname="O'Brien", firstname="Patrick"
- 返回: "O''Brien", "Patrick"

DTW_CACHE_PAGE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X							

目的

把所有 HTML 输出，从此函数在宏文件中的位置开始作高速缓存。在调用此函数时，它试图从这个高速缓存开始检索指定的页面，并将它送至 Web 浏览器，就象它是从宏中生成的输出页面一样。如果找到这个页面并且它还没有过期，则 Net.Data 停止处理宏，退出宏文件，并把高速缓存中的页面送给 Web 浏览器。

如果请求的页不在高速缓存中，或者现有的高速缓存中的页比 *age* 的值要早，则 Net.Data 生成一个新的输出页。当宏成功完成后，Net.Data 把新的页发送给浏览器并放在高速缓存中。

确定 DTW_CACHE_PAGE 函数在宏文件中的位置:

- 对于大部分高速缓存应用程序，在宏的顶端开始指定 DTW_CACHE_PAGE 以将所有 HTML 输出放入高速缓存。这一技术使得在添加新的报表块时能更容易地维护宏文件。例如，当函数在宏的中间时，则如果一个 HTML 报表段添加到宏中前面的部分，则它可能不被注意到。Net.Data 不对新的报表输出作高速缓存。另外，因为在 Net.Data 确定此页面要作高速缓存时停止所有进一步的处理，所以这种方法提高了性能。
- 对于高级的高速缓存应用程序，当您需要在处理期间决定在某个特定点作高速缓存时，可以把此函数放在 HTML 输出段，而不是在宏文件的开始。例如，您可能需要根据从查询或函数调用返回的行数来作高速缓存决定。

格式

@DTW_CACHE_PAGE(cacheid, url, age, status)

值

表 26. DTW_CACHE_PAGE 参数

参数	用法	说明
<i>cache_id</i>	IN	标志放置此页的高速缓存的字符串变量。
<i>cached_page_ID</i>	IN	包含一个标识符的字符串变量，该标识符用于找出后继 DTW_CACHE_PAGE 高速缓存请求中的高速缓存页。此字符串可以是一个 URL。
<i>age</i>	IN	包含时间长度(以秒计)的字符串变量。这个参数确定页面是否过期。如果此页比 <i>age</i> 早，则它不发送给浏览器。 如果 <i>age</i> 指定为 -1，并且此页面存在于高速缓存中，则 Net.Data 不管它的年龄如何，直接把它从高速缓存中发送给 Web 浏览器。Net.Data 不替换高速缓存中的页。

表 26. DTW_CACHE_PAGE 参数 (续)

参数	用法	说明
<i>status</i>	OUT	<p>指出高速缓存页的状态的字符串变量。可能的值是小写形式的:</p> <ul style="list-style-type: none"> • ok: 输出页将在宏执行终止时作高速缓存。 • new: 此页不在高速缓存中。 • renew: 此页在高速缓存中, 但是已过期。 • no_cache: 指定的高速缓存标识符不存在。它必须定义在高速缓存配置文件中。您的宏可以继续执行而不作页面高速缓存。 • inactive: 指定的高速缓存已经标记为非活动的。您的宏可以继续执行而不作页面高速缓存。 • busy: 您的宏已经在此次执行前发出 DTW_CACHE_PAGE 内部函数。此宏可以继续执行。 • error: 在试图和高速缓存进行通信时发生错误。

例

例 1: 把 DTW_CACHE_PAGE 函数放在宏文件的开始, 以捕捉所有 HTML 输出

```
%IF (customer_status == "Classic")
@DTW_CACHE_PAGE("mymacro.mac", "http://www.mypage.org", "-1", status)
%ENDIF
% DEFINE { ...%}

...

%HTML(OUTPUT) {
<title>This is the page title
</head>
<body>
<center>
This is the Main Heading
<p>It is $(time). Have a nice day!
</body>
</html>

%}
```

例 2: 因为作高速缓存的决定依赖于 HTML 输出的期望尺寸, 所以把函数放在 HTML 块中

```
%DEFINE { ...%}

...

%FUNCTION(DTW_SQL) count_rows(){
select count(*) from customer
%REPORT{
%ROW{
@DTW_ASSIGN(ALL_ROWS, V1)
%}
%}
%}

%FUNCTION(DTW_SQL) all_customers(){
select * from customer
%}
```

```
%HTML(OUTPUT) {
<html>
<head>
<title>This is the customer list
</head>
<body>

@count_rows()

  %IF ($(ALL_ROWS) > "100")
@DTW_CACHE_PAGE("mymacro.mac", "http://www.mypage.org", "-1", status)
%ENDIF

@all_customers()

  </body>
</html>
%}
```

在此例中，此页根据 HTML 输出的期望尺寸作高速缓存或检索。只有当数据库表格包含多于 100 行时，才认为 HTML 输出页是值得作高速缓存的。Net.Data 总是在执行这个宏之后，把 OUTPUT 块中的文本(This is the customer list)发送给浏览器；此文本永不作高速缓存。跟在函数调用后的行 (@count_rows()) 在满足 IF 块的条件时作高速缓存或检索。两部分共同形成一个完整的 Net.Data 输出页。

例 3：动态地检索高速缓存标识符和放在高速缓存的页面标识符

```
%HTML(OUTPUT) {
  %IF (customer == "Joe Smith")

@DTW_CACHE_PAGE(@DTW_rGETENV("DTW_MACRO_FILENAME"), @DTW_rGETENV("URL"), "-1", status)

%ENDIF

...

<html>
<head>
<title>This is the page title</title>
</head>
<body>
<center>
<h3>This is the Main Heading</h3>
<p>It is @DTW_rDATE(). Have a nice day!
</body>
</html>

%}
```

DTW_DATE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

以指定的格式返回当前的系统日期。

格式

```
@DTW_DATE(format, stringOut)
@DTW_DATE(stringOut)
@DTW_rDATE(format)
@DTW_rDATE()
```

值

表 27. DTW_DATE 参数

数据类型	参数	用法	说明
字符串	<i>format</i>	IN	一个用于指定数据格式的变量或文字串。有效的格式包括： D - 一年中的日 (001-366) E - 欧洲日期格式 (dd/mm/yy) N - 常规日期格式 (dd mon yyyy) O - 顺序日期格式 (yy/mm/dd) S - 标准日期格式 (yyyymmdd) U - 美国日期格式 (mm/dd/yy) 缺省值为 N。
字符串	<i>stringOut</i>	OUT	一个包含有指定格式的日期的变量。

例

```
例 1: 常规日期格式
@DTW_DATE(results)
• 返回: results = "25 Apr 1997"

例 2: 欧洲日期格式
@DTW_DATE("E", results)
• 返回: results="25/04/97"

例 3: 美国日期格式
%HTML (report){
<P>This report created on @DTW_rDATE("U").
• 返回: 04/25/97
```

DTW_EXIT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

指定立即离开宏。Net.Data 保证这个宏到现在为止所创建的页将发送给浏览器。

性能提示：使用 DTW_EXIT，在生成输出后停止宏文件的处理，以节省 Net.Data 必须处理整个文件的时间。

重要事项！在添加 DTW_EXIT 函数之前，请确保整个宏在句法上是正确的。使用 DTW_EXIT() 可以使 Net.Data 在遇到对该函数的调用时停止对宏文件的处理，这样可以避免 DTW_EXIT() 函数处理之后所出现的错误。

格式

@DTW_EXIT()

例

例 1：退出宏

```
%HTML(cache_example) {  
  
<html>  
<head>  
<title>This is the page title</title>  
</head>  
<body>  
<center>  
<h3>This is the Main Heading</h3>  
<!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
<! Joe Smith sees a very short page                               !>  
<!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
%IF (customer == "Joe Smith")  
  
@DTW_EXIT()  
  
%ENDIF  
  
...  
  
</body>  
</html>  
%}
```

DTW_GETCOOKIE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X			X	X	X

目的

指定要阅读的 cookie 的名称，并返回它的值。

提示： 在两个独立的 HTTP 请求中定义并检索 cookie。因为只有在 cookie 被送给客户之后它才可见，所以宏试图获取在相同 HTTP 请求中已定义过的 cookie，您就可能收到意料之外的结果。

格式

@DTW_GETCOOKIE(IN cookie_name, OUT cookie_value)

@DTW_rGETCOOKIE(IN cookie_name)

值

表 28. DTW_GETCOOKIE 参数

数据类型	参数	用法	说明
字符串	cookie_name	IN	指定 cookie 名称的变量或文字串。
字符串	cookie_value	OUT	包含函数检索的 cookie 的值(例如用户状态信息)的变量。

使用法

如果未找到此 cookie，则返回用法注解 8000。可能由于下列原因找不到这个 cookie：

- 从未设置过这个 cookie。
- 此 cookie 已过期。
- 此 cookie 没有截止日期因此不持久；接收此 cookie 的 Web 浏览器已退出或者被杀死。
- 此 cookie 以一个安全选项来设置，而当前的 HTTP 请求在非安全通道更改上发送。
- 在设置 cookie 请求被提交时，Web 浏览器不接受 cookie 或不执行 JavaScript 程序。
- 此 cookie 已被 Web 浏览器删除。在 cookie 的数目超过浏览器的限度时可能发生这事。Netscape 的说明中描述了限制条件，到出版时，这些限制为：
 - 共 300 个 cookie
 - 每个 cookie 4 千字节(KB)，其中名称和值组合形成 4 千字节限制。
 - 每个服务器或域 20 个 cookie。(注意，完整指定的主机和域被看成是独立的实体，各自而不是组合起来有 20 个 cookie 的限制。)

服务器不应当期望客户超过这些限制。当超过 300 个 cookie 限制或每台服务器 20 个 cookie 的限制时，客户应当删除近期最少使用的 cookie。当遇到一个 cookie 大于 4 千字节时，此 cookie 应适当去尾，但是只要它小于 4 千字节，名称应保持原样。

参阅 Netscape 的说明，获取『Persistent Client State HTTP Cookies』中的最新信息，它在下面的网址可用：

http://search.netscape.com/newsref/std/cookie_spec.html

例

例 1: 检索包含用户标识符和口令信息的 cookie

```
@DTW_GETCOOKIE("mycookie_name_for_userID", userID)
@DTW_GETCOOKIE("mycookie_name_for_password", password)
```

例 2: 确定在收集用户信息之前某个用户的 cookie 是否已存在

```
%MESSAGE {
    8000 : "" : continue
}%

%HTML(welcome) {
<html>
<body>
    <h1>Net.Data Club</h1>
    @DTW_GETCOOKIE("NDC_name", name)
    %IF ( $(RETURN_CODE) == "8000" ) %{ The cookie is not found. %}
    <form method="post" action="remember">
    <p>Welcome to the club. Please enter your name.<br>
    <input name="name">
    <input type="submit" value="submit"><br>
</form>
%ELSE
    <p>Hi, $(name). Welcome back.
%ENDIF
</body>
</html>
%}
```

HTML 欢迎段检查 cookie NDC_name 是否存在。如果 cookie 存在，则浏览器显示个人的问候。如果此 cookie 不存在，则浏览器提示要求用户的名字，把它投递到 HTML 记忆段，把用户的名字设置到 cookie NDC_name 里，如下所示：

```
%HTML(remember) {
<html>
<body>
    <H1>Net.Data Club</H1>
    @DTW_SETCOOKIE("NDC_name", name, "expires=Wednesday, 01-Dec-2010 00:00:00;path=/")
    <p>Thank you.
    <p><a href="welcome">Come back</a>
</body>
</html>
%}
```

DTW_GETENV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回指定的环境变量的值。同样可以使用 ENVVAR 来引用环境变量的值。有关的更多信息，请参阅第12页的『ENVVAR 语句』。

格式

```
@DTW_GETENV(envVarName, envVarValue)
@DTW_rGETENV(envVarName)
```

值

表 29. DTW_GETENV 参数

数据类型	参数	用法	说明
字符串	envVarName	IN	一个变量或文字串。
字符串	envVarValue	OUT	正在 envVarName 中指定的环境变量的值。如果找不到这个值则返回空串。

例

例 1: 返回 PATH 语句 OUT 参数上的值

```
@DTW_GETENV(myEnvVarName, myEnvVarValue)
• 输入: myEnvVarName = "PATH"
• 返回: myEnvVarValue = "/usr/path"
```

例 2: 返回 PATH 语句的值

```
@DTW_rGETENV(myPath)
• 输入: myPath = "PATH"
• 返回: "/usr/path"
```

例 3: 返回服务器的名称值

```
The server is @DTW_rGETENV("SERVER_NAME").
• 返回: "www.software.ibm.com"
```

DTW_GETINIDATA

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回指定的配置变量的值。如果找不到这个值则返回空串。

| 限制: 对于非 OS/400 操作系统, 不能用这个调用检索配置路径变量 (MACRO_PATH、
| EXEC_PATH、INCLUDE_PATH) 以及 ENVIRONMENT 语句。在 OS/400 操作系统上,
| 这个限制只适用于 ENVIRONMENT 语句。

格式

@DTW_GETINIDATA(iniVarName, iniVarValue)

@DTW_rGETINIDATA(iniVarName)

值

表 30. DTW_GETINIDATA 参数

数据类型	参数	用法	说明
字符串	<i>iniVarName</i>	IN	一个变量或文字串。
字符串	<i>iniVarValue</i>	OUT	<i>iniVarName</i> 中指定的配置变量的值。

例

例 1: 返回 Net.Data 路径变量值

@DTW_GETINIDATA(myEnvVarName, myEnvVarValue)

- 输入: myEnvVarName = "FFI_PATH"
- 返回: myEnvVarValue = "D:\FFI"

DTW_HTMLENCODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

大多数(但非所有)字符的编码字符(使用标准 HTML 十进制转义代码。)您可以使用此函数对不想让 Web 浏览器解释为 HTML 的数据进行编码。例如，通过使用适当的转义字符，可以显示小于(<)和大于(>)符号，这些符号通常作为 HTML 标记保留。

在第二个例子中，HTML 中的以下字符串在每个数字之间只显示一个空格。

1 2 3

使用 DTW_HTMLENCODE 来确保显示正确个数的空格。

表31显示由 DTW_HTMLENCODE 函数编码的字符。

表 31. HTML 十进制转义字符

字符	名称	代码
SPACE	空格	
"	双引号	"
#	数值符号	#
%	百分号	%
&	&符号	&
[左方括号	(
]	右方括号)
+	加号	+
\	斜杠	/
:	冒号	:
;	分号	;
<	小于	<
=	等于	=
>	大于	>
?	问号	?
@	@符号	@
/	反斜杠	\
^	^符号	^
{	左花括号	{
	竖线	|
}	右花括号	}
~	~符号	~

格式

@DTW_HTMLENCODE(stringIn, stringOut)
@DTW_rHTMLENCODE(stringIn)

值

表 32. DTW_HTMLENCODE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
字符串	<i>stringOut</i>	OUT	一个包含已修改的输入字符串的变量，其中输入字符串中的某些字符已经被编码过的 HTML 转义字符所替换。

例

例 1: 编码的空格字符

```
@DTW_HTMLENCODE(string1,string2)
• 输入: string1 = "Jim's dog"
• 返回: string2 = "Jim's&#32;dog"
```

例 2: 编码空格，小于符号和等号

```
@DTW_rHTMLENCODE("X <= 10")
• 返回: "X&#32;&#60;&#61;&#32;10"
```

DTW_QHTMLENCODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

| 执行和 @DTW_HTMLENCODE 相同的函数，但把单引号字符(')编码成 '。
| DTW_QHTMLENCODE 所使用的 HTML 十进制转义字符显示在第117页的表31中。

格式

```
@DTW_QHTMLENCODE(stringIn, stringOut)
@DTW_rQHTMLENCODE(stringIn)
```

值

表 33. DTW_QHTMLENCODE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
字符串	<i>stringOut</i>	OUT	一个包含 <i>stringIn</i> 已修改格式的变量，其中输入字符串中的某些字符已经被编码过的 HTML 转义字符所替换。

例

例 1: 编码一个单引号和一个空格

```
@DTW_QHTMLENCODE(string1,string2)
• 输入: string1 = "Jim's dog"
• 返回: string2 = "Jim&#39;s&#32;dog"
```

例 2: 编码单引号、空格和一个与符号

```
@DTW_rQHTMLENCODE("John's & Jane's")
• 返回: "John&#39;s&#32;&#38;&#32;Jane&#39;s"
```

DTW_SENDMAIL

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X			X	X	X

目的

动态地构建并传输电子邮件(e-mail)消息。

此函数与一个可选的配置变量 (DTW_SMTP_SERVER) 一起工作，它指定 SMTP 服务器用于传送电子邮件消息。这个参数的值可以是一个主机名，或是一个 IP 地址。如果没有定义这个变量，则 Net.Data 把本地主机用作 SMTP 服务器。请参阅 *Net.Data 管理与程序设计指南* 中的配置一章，进一步学习这些变量。

国家语言问题: 标准简单邮件传送协议(SMTP)服务器只接受 7 位(例如美国 ASCII 码字符)数据。如果您的消息有 8 位字符，则建议指定一个扩展的简单邮件传送协议(ESMTP)服务器；ESMTP 服务器接受 8 位字符。Net.Data 不把 8 位数据编码成 7 位数据。如果没有对 ESMTP 服务器的访问权，则从电子邮件消息中除去所有 8 位的字符。

故障排除: 下面的列表描述了 Net.Data 不发送电子邮件消息的几种情况:

- 无法到达指定的 SMTP 服务器。
- 指定的 SMTP 服务器不支持扩展的简单邮件传输 (ESMTP)，但指定的电子邮件消息中包含非美国的 ASCII 字符。

格式

```
@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy,
IN BlindCarbonCopy, IN ReplyTo, IN Organization)

@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy,
IN BlindCarbonCopy, IN ReplyTo)

@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy,
IN BlindCarbonCopy)

@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy)

@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject)

@DTW_SENDMAIL(IN Sender, IN Recipient, IN Message)
```

值

表 34. DTW_SENDMAIL 参数

数据类型	参数	用法	说明
字符串	<i>sender</i>	IN	指定作者地址的变量或文字串。此参数是必需的。 有效的格式是: <ul style="list-style-type: none">• Name <user@domain>• <user@domain>• user@domain

表 34. DTW_SENDMAIL 参数 (续)

数据类型	参数	用法	说明
字符串	<i>recipient</i>	IN	指定发送此消息的目标电子邮件地址的变量或文字串。这个值包含多个收件人，之间用逗号(,)隔开。此参数是必需的。有效的 <i>recipient</i> 格式是： <ul style="list-style-type: none">• Name <user@domain>• <user@domain>• user@domain
字符串	<i>message</i>	IN	包含电子邮件消息文本的变量或文字串。此参数是必需的。
字符串	<i>subject</i>	IN	包含主题行文本的变量或文字串。
字符串	<i>CarbonCopy</i>	IN	包含电子邮件地址或者附加收件人的名称和电子邮件地址的变量或文字串。这个值包含多个附加的收件人，之间用逗号(,)隔开。参阅 <i>Recipient</i> 参数的有效收件人格式。
字符串	<i>BlindCarbonCopy</i>	IN	包含电子邮件地址，或者附加收件人的名称和电子邮件地址，但是收件人不出现在电子邮件标题中的变量或文字串。这个值包含多个附加的收件人，之间用逗号(,)隔开。参阅 <i>Recipient</i> 参数的有效收件人格式。
字符串	<i>ReplyTo</i>	IN	包含答复这条发送消息的电子邮件地址的变量或者文字串。有效的 <i>ReplyTo</i> 格式是： <ul style="list-style-type: none">• Name <user@domain>• <user@domain>• user@domain
字符串	<i>Organization</i>	IN	包含 <i>sender</i> 机构名称的变量或者文字串。

例

例 1: 构建并发送一个简单电子邮件消息的函数调用

```
@DTW_SENDMAIL("<ibmuser1@ibm.com>", "<ibmuser2@ibm.com>", "There is a meeting at 9:30.", "Status meeting")
```

DTW_SENDMAIL 函数发送一条具有下列信息的电子邮件消息:

```
Date: Mon, 3 Apr 1998 09:54:33 PST
To: <ibmuser2@ibm.com>
From: <ibmuser1@ibm.com>
Subject: Status meeting

There is a meeting at 9:30.
```

Date 的信息通过使用系统日期和时间函数来构造，并且以 SMTP 特定数据格式格式化。

例 2: 构建并发送一条电子邮件消息的函数调用，该消息具有多个收件人、复写拷贝和盲目复写拷贝收件人以及公司名称

```
@DTW_SENDMAIL("IBM User 1 <ibmuser1@ibm.com>", "IBM User 2 <ibmuser2@ibm.com>, IBM User 3 <ibmuser3@ibm.com>, IBM User 4 <ibmuser4@ibm.com>, IBM User 5 <ibmuser5@ibm.com>", "Status meeting", "Status meeting")
```

DTW_SENDMAIL 函数发送一条具有下列信息的电子邮件消息:

```
Date: Mon, 3 Apr 1998 09:54:33 PST
To: IBM User 2 <ibmuser2@ibm.com>, IBM User 3 <ibmuser3@ibm.com>, IBM User 4 <ibmuser4@ibm.com>
CC: IBM User 5 <ibmuser5@ibm.com>
```

```
BCC: IBM User 6 <ibmuser6@ibm.com>
From: IBM User 1 <ibmuser1@ibm.com>
ReplyTo: meeting@ibm.com
Organization: IBM
Subject: Status meeting
```

There is a meeting at 9:30.

例 3: 通过 Web 格式的界面构建和发送电子邮件的宏

```
%HTML(start) {
<html>
<body>
<h1>Net.Data E-Mail Example</h1>
<form method="post" action="sendemail">
<p>To:<br><input name="recipient"><p>
Subject:<br><input name="subject"><p>
Message:<br><textarea name="message" rows=20 cols=40>
</textarea><p>
<input type="submit" value="Send E-mail"><br>
</form>
</body>
</html>
%}

%HTML(sendemail) {
<html>
<body>
<h1>Net.Data E-Mail Example</h1>
@DTW_SENDMAIL("Net.Data E-mail Service <netdata@us.ibm.com>", recipient, message, subject)
<p>E-mail has been sent out.
</body>
</html>
%}
```

这个宏通过一个 Web 格式界面发送电子邮件。HTML 开始段显示一张表，在其中可以输入收件人的电子邮件地址、主题和消息。当用户单击**发送电子邮件**按钮时，此条消息发送给 HTML(vsendemail) 段中指定的收件人。这一段调用 DTW_SENDMAIL 并使用从 Web 表中获取的参数，确定电子邮件消息的内容，以及发送者和收件人。一旦发出电子邮件消息，显示一条确认通知。

例 4: 使用 SQL 查询确定收件人列表的一个宏

```
%Function(DTW_SQL) mailing_list(IN message) {
  SELECT EMAIL_ADDRESS FROM CUSTOMERS WHERE ZIPCODE='CA'
  %REPORT {
    Sending out product information to all customers who live in California...<P>
    %ROW {
      @DTW_SENDMAIL("John Doe Corp. <John.Doe@doe.com>", V1, message, "New Product Release")
      E-mail sent out to customer ${V1}.<BR>
    }
  }
%}
```

这个宏根据客户数据中 SQL 查询的结果，向特定的客户组发送一个自动化的电子邮件消息。此 SQL 查询还检索这些客户的电子邮件地址。电子邮件内容由 *message* 的值确定，可以是静态的，也可以是动态的(例如，可以使用另一个 SQL 查询动态地指定产品的版本号或不同的提供价格)。

DTW_SETCOOKIE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X			X	X	X

目的

定义一个 cookie 名称、值和选项，例如到期日和安全性要求。

要检索一个 cookie，使用 DTW_SETCOOKIE() 函数。参阅第113页的『DTW_GETCOOKIE』学习如何定义 cookie。

如果未指定 secure 要求，则此 cookie 可能在不保证安全的通道中发送。安全选项不要求浏览器对 cookie 进行加密，也不保证包含 DTW_SETCOOKIE 语句的页面在 SSL 中传输。

提示:

- 在两个独立的 HTTP 请求中定义并检索 cookie。因为只有在 cookie 被送给客户之后它才可见，所以宏试图获取在相同 HTTP 请求中已定义过的 cookie，您就可能收到意料之外的结果。
- 要获得简明性，cookie 中避免使用分号、逗号和空格。如果它们是必需的，则使用 Net.Data 函数 DTW_rURLESCSEQ，在把包含特殊字符的字符串送给 DTW_SETCOOKIE 之前先处理它。例如，
@DTW_SETCOOKIE("my_cookie_name", @DTW_rURLESCSEQ("my cookie value"))

限制:

- 如果客户 Web 浏览器不支持 Java Script，则浏览器不设置 cookie。
- 因为 DTW_SETCOOKIE 生成 Java Script 代码，不要在 <SCRIPT> 或 <NOSCRIPT> HTML 元素内调用 DTW_SETCOOKIE。

格式

```
@DTW_SETCOOKIE(IN cookie_name, IN cookie_value, IN advanced_options)
@DTW_SETCOOKIE(IN cookie_name, IN cookie_value)
```

值

表 35. DTW_SETCOOKIE 参数

数据类型	参数	用法	说明
字符串	cookie_name	IN	指定 cookie 名称的变量或文字串。
字符串	cookie_value	IN	指定 cookie 值的变量或文字串。

表 35. DTW_SETCOOKIE 参数 (续)

数据类型	参数	用法	说明
字符串	<i>advanced_options</i>	IN	<p>包含用于定义 cookie 的可选属性，属性间用分号分隔的字符串。这些属性是：</p> <p>expires = date 指定一个日期字符串，它定义 cookie 的有效寿命。在此日期期满后，将不再存储或检索这个 cookie。语法： <i>weekday, DD-month-YYYY HH:MM:SS GMT</i></p> <p>其中：</p> <p><i>weekday</i> 指定星期中各天的全名。</p> <p><i>DD</i> 指定月份的数值日期。</p> <p><i>month</i> 指定月份的三字符缩写。</p> <p><i>YYYY</i> 指定四个字符的数字年份。</p> <p><i>HH:MM:SS</i> 用小时、分钟和秒指定时间戳记。</p> <p>domain = domain_name 指定 cookie 的域属性，供域属性匹配使用。</p> <p>path = path 指定针对此 cookie 有效的域中 URL 的子集。</p> <p>secure 指定只在安全通道中才把此 cookie 传输到 HTTP 服务器。</p>

例

例 1: 定义具有“安全”增强选项的、包含用户标识符和口令信息的 cookie

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT", "secure")
```

例 2: 定义包含过期日期增强选项的 cookie

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1", "expires=Wednesday,
01-Dec-2010 00:00:00")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT", "expires=Wednesday,
01-Dec-2010 00:00:00; secure")
```

例 3: 确定在收集用户信息之前某个用户的 cookie 是否已存在

```
%HTML(welcome) {
<html>
<body>
<h1>Net.Data Club</h1>
@DTW_GETCOOKIE("NDC_name", name)
%IF (%RETURN_CODE) == "8000") %{ The cookie is not found. %{
<form method="post" action="remember">
<p>Welcome to the club. Please enter your name.<br>
<input name="name">
```



```

        <input type="submit" value="submit"><br>
    </form>
%ELSE
    <p>Hi, $(name). Welcome back.
%ENDIF
</body>
</html>
%}

```

HTML(welcome) 段检查 cookie NDC_name是否存在。如果 cookie 存在，则浏览器显示个人的问候。如果此 cookie 不存在，则浏览器提示要求用户的名字，把它投递到 HTML(remember) 记忆段。这一段把用户的名字记录到 cookie NDC_name 中，如下所示：

```

%HTML(remember) {
<html>
<body>
    <H1>Net.Data Club>
    @DTW_SETCOOKIE("NDC_name", name, "expires=Wednesday, 01-Dec-2010 00:00:00;path=/")
    <p>Thank you.
    <p><a href="welcome">Come back</a>
</body>
</html>
%}

```

DTW_SETENV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

用指定的值指定一个环境变量并返回先前的值。如果找不到这个值则返回空串。

格式

```
@DTW_SETENV(envVarName, envVarValue, prevValue)
@DTW_rSETENV(envVarName, envVarValue)
```

值

表 36. DTW_SETENV 参数

数据类型	参数	用法	说明
字符串	<i>envVarName</i>	IN	一个表示环境变量的变量或文字串。
字符串	<i>envVarValue</i>	IN	具有要赋给此环境变量的值的变量或者文字串。
字符串	<i>prevValue</i>	OUT	一个包含有环境变量先前值的变量。

例

例 1: 返回先前路径的值

```
@DTW_SETENV("PATH", "myPath", prevValue)
• 输入: envVarName = "PATH", envVarValue = "myPath"
• 返回: prevValue = "myPreviousPath"
```

例 2: 返回先前路径的值，并指定 PATH 值的值

```
@DTW_rSETENV("PATH", "myPath")
• 输入: envVarName = "PATH", envVarValue = "myPath"
• 返回: "myPreviousPath", PATH = "myPath"
```

DTW_TIME

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

以指定的格式返回当前的系统时间。

格式

```
@DTW_TIME(stringIn, stringOut)
@DTW_TIME(stringOut)
@DTW_rTIME(stringIn)
@DTW_rTIME()
```

值

表 37. DTW_TIME 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个用于指定时间格式的变量或文字串。 有效的格式是: C - 民用时间 (使用12小时时钟的 hh:mmAM/PM) L - 本地时间 (hh:mm:ss) N - 常规时间 (使用24小时时钟的 hh:mm:ss) H - 自子夜以来的小时数 M - 自子夜以来的分钟数 S - 自子夜以来的秒数
字符串	<i>stringOut</i>	OUT	一个包含有指定格式的时间的变量。

例

```
例 1: 24 小时时钟格式
@DTW_TIME(results)
• 返回: results = "10:30:53"

例 2: 民用时间格式
@DTW_TIME("C", results)
• 返回: results = "10:30AM"

例 3: 用函数调用返回自子夜以来的分钟数
@DTW_rTIME("M")
• 返回: "630"

例 4: 用函数调用返回缺省时间和日期格式
%REPORT{
<P>This report was created at @DTW_rTIME(), @DTW_rDATE().
%}
```

- 返回: This report was created 15:04:39, 01 May 1997.

DTW_URLESCSEQ

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

用转换值替换 URL 中不允许的字符，也称 URL 编码值。必须使用此函数把在表38中列出的字符传递给另一个宏文件或 HTML 模块。

表 38. URL 中不允许的字符

字符	名称	代码
SPACE	空格	%20
"	双引号	%22
#	数值符号	%23
%	百分号	%25
&	&符号	%26
+	加号	%2B
\	反斜杠	%2F
:	冒号	%3A
;	分号	%3B
<	小于	%3C
=	等于	%3D
>	大于	%3E
?	问号	%3F
@	@符号	%40
[左方括号	%5B
/	斜杠	%5C
]	右方括号	%5D
^	^符号	%5E
{	左花括号	%7B
	竖线	%7C
}	右花括号	%7D
~	~符号	%7E

格式

@DTW_URLESCSEQ(stringIn, stringOut)

@DTW_rURLESCSEQ(stringIn)

值

表 39. DTW_URLESCSEQ 参数

数据类型	参数	用法	说明
字符串	stringIn	IN	一个变量或文字串。

表 39. DTW_URLESCSEQ 参数 (续)

数据类型	参数	用法	说明
字符串	<i>stringOut</i>	OUT	一个包含输入字符串的变量，其中输入字符串中不允许在 URL 中出现的字符用其十六进制的转义值替换。

例

例 1: 用其 URL 转换代码替换 *string1* 中的空格和与符号，并把结果分配给 *string2*

```
@DTW_URLESCSEQ(string1,string2)
```

- 输入: `string1 = "Guys & Dolls"`
- 返回: `string2 = "Guys%20%26%20Dolls"`

例 2: 把空格和一个与符号转换成 URL 编码格式

```
@DTW_rURLESCSEQ("Guys & Dolls")
```

- 返回: `"Guys%20%26%20Dolls"`

例 3: 在 ROW 块中使用 DTW_rURLESCSEQ，并把空格和 “[@0000]” 符号转换成 URL 编码格式

```
%ROW{
<P><a href="fullrpt.mac/input?name=@DTW_rURLESCSEQ(V1)&email=@DTW_rULRESCSEQ(V2)">
$(V1)</a>
%}
```

- 输入: `V1="Patrick O'Brien", V2="obrien@ibm.com"`
- 返回:

```
<P><a href="fullrpt.mac/input?name=Patrick%20'O'Brien&email="obrien%40ibm.com">
Patrick O'Brien</a>
```

当应用程序用户单击名称时，名称和 e-mail 地址都被发送到 Net.Data 宏 fullrpt.mac 的输入模块中，编码后的值作为变量 *name* 和 *email*。

数学函数

这些函数可以让您完成数学运算。

针对 UNIX、Windows NT 和 OS/2 的性能提示： 为了优化算术函数的性能，把 Net.Data 初始化文件或宏文件中的 DTW_OPTIMIZE_MATH 配置值设置成 YES 即可。

- 如果设置成 YES，Net.Data 使用 C 数学格式化，函数能更快地运行；但是，输出格式和没有这个变量的格式不同。小数点后的尾随零不显示。
- 如果 DTW_OPTIMIZE_MATH 设置成 NO，Net.Data 使用 REXX 数学格式。函数运行得较慢，但是输出格式和由先前版本的 Net.Data 生成的输出格式一致。缺省值是 NO。

请参阅 *Net.Data 管理与程序设计指南* 中的配置变量章节，学习如何配置此变量。

NLS 针对数学函数的考虑： Net.Data 根据运行 Net.Data 的 Web 服务器上指定的区域设置在数值中显示小数点。例如，如果 Web 服务器上指定小数点为一个逗号 (,)，则 Net.Data 使用逗号将十进制数据格式化。Net.Data 使用以下设置确定哪个字符用于指定小数点：

对于 OS/390、Windows NT、OS/2 和 UNIX 操作系统：

Web 服务器上的 LOCALE 设置

对于 OS/400 操作系统：

- V4R2 或后继发行版：由用来运行这个进程的用户简要表指定
- V4R1 或前发行版：从 QDECFMT 系统值进行检索。

对于数学计算，有下列函数可用：

- 第132页的『DTW_ADD』
- 第133页的『DTW_DIVIDE』
- 第134页的『DTW_DIVREM』
- 第135页的『DTW_FORMAT』
- 第138页的『DTW_INTDIV』
- 第139页的『DTW_MULTIPLY』
- 第140页的『DTW_POWER』
- 第141页的『DTW_SUBTRACT』

DTW_ADD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将两个参数的值相加。

格式

@DTW_ADD(number1, number2, precision, result)

@DTW_ADD(number1, number2, result)

@DTW_rADD(number1, number2, precision)

@DTW_rADD(number1, number2)

值

表 40. DTW_ADD 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 与 <i>number2</i> 之和的变量。

例

例 1:

@DTW_ADD(NUM1, NUM2, "2", result)

- 输入: NUM1 = "105", NUM2 = "3"
- 返回: result = "1.1E+2"

例 2:

@DTW_rADD("12", NUM2, "5")

- 输入: NUM2 = "7.00"
- 返回: "19.00"

DTW_DIVIDE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

第一个参数的值除以第二个参数的值。

格式

```
@DTW_DIVIDE(number1, number2, precision, result)
@DTW_DIVIDE(number1, number2, result)
@DTW_rDIVIDE(number1, number2, precision)
@DTW_rDIVIDE(number1, number2)
```

值

表 41. DTW_DIVIDE 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 除以 <i>number2</i> 的结果的变量。

例

```
例 1:
@DTW_DIVIDE("8.0", NUM2, result)
• 输入: NUM2 = "2"
• 返回: result = "4"

例 2:
@DTW_rDIVIDE("1", NUM2, "5")
• 输入: "1", NUM2 = "3"
• 返回: "0.33333"

例 3:
@DTW_rDIVIDE(NUM1, "2", "5")
• 输入: NUM1 = "5"
• 返回: "2.5"
```

DTW_DIVREM

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

第一个参数除以第二个参数，并返回余数。如果余数非零，则余数的符号与第一个参数相同。

格式

```
@DTW_DIVREM(number1, number2, precision, result)
@DTW_DIVREM(number1, number2, result)
@DTW_rDIVREM(number1, number2, precision)
@DTW_rDIVREM(number1, number2)
```

值

表 42. DTW_DIVREM 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或文字串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或文字串。
整数	<i>precision</i>	IN	一个表示正整数的变量或文字串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 除以 <i>number2</i> 的余数的变量。

例

例 1:

```
@DTW_DIVREM(NUM1, NUM2, result)
```

- 输入: NUM1 = "2.1", NUM2 = "3"
- 返回: result = "2.1"

例 2:

```
@DTW_rDIVREM("10", NUM2)
```

- 输入: NUM2 = "0.3"
- 返回: "0.1"

例 3:

```
@DTW_rDIVREM("3.6", "1.3")
```

- 返回: "1.0"

例 4:

```
@DTW_rDIVREM("-10", "3")
```

- 返回: "-1"

DTW_FORMAT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

定制一个数的格式。如果只指定 *number* 参数，则其格式化结果与执行 @DTW_rADD(number, 『0』) 语句相同。如果指定了其它某个选项，那么将按照以下规则来格式化 *number*:

- *before* 和 *after* 参数描述了分别有多少个字符用于 *result* 的整数和小数部分。如果省略了这些参数，则用于该部分的字符个数与结果所需的长度相同。
- 如果 *before* 的值不够大，无法包含此数的整数部分(加上正数的符号)，则产生一个错误的结果。如果 *before*参数大于那部分所需的大小，则 *number* 参数值左边用空格填补。如果 *after* 参数的大小与 *number* 参数的小数部分不同，则此数将被舍入(或以 0 来扩充)，使其结果为 *after* 的指定大小。如果指定 0，这将引起此数被舍入成为一个整数。
- 另外，*expp* 和 *expt* 参数控制结果的指数部分。*expp* 参数设置指数部分的位数；缺省情况下，将使用所需的位数(可能为 0)。*expt* 参数设置使用指数表示法的触发点。其缺省值是精度参数的缺省值。
- 如果 *expp* 是 0，则不提供指数表示，而以简单形式来表示此数，需要时添加 0。如果 *expp* 不能足以包含指数，则结果错误。
- 如果整数或小数部分所需的位数分别超过 *expt* 或 *expt* 的两倍，则使用指数部分。如果 *expt* 是 0，则总是使用指数表示法，除非指数为 0。(如果 *expt* 为 0，这将覆盖 *expt* 值为 0 的情况。)如果 *expp* 为非零时指数为 0，那么将在结果的指数部分提供 *expp*+2 个空格。如果指数是 0，并且没有指定 *expp*，则使用简单表示形式。

格式

```
@DTW_FORMAT(number, before, after, expp, expt, precision, result)
@DTW_FORMAT(number, before, after, expp, expt, result)
@DTW_FORMAT(number, before, after, expp, result)
@DTW_FORMAT(number, before, after, result)
@DTW_FORMAT(number, before, result)
@DTW_FORMAT(number, result)
@DTW_rFORMAT(number, before, after, expp, expt, precision)
@DTW_rFORMAT(number, before, after, expp, expt)
@DTW_rFORMAT(number, before, after, expp)
@DTW_rFORMAT(number, before, after)
@DTW_rFORMAT(number, before)
@DTW_rFORMAT(number)
```

值

表 43. DTW_FORMAT 参数

数据类型	参数	用法	说明
浮点数	<i>number</i>	IN	一个表示数值的变量或文字串。
整数	<i>before</i>	IN	一个表示正整数的变量或文字串。这是一个可选参数。 必须输入一个空串(""), 以使用另外的参数。
整数	<i>after</i>	IN	一个表示正整数的变量或文字串。这是一个可选参数。 必须输入一个空串(""), 以指定另外的参数。
整数	<i>expp</i>	IN	一个表示正整数的变量或文字串。必须指定一个空串 ("")来指定附加的参数。
整数	<i>expt</i>	IN	一个表示正整数的变量或文字串。必须输入一个空串(""), 以指定另外的参数。
整数	<i>precision</i>	IN	一个表示正整数的变量或文字串, 此整数指定了结果的精确度。 缺省值是 9。
浮点数	<i>result</i>	OUT	一个包含指定舍入和格式状态的数的变量。

例

例 1:

```
@DTW_FORMAT(NUM, BEFORE, result)
• 输入: NUM = "3", BEFORE = "4"
• 返回: result= " 3"
```

例 2:

```
@DTW_FORMAT("1.73", "4", "0", result)
• 返回: result = "2"
```

例 3:

```
@DTW_FORMAT("1.73", "4", "3", result)
• 返回: result = " 1.730"
```

例 4:

```
@DTW_FORMAT(" - 12.73", "", "4", result)
• 返回: result = "-12.7300"
```

例 5:

```
@DTW_FORMAT("12345.73", "", "", "2", "2", result)
• 返回: result = "1.234573E+04"
```

例 6:

```
@DTW_FORMAT("1.234573", "", "3", "", "0", result)
• 返回: result = "1.235"
```

例 7:

```
@DTW_rFORMAT(" - 12.73")
• 返回: " - 12.73"
```

例 8:

@DTW_rFORMAT("0.000")

- 返回: "0"

例 9:

@DTW_rFORMAT("12345.73","", "", "3", "6")

- 返回: "12345.73"

例 10:

@DTW_rFORMAT("1234567e5","", "3", "0")

- 返回: "123456700000.000"

例 11:

@DTW_rFORMAT("12345.73","", "3", "", "0")

- 返回: "1.235E+4"

DTW_INTDIV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

第一个参数除以第二个参数，并返回结果的整数部分。

格式

```
@DTW_INTDIV(number1, number2, precision, result)
@DTW_INTDIV(number1, number2, result)
@DTW_rINTDIV(number1, number2, precision)
@DTW_rINTDIV(number1, number2)
```

值

表 44. DTW_INTDIV 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 除以 <i>number2</i> 整数部分的的变量。

例

例 1:

```
@DTW_INTDIV(NUM1, NUM2, result)
• 输入: NUM1 = "10", NUM2 = "3"
• 返回: result = "3"
```

例 2:

```
@DTW_rINTDIV("2", NUM2)
• 输入: NUM2 = "3"
• 返回: "0"
```

DTW_MULTIPLY

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

两个参数相乘并返回结果。

格式

```
@DTW_MULTIPLY(number1, number2, precision, result)
@DTW_MULTIPLY(number1, number2, result)
@DTW_rMULTIPLY(number1, number2, precision)
@DTW_rMULTIPLY(number1, number2)
```

值

表 45. DTW_MULTIPLY 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 和 <i>number2</i> 的乘积的变量。

例

例 1:

```
@DTW_MULTIPLY(NUM1, NUM2, result)
• 输入: NUM1 = "4", NUM2 = "5"
• 返回: result = "20"
```

例 2:

```
@DTW_rMULTIPLY("0.9", NUM2)
• 输入: NUM2 = "0.8"
• 返回: "0.72"
```

DTW_POWER

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

计算第一个参数的第二个参数的次方，返回其结果。

格式

```
@DTW_POWER(number1, number2, precision, result)
@DTW_POWER(number1, number2, result)
@DTW_rPOWER(number1, number2, precision)
@DTW_rPOWER(number1, number2)
```

值

表 46. DTW_POWER 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 的 <i>number2</i> 次方的结果的变量。

例

```
例 1:
@DTW_POWER(NUM1, NUM2, result)
• 输入: NUM1 = "2", NUM2 = "-3"
• 返回: result = "0.125"

例 2:
@DTW_rPOWER("1.7", NUM2, precision)
• 输入: NUM2 = "8", precision = "5"
• 返回: "69.758"
```


DTW_SUBTRACT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

计算第一个参数的值减去第二个参数的值并返回结果。

格式

```
@DTW_SUBTRACT(number1, number2, precision, result)
@DTW_SUBTRACT(number1, number2, result)
@DTW_rSUBTRACT(number1, number2, precision)
@DTW_rSUBTRACT(number1, number2)
```

值

表 47. DTW_SUBTRACT 参数

数据类型	参数	用法	说明
浮点数	<i>number1</i>	IN	一个表示数值的变量或字符串。
浮点数	<i>number2</i>	IN	一个表示数值的变量或字符串。
整数	<i>precision</i>	IN	一个表示正整数的变量或字符串，此整数指定了结果的精确度。缺省值是 9。
浮点数	<i>result</i>	OUT	包含 <i>number1</i> 和 <i>number2</i> 之差的变量。

例

例 1:

```
@DTW_SUBTRACT(NUM1, NUM2, comp)
%IF(comp > "0")
<P>$(NUM1) is larger than $(NUM2).
%ENDIF
```

- 输入: NUM2 = "2.07"
- 返回: "-0.77"

此例子说明了一种比较两个数值(Net.Data 中的字符串)大小的方法。

例 2:

```
@DTW_SUBTRACT(NUM1, NUM2, result)
• 输入: NUM1 = "1.3, NUM2 = "1.07"
• 返回: result = "0.23"
```

例 3:

```
@DTW_rSUBTRACT("1.3", NUM2)
• 输入: NUM2 = "2.07"
• 返回: "-0.77"
```

字符串函数

下列函数是 Net.Data 支持的标准字符串函数集:

- 第143页的『DTW_ASSIGN』
- 第144页的『DTW_CONCAT』
- 第145页的『DTW_DELSTR』
- 第146页的『DTW_INSERT』
- 第148页的『DTW_LASTPOS』
- 第149页的『DTW_LENGTH』
- 第150页的『DTW_LOWERCASE』
- 第151页的『DTW_POS』
- 第152页的『DTW_REVERSE』
- 第153页的『DTW_STRIP』
- 第154页的『DTW_SUBSTR』
- 第155页的『DTW_TRANSLATE』
- 第157页的『DTW_UPPERCASE』

MBCS 支持 OS/390、OS/2、Windows NT 和 UNIX: 可以用 DTW_MBMODE 配置值指定多字节字符集(MBCS)对单词和字符串的支持。在 Net.Data 初始化文件中指定这个值; 缺省是不支持的。可以通过在 Net.Data 宏文件中设置 DTW_MBMODE 变量来覆盖初始化文件中的值。参阅 *Net.Data 管理与程序设计指南* 中的配置变量章节和 第81页的『DTW_MBMODE』, 获取更多的信息。

MBCS 支持 OS/400: 自动提供 DBCS 支持, 且不要求这个变量。

DTW_ASSIGN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将一个输入变量的值赋给输出变量。同样可以使用这个函数在宏中更改一个变量。

格式

```
@DTW_ASSIGN(stringOut, stringIn)
```

值

表 48. DTW_ASSIGN 参数

数据类型	参数	用法	说明
字符串	<i>stringOut</i>	OUT	包含和 <i>stringIn</i> 完全相同的文字的变量。
字符串	<i>stringIn</i>	IN	一个变量或文字。

例

例 1:

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

- 将 RC 设置为 "0"。

例 2:

```
@DTW_ASSIGN(string1, string2)
```

- 把 *string1* 设置成 *string2* 的值。

DTW_CONCAT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

连接两个字符串。

格式

@DTW_CONCAT(stringIn1, stringIn2, stringOut)

@DTW_rCONCAT(stringIn1, stringIn2)

值

表 49. DTW_CONCAT 参数

数据类型	参数	用法	说明
字符串	<i>stringIn1</i>	IN	一个变量或文字串。
字符串	<i>stringIn2</i>	IN	一个变量或文字串。
字符串	<i>stringOut</i>	OUT	包含字符串 ' <i>stringIn1stringIn2</i> ' 的变量, 其中 <i>string1</i> 后连接 <i>string2</i> 。

例

例 1:

```
@DTW_CONCAT("This", " is a test.", result)
```

- 返回: result = "This is a test."

例 2:

```
@DTW_CONCAT(string1, "1-2-3", result)
```

- 输入: string1 = "Testing "
- 返回: result = "Testing 1-2-3"

例 3:

```
@DTW_rCONCAT("This", " is a test.")
```

- 返回: "This is a test."

DTW_DELSTR

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从指定的字符串删除从第 *n* 个字符开始，长度为 *length* 的一个子串。

格式

```
@DTW_DELSTR(stringIn, n, length, stringOut)
@DTW_DELSTR(stringIn, n, stringOut)
@DTW_rDELSTR(stringIn, n, length)
@DTW_rDELSTR(stringIn, n)
```

值

表 50. DTW_DELSTR 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	要删除的子串所开始的字符位置。如果 <i>n</i> 的值大于 <i>stringIn</i> 的长度，则将 <i>stringOut</i> 设置成 <i>stringIn</i> 的值。
整数	<i>length</i>	IN	要删除的子串的长度。缺省是删除到 <i>stringIn</i> 结尾为止的所有字符。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 的修改格式的变量。

例

```
例 1:
@DTW_DELSTR("abcde", "3", "2", result)
• 返回: result = "abe"

例 2:
@DTW_rDELSTR("abcde", "4", "1")
• 返回: "abce"
```

DTW_INSERT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将一个字符串插入到另一个字符串的第 *n* 个字符后。

格式

```
@DTW_INSERT(stringIn1, stringIn2, n, length, pad, stringOut)
@DTW_INSERT(stringIn1, stringIn2, n, length, stringOut)
@DTW_INSERT(stringIn1, stringIn2, n, stringOut)
@DTW_INSERT(stringIn1, stringIn2, stringOut)
@DTW_rINSERT(stringIn1, stringIn2, n, length, pad)
@DTW_rINSERT(stringIn1, stringIn2, n, length)
@DTW_rINSERT(stringIn1, stringIn2, n)
@DTW_rINSERT(stringIn1, stringIn2)
```

值

表 51. DTW_INSERT 参数

数据类型	参数	用法	说明
字符串	<i>stringIn1</i>	IN	要插入到 <i>stringIn2</i> 中的变量或者文字串。
字符串	<i>stringIn2</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	<i>stringIn2</i> 中的字符位置，在其后插入 <i>stringIn1</i> 。如果 <i>n</i> 大于 <i>stringIn2</i> 的长度，则在其中填满填充字符(<i>pad</i>)，直至有足够个数的字符为止。缺省是插入在 <i>stringIn2</i> 的开始位置。
整数	<i>length</i>	IN	要插入的 <i>stringIn1</i> 的字符数。如果此参数的值大于 <i>stringIn1</i> 的长度，则用填充字符 <i>pad</i> 来填充。缺省是 <i>stringIn1</i> 的长度。
整数	<i>pad</i>	IN	填充字符，如对 <i>n</i> 和 <i>length</i> 的描述。缺省填充字符是一个空格。
字符串	<i>stringOut</i>	OUT	一个包含由于插入部分或全部 <i>stringIn1</i> 而修改过的 <i>stringIn2</i> 的变量。

例

```
例 1:
@DTW_INSERT("123", "abc", result)
• 返回: result = "123abc"

例 2:
@DTW_INSERT("123", "abc", "5", result)
• 返回: result = "abc 123 "

例 3:
```

```
@DTW_INSERT("123", "abc", "5", "6", result)
```

- 返回: result = "abc 123 "

例 4:

```
@DTW_INSERT("123", "abc", "5", "6", "/", result)
```

- 返回: result = "abc//123///"

例 5:

```
@DTW_rINSERT("123", "abc", "5", "6", "+")
```

- 返回: "abc++123++"

DTW_LASTPOS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个字符串在另一个字符串中最后一次出现的位置，从第 *n* 个字符开始往字符串开始方向查找(从右到左)。

格式

```
@DTW_LASTPOS(stringIn1, stringIn2, n, position)
@DTW_LASTPOS(stringIn1, stringIn2, position)
@DTW_rLASTPOS(stringIn1, stringIn2, n)
@DTW_rLASTPOS(stringIn1, stringIn2)
```

值

表 52. DTW_LASTPOS 参数

数据类型	参数	用法	说明
字符串	<i>stringIn1</i>	IN	在 <i>stringIn2</i> 中搜索的变量或者文字串。
字符串	<i>stringIn2</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	在 <i>stringIn2</i> 中搜索 <i>stringIn1</i> 的开始字符位置。缺省是从最后一个字符开始搜索并向后扫描(从右到左)。
整数	<i>position</i>	OUT	<i>stringIn1</i> 在 <i>stringIn2</i> 中最后一次出现的位置。如果找不到这种串，就返回 0。

例

```
例 1:
@DTW_LASTPOS(" ", "abc def ghi", result)
• 返回: result = "8"

例 2:
@DTW_LASTPOS(" ", "abc def ghi", "10", result)
• 返回: result = "8"

例 3:
@DTW_rLASTPOS(" ", "abc def ghi", "7")
• 返回: "4"
```


DTW_LENGTH

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回字符串的长度。

格式

@DTW_LENGTH(stringIn, length)

@DTW_rLENGTH(stringIn)

值

表 53. DTW_LENGTH 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>length</i>	OUT	包含 <i>stringIn</i> 中字符个数的符号。

例

例 1:

@DTW_LENGTH("abcdefgh", result)

- 返回: result = "8"

例 2:

@DTW_rLENGTH("")

- 返回: "0"

DTW_LOWERCASE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个所有字符全为小写的字符串。

格式

```
@DTW_LOWERCASE(stringIn, stringOut)
@DTW_rLOWERCASE(stringIn)
@DTW_mLOWERCASE(stringMult1, stringMult2, ..., stringMultn)
```

值

表 54. DTW_LOWERCASE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串，其中的字符可以是大写或小写。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> (其中所有字符都是小写的)的变量。
字符串	<i>stringMult</i>	INOUT	<ul style="list-style-type: none">输入： 一个包含字符串的变量。输出： 一个包含输入字符串的变量，其中的所有字符都已转换为小写。

例

```
例 1:
@DTW_LOWERCASE("This", stringOut)
• 返回: stringOut = "this"

例 2:
@DTW_rLOWERCASE(string1)
• 输入: string1 = "Hello"
• 返回: "hello"

例 3:
@DTW_mLOWERCASE(string1, string2, string3)
• 输入: string1 = "THIS", string2 = "IS", string3 = "LOWERCASE"
• 返回: string1 = "this", string2 = "is", string3 = "lowercase"
```

DTW_POS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

使用向前搜索模式，返回一个字符串在另一个字符串中第一次出现的位置。

格式

```
@DTW_POS(stringIn1, stringIn2, n, nOut)
@DTW_POS(stringIn1, stringIn2, nOut)
@DTW_rPOS(stringIn1, stringIn2, n)
@DTW_rPOS(stringIn1, stringIn2)
```

值

表 55. DTW_POS 参数

数据类型	参数	用法	说明
字符串	<i>stringIn1</i>	IN	一个要搜索的变量或文字串。
字符串	<i>stringIn2</i>	IN	一个被搜索(在其中搜索另一个字符串)的变量或文字串。
整数	<i>n</i>	IN	<i>stringIn2</i> 中开始搜索的字符位置。缺省值是从 <i>stringIn2</i> 的第一个字符开始搜索。
整数	<i>nOut</i>	OUT	包含 <i>stringIn1</i> 在 <i>stringIn2</i> 中第一次出现的位置值的一个变量。如果找不到这种串，就返回 0。

例

```
例 1:
@DTW_POS("day", "Saturday", result)
• 返回: result = "6"

例 2:
@DTW_POS("a", "Saturday", "3", result)
• 返回: result = "7"

例 3:
@DTW_rPOS(" ", "abc def ghi", "5")
• 返回: "8"
```

DTW_REVERSE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将输入字符串反转过来。

格式

@DTW_REVERSE(stringIn, stringOut)

@DTW_rREVERSE(stringIn)

值

表 56. DTW_REVERSE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	要进行反转的变量或文字串。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 的反转格式的变量。

例

例 1:

```
@DTW_REVERSE("This is it.", result)
```

- 返回: result = ".ti si sihT"

例 2:

```
@DTW_rREVERSE(string1)
```

- 输入: string1 = "reversed"
- 返回: "desrever"

DTW_STRIP

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从输入字符串删除前导空格或尾随空格(或同时删除)。

格式

```
@DTW_STRIP(stringIn, option, stringOut)
@DTW_STRIP(stringIn, stringOut)
@DTW_rSTRIP(stringIn, option)
@DTW_rSTRIP(stringIn)
```

值

表 57. DTW_STRIP 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
字符串	<i>option</i>	IN	指定从 <i>stringIn</i> 删除哪些空格。 缺省值是 B。 B 或 b - 同时删除前导和尾随空格 L 或 l - 只删除前导空格 T 或 t - 只删除尾随空格
字符串	<i>stringOut</i>	OUT	一个包含按照选项所指去除了空格的 <i>stringIn</i> 的变量。

例

```
例 1:
@DTW_STRIP(" day ", result)
• 返回: result = "day"

例 2:
@DTW_STRIP(" day ", "T", result)
• 返回: result = "day"

例 3:
@DTW_rSTRIP(" a day ", "L")
• 返回: "a day "
```

DTW_SUBSTR

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回输入字符串的一个子串，其中可能带填充字符。

格式

```
@DTW_SUBSTR(stringIn, n, length, pad, stringOut)
@DTW_SUBSTR(stringIn, n, length, stringOut)
@DTW_SUBSTR(stringIn, n, stringOut)
@DTW_rSUBSTR(stringIn, n, length, pad)
@DTW_rSUBSTR(stringIn, n, length)
@DTW_rSUBSTR(stringIn, n)
```

值

表 58. DTW_SUBSTR 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个被搜索(在其中搜索另一个字符串)的变量或文字串。
整数	<i>n</i>	IN	子字符串第一个字符的位置。缺省是从 <i>stringIn</i> 的开始位置开始
整数	<i>length</i>	IN	子串的字符个数。缺省值是字符串的剩余部分。
字符串	<i>pad</i>	IN	如果 <i>n</i> 大于 <i>stringIn</i> 的长度或者如果 <i>length</i> 比 <i>stringIn</i> 长的时候所使用的填充字符。缺省值是一个空格。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 的子字符串的变量。

例

```
例 1:
@DTW_SUBSTR("abc", "2", result)
• 返回: result = "bc "
```

```
例 2:
@DTW_SUBSTR("abc", "2", "4", result)
• 返回: result = "bc "
```

```
例 3:
@DTW_SUBSTR("abc", "2", "4", ".", result )
• 返回: result = "bc.."
```

```
例 4:
@DTW_rSUBSTR("abc", "2", "6", ".")
• 返回: "bc...."
```

DTW_TRANSLATE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

使用输入和输出转换表 *tableI* 和 *tableO* 来转换输入字符串中的字符。如果 *tableI*、*tableO* 和 *default* 字符不在参数表中，则 *stringIn* 参数转换成大写。如果列表中存在 *tableI* 和 *tableO*，则在 *tableI* 中搜索输入字符串中的每个字符，并在 *tableO* 中转换成相应的字符。如果 *tableI* 中的字符和 *tableO* 中的字符无对应关系，则使用 *default* 字符。

格式

```
@DTW_TRANSLATE(stringIn, tableO, tableI, default, stringOut)
@DTW_TRANSLATE(stringIn, tableO, tableI, stringOut)
@DTW_TRANSLATE(stringIn, tableO, stringOut)
@DTW_TRANSLATE(stringIn, stringOut)
@DTW_rTRANSLATE(stringIn, tableO, tableI, default)
@DTW_rTRANSLATE(stringIn, tableO, tableI)
@DTW_rTRANSLATE(stringIn, tableO)
@DTW_rTRANSLATE(stringIn)
```

值

表 59. DTW_TRANSLATE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
字符串	<i>tableO</i>	IN	一个作为转换表使用的变量或文字串。使用空("")指定 <i>tableI</i> 或 <i>default</i> ; 否则这个参数是可选的。
字符串	<i>tableI</i>	IN	在 <i>stringIn</i> 中搜索的变量或者文字串。使用空("")指定 <i>default</i> ; 否则这个参数是可选的。
字符串	<i>default</i>	IN	要使用的缺省字符。缺省值是一个空格。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 的转换结果的变量。

例

例 1:

```
@DTW_TRANSLATE("abbc", result)
• 返回: result = "ABBC"
```

例 2:

```
@DTW_TRANSLATE("abbc", "R", "bc", result)
• 返回: result = "aRR "
```

例 3:

```
@DTW_rTRANSLATE("abcdef", "12", "abcd", ".")
```

- 返回: "12..ef"

例 4:

```
@DTW_rTRANSLATE("abbc", "", "", "")
```

- 返回: "abbc"

DTW_UPPERCASE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个大写的字符串。

格式

```
@DTW_UPPERCASE(stringIn, stringOut)
@DTW_rUPPERCASE(stringIn)
@DTW_mUPPERCASE(stringMult1, stringMult2, ..., stringMultn)
```

值

表 60. DTW_UPPERCASE 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串，其中的字符可以是大写或小写。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> (其中所有字符都是大写的)的变量。
字符串	<i>stringMult</i>	INOUT	<ul style="list-style-type: none">输入： 一个包含字符串的变量。输出： 一个包含输入字符串的变量，其中的所有字符都已转换为大写。

例

```
例 1:
@DTW_UPPERCASE("Test", result)
• 返回: result = "TEST"

例 2:
@DTW_rUPPERCASE(string1)
• 输入: string1 = "Web pages"
• 返回: "WEB PAGES"

例 3:
@DTW_mUPPERCASE(string1, string2, string3)
• 输入: string1 = "This", string2 = "is", string3 = "uppercase"
• 返回: string1 = "THIS", string2 = "IS", string3 = "UPPERCASE"
```

字处理函数

这些函数是字符串函数的补充，用于修改单词或设置单词。Net.Data 将单词(字)解释为由空格分隔的字符串，或两边都有空格的字符串。例子如下：

字符串值	单词个数
one two three	3
one , two , three	5
Part 2: Internet Sales Grow	5

MBCS 支持 OS/390、OS/2、Windows NT 和 UNIX: 可以用 DTW_ MBMODE 配置值指定多字节字符集(MBCS)对单词和字符串的支持。在 Net.Data 初始化文件中指定这个值；缺省是不支持的。可以通过在 Net.Data 宏文件中设置 DTW_ MBMODE 变量来覆盖初始化文件中的值。参阅 *Net.Data 管理与程序设计指南* 中的配置变量章节和 第 81 页的『DTW_ MBMODE』，获取更多的信息。

MBCS 支持 OS/400: 自动提供 DBCS 支持，且不要求这个变量。

下列是 Net.Data 支持的字处理函数：

- 第159页的『DTW_ DELWORD』
- 第160页的『DTW_ SUBWORD』
- 第161页的『DTW_ WORD』
- 第162页的『DTW_ WORDINDEX』
- 第163页的『DTW_ WORDLENGTH』
- 第164页的『DTW_ WORDPOS』
- 第165页的『DTW_ WORDS』

DTW_DELWORD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回输入字符串的一个子串。从第 *n* 个单词开始，删除由 *length* 指定个数的单词。

格式

```
@DTW_DELWORD(stringIn, n, length, stringOut)
@DTW_DELWORD(stringIn, n, stringOut)
@DTW_rDELWORD(stringIn, n, length)
@DTW_rDELWORD(stringIn, n)
```

值

表 61. DTW_DELWORD 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	要删除的第一个单词的单词位置。
整数	<i>length</i>	IN	要删除的单词个数。缺省情况是删除从 <i>n</i> 到 <i>stringIn</i> 结尾为止的所有单词。这是可选的参数。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 的修改格式的变量。

例

```
例 1:
@DTW_DELWORD("Now is the time", "5", result)
• 返回: result = "Now is the time"

例 2:
@DTW_DELWORD("Now is the time", "2", result)
• 返回: result = "Now"

例 3:
@DTW_DELWORD("Now is the time", "2", "2", result)
• 返回: result = "Now time"

例 4:
@DTW_rDELWORD("Now is the time.", "3")
• 返回: "Now is"
```

DTW_SUBWORD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回输入字符串的一个子串。此子串从第 *n* 个单词开始，后面跟由 *length* 指定个数的单词。

格式

```
@DTW_SUBWORD(stringIn, n, length, stringOut)
@DTW_SUBWORD(stringIn, n, stringOut)
@DTW_rSUBWORD(stringIn, n, length)
@DTW_rSUBWORD(stringIn, n)
```

值

表 62. DTW_SUBWORD 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	子串的第一个单词的单词位置。如果此值大于 <i>stringIn</i> 中的单词个数，则返回空。
整数	<i>length</i>	IN	子串中的单词个数。如果这个值大于从第 <i>n</i> 个单词到 <i>stringIn</i> 结尾为止的单词总个数，则返回到 <i>stringIn</i> 结尾为止的所有单词。缺省情况是返回从第 <i>n</i> 个单词到 <i>stringIn</i> 结尾为止的所有单词。
字符串	<i>stringOut</i>	OUT	包含由 <i>n</i> 和 <i>length</i> 指定的 <i>stringIn</i> 的子字符串的变量。

例

```
例 1:
@DTW_SUBWORD("Now is the time", "5", result)
• 返回: result = ""

例 2:
@DTW_SUBWORD("Now is the time", "2", result)
• 返回: result = "is the time"

例 3:
@DTW_SUBWORD("Now is the time", "2", "2", result)
• 返回: result = "is the"

例 4:
@DTW_rSUBWORD("Now is the time", "3")
• 返回: "the time"
```

DTW_WORD

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从输入字符串的指定位置返回一个单词。

格式

```
@DTW_WORD(stringIn, n, stringOut)
@DTW_rWORD(stringIn, n)
```

值

表 63. DTW_WORD 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	要返回的单词的单词位置。如果此值大于 <i>stringIn</i> 中的单词个数，则返回空。
字符串	<i>stringOut</i>	OUT	包含第 <i>n</i> 个单词位置的单词的一个变量。

例

```
例 1:
@DTW_WORD("Now is the time", "3", result)
• 返回: result = "the"

例 2:
@DTW_WORD("Now is the time", "5", result)
• 返回: result = ""

例 3:
@DTW_rWORD("Now is the time", "4")
• 返回: "time"
```

DTW_WORDINDEX

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回输入字符串的第 n 个单词的第一个字符的字符位置。

格式

@DTW_WORDINDEX(stringIn, n, stringOut)

@DTW_rWORDINDEX(stringIn, n)

值

表 64. DTW_WORDINDEX 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	要索引的单词的单词位置。如果此值大于输入字符串中的单词个数，则返回 0。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 中第 n 个单词的字符位置的变量。

例

例 1:

```
@DTW_WORDINDEX("Now is the time", "3", result)
```

- 返回: result = "8"

例 2:

```
@DTW_WORDINDEX("Now is the time", "6", result)
```

- 返回: result = "0"

例 3:

```
@DTW_rWORDINDEX("Now is the time", "2")
```

- 返回: "5"

DTW_WORDLENGTH

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回输入字符串的第 n 个单词的长度。

格式

@DTW_WORDLENGTH(stringIn, n, stringOut)

@DTW_rWORDLENGTH(stringIn, n)

值

表 65. DTW_WORDLENGTH 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
整数	<i>n</i>	IN	要查询长度的那个单词的单词位置。如果此值大于输入字符串中的单词个数，则返回 0。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 中第 n 个单词的长度值的一个变量。

例

例 1:

@DTW_WORDLENGTH("Now is the time", "1", result)

- 返回: result = "3"

例 2:

@DTW_rWORDLENGTH("Now is the time", "6")

- 返回: "0"

DTW_WORDPOS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个字符串在另一个字符串内第一次出现的单词位置。在比较时将多个空格看作一个空格。这种比较是区别大小写的。

格式

```
@DTW_WORDPOS(stringIn1, stringIn2, n, stringOut)
@DTW_WORDPOS(stringIn1, stringIn2, stringOut)
@DTW_rWORDPOS(stringIn1, stringIn2, n)
@DTW_rWORDPOS(stringIn1, stringIn2)
```

值

表 66. DTW_WORDPOS 参数

数据类型	参数	用法	说明
字符串	<i>stringIn1</i>	IN	一个变量或文字串。
字符串	<i>stringIn2</i>	IN	一个被搜索(在其中搜索另一个字符串)的变量或文字串。
整数	<i>n</i>	IN	<i>stringIn2</i> 中开始搜索的单词位置。如果此值大于 <i>stringIn2</i> 中的单词个数, 则返回 0。缺省情况是从 <i>stringIn2</i> 的开头开始搜索。
字符串	<i>stringOut</i>	OUT	<i>stringIn1</i> 在 <i>stringIn2</i> 中的单词。

例

```
例 1:
@DTW_WORDPOS("the", "Now is the time", result)
• 返回: result = "3"

例 2:
@DTW_WORDPOS("The", "Now is the time", result)
• 返回: result = "0"

例 3:
@DTW_WORDPOS("The", "Now is the time", "5", result)
• 返回: result = "0"

例 4:
@DTW_WORDPOS("is the", "Now is the time", result)
• 返回: result = "2"

例 5:
@DTW_rWORDPOS("be", "To be or not to be", "3")
• 返回: "6"
```


DTW_WORDS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个字符串中的单词个数。

格式

@DTW_WORDS(stringIn, stringOut)

@DTW_rWORDS(stringIn)

值

表 67. DTW_WORDS 参数

数据类型	参数	用法	说明
字符串	<i>stringIn</i>	IN	一个变量或文字串。
字符串	<i>stringOut</i>	OUT	包含 <i>stringIn</i> 中单词个数的变量。

例

例 1:

@DTW_WORDS("Now is the time", result)

- 返回:

result = "4"

例 2:

@DTW_rWORDS(" ")

- 返回: "0"

表格函数

这些函数用于简化 Net.Data 表格的处理，它比您使用 REXX、C 或 Perl 编写的函数更为有效。

- 第167页的『DTW_TB_APPENDROW』
- 第168页的『DTW_TB_COLS』
- 第169页的『DTW_TB_DELETEROW』
- 第170页的『DTW_TB_DLIST』
- 第172页的『DTW_TB_DUMP』
- 第173页的『DTW_TB_DUMPV』
- 第174页的『DTW_TB_GETN』
- 第175页的『DTW_TB_GETV』
- 第176页的『DTW_TB_HTMLLENCODE』
- 第177页的『DTW_TB_INPUT_CHECKBOX』
- 第178页的『DTW_TB_INPUT_RADIO』
- 第179页的『DTW_TB_INPUT_TEXT』
- 第180页的『DTW_TB_INSERTCOL』
- 第181页的『DTW_TB_INSERTROW』
- 第182页的『DTW_TB_LIST』
- 第183页的『DTW_TB_MAXROWS』
- 第184页的『DTW_TB_QUERYCOLNONJ』
- 第185页的『DTW_TB_ROWS』
- 第186页的『DTW_TB_SELECT』
- 第187页的『DTW_TB_SETCOLS』
- 第188页的『DTW_TB_SETN』
- 第189页的『DTW_TB_SETV』
- 第190页的『DTW_TB_TABLE』
- 第192页的『DTW_TB_TEXTAREA』

DTW_TB_APPENDROW

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

在表格结尾添加一行或多行。向表格中追加行之后，使用 DTW_TB_SETV() 函数来指定新行的值。

调用 DTW_TB_APPENDROW() 之前必须先设置表中的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

如果表中允许的总行数有限制，并且要追加的行数将使得总行数超过限制，则将向调用者返回错误。

格式

```
@DTW_TB_APPENDROW(table, rows)
```

值

表 68. DTW_TB_APPENDROW 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	针对追加的行的宏表格变量。
整数	<i>rows</i>	IN	要追加到 <i>table</i> 的行数。

例

```
例 1: 向表中追加十行
%DEFINE myTable = %TABLE
@DTW_TB_APPENDROW(myTable, "10")
```

DTW_TB_COLS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

返回表格中当前的列数。

格式

```
@DTW_TB_COLS(table, cols)
@DTW_TB_rCOLS(table)
```

值

表 69. DTW_TB_COLS 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	针对返回的列数的宏表格变量。
整数	<i>cols</i>	OUT	包含 <i>table</i> 中列数的变量。

例

例 1: 检索列数，并把值赋给 *cols*

```
%DEFINE myTable = %TABLE
%DEFINE cols = ""
...
@FillTable()
...
@DTW_TB_COLS(myTable, cols)
```

例 2: 检索并显示表格中当前列数的值

```
%DEFINE myTable = %TABLE
...
@FillTable()
...
<P>My table contains @DTW_TB_rCOLS(myTable) columns.
```

DTW_TB_DELETEROW

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

删除以 start_row 中指定的行开头的一行或多行。

调用 DTW_TB_DELETEROW() 之前必须先设置表中的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

@DTW_TB_DELETEROW(table, start_row, rows)

值

表 70. DTW_TB_DELETEROW 参数

数据类型	参数	用法	说明
表格	table	IN	要从中删除行的宏表格变量。
整数	start_row	IN	table 中要删除的第一行的行号。
整数	rows	IN	要从 table 中删除的行数。

例

例 1: 从表中删除从第 10 行开始的五行

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETEROW(myTable, "10", "5")
```

例 2: 删除表中的所有行

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETEROW(myTable, "1", @DTW_TB_rROWS(myTable))
```

DTW_TB_DLIST

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从宏表格中返回一个 HTML 定义列表。

格式

@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u, image, image_u)

值

表 71. DTW_TB_DLIST 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	一个符号，指定表格变量显示成一张 HTML 定义列表。
整数	<i>term</i>	IN	<i>table</i> 中的列号，它包含 <i>term</i> 名称值 (<DT> 标记后转向的文本)。缺省情况是使用第一列。
整数	<i>def</i>	IN	<i>table</i> 中包含项目定义值 (<DD 标记后面的文本)的列号。缺省情况是使用第二列。
字符串	<i>termstyle</i>	IN	包含针对 <i>term</i> 名称值的一张 HTML 元素列表的变量或者文字串。缺省情况是不使用任何风格标记。
字符串	<i>defstyle</i>	IN	包含针对 <i>term</i> 定义值的一张 HTML 元素列表的变量或者文字串。缺省情况是不使用任何风格标记。
字符串	<i>link</i>	IN	指定针对哪些 HTML 元素生成 HTML 链接。有效值是 DT 和 DD。缺省是不生成 HTML 链接。
整数	<i>link_u</i>	IN	<i>table</i> 中的列号，它包含 HTML 引用的 URL。缺省是不生成 HTML 链接。
字符串	<i>image</i>	IN	指定要为哪些 HTML 元素生成内嵌图象 (inline image)。有效值是 DT 和 DD。缺省是不生成内嵌图象(DT)。
整数	<i>image_u</i>	IN	<i>table</i> 中的列号，它包含内嵌图象的 URL。缺省情况是不生成内嵌图象。

例

例 1: 创建一张定义列表，按照表格数据产生下面所示的 HTML

@DTW_TB_DLIST(Mytable,"3","4","b i","strong","DD","2","DT","1")

结果:

```
<DL>
<DT>
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b><i>image1text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link1.html"><strong>link1text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b><i>image2text</i></b>
```

```
<DD>
<A HREF="http://www.mycompany.com/link2.html"><strong>link2text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b><i>image3text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link3.html"><strong>link3text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT=""><b><i>image4text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link4.html"><strong>link4text</strong></A>
</DT>
</DL>
```

DTW_TB_DUMP

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个宏表格变量的内容。表格中的每一行显示在不同的行中。整个表格要用 <PRE></PRE> 标记括起来。

格式

@DTW_TB_DUMP(table)

值

表 72. DTW_TB_DUMP 参数

数据类型	参数	用法	说明
表格	table	IN	指定要显示的宏表格变量的符号。

例

例 1:

@DTW_TB_DUMP(Mytable)

此例生成的 HTML 看起来象:

```
<PRE>
Name           Department      Position
Jack Smith     Internet Technologies  Software Engineer
Helen Williams Database           Development Manager
Alex Jones     Manufacturing      Industrial Engineer
Tom Baker      Procurement        Sales Rep
</PRE>
```


DTW_TB_DUMPV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

| 返回一个宏表格变量的内容。每个表格值在不同的行中。整个表格要用 <PRE></PRE>
| 标记括起来。

格式

@DTW_TB_DUMPV(table)

值

表 73. DTW_TB_DUMPV 参数

数据类型	参数	用法	说明
表格	table	IN	指定要显示的宏表格变量的符号。

例

例 1:

@DTW_TB_DUMPV(Mytable)

这个例子生成的 HTML 如下所示:

```
<PRE>
http://www.mycompany.com/images/image1.gif
http://www.mycompany.com/link1.html
image1text
link1text
http://www.mycompany.com/images/image2.gif
http://www.mycompany.com/link2.html
image2text
link2text
http://www.mycompany.com/images/image3.gif
http://www.mycompany.com/link3.html
image3text
link3text
http://www.mycompany.com/images/image4.gif
http://www.mycompany.com/link4.html
image4text
link4text
</PRE>
```

DTW_TB_GETN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

检索 *col* 中指定的列号的列标题。

必须在调用 DTW_TB_GETN() 之前设置表格的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

```
@DTW_TB_GETN(table, col, name)
@DTW_TB_rGETN(table, col)
```

值

表 74. DTW_TB_GETN 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	从中返回一个列名称的宏表格变量。
整数	<i>col</i>	IN	要返回的那个列的列号。
字符串	<i>name</i>	OUT	包含 <i>col</i> 中指定的列名称的变量。

例

例 1: 检索第 4 列的列名

```
%DEFINE myTable = %TABLE
%DEFINE name = ""
...
@FillTable()
...
@DTW_TB_GETN(myTable, "4", name)
```

例 2: 检索表格最后一列的列名

```
%DEFINE myTable = %TABLE
...
@FillTable()
...
<P>The column name of the last column is @DTW_TB_rGETN(myTable, @DTW_TB_rCOLS(myTable))
```

DTW_TB_GETV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

检索 *row* 和 *col* 所指定的行和列的表格值。

必须在调用 DTW_TB_GETV() 之前设置表格的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

```
@DTW_TB_GETV(table, row, col, value)
@DTW_TB_rGETV(table, row, col)
```

值

表 75. DTW_TB_GETV 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	宏表格变量，它返回一个表格值。
整数	<i>row</i>	IN	要返回的行号值。
整数	<i>col</i>	IN	要返回的列号值。
字符串	<i>value</i>	OUT	一个变量，包含 <i>row</i> 和 <i>col</i> 所指定的行和列上的值。

例

例 1: 检索第 6 行、第 3 列上的表格值

```
%DEFINE myTable = %TABLE
%DEFINE value = ""
...
@FillTable()
...
@DTW_TB_GETV(myTable, "6", "3", value)
```

例 2: 检索第 1 行、第 1 列上的表格值

```
%DEFINE myTable = %TABLE
...
@FillTable()
...
<P>The table value of row 1, column 1 is @DTW_TB_rGETV(myTable, "1", "1").
```

DTW_TB_HTMLENCODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回带有已编码的 HTML 字符的输入宏表格。

名称	字符	代码
&符号	&	&
双引号	"	"
大于	>	>
小于	<	<

格式

@DTW_TB_HTMLENCODE(table, collist)

值

表 76. DTW_TB_HTMLENCODE 参数

数据类型	参数	用法	说明
表格	<i>table</i>	INOUT	要修改的宏表格变量。
字符串	<i>collist</i>	IN	<i>table</i> 中要编码的列号。缺省情况是对所有列进行编码。

例

例 1:

@DTW_TB_HTMLENCODE(Mytable, "3 4")

在指定表格中第 3 和第 4 列中的特殊字符被它们的编码格式所替换。

DTW_TB_INPUT_CHECKBOX

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从一个宏表格变量返回一个或多个 HTML 校验框输入标记。

格式

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol, valuecol, rows, checkedrows)

值

表 77. DTW_TB_INPUT_CHECKBOX 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要作为校验框输入标记来显示的宏表格变量。
字符串	<i>prompt</i>	IN	<i>table</i> 中的列号，或是一个包含要在校验框旁边显示的文本的字符串。此参数是必需的，但可以是空("")值。如果 <i>prompt</i> 为空，则使用的值是定义给 <i>namecol</i> 的值。
字符串	<i>namecol</i>	IN	<i>table</i> 中的列号，或一个包含输入字段名的字符串。
字符串	<i>valuecol</i>	IN	<i>table</i> 中的列号，或是一个包含输入字段值的字符串。缺省值是 1。
整数	<i>rows</i>	IN	从中生成输入字段的 <i>table</i> 中的一张行的列表。缺省情况是使用所有的行。
整数	<i>checkedrows</i>	IN	一张行列表，指定选择 <i>table</i> 中的哪些 <i>rows</i> 。缺省情况是不选择字段。

例

例 1: 生成三个校验框输入标记的 HTML

@DTW_TB_INPUT_CHECKBOX(Mytable,"3","4","", "2 3 4", "1 3 4")

结果:

```
<INPUT TYPE="CHECKBOX" NAME="link2text" VALUE="1">image2text<BR>
<INPUT TYPE="CHECKBOX" NAME="link3text" VALUE="1" CHECKED>image3text<BR>
<INPUT TYPE="CHECKBOX" NAME="link4text" VALUE="1" CHECKED>image4text<BR>
```

DTW_TB_INPUT_RADIO

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从一个宏表格变量返回一个或多个 HTML 选项按钮输入标记。

格式

@DTW_TB_INPUT_RADIO(table, prompt, namecol, valuecol, rows, checkedrows)

值

表 78. DTW_TB_INPUT_RADIO 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	显示成选项按钮输入标记的宏表格变量。
字符串	<i>prompt</i>	IN	<i>table</i> 中的列号，或是一个包含要在选项按钮旁边显示的文本的字符串。此参数是必需的，但可以包含空("")值。如果 <i>prompt</i> 为空，则使用 <i>valuecol</i> 的值。
字符串	<i>namecol</i>	IN	<i>table</i> 中的列号，或一个包含输入字段名的字符串。
字符串	<i>valuecol</i>	IN	<i>table</i> 中的列号，或是一个包含输入字段值的字符串。
字符串	<i>rows</i>	IN	从中生成输入字段的 <i>table</i> 中的一张行的列表。缺省情况是使用所有的行。
整数	<i>checkedrows</i>	IN	<i>table</i> 中的行号，显示所选择的相应选项按钮。只允许指定一个值。

例

例 1: 生成三个选项按钮输入标记的 HTML

@DTW_TB_INPUT_RADIO(Mytable,"3","Radio4","4","2 3 4","4")

结果:

```
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link2text">image2text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link3text">image3text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link4text" CHECKED>image4text<BR>
```

DTW_TB_INPUT_TEXT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

针对 Net.Data 表格中指定的行返回 HTML <INPUT> 标记。例如， Net.Data 可以生成具有 VALUE、SIZE 和 MAXLENGTH 属性的 HTML <INPUT> 标记。

<INPUT TYPE="TEXT" NAME="someName" VALUE="someValue" SIZE="20" MAXLENGTH="40">

格式

@DTW_TB_INPUT_TEXT(table, prompt, namecol, valuecol, size, maxlen, rows)

值

表 79. DTW_TB_INPUT_TEXT 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	显示成文本输入标记的宏表格变量。
字符串	<i>prompt</i>	IN	<i>table</i> 中的列号，或是一个包含要在输入字段旁边显示的文本的字符串。如果 <i>prompt</i> 为空，则不显示文本。
字符串	<i>namecol</i>	IN	<i>table</i> 中的列号，或一个包含输入字段名的字符串。
字符串	<i>valuecol</i>	IN	<i>table</i> 中的列号，或一个包含缺省输入字段值的字符串，这是给 INPUT 标记上的 VALUE 属性指定的。缺省情况是不生成 VALUE 属性值。
整数	<i>size</i>	IN	输入字段的字符个数，这是为 INPUT 标记上的 SIZE 属性指定的。缺省值是最长的缺省输入值的长度，如果不存在缺省输入，则为 10。
整数	<i>maxlen</i>	IN	输入字符串的最大长度，这是为 INPUT 标记上的 MAXLENGTH 属性指定的。缺省是不生成MAXLENGTH 属性值。
整数	<i>rows</i>	IN	从中生成输入字段的 <i>table</i> 中的一张行的列表。缺省情况是使用所有的行。

例

例 1: 返回三个 HTML <INPUT> 标记

@DTW_TB_INPUT_TEXT(Mytable,"3","3","4","35","40","1 2 3")

结果:

<P>image1text
<INPUT TYPE="TEXT" NAME="image1text" VALUE="link1text" SIZE="35" MAXLENGTH="40">
<P>image2text
<INPUT TYPE="TEXT" NAME="image2text" VALUE="link2text" SIZE="35" MAXLENGTH="40">
<P>image3text
<INPUT TYPE="TEXT" NAME="image3text" VALUE="link3text" SIZE="35" MAXLENGTH="40">

DTW_TB_INSERTCOL

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

在 *after_col* 中指定的列后插入一行或多行。

格式

@DTW_TB_INSERTCOL(table, after_col, cols)

值

表 80. DTW_TB_INSERTCOL 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要在其中插入列的宏表格变量。
整数	<i>after_col</i>	IN	新列插入后的列号。要在表的开头插入列，请指定 0。
整数	<i>cols</i>	IN	要插入 <i>table</i> 的列数。

例

例 1: 在表尾插入五列

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTCOL(myTable, @DTW_TB_rCOLS(myTable), "5")
```

例 2: 在表首插入一列

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTCOL(myTable, "0", "1")
```


DTW_TB_INSERTROW

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

在 *after_row* 中指定的行之后插入一行或多行。

调用 DTW_TB_INSERTROW() 之前必须先设置表中的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

@DTW_TB_INSERTROW(table, after_row, rows)

值

表 81. DTW_TB_INSERTROW 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要在其中插入行的宏表格变量。
整数	<i>after_row</i>	IN	新行插入之后的行号。要在表格的开头插入行，请指定 0。
整数	<i>rows</i>	IN	要插入 <i>table</i> 的行数。

例

例 1: 在表的第五行后插入一行

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTROW(myTable, "5", "1")
```

例 2: 在表的开始处插入三行

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTROW(myTable, "0", "3")
```

DTW_TB_LIST

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个 HTML 列表。

格式

@DTW_TB_LIST(table, listtype, listitem, itemstyle, link_u, image_u)

值

表 82. DTW_TB_LIST 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	指定显示成 HTML 列表的宏表格变量的符号。
字符串	<i>listtype</i>	IN	要生成的列表类型。可接受的列表类型包括: DIR MENU OL UL
整数	<i>listitem</i>	IN	<i>table</i> 中的列号，它包含列表值(标记后所转向的文本)。缺省情况是使用第一列。
字符串	<i>itemstyle</i>	IN	一个变量或文字串，其中包含用于项目名称值的 HTML 元素列表。缺省情况是不使用任何风格标记。
整数	<i>link_u</i>	IN	<i>table</i> 中的列号，它包含 HTML 链接的 URL。如果没有指定这个值，则不生成 HTML 链接。
整数	<i>image_u</i>	IN	<i>table</i> 中的列号，它包含内嵌图象的 URL。如果没有指定这个值，则不生成嵌入图象。

例

例 1: 生成一张顺序列表的 HTML 标记

@DTW_TB_LIST(Mytable,"OL","4","TT U","2","1")

结果:

```
<TT><U>
<OL>
<LI><A HREF="http://www.mycompany.com/link1.html">
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT="">link1text</A>
<LI><A HREF="http://www.mycompany.com/link2.html">
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT="">link2text</A>
<LI><A HREF="http://www.mycompany.com/link3.html">
<IMG SRC="http://www.mycompany.com/images/image3.gif" ALT="">link3text</A>
<LI><A HREF="http://www.mycompany.com/link4.html">
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT="">link4txt</A>
</OL>
</U></TT>
```

DTW_TB_MAXROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

返回表中允许的最大行数。

格式

```
@DTW_TB_MAXROWS(table, maxrows)
@DTW_TB_rMAXROWS(table)
```

值

表 83. DTW_TB_MAXROWS 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	返回最大行数的宏表格变量。
整数	<i>maxrows</i>	OUT	一个变量，包含 <i>table</i> 中所允许的最大行数。如果表定义时没有行数的限制，则返回 0。

例

例 1: 检索表中允许的最大行数并返回值

```
%DEFINE myTable = %TABLE
%DEFINE maxrows = ""

@DTW_TB_MAXROWS(myTable, maxrows)
%IF (maxrows != "0")
    允许的最大行数是 $(maxrows)。
%ELSE
    允许的行数没有限制。
%ENDIF
```

例 2: 检索并显示最大行数的值

```
%DEFINE myTable = %TABLE

%IF (@DTW_TB_rMAXROWS(myTable) != "0")
    允许的最大行数是 @DTW_TB_rMAXROWS(myTable)。
%ELSE
    允许的行数没有限制。
%ENDIF
```

DTW_TB_QUERYCOLNONJ

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

返回与列标题相关联的列号。 如果表格中不存在列标题，则返回 0。

格式

```
@DTW_TB_QUERYCOLNONJ(table, name, col)
@DTW_TB_rQUERYCOLNONJ(table, name)
```

值

表 84. DTW_TB_QUERYCOLNONJ 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要从中返回列号的宏表格变量。
字符串	<i>name</i>	IN	列标题的名称(对于这个列标题返回了列号)。如果表中不存在该列标题，则返回 0。
整数	<i>col</i>	OUT	包含某一列列号的变量(该列的名称在 <i>name</i> 中指定)。

例

例 1: 检索列名为 SERIAL_NUMBER 的列的列号

```
%DEFINE myTable = %TABLE
%DEFINE col = ""

@DTW_TB_QUERYCOLNONJ(myTable, "SERIAL_NUMBER", col)
```

例 2检索列名为 SERIAL_NUMBER 的列的列号

```
%DEFINE myTable = %TABLE
<P>The "SERIAL_NUMBER" column is column number @DTW_TB_rQUERYCOLNONJ(myTable, "SERIAL_NUMBER")
```

DTW_TB_ROWS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

返回表格中当前的行数。

格式

```
@DTW_TB_ROWS(table, rows)
@DTW_TB_rROWS(table)
```

值

表 85. DTW_TB_ROWS 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	宏表格变量，它返回当前的行数。
整数	<i>rows</i>	OUT	包含 <i>table</i> 中当前行数的变量。

例

例 1: 检索表格中当前的行数，并把值赋给 *rows*

```
%DEFINE myTable = %TABLE
%DEFINE rows = ""
...
@FillTable()
...
@DTW_TB_ROWS(myTable, rows)
```

例 2: 检索并显示当前行号的值

```
%DEFINE myTable = %TABLE
...
@FillTable()
...
<P>The table value of row 1, column 1 is @DTW_TB_rROWS(myTable, "1", "1").
```

DTW_TB_SELECT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回一个 HTML SELECT 菜单。

格式

@DTW_TB_SELECT(table, name, optioncol, size, multiple, rows, selectrows)

值

表 86. DTW_TB_SELECT 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要作为 SELECT 字段来显示的宏表格变量。
字符串	<i>name</i>	IN	SELECT 字段中 NAME 属性的值。
整数	<i>optioncol</i>	IN	<i>table</i> 中的列号，它的值将在 SELECT 字段的 OPTION 标记中使用。缺省情况是使用第一列。
整数	<i>size</i>	IN	<i>table</i> 中的行数，用于 SELECT 字段中的 OPTION 标记。缺省情况是使用所有行。
字符串	<i>multiple</i>	IN	指定是否允许多个选择。缺省值是 N，即不允许多个选择。
字符串	<i>selectedrows</i>	IN	<i>table</i> 中的行数，它用在 SELECT 字段中。缺省情况是使用所有行。
字符串	<i>rows</i>	IN	表格中的行列表，这些行的 OPTION 标记都将被选择。如果要指定多行，必须将“多个参数”选项设置为 Y。缺省值是使用第一个项目。

例

生成一个具有多项选择的 HTML SELECT 菜单

@DTW_TB_SELECT(Mytable,"URL6","3","","y","1 2 4","1 4")

结果:

```
<SELECT NAME="URL6" SIZE="3" MULTIPLE>
<OPTION SELECTED>image1text
<OPTION>image2text
<OPTION SELECTED>image4text
</SELECT>
```

DTW_TB_SETCOLS

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

设置表中的列数。用 DTW_TB_SETN() 函数指定列标题。对于每个表，您只能使用 DTW_TB_SETCOLS() 函数一次。接下去，使用 DTW_TB_DELETECOL() 或 DTW_TB_INSERTCOL() 函数来更改表中的列数。

格式

@DTW_TB_SETCOLS(table, cols)

值

表 87. DTW_TB_SETCOLS 参数

数据类型	参数	用法	说明
表格	table	IN	设置了列数的宏表格变量。
整数	cols	IN	要在 table 中分配的初始列数。

例

例 1: 为表分配三个列并指定列名

```
%DEFINE myTable = %TABLE

@DTW_TB_SETCOLS(myTable, "3")
@DTW_TB_SETN(myTable, "Name", "1")
@DTW_TB_SETN(myTable, "Address", "2")
@DTW_TB_SETN(myTable, "Phone", "3")
```

DTW_TB_SETN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

给列标题指定一个名称。要删除一个列标题，可将列标题指定为 NULL。

调用 DTW_TB_SETN() 之前必须先设置表中的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

```
@DTW_TB_SETN(table, name, col)
```

值

表 88. DTW_TB_SETN 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要在其中设置列名的宏表格变量。
字符串	<i>name</i>	IN	指定给列标题的字符串(该列在 <i>col</i> 中指定)。
整数	<i>col</i>	IN	要设置标题的列的列号。

例

例 1: 为列标题 1 至 3 指定一个名称

```
%DEFINE myTable = %TABLE

@DTW_TB_SETCOLS(myTable, "3")
@DTW_TB_SETN(myTable, "Name", "1")
@DTW_TB_SETN(myTable, "Address", "2")
@DTW_TB_SETN(myTable, "Phone", "3")
```

例 2: 为列 2 删除列标题。这可以通过在未定义的函数调用时传递一个变量来实现。缺省情况下，这个变量的值为 NULL

```
%DEFINE myTable = %TABLE

@DTW_TB_SETN(myTable, nullVar, "2")
```


DTW_TB_SETV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

在表格中指定一个值。要删除一个表格值，可将该值指定为 NULL。

调用 DTW_TB_SETV() 之前必须先设置表中的列数。用 DTW_TB_SETCOLS() 或 DTW_TB_INSERTCOL() 函数设置列数，或者通过把表格传送给一个语言环境来设置。

格式

@DTW_TB_SETV(table, value, row, col)

值

表 89. DTW_TB_SETV 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	要在其中设置表格值的宏表格变量。
字符串	<i>value</i>	IN	指定给表格值的字符串(其行和列在 <i>row</i> 和 <i>col</i> 中指定)。
整数	<i>row</i>	IN	要设置的行号值。
整数	<i>col</i>	IN	要设置的列号值。

例

例 1: 为第 3 行第 3 列指定一个值

```
%DEFINE myTable = %TABLE
@DTW_TB_SETV(myTable, "value3.3", "3", "3")
```

例 2: 删除第 4 行、第 2 列的表格值。这可以通过在未定义的函数调用时传递一个变量来实现。缺省情况下，这个变量的值为 NULL。

```
%DEFINE myTable = %TABLE
@DTW_TB_SETV(myTable, nullVar, "4", "2")
```

DTW_TB_TABLE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从一个宏表格变量返回一个 HTML 表格。

格式

@DTW_TB_TABLE(table, options, collist, cellstyle, link_u, image_u, url_text, url_style)

值

表 90. DTW_TB_TABLE 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	显示成 HTML 表格的宏表格变量。
字符串	<i>options</i>	IN	TABLE 标记内部的表格属性。缺省情况是不使用任何属性。有效值包括: <ul style="list-style-type: none">• BORDER• CELLSPACING• WIDTH
字符串	<i>collist</i>	IN	<i>table</i> 中用在 HTML 表格中的列号。缺省情况是使用所有列。
字符串	<i>cellstyle</i>	IN	在每个 TD 标记中的文本附近出现的 HTML 风格元素的一个列表, 例如 B 和 I。缺省情况是不使用风格标记。
整数	<i>link_u</i>	IN	<i>table</i> 中的列号, 它包含用于创建 HTML 链路的 URL。同样必须在 <i>collist</i> 中指定这一列。缺省是不生成 HTML 链接。
整数	<i>image_u</i>	IN	<i>table</i> 中的列号, 它包含用于创建内嵌图象的 URL。同样必须在 <i>collist</i> 中指定这一列。缺省是不生成图象标记。
整数	<i>url_text</i>	IN	<i>table</i> 中的列号, 它包含显示给 HTML 链接或内嵌图象的文本。缺省情况是使用 URL 本身。
字符串	<i>url_style</i>	IN	HTML 风格元素的一个列表, 这些风格用于在 <i>url_text</i> 中指定的文本。缺省情况是不生成风格标记。

例

例 1: 为一张具有一个边框并使用 B (黑体)和 I (斜体字)标记的表格生成 HTML 标记
@DTW_TB_TABLE(Mytable,"BORDER","4 2 1","i","2","1","4","b")

结果:

```
<TABLE BORDER>
<TR>
<TH>TITLE
<TH>LINKURL
<TH>IMAGEURL
<TR>
```

```

<TD><i>link1text</i>
<TD><A HREF="http://www.mycompany.com/link1.html"><b>link1text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b>link1text</b>
<TR>
<TD><i>link2text</i>
<TD><A HREF="http://www.mycompany.com/link2.html"><b>link2text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b>link2text</b>
<TR>
<TD><i>link3text</i>
<TD><A HREF="http://www.mycompany.com/link3.html"><b>link3text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b>link3text</b>
</TABLE>

```

DTW_TB_TEXTAREA

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从一个宏表格变量返回 HTML TEXTAREA 标记。

格式

@DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol, rows)

值

表 91. DTW_TB_TEXTAREA 参数

数据类型	参数	用法	说明
表格	<i>table</i>	IN	作为 TEXTAREA 标记显示的宏表格变量。
字符串	<i>name</i>	IN	文本区域的名称。
整数	<i>numrows</i>	IN	在 rows 中指定的文本区域的高度。缺省是 <i>table</i> 中的行数。
整数	<i>numcols</i>	IN	在 columns 中指定的文本区域的宽度。缺省是 <i>table</i> 中最长行的长度。
整数	<i>valuecol</i>	IN	<i>table</i> 中的列号，其值显示在文本区域中。缺省值是第一列。
字符串	<i>rows</i>	IN	<i>table</i> 的一张行列表，用于生成 TEXTAREA 标记。缺省情况是使用所有的行。

例

例 1: 生成 HTML TEXTAREA 标记并指定要包含哪些行

@DTW_TB_TEXTAREA(Mytable,"textarea5","3","70","4","1 3 4")

结果:

```
<TEXTAREA NAME="textarea5" ROWS="3" COLS="70">
link1text
link3text
link4text
</TEXTAREA>
```

平面文件接口函数

平面文件接口(FFI)允许您打开、读取和处理平面文件源(文本文件)中的数据，也可以将数据存储到平面文件中。以下平面文件接口内部函数可用：

- 第197页的『DTWF_APPEND』
- 第199页的『DTWF_CLOSE』
- 第200页的『DTWF_DELETE』
- 第200页的『DTWF_DELETE』
- 第204页的『DTWF_OPEN』
- 第205页的『DTWF_READ』
- 第207页的『DTWF_REMOVE』
- 第208页的『DTWF_SEARCH』
- 第210页的『DTWF_UPDATE』
- 第212页的『DTWF_WRITE』

以下章节将讨论如何使用 FFI 内部函数以及访问平面文件源：

- 『访问平面文件数据源』
- 第195页的『平面文件接口定界符』
- 第196页的『锁定文件』

访问平面文件数据源

您可以在 `Net.Data` 初始化文件中使用 `FFI_PATH` 路径配置语句来列出使用 FFI 函数时允许指定的目录和子目录，同时为那些不在该路径语句所包括的目录中的文件提供安全性。`Net.Data` 初始化文件在发行时是没有 `FFI_PATH` 的。请参阅 *Net.Data 管理与程序设计指南*，以了解如何配置路径。

`FFI_PATH` 使用以下语法：

```
FFI_PATH /path1;/path2;/path3...
```

在宏函数中调用 FFI 语言环境时，您可以使用 FFI 函数的 *filename* 参数来指定与 FFI 函数一起使用的平面文件的路径。例如：

```
@DTWF_READ("/macros/myfile.txt", ...)
```

以下章节将讨论：

- 第194页的『`Net.Data` 如何确定平面文件的位置』
- 第194页的『平面文件配置规则』
- 第195页的『安全性建议』
- 第195页的『权限要求』

Net.Data 如何确定平面文件的位置

Net.Data 使用 FFI 函数的 *filename* 参数中的信息在 Net.Data 初始化文件中搜索 FFI_PATH 语句，并确定是使用指定的目录还是使用当前目录。

当 FFI 函数中指定文件名时，Net.Data 将通过搜索 FFI_PATH 中列出的每个路径来试图找出该文件(从指定的第一个路径开始)。Net.Data 使用找到的第一个副本。如果文件没有找到，Net.Data 将试图在它正在运行的进程或线程的当前工作目录中查找该文件。

例: Net.Data 使用 FFI_PATH 配置语句来定位文件

FFI_PATH 包含以下目录:

```
FFI_PATH /macros;/macros/org1;/macros/org2
```

如果使用 DTWF_READ 函数来读取一个现存文件，并且指定文件名 *myfile.txt*，那么，Net.Data 将在目录 */macros*、*/macros/org1* 和 */macros/org2* 中搜索该文件(假定 FFI_PATH 包含了上述指定的路径列表)。如果所有的目录中都没有找到文件，则 Net.Data 将在当前工作目录中搜索该文件。

如果指定文件名 */mydir/myfile.txt*，Net.Data 将生成一个错误，因为目录 */mydir* 与 FFI_PATH 中指定的允许目录都不匹配。

确定当前目录:

当 FFI 函数中指定的文件不具有目录路径并且在 FFI_PATH 指定的任何路径中都没有找到文件，则 Net.Data 将使用当前目录。

Net.Data 的当前目录取决于您的 Web 服务器的配置:

- 如果您使用 CGI，那么当前目录就是运行 Net.Data 的目录，通常是 *\www\cgi-bin*。
- 如果您使用 Web 服务器 API，那么当前目录就可能各不相同。如果服务器的缺省请求路由选择或资源映射被更改了，那么当前目录也可能更改。

指定平面文件访问的建议:

使用以下建议以确保 Net.Data 能够访问平面文件数据源。

- 在使用 DTWF_CREATE 函数创建平面文件时，请确保 FFI_PATH 中有您指定的目录路径，或者您知道哪一个是当前目录。如果不指定目录，Net.Data 将试图在当前工作目录中创建文件。
- 如果您将目录包含在 *filename* 参数中，则请指定与 FFI_PATH 中某一路径相匹配的全路径，因为 Net.Data 不会在 FFI_PATH 指定的目录中对子目录进行搜索。
- 对 *filename* 参数使用绝对路径，特别是在使用 Web 服务器 API 时。

平面文件配置规则

在 Net.Data 初始化文件中添加或更新 FFI_PATH 时使用以下规则:

- FFI_PATH 中的路径语句必须包含有效的可打印字符。FFI 不允许使用包含问号 (?) 或双引号 (") 的路径。
- 在宏文件中与 *filename* 参数一起使用的所有目录和子目录必须在 FFI_PATH 中指定。不搜索 *filename* 中列出的路径的子目录，除非在 FFI_PATH 中明确指定。
- 使用 FFI_PATH 语句的绝对路径。

安全性建议

您可以指定 FFI 函数可以使用 Net.Data 初始化文件中的 FFI_PATH 语句来访问哪些文件。FFI 只搜索该语句中列出的路径，因此其它目录中的文件不会遭到未授权的访问。

例如，您可以指定一个类似于此的 FFI_PATH，从而为 public 或 guest 用户 ID 指定目录。

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

以下列表提供了保证平面文件安全的建议：

- 选择那些适于平面文件操作的目录。这些目录需要添加到 FFI_PATH 中以限制对那些目录的搜索。
- 在宏中执行 DTWF_REMOVE 或其它调出操作的人必须小心，以免除去或改变当前目录中可能存在的扩展名为 .dll 和 .cmd 的文件。
- 对添加到系统中的宏进行合理的控制，采取适当的步骤来保护系统中的文件。
- 不要在 FFI_PATH 中指定允许匿名 FTP 用户写入的路径。如果指定了这样的路径，那么别人就可以在系统中放入某个 Net.Data 宏，从而可以执行原先所不允许的操作。
- 不要将 Net.Data 初始化文件的路径添加到 FFI_PATH 中。

权限要求

请确保执行 Net.Data 的用户 ID 对 FFI 内部函数所使用的文件具有访问权限。请参阅 *Net.Data 管理和程序设计指南* 的配置一章中关于指定 Web 服务器对 Net.Data 文件的访问权限这一部分，以获取更多信息。

平面文件接口定界符

为了改进性能，您可以把来自一系列 SQL 请求的 Net.Data 表格输出保留在一个平面文件中。然后就可以在后继的请求中检索此平面文件，而不需要重新发出 SQL 请求。

可以从 Net.Data 表格创建 Net.Data 平面文件；也可以从平面文件来构建 Net.Data 表格。为了实现表格和平面文件之间的转换，必须在表格列和平面文件的记录之间定义一种映射。定界符是 FFI 在根据请求的变换将文件分割为几个部分(如一行中的几列)时所使用的标志或分隔符。定界符提供了一种方法，用于定义如何划分平面文件中的记录，如何映射到表格列中；以及如何将表格列映射成平面文件中的记录。

有两种类型的定界符：

换行字符 (ASCITEXT)

当表格仅由一列组成时，可以使用这种转换。Net.Data 将相应的平面文件中的每个记录映射成表格中的一行。在这种情况下，仅使用正常的换行字符作为定界符，平面文件的记录之间就是用换行字符分隔的。

换行字符和定界字符串 (DELIMITED)

当表格由多列组成时，可以使用这种转换。当 Net.Data 从一个表格行创建一个平面文件记录时，它在项目之间放置定界字符串，以此作为分隔符。当 Net.Data 从平面文件重新建立表格时，它利用定界字符串来确定每行中有多少信息是要放入表格的一列中。在此情况下，用于在平面文件中划分不同记录的正常换行字符对应于表格中的行，而定界字符串划分了同一个记录中的不同项目。

对于读出操作来说，分界符将文件的内容分成表格中的行与列。对于写操作来说，分界符则表示表格行与表格列中一个值的结尾。Net.Data 将分界符作为宏字符串传送给 FFI，并且在字符串的末尾不包含空字符，除非在 DELIMITER 参数中明确地列出。

要在分界符中使用空字符，需要将 DELIMITER 参数指定为一对双引号中的反斜杠与一个 0 (『/0』)，而不是双引号中的空字符串 (『"』)。如果您指定 ASCIITEXT 变换，则 Net.Data 将使用新行字符作为分界符，并忽略任何请求的分界符。

如果您在写操作中使用与读出操作不同的分界符，则将对这个文件进行不合要求的更改。Net.Data 使用新的定界符写文件。

分界符的最大长度是 256 个字符。

锁定文件

您可以使用 DTWF_OPEN 和 DTWF_CLOSE 函数锁定平面文件。这些函数允许您对需要访问该文件的 Net.Data 进程保留这个文件，这样，其它的进程就不能读取或更新这个文件。

要锁定一个文件，可以使用 DTWF_OPEN 函数。这个函数确保其它应用程序不可使用该文件，并可防止文件在读和更新之间被更改。

要释放文件，可以使用 DTWF_CLOSE 函数。这个函数将文件从 Net.Data 进程的控制中释放出来，并允许其它进程读或更新该文件。

DTWF_APPEND

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将表格变量的内容写到一个文件的结尾。

文件中的当前内容将影响使用 DTWF_APPEND 的结果，特别是最后一行最后一列的内容。如果文件最后一行最后一列的值后面跟有一个换行字符，则附加的数据放入新行中。否则，附加的数据成为文件最后一行的一部分。

格式

```
@DTWF_APPEND(filename, transform, delimiter, table, retry, rows)
@DTWF_APPEND(filename, transform, delimiter, table, retry)
@DTWF_APPEND(filename, transform, delimiter, table)
```

值

表 92. DTWF_APPEND 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	向它添加变量内容的文件的名称。此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCII TEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCII TEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCII TEXT，则忽略此参数。
表格	<i>table</i>	IN	用于读取记录的表格变量。 对于非 OS/400 用户：FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
整数	<i>retry</i>	IN	不能立即追加文件时需要的重试次数。缺省情况是不重试。
整数	<i>rows</i>	IN	<i>table</i> 中要追加的最大行数。缺省情况是追加所有的行。指定 0 则添加所有的行。

例

例 1:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "DELIMITED", " ;", myTable)
```

例 2:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCIITEXT", " ;", myTable)
```

例 3:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCIITEXT", " ;", myTable, "0", "10")
```

DTWF_CLOSE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

关闭由 DTWF_OPEN 打开的文件。

格式

```
@DTWF_CLOSE(filename, retry)
@DTWF_CLOSE(filename)
```

值

表 93. DTWF_CLOSE 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要关闭的文件名称。此函数成功完成时，这个参数返回全限定文件名。
整数	<i>retry</i>	IN	不能立即关闭文件时需要的重试次数。缺省情况是不重试。

例

例 1:

```
@DTWF_CLOSE(myFile, "5")
```

DTWF_DELETE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

删除文件中的记录。(不删除空文件)。

格式

```
@DTWF_DELETE(filename, transform, delimiter, retry, rows, startrow)
@DTWF_DELETE(filename, transform, delimiter, retry, rows)
@DTWF_DELETE(filename, transform, delimiter, retry)
@DTWF_DELETE(filename, transform, delimiter)
```

值

表 94. DTW_DELETE 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要删除记录的那个文件的名称。 此函数成功完成时， 这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCII TEXT - 将表格写入文件， 在两个列值之间使用换行字符， 并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件， 并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCII TEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。 这个参数是区别大小写的。 如果 <i>transform</i> 为 ASCII TEXT， 则忽略此参数。
整数	<i>retry</i>	IN	不能立即删除记录时所需要的重试次数。 缺省情况是不重试。
整数	<i>rows</i>	IN	要删除的最大行数。 缺省情况是删除所有的行。 指定 0 则删除所有的行。
整数	<i>startrow</i>	INOUT	开始执行删除操作的行号。 值 1 表示从第一个行开始删除。 如果这个值大于文件中的总行数， 则将这个值更改为最后一个记录， 并返回一个错误。 缺省情况是从 1 开始。

例

例 1:

```
%DEFINE{
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "5000"
```

```
        myRows = "2"  
    %}  
    @DTWF_DELETE(myFile, "Delimited", "|", myWait, myRows)
```

例 2:

```
%DEFINE{  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myStart = "1"  
    myRows = "2"  
%}  
@DTWF_DELETE(myFile, "Asciitext", "|", "0", myRows, myStart)
```

DTWF_INSERT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

在文件中插入记录。

格式

```
@DTWF_INSERT(filename, transform, delimiter, table, retry, rows, startrow)
@DTWF_INSERT(filename, transform, delimiter, table, retry, rows)
@DTWF_INSERT(filename, transform, delimiter, table, retry)
@DTWF_INSERT(filename, transform, delimiter, table)
```

值

表 95. DTWF_INSERT 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	用于插入记录的文件名称。 此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCII TEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCII TEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCII TEXT，则忽略此参数。
表格	<i>table</i>	IN	从中把记录插入到文件的表格变量。 对于非 OS/400 用户：FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
整数	<i>retry</i>	IN	不能立即写文件时需要的重试次数。 缺省情况是不重试。
整数	<i>rows</i>	IN	从 <i>table</i> 中取出进行插入的最大行数。缺省情况是插入所有的行。 值 0 表示插入所有的行。
整数	<i>startrow</i>	INOUT	开始执行插入操作的行号。值 1 表示在第一个行开始插入。 如果这个值大于文件中的总行数，则将这个值更改为最后一个记录，并返回一个错误。缺省情况是从 1 开始。

例

例 1:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "3000"
}%
@DTWF_INSERT(myFile, "Delimited", "|", myTable, myWait)
```

例 2:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myStart = "1"
    myRows = "2"
}%
@DTWF_INSERT(myFile, "Asciitext", "|", myTable, "0", myRows, myStart)
```

DTWF_OPEN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

显式地打开一个文件。当该文件不存在时，应指定文件名的绝对路径，并且要创建该文件的目录必须与 FFI_PATH 中指定的某一目录相匹配。如果不使用绝对路径，则将在当前工作目录中打开该文件。DTWF_OPEN 保持文件的打开，否则此文件在每次平面文件操作结束后关闭。

性能提示： 使用 DTWF_OPEN 减少文件打开的次数。

文件直到使用 DTWF_CLOSE 或宏处理结束后才关闭。

格式

```
@DTWF_OPEN(filename, mode, retry)
@DTWF_OPEN(filename, mode)
```

值

表 96. DTWF_OPEN 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要打开的文件名称。此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>mode</i>	IN	请求的访问类型： <ul style="list-style-type: none">• r - 打开一个现有的文件，用于读。• w - 创建一个文件，用于写。（如果存在同名文件，则将破坏此文件）• a - 打开一个文件，用于追加。 如果找不到此文件，Net.Data 将创建此文件。• r+ - 打开一个现有文件，用于读和写。• w+ - 创建一个文件，用于读和写。（如果存在同名文件，则将破坏此文件）• a+ - 以追加方式打开一个文件，用于读和写。 如果找不到此文件，Net.Data 将创建此文件。
整数	<i>retry</i>	IN	不能立即打开文件时需要的重试次数。 缺省情况是不重试。

例

```
例 1:
%DEFINE{
    myFile = "c:/private/myfile"
    myMode = "r+"
}%
@DTWF_OPEN(myFile, myMode, "1000")
```


DTWF_READ

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从一个文件中将记录读入一表格变量。

格式

```
@DTWF_READ(filename, transform, delimiter, table, retry, rows, startrow, columns)
@DTWF_READ(filename, transform, delimiter, table, retry, rows, startrow)
@DTWF_READ(filename, transform, delimiter, table, retry, rows)
@DTWF_READ(filename, transform, delimiter, table, retry)
@DTWF_READ(filename, transform, delimiter, table)
```

值

表 97. DTWF_READ 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要把记录读入表格变量的那个文件的名称。此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCII TEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCII TEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCII TEXT，则忽略此参数。
表格	<i>table</i>	OUT	一个表格变量，用于写入从文件中读取的记录。 对于非 OS/400 用户：FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
整数	<i>retry</i>	IN	不能立即读取文件时需要的重试次数。缺省情况是不重试。
整数	<i>rows</i>	INOUT	要读到表格中的文件记录的最大个数。缺省情况是读取所有记录，或直到表格填满为止。0 值表示要读到文件结束。返回所生成的表格的行数。
整数	<i>startrow</i>	IN	文件中开始读取记录的位置。缺省情况是从第一个记录开始。

DTWF_REMOVE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

删除整个文件。

格式

```
@DTWF_REMOVE(filename, retry)
@DTWF_REMOVE(filename)
```

值

表 98. DTW_REMOVE 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要删除的文件名称。此函数成功完成时，这个参数返回全限定文件名。
整数	<i>retry</i>	IN	不能立即删除文件时需要的重试次数。缺省情况是不重试。

例

```
例 1:
%DEFINE myFile = "c:/private/myfile"
@DTWF_REMOVE(myFile)
```

```
例 2:
%DEFINE{
    myFile = "c:/private/myfile"
    myWait = "2000"
}%
@DTWF_REMOVE(myFile, myWait)
```

DTWF_SEARCH

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

返回对一个表格变量的字符串搜索结果。

格式

```
@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows, startrow)
@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows)
@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry)
@DTWF_SEARCH(filename, transform, delimiter, table, searchFor)
```

值

表 99. DTWF_SEARCH 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要搜索的文件名称。此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCII TEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCII TEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCII TEXT，则忽略此参数。
表格	<i>table</i>	OUT	用于存放搜索结果的表格变量。 如果 <i>transform</i> 为 DELIMITED，则返回三列： <ul style="list-style-type: none">• 找到匹配的行号。• 找到匹配的列号。• 文件中匹配的列。 对于非 OS/400 用户: FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
字符串	<i>searchFor</i>	IN	要查找的字符串。
整数	<i>retry</i>	IN	不能立即搜索文件时需要的重试次数。 缺省情况是不重试。

表 99. DTWF_SEARCH 参数 (续)

数据类型	参数	用法	说明
整数	<i>rows</i>	INOUT	读至 <i>table</i> 中的最大行数。缺省情况是读取所有的行，或直到 <i>table</i> 填满为止。指定 0 表示要读到文件结束。此参数返回结果表格中的行数。
整数	<i>startrow</i>	IN	文件中开始执行搜索操作的位置。缺省值是 1，表示从第一个记录开始搜索。

使用法

- DTWF_SEARCH 所返回的表格有三列。前两行中包含找到匹配的行号与列号；最后一行的列值中包含 *SearchFor* 参数中指定的字符。例如，如果文件第四行、第三列中包含匹配的字符，则返回表第一列的值为 4，表示文件中的行号；第二列的值为 3，表示匹配值在文件中的列号；第三列中则是完整的列值。
- SearchFor* 参数不能包括 *delimiter* 参数的内容。

例

例 1:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1000"
    mySearch = "0123456789abcdef"
@DTWF_SEARCH(myFile, "DELIMITED", ";",
    myTable, mySearch, myWait)
```

例 2:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    mySearch = "answer:"
    myWait = "0"
    myRows = "0"
    myStartrow = "1"
}%
@DTWF_SEARCH(myFile, "DELIMITED", ";", myTable,
    mySearch, myWait, myRows, myStartrow)
```

DTWF_UPDATE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

从表格变量更新文件中的记录。当该文件不存在时，应指定文件名的绝对路径，并且要创建该文件的目录必须与 FFI_PATH 中指定的某一目录相匹配。如果不使用绝对路径，则将在当前工作目录中打开该文件。

格式

```
@DTWF_UPDATE(filename, transform, delimiter, table, retry, rows, startrow)
@DTWF_UPDATE(filename, transform, delimiter, table, retry, rows)
@DTWF_UPDATE(filename, transform, delimiter, table, retry)
@DTWF_UPDATE(filename, transform, delimiter, table)
```

值

表 100. DTWF_UPDATE 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	要从表格变量更新其记录的文件名称。 此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCIITEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCIITEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCIITEXT，则忽略此参数。
表格	<i>table</i>	IN	用于更新文件记录的表格变量。 对于非 OS/400 用户: FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
整数	<i>retry</i>	IN	不能立即写文件时需要的重试次数。 缺省情况是不重试。
整数	<i>rows</i>	IN	<i>table</i> 中要更新的最大记录数。缺省情况是更新所有的记录。0 值表示更新文件中所有行。

表 100. DTWF_UPDATE 参数 (续)

数据类型	参数	用法	说明
整数	<i>startrow</i>	INOUT	要更新的第一个文件记录。缺省值是 1，表示要从文件开头开始更新。如果这个值大于文件中的记录个数，则将这个值更改为指向文件中的最后一个记录，并返回一个错误。

例

例 1:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1500"
    myRows = "2"
}%
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

例 2:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myStart = "1"
    myRows = "2"
}%
@DTWF_UPDATE(myFile, "Asciitext", "|", myTable, "0", myRows, myStart)
```

DTWF_WRITE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

目的

将一个表格变量的内容写入一个文件。当该文件不存在时，应指定文件名的绝对路径，并且要创建该文件的目录必须与 FFI_PATH 中指定的某一目录相匹配。如果不使用绝对路径，则将在当前工作目录中打开该文件。

格式

```
@DTWF_WRITE(filename, transform, delimiter, table, retry, rows, startrow)
@DTWF_WRITE(filename, transform, delimiter, table, retry, rows)
@DTWF_WRITE(filename, transform, delimiter, table, retry)
@DTWF_WRITE(filename, transform, delimiter, table)
```

值

表 101. DTWF_WRITE 参数

数据类型	参数	用法	说明
字符串	<i>filename</i>	INOUT	一个文件名称，用于写入表格变量中的记录。此函数成功完成时，这个参数返回全限定文件名。
字符串	<i>transform</i>	IN	文件的格式： <ul style="list-style-type: none">• ASCIITEXT - 将表格写入文件，在两个列值之间使用换行字符，并忽略 <i>delimiter</i> 参数。• DELIMITED - 将表格写入文件，并在 <i>delimiter</i> 参数中指定定界符。 文件中的新行字符在 ASCIITEXT 和 DELIMITED 变换过程中指出 Net.Data 宏表格中行的结尾。
字符串	<i>delimiter</i>	IN	用于表示一个值结束的字符串。这个参数是区别大小写的。如果 <i>transform</i> 为 ASCIITEXT，则忽略此参数。
表格	<i>table</i>	IN	表格变量，用于把其中的行调出到文件。 对于非 OS/400 用户：FFI 表中一行的最大的长度为 16383 个字符。这个限制包括对 Net.Data 宏表格中每一列所使用的一个空字符。
整数	<i>retry</i>	IN	不能立即写文件时需要的重试次数。缺省情况是不重试。
整数	<i>rows</i>	IN	要写入的文件记录的最大个数。缺省情况是写整个表格。0 值表示把所有记录写到文件结束。

表 101. DTWF_WRITE 参数 (续)

数据类型	参数	用法	说明
整数	<i>startrow</i>	INOUT	开始写到文件中的第一个记录。缺省值是 1，表示要从第一个记录开始。如果指定的值超出了文件结束范围，则返回文件中的最后一行，并返回一个错误。

例

例 1:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "DELIMITED", ";", myTable)
```

例 2:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000")
```

例 3:

```
%DEFINE{
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000", "10", "50")
```

Web 注册表函数

Web 注册表是一个文件，由 Net.Data 维护此文件的一个关键字，允许您方便地添加、检索和删除其中的条目。您可以在一个系统中创建多个 Net.Data Web 注册表。每个注册表有自己的名称，可以包含多个条目。Net.Data 提供了一些函数，用于维护注册表及其包含的条目。

- 第215页的『DTWR_ADDENTRY』
- 第216页的『DTWR_CLEARREG』
- 第217页的『DTWR_CLOSEREG』
- 第218页的『DTWR_CREATEREG』
- 第219页的『DTWR_DELENTY』
- 第220页的『DTWR_DELREG』
- 第221页的『DTWR_LISTREG』
- 第222页的『DTWR_LISTSUB』
- 第223页的『DTWR_OPENREG』
- 第224页的『DTWR_RTVENTRY』
- 第225页的『DTWR_UPDATEENTRY』

限制：当使用 OS/2 时，不要把星号(*)作为 *registry*、*registryVariable* 和 *registryData* 参数的值。

DTWR_ADDENTRY

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

将一个条目添加至 Web 注册表。

格式

```
@DTWR_ADDENTRY(registry, registryVariable, registryData, index)
@DTWR_ADDENTRY(registry, registryVariable, registryData)
```

值

表 102. DTWR_ADDENTRY 参数

数据类型	参数	用法	说明
字符串	registry	IN	要向其中添加条目的注册表名称。
字符串	registryVariable	IN	要添加的注册表条目中 registryVariable 字符串部分的值。
字符串	registryData	IN	要添加的注册表条目中 registryData 字符串部分的值。
字符串	index	IN	要添加的索引项中 registryVariable 字符串索引部分的值。此参数是可选的。 如果指定此参数，则将一个索引项添加到指定的注册表。

例

```
例 1:
@DTWR_ADDENTRY("Myregistry", "Jones", "http://Advantis.com/~Jones/webproj")
```

```
例 2:
@DTWR_ADDENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/",
"WORK_URL,")
```

DTWR_CLEARREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

从 Web 注册表中清除条目。

格式

@DTWR_CLEARREG(registry)

值

表 103. DTWR_CLEARREG 参数

数据类型	参数	用法	说明
字符串	registry	IN	要清除的注册表名称。

例

例 1:

@DTWR_CLEARREG("Myregistry")

DTWR_CLOSEREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

关闭一个 Web 注册表

格式

@DTWR_CLOSEREG(registry)

值

表 104. DTWR_CLOSEREG 参数

数据类型	参数	用法	说明
字符串	registry	IN	要关闭的注册表的名称。 限制: Web 注册表名称中不要使用例如星号 (*)和反斜线(\)等的特殊字符。

例

例 1: 关闭注册表

@DTWR_CLOSEREG("/qsys.lib/mylib.lib/myreg.file")

DTWR_CREATEREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

创建一个新的 Web 注册表。

Re

格式

```
@DTWR_CREATEREG(registry, security)
@DTWR_CREATEREG(registry)
```

值

表 105. DTWR_CREATEREG 参数

数据类型	参数	用法	说明
字符串	<i>registry</i>	IN	要创建的注册表名称。 限制: Web 注册表名称中不要使用例如星号(*)和反斜线(\)等的特殊字符。
字符串	<i>security</i>	IN	用以创建 <i>registry</i> 的安全性类型。在 UNIX 操作系统中, 缺省的安全性与注册表所在目录的安全性相同。必须为三个安全性组指定安全性: 用户、组和公共。R 表示具有读许可权, W 表示具有写许可权, 而 X 表示具有执行许可权。 例如, 如果要对三个组都指定全部权限, 可以指定此参数为 *RWX, *RWX, *RWX。 此参数是可选的。

例

例 1:

```
@DTWR_CREATEREG("myRegistry")
```

例 2:

```
@DTWR_CREATEREG("URLLIST", "*RWX, *RWX, *R")
```

DTWR_DELENTY

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

将 Web 注册表删除一个条目。

格式

@DTWR_DELENTY(registry, registryVariable, index)

@DTWR_DELENTY(registry, registryVariable)

值

表 106. DTWR_DELENTY 参数

数据类型	参数	用法	说明
字符串	registry	IN	要从中移去条目的注册表名称。
字符串	registryVariable	IN	要移去的条目中 registryVariable 字符串部分的值。
字符串	index	IN	索引项中 registryVariable 字符串索引部分的值。这是一个可选参数。如果指定此参数，则从注册表移去索引项。

例

例 1:

@DTWR_DELENTY("Myregistry", "Jones")

例 2:

@DTWR_DELENTY("URLLIST", "SMITH", "WORK_URL")

DTWR_DELREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

删除一个 Web 注册表。

格式

@DTWR_DELREG(registry)

值

表 107. DTWR_DELREG 参数

数据类型	参数	用法	说明
字符串	<i>registry</i>	IN	要删除的注册表名称。

例

例 1:

```
@DTWR_DELREG("Myregistry")
```


DTWR_LISTREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

列出整个 Web 注册表。DTWR_LISTREG 返回由用户传递的 OUT 表格变量中的注册表条目信息。此表格变量是在作为参数传递到 FUNCTION 模块执行 LISTREG 注册表操作以前，在用户宏中定义的。

如果对于表格中的最大行数，用户使用 ALL 选项来定义表格变量，则此操作列出了表格中所有可用的注册表条目，每个表格行一个。另一方面，如果用户指定表格中的最大行数为值 X，但在指定的注册表中有多于 X 个条目，则只列出前面的 X 个条目，并返回一个错误代码，表示由于没有足够的表格行用于列出更多的条目，现在只能列出一个部分列表。如果值 X 超过了指定的注册表中的可用条目数，则列出所有条目。

在表格中总是有两列。由 Web 注册表语言环境指定表格的列标题为 "REGISTRY_VARIABLE" 和 "REGISTRY_DATA"。

格式

@DTWR_LISTREG(registry, registryTable)

值

表 108. DTWR_LISTREG 参数

数据类型	参数	用法	说明
字符串	registry	IN	要列出的注册表名称。
字符串	registryTable	OUT	用于存放注册表条目的表格变量的名称。

例

例 1:

```
%DEFINE RegistryTable = %TABLE(ALL)
@DTWR_LISTREG("URLLIST", RegistryTable)
```

DTWR_LISTSUB

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
							X

目的

列出 Web 注册表中的直接子关键字条目。DTWR_LISTSUB 返回由用户传递的 OUT 表格参数中的注册表条目信息。此表格变量是在作为参数传递到 LISTSUBG 注册表操作以前，在用户宏中定义的。

如果对于表格中的最大行数，用户使用 ALL 选项来定义表格变量，则此操作列出了表格中所有可用的注册表条目，每个表格行一个条目。另一方面，如果用户指定表格中的最大行数为值 X，但在指定的注册表中有多于 X 个条目，则只列出前面的 X 个条目，并发回一个错误代码，表示由于没有足够的表格行用于列出更多的条目，现在只能列出一个部分列表。如果值 X 超过了指定的注册表中的可用条目数，则列出所有条目。在表格中总是只有一列。

表格的列标题设置为 "REGISTRY_SUBKEY"。

只有在与 Windows95 “系统注册表” 兼容的操作系统中，此函数才有效。

格式

@DTWR_LISTSUB(registry, registryTable)

值

表 109. DTWR_LISTSUB 参数

数据类型	参数	用法	说明
字符串	registry	IN	要列出的注册表名称。
字符串	registryTable	OUT	用于存放注册表条目的表格变量的名称。

例

```
例 1:
%DEFINE RegistryTable = %TABLE(ALL)
@DTWR_LISTSUB("URLLIST", RegistryTable)
```

DTWR_OPENREG

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

打开一个 Web 注册表。

格式

```
@DTWR_OPENREG(registry, commit)
@DTWR_OPENREG(registry)
```

值

表 110. DTWR_OPENREG 参数

数据类型	参数	用法	说明
字符串	<i>registry</i>	IN	要打开的注册表的名称。
字符串	<i>commit</i>	IN	一个单独的符号或文字串，指定注册表在确认控制下是否打开。可能的值有： Y 在确认控制下打开注册表。 N 不在确认控制下打开注册表。 缺省值为 N

例

```
例 1: 在确认控制下打开注册表
@DTWR_OPENREG("/qsys.lib/mylib.lib/myreg.file", "Y")
```

DTWR_RTVENTRY

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

从 Web 注册表条目中检索 registryData 字符串。

格式

```
@DTWR_RTVENTRY(registry, registryVariable, registryData, index)
@DTWR_RTVENTRY(registry, registryVariable, registryData)
@DTWR_rRTVENTRY(registry, registryVariable, index)
@DTWR_rRTVENTRY(registry, registryVariable)
```

值

表 111. DTWR_RTVENTRY 参数

数据类型	参数	用法	说明
字符串	registry	IN	要在其中检索条目的注册表名称。
字符串	registryVariable	IN	针对检索其 registerData 字符串的注册表项，它的 registryVariable 字符串部分的值。
字符串	registryData	OUT	返回注册表项 registryData 字符串部分的值，此注册表项和 registryVariable匹配。
字符串	index	IN	一个索引项中 registryVariable 字符串的索引部分的值，此索引项的 registryData 字符串被返回。这是一个可选参数。如果指定此参数，则返回索引项的 registryData 字符串。

例

例 1:

```
%DEFINE RegistryData = ""
@DTWR_RTVENTRY("Myregistry", "Jones", RegistryData)
```

例 2:

```
@DTWR_RTVENTRY("URLLIST", "SMITH", RegistryData, "WORK_URL")
```

例 3:

```
@DTWR_rRTVENTRY("Myregistry", "Jones")
```

例 4:

```
@DTWR_rRTVENTRY("URLLIST", "SMITH", "WORK_URL")
```

DTWR_UPDATEENTRY

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X		X	X	X	X

目的

将指定的注册表条目中现有的 *registryData* 字符串替换为由调用者指定的新值。
registerVariable 字符串不能更改。

格式

```
@DTWR_UPDATEENTRY(registry, registryVariable, newData, index)
@DTWR_UPDATEENTRY(registry, registryVariable, newData)
```

值

表 112. DTWR_UPDATEENTRY 参数

数据类型	参数	用法	说明
字符串	<i>registry</i>	IN	要更新其条目的注册表的名称。
字符串	<i>registryVariable</i>	IN	要更新的注册表条目 <i>registryVariable</i> 字符串部分的值。
字符串	<i>newData</i>	IN	给要更新的注册表条目 <i>registryData</i> 字符串部分的新值。
字符串	<i>index</i>	IN	在要更新的索引项中 <i>registryVariable</i> 字符串索引部分的值。这是一个可选参数。 如果指定此参数，则更新索引条目。

例

例 1:

```
@DTWR_UPDATEENTRY("Myregistry", "Jones", "http://advantis.com/~Jones/personal")
```

例 2:

```
@DTWR_UPDATEENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/personal", "WORK_URL")
```

持久性宏函数

持久性宏函数支持在 `Net.Data` 中进行事务处理，这是通过帮助您定义哪些宏块在单个事务中是持久性的来实现的。使用这些函数来定义一个事务的开头与结尾 (HTML 块在该事务中是持久性的)、变量在事务中的作用域以及在事务中提交还是撤消更改。

- 第227页的『DTW_ACCEPT』
- 第228页的『DTW_COMMIT』
- 第229页的『DTW_ROLLBACK』
- 第230页的『DTW_RTVHANDLE』
- 第231页的『DTW_STATIC』
- 第232页的『DTW_TERMINATE』

DTW_ACCEPT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

定义用于调用持久性宏的事务句柄。Net.Data 需要将事务句柄包括在调用宏的 URL 中，作为来自 Web 浏览器的响应。当请求进入 Web 服务器时，服务器将使用事务句柄将请求路由到处理事务的 CGI 进程。

事务句柄必须在宏中每个 HTML 块开始时调用，直至最后的逻辑块，其中包含对 DTW_TERMINATE() 的调用。如果在文本输出到浏览器之前没有找到对 DTW_ACCEPT() 或 DTW_TERMINATE() 的调用，则将出现 Net.Data 错误。

您可以对这一页指定一个超时值，从而覆盖 @DTW_STATIC() 函数中指定的超时值。Web 服务器在指定的时间(以秒为单位)内等待用户对这个请求作出响应。

如果当宏处于持久性状态时调用此函数，则将出现 Net.Data 错误。

技巧: 包含事务句柄的 URL 可以被编码为表单按钮上的操作或显示在浏览器上的页面中的超文本链。

格式

```
@DTW_ACCEPT(handle, timeout)
@DTW_ACCEPT(handle)
```

值

表 113. DTW_ACCEPT 参数

数据类型	参数	用法	说明
字符串	<i>handle</i>	IN	一个变量或文字串，指定用于此持久性事务中后继宏调用的 URL 中的事务句柄。
整数	<i>timeout</i>	IN	一个变量或文字串，以秒为单位指定服务于此端口的作业等待响应的的时间。此值将覆盖 DTW_STATIC() 函数中指定的任何超时值。

例

例 1:

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

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

DTW_COMMIT

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

将自上一次确认界限后对确认控制资源的全部更改变为永久的更改，然后建立一个新的确认界限。

格式

@DTW_COMMIT()

值

无。

例

例 1: 指定一个提交

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


DTW_ROLLBACK

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

重新建立最后一次确认界限，作为当前的确认界限。自上一次确认界限后正在运行的 Net.Data 进行对确认控制资源的全部更改将变为永久的更改。

格式

@DTW_ROLLBACK()

值

无。

例

例 1: 指定一个撤消

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

DTW_RTVHANDLE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

生成并返回一个此宏(跨几个分开的调用)所唯一的、并根据线程信息、时间戳记和当前用户(如果有的话)的组合计算而得的事务句柄。事务句柄可用于确保作为持久性事务指定的 URL 对于 HTTP 服务器是唯一的，并且可以安全地标识为有效的请求。

格式

@DTW_RTVHANDLE(handle)

值

表 114. DTW_RTVHANDLE 参数

数据类型	参数	用法	说明
字符串	handle	OUT	一个包含当前持久性宏的唯一事务句柄的变量。

例

例 1: 定义 handle 变量用于检索事务句柄

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

DTW_STATIC

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

表示整个宏是持久性的。它应是宏中的第一个语句。这个函数调用之后，除非另外指定，宏中定义的所有变量将跨越多个宏调用(持久性的)，直至调用 DTW_TERMINATE() 或进程结束。

函数调用时可以指定一个超时值(以秒为单位)，用以指出 Net.Data 进程等待浏览器响应的 时间。如果到达超时值，则进程终止，并且撤消自上一次确认界限后对确认控制资源所作的全部更改。

如果在后继的 @DTW_ACCEPT() 调用中指定了一个超时值，则 Net.Data 将用后继调用中的值覆盖此值。如果这个调用或后继 @DTW_ACCEPT() 调用中没有指定超时值，则将使用 Web 服务器的缺省超时值。

格式

```
@DTW_STATIC(timeout)
@DTW_STATIC()
```

值

表 115. DTW_STATIC 参数

数据类型	参数	用法	说明
整数	<i>timeout</i>	IN	一个变量或文字串，以秒为单位指定处理此事务的进程应等待响应的 时间。

例

```
例 1: 对 DTW_STATIC() 的调用指定超时值为 60 秒。
@DTW_STATIC("60")
```

DTW_TERMINATE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
				X			

目的

结束一个持久性的事务。自上一次确认界限后对确认控制资源的全部更改将变为永久的更改。

在任何文本输出到浏览器之前，将在持久性事务逻辑上最后一个 HTML 块的开始处调用 DTW_TERMINATE 函数。如果在函数前、块中出现任何文本输出，则将出现 Net.Data 错误。请注意，根据应用程序的编写，可能有多个逻辑上最后的 HTML 块。如果当宏处于持久性状态时调用此函数，则将出现 Net.Data 错误。

格式

@DTW_TERMINATE()

值

无

例

例 1: 终止持久性事务

```
%HTML(QUIT){  
@DTW_TERMINATE()  
...  
%}
```

附录A. DB2 WWW Connection

如果您已经有 DB2 WWW Connection, 则可以用 Net.Data 运行现有的应用程序。建议更新您的应用程序, 充分利用 Net.Data 版本 2 的功能特征。

DB2 WWW 语言结构:

- 『EXEC_SQL』
- 『HTML_INPUT』
- 『HTML_REPORT』
- 『SQL』
- 第236页的『SQL_MESSAGE』
- 第236页的『SQL_REPORT』
- 第236页的『SQL_CODE』

EXEC_SQL

此语言结构调用一个 SQL 块。但建议您把 SQL 语句当作函数来调用。请参阅第15页的『FUNCTION 块』, 以获取更多信息。

HTML_INPUT

此语言结构和称为 INPUT 的 HTML 块相同。请参阅第23页的『HTML 块』, 以获取更多信息。

HTML_REPORT

此语言结构和称为 REPORT 的 HTML 块相同。请参阅第23页的『HTML 块』, 以获取更多信息。

SQL

此语言结构等价于 Net.Data 中用 FUNCTION(DTW_SQL) 调用的一个函数。

它包含 SQL_REPORT 和 SQL_MESSAGE 语句, 这些语句也来自 DB2 WWW Connection。DB2 WWW Connection 不支持名为 %SQL 的模块。

例子:

例 1: 一个 DB2 WWW Connection 宏

```
%SQL{  
UPDATE $(dbtb1) SET URL='$(URL)' WHERE ID=$(ID)  
%SQL_MESSAGE{
```

```

100: "<B>The selected URL no longer exists in the table</B>." : continue
%}
%}

%HTML_INPUT{
<HTML>
...
%EXEC_SQL
</HTML>
%}

%HTML_REPORT{
<HTML>
...
</HTML>
%}

```

例 1: 一个等价的 Net.Data 宏

```

%FUNCTION(DTW_SQL) URLQuery(){
UPDATE $(dbtbl) SET URL='$(URL)' WHERE ID=$(ID)
%MESSAGE{
100: "<B>The selected URL no longer exists in the table</B>." : continue
%}
%}

%HTML(INPUT) {
<HTML>
...
@URLQuery
</HTML>
%}

%HTML(REPORT){
<HTML>
...
</HTML>
%}

```

SQL_MESSAGE

此语言结构等价于 Net.Data MESSAGE 语句。请参阅第39页的『MESSAGE 块』中的例子。

SQL_REPORT

此语言结构等价于 Net.Data REPORT 语句。请参阅第43页的『REPORT 块』中的例子。

SQL_CODE

此语言结构来自 DB2 WWW Connection, Net.Data 仍支持它以保持兼容性。等价于第102页的『RETURN_CODE』。

附录B. Net.Data 操作系统参考

并非每个操作系统都支持全部的 Net.Data 功能特征。本章节说明您的操作系统支持哪些功能特征。**X** 表示支持相应的功能。

这里列出的某些功能并不是普遍适用的。

表 116. Net.Data 语言环境

语言环境	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
缺省	X	X	X	X	X	X	X	X
平面文件接口	X	X	X	X	X	X	X	X
IMS Web	X			X				X
Java 小应用程序	X	X	X	X		X	X	X
Java 应用程序	X		X				X	X
ODBC	X	X	X	X		X	X	X
Oracle	X							X
Perl	X		X	X				X
REXX	X		X	X	X	X	X	X
SQL	X	X	X	X	X	X	X	X
Sybase	X							X
System	X	X	X	X	X	X	X	X
Web 注册表	X	X	X		X	X	X	X

表 117. Net.Data 存储过程数据类型

数据类型	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
BIGINT	X	X	X			X	X	X
BLOB	X	X	X			X	X	X
CHAR	X	X	X	X	X	X	X	X
CLOB	X	X	X			X	X	X
DATE	X	X	X		X	X	X	X
DBCLOB	X	X	X			X	X	X
DECIMAL	X	X	X	X	X	X	X	X
DOUBLE	X	X	X	X	X	X	X	X
DOUBLEPRECISION	X	X	X	X	X	X	X	X
FLOAT	X	X	X	X	X	X	X	X
INTEGER	X	X	X	X	X	X	X	X
GRAPHIC	X	X	X	X	X	X	X	X
LONGVARCHAR	X	X	X		X	X	X	X
LONGVARGRAPHIC	X	X	X		X	X	X	X
REAL	X	X	X		X	X	X	X
SMALLINT	X	X	X	X	X	X	X	X
TIME	X	X	X		X	X	X	X
TIMESTAMP	X	X	X		X	X	X	X

表 117. Net.Data 存储过程数据类型 (续)

数据类型	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
VARCHAR	X	X	X	X	X	X	X	X
VARGRAPHIC	X	X	X	X	X	X	X	X

表 118. Net.Data 配置变量

配置变量	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
CACHE_MACHINE	X							X
CACHE_PORT	X							X
DefaultDBCp				X				
DB2INSTANCE	X	X	X			X	X	X
DB2MSGs				X				
DB2PLAN				X				
DB2SSID				X				
DSNAOINI				X				
DTW_CM_PORT	X	X	X			X	X	X
DTW_INST_DIR	X	X	X			X	X	X
DTW_LOG_DIR	X	X	X			X	X	X
DTW_MBMODE	X	X	X	X		X	X	X
DTW_OPTIMIZE_MATH	X	X	X			X	X	X
DTW_REMOVE_WS				X				
DTW_SMTP_SERVER	X	X	X			X	X	X
DTW_SQL_ISOLATION					X			
DTW_SQL_NAMING_MODE					X			
DTWR_CLOSE_REGISTRIES					X			

表 119. Net.Data 变量

变量	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
ALIGN	X	X	X	X	X	X	X	X
DATABASE	X	X	X		X	X	X	X
DB_CASE	X	X	X	X	X	X	X	X
DB2PLAN				X				
DB2SSID				X				
DTW_APPLET_ALTTEXT	X	X	X	X		X	X	X
DTW_CURRENT_FILENAME	X	X	X	X	X	X	X	X
DTW_CURRENT_LAST_MODIFIED	X	X	X	X	X	X	X	X
DTW_DEFAULT_MESSAGE					X			
DTW_DEFAULT_REPORT	X	X	X	X	X	X	X	X
DTW_EDIT_CODES					X			
DTW_HTML_TABLE	X	X	X	X	X	X	X	X
DTW_LOG_LEVEL	X	X	X			X	X	X
DTW_MACRO_FILENAME	X	X	X	X	X	X	X	X
DTW_MACRO_LAST_MODIFIED	X	X	X	X	X	X	X	X

表 119. Net.Data 变量 (续)

变量	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_MBMODE	X	X	X	X		X	X	X
DTW_MP_PATH	X	X	X	X	X	X	X	X
DTW_MP_VERSION	X	X	X	X	X	X	X	X
DTW_PRINT_HEADER	X	X	X	X	X	X	X	X
DTW_REMOVE_WS	X	X	X	X	X	X	X	X
DTW_SAVE_TABLE_IN	X	X	X	X	X	X	X	X
DTW_SET_TOTAL_ROWS	X	X	X		X	X	X	X
LOCATION				X				
LOGIN	X	X	X		X	X	X	X
Nn	X	X	X	X	X	X	X	X
NLIST	X	X	X	X	X	X	X	X
NULL_RPT_FIELD					X			
NUM_COLUMNS	X	X	X	X	X	X	X	X
NUM_ROWS					X			
PASSWORD	X	X	X		X	X	X	X
RETURN_CODE	X	X	X	X	X	X	X	X
ROW_NUM	X	X	X	X	X	X	X	X
RPT_MAX_ROWS	X	X	X	X	X	X	X	X
SHOWSQL	X	X	X	X	X	X	X	X
SQL_CODE	X	X	X	X	X	X	X	X
SQL_STATE	X	X	X	X	X	X	X	X
START_ROW_NUM	X	X	X		X	X	X	X
TOTAL_ROWS	X	X	X		X	X	X	X
TRANSACTION_SCOPE	X	X	X	X	X	X	X	X
V_columnName	X	X	X	X	X	X	X	X
VLIST	X	X	X	X	X	X	X	X
Vn	X	X	X	X	X	X	X	X

表 120. Net.Data 函数

函数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_ACCEPT					X			
DTW_ADD	X	X	X	X	X	X	X	X
DTW_ADDQUOTE	X	X	X	X	X	X	X	X
DTW_ASSIGN	X	X	X	X	X	X	X	X
DTW_CACHE_PAGE	X							X
DTW_COMMIT					X			
DTW_CONCAT	X	X	X	X	X	X	X	X
DTW_DATE	X	X	X	X	X	X	X	X
DTW_DELSTR	X	X	X	X	X	X	X	X
DTW_DELWORD	X	X	X	X	X	X	X	X

表 120. Net.Data 函数 (续)

函数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_DIVIDE	X	X	X	X	X	X	X	X
DTW_DIVREM	X	X	X	X	X	X	X	X
DTW_EXIT	X	X	X		X	X	X	X
DTW_FORMAT	X	X	X	X	X	X	X	X
DTW_GETCOOKIE	X	X	X			X	X	X
DTW_GETENV	X	X	X	X	X	X	X	X
DTW_GETINIDATA	X	X	X	X	X	X	X	X
DTW_HTMLENCODE	X	X	X	X	X	X	X	X
DTW_INSERT	X	X	X	X	X	X	X	X
DTW_INTDIV	X	X	X	X	X	X	X	X
DTW_LASTPOS	X	X	X	X	X	X	X	X
DTW_LENGTH	X	X	X	X	X	X	X	X
DTW_LOWERCASE	X	X	X	X	X	X	X	X
DTW_MULTIPLY	X	X	X	X	X	X	X	X
DTW_POS	X	X	X	X	X	X	X	X
DTW_POWER	X	X	X	X	X	X	X	X
DTW_QHTMLENCODE	X	X	X	X	X	X	X	X
DTW_REVERSE	X	X	X	X	X	X	X	X
DTW_ROLLBACK					X			
DTW_RVTHANDLE					X			
DTW_SENMAIL	X	X	X			X	X	X
DTW_SETCOOKIE	X	X	X			X	X	X
DTW_SETENV	X	X	X	X	X	X	X	X
DTW_STATIC					X			
DTW_STRIP	X	X	X	X	X	X	X	X
DTW_SUBSTR	X	X	X	X	X	X	X	X
DTW_SUBTRACT	X	X	X	X	X	X	X	X
DTW_SUBWORD	X	X	X	X	X	X	X	X
DTW_TB_APPENDROW					X			
DTW_TB_COLS	X	X	X		X	X	X	X
DTW_TB_DELETEROW					X			
DTW_TB_DLIST	X	X	X	X	X	X	X	X
DTW_TB_DUMPV	X	X	X	X	X	X	X	X
DTW_TB_GETN	X	X	X		X	X	X	X
DTW_TB_GETV	X	X	X		X	X	X	X
DTW_TB_HTMLENCODE	X	X	X	X	X	X	X	X
DTW_TB_INPUT_CHECKBOX	X	X	X	X	X	X	X	X
DTW_TB_INPUT_RADIO	X	X	X	X	X	X	X	X
DTW_TB_INPUT_TEXT	X	X	X	X	X	X	X	X

表 120. Net.Data 函数 (续)

函数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_TB_INSERTCOL					X			
DTW_TB_INSERTROW					X			
DTW_TB_LIST	X	X	X	X	X	X	X	X
DTW_TB_MAXROWS					X			
DTW_TB_QUERYCOLNONJ					X			
DTW_TB_ROWS	X	X	X		X	X	X	X
DTW_TB_SELECT	X	X	X	X	X	X	X	X
DTW_TB_SETCOLS					X			
DTW_TB_SETN					X			
DTW_TB_SETV					X			
DTW_TB_TABLE	X	X	X	X	X	X	X	X
DTW_TB_TEXTAREA	X	X	X	X	X	X	X	X
DTW_TERMINATE					X			
DTW_TIME	X	X	X	X	X	X	X	X
DTW_TRANSLATE	X	X	X	X	X	X	X	X
DTW_UPPERCASE	X	X	X	X	X	X	X	X
DTW URLESCSEQ	X	X	X	X	X	X	X	X
DTW_WORD	X	X	X	X	X	X	X	X
DTW_WORDINDEX	X	X	X	X	X	X	X	X
DTW_WORDLENGTH	X	X	X	X	X	X	X	X
DTW_WORDPOS	X	X	X	X	X	X	X	X
DTW_WORDS	X	X	X	X	X	X	X	X
DTWF_APPEND	X	X	X	X	X	X	X	X
DTWF_CLOSE	X	X	X	X	X	X	X	X
DTWF_DELETE	X	X	X	X	X	X	X	X
DTWF_INSERT	X	X	X	X	X	X	X	X
DTWF_OPEN	X	X	X	X	X	X	X	X
DTWF_READ	X	X	X	X	X	X	X	X
DTWF_REMOVE	X	X	X	X	X	X	X	X
DTWF_SEARCH	X	X	X	X	X	X	X	X
DTWF_UPDATE	X	X	X	X	X	X	X	X
DTWF_WRITE	X	X	X	X	X	X	X	X
DTWR_ADDENTRY	X	X	X		X	X	X	X
DTWR_CLEARREG	X	X	X		X	X	X	X
DTWR_CLOSEREG					X			
DTWR_CREATEREG	X	X	X		X	X	X	X
DTWR_DELENTY	X	X	X		X	X	X	X
DTWR_DELREG	X	X	X		X	X	X	X
DTWR_LISTREG	X	X	X		X	X	X	X
DTWR_LISTSUB	X	X	X			X	X	X

表 120. Net.Data 函数 (续)

函数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTWR_OPENREG					X			
DTWR_RTVENTRY	X	X	X		X	X	X	X
DTWR_UPDATEENTRY	X	X	X		X	X	X	X

表 121. Net.Data 接口

接口类型	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
FastCGI	X							
CGI	X	X	X	X	X	X	X	X
Java Beans								X
Internet Connection API (ICAPI)	X		X	X				X
Internet Server API (ISAPI)								X
现场连接	X	X	X				X	X
Lotus Domino Go Web Server (GWAPI)	X		X	X				X
Netscape API (NSAPI)	X						X	X
Servlets	X							X

表 122. Net.Data 工具

工具	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
管理工具	X		X					X
NetObjects Fusion Plug-ins								X
Wizards	X	X	X			X	X	X

附录C. 注意事项

本信息是为在美国提供的产品和服务而开发的。IBM 在其它国家也许没有提供本文档中所讨论的产品、服务或功能部件。关于您所在区域目前可用的产品及服务的信息，请向当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来替代 IBM 产品、程序或服务。当然，评估和验证非 IBM 产品、程序或服务均由用户自行负责。

本文档的议题可能涉及 IBM 的某些专利或正在申请中的专利的应用。提供本文档并不表示允许您使用这些专利。您可以用书面形式将特许查询寄往：

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

关于双字节 (DBCS) 许可证查询的信息，请与您所在国家的 IBM 知识产权部分联系，或通过写信将查询邮寄至：

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于英国与其它当地法律不允许这种供应方式的国家：国际商用机器公司『按现在的样子』出版此书，不做任何明确或暗示的担保，包括但不限于可销售性或适用于特殊目的暗示担保。一些地区在某些事务中不允许放弃明确或暗示的担保，因此本条款可能不适合您。

本信息中可能有技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些信息将包含在本书新的版本中。IBM 可以在任何时间对本书中说明的产品或程序进行改进，而不必通知您。

已经获得这个程序许可证的用户，如果希望得到有关的更多信息，以允许：(i) 在独立创建的程序和其它程序(包括本程序)之间交换信息，以及 (ii) 相互使用已经交换的信息，请联络：

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

这些信息可以通过遵循相应的条款和条件来获取，在某些情况下，需支付一定的费用。

这些信息中描述的特许程序及其所有可用的特许资料，按 IBM 客户协议 (IBM Customer Agreement) 或任何等价的协议中的条款，由 IBM 提供。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版宣布或其它公众可用源得到。IBM 未测试这些产品，因此不能确认性能、兼容性或其它关于非 IBM 产品承诺等的准确度。有关非 IBM 产品功能方面的问题可向它们的供应商提出。

版权许可证:

本信息中包含用源语言编写的示例应用程序，它们说明了各种不同的操作平台上的程序设计技术。您可以为了开发、使用、市场营销或分发应用程序(这些应用程序遵守编写这些示例程序的操作平台的应用程序接口)的目的，以任何形式复制、修改和分发这些示例程序，不用向 IBM 付费。这些例子未经所有条件下的完整测试。因此，IBM 不能保证或暗示其可靠性、可用性或这些程序的功能。您可以为了开发、使用、市场营销或分发应用程序(这些应用程序遵守编写这些示例程序的操作平台的应用程序接口)的目的，以任何形式复制、修改和分发这些示例程序，不用向 IBM 付费。

商标

以下术语是 IBM 公司在美国或其他国家的注册商标:

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

以下术语是其它公司的商标:

Java 和 HotJava 是 SUN 公司的商标。

Microsoft、Windows、Windows NT® 和 Windows 95 标志都是 Microsoft 公司的注册商标。

UNIX 是在美国和其它国家的注册商标，许可权属于 X/Open 有限公司专有。

其它公司、产品和服务名称可能是其它公司的商标或服务标志。

词汇表

absolute path (绝对路径). 对象的全路径名。绝对路径名从最高一级开始, 或者说从“根”目录(由斜杠 (/) 或反斜杠 (\) 字符标识)开始。

API. Application programming interface 的缩写, 应用程序设计接口。Net.Data 支持三个专用的 API, 以改进在 CGI 处理上的性能。

applet (小应用程序). 包含在 HTML 页中的一段 Java 程序。在支持 Java 的浏览器(例如 Netscape) 中可以运行小应用程序, 它是在装入 HTML 页时装入的。

application programming interface (API, 应用程序设计接口). 由操作系统或可单独订购的特许应用程序提供的一个功能接口, 它允许以高级语言编写的应用程序可以使用特定于操作系统或特许程序的数据。Net.Data 支持以下这些专门的 Web 服务器 API, 这些 API 用于 CGI 进程中的改进性能: ICAP、GWA、ISAPI 和 NSAPI。

cache (高速缓存). 一部分包含最近访问过的数据的内存或磁盘空间, 是为了加快对相同数据的后继访问而设计的。高速缓存经常是用来对网络中可以访问的、频繁使用的数据保留一个本地副本。

caching (高速缓存). 将频繁使用的结果(来自对 Web 服务器的请求)存储在本地以备快速检索的过程, 直到刷新信息时为止。

Cache Manager (高速缓存管理器). 为一台机器管理高速缓存的程序。它可以管理多个高速缓存。

CGI. Common Gateway Interface 的缩写, 公共网关接口。

cliette. Net.Data 现场连接中一个长时间运行的进程, 为来自 Web 服务器的请求提供服务。连接管理器负责调度 cliette 进程, 使其为这些请求提供服务。

commitment control (确认控制). Net.Data 正在运行的进程内的边界创建, 其中对资源的操作是工作单元的一部分。

Common Gateway Interface (公共网关接口). Web 服务器将控制传递给一个应用程序以及接收回数据的一种标准方法。

Connection Manager (连接管理器). Net.Data 中的一个可执行文件 dtwcm, 用于支持“现场连接”。

cookie. 一个信息包, 由 HTTP 服务器发送给 Web 浏览器, 然后在浏览器每次访问该服务器时发回。Cookies 中可以包含服务器所选择的任意信息, 用于维持否则将没有状态的 HTTP 事务之间的状态。*计算的自由联机字典*

current working directory (当前工作目录). 进程的缺省目录, 从该目录分辨所有的相对路径名。

database (数据库). 表格的一个集合, 或表格空间和索引空间的一个集合。

database management system (DBMS, 数据库管理系统). 用于控制创建、组织和修改一个数据库, 并控制对其存储的数据进行访问的一个软件系统。

data type (数据类型). 列和字面量的属性。

DBMS. Database management system 的缩写, 数据库管理系统。

firewall (防火墙). 一台装有软件的计算机, 用于防止外部未经授权计算机侵入内部网络。

flat file interface (平面文件接口). 一系列 Net.Data 内部函数, 可让您在明文文件中读写数据。

HTML. Hypertext markup language 的缩写, 超文本标记语言。

HTTP. Hypertext transfer protocol 的缩写, 超文本传送协议。

hypertext markup language (超文本标记语言). 一种用于编写 Web 文档的标记语言。

hypertext transfer protocol (超文本传送协议). 一种在 Web 服务器和浏览器之间使用的通信协议。

ICAP. Internet Connection API 的缩写。

ICS. Internet Connection Server 的缩写。

ICSS. Internet Connection Secure Server 的缩写。

Internet. 国际公用 TCP/IP 计算机网络。

Internet Connection Server. IBM 公司的公开 Web 服务器。

Internet Connection Secure Server. IBM 公司的安全 Web 服务器。

Intranet. 在公司防火墙内部的 TCP/IP 网络。

ISAPI. Microsoft 公司的 Internet Server API。

Java. 一种独立于操作系统的面向对象的程序设计语言, 特别适用于 Internet 应用程序。

language environment (语言环境). 一个模块, 提供从 Net.Data 宏到外部数据源(例如 DB2 或诸如 Perl 等程序设计语言)的访问。有一些语言环境是与 Net.Data 一起提供的, 例如 REXX、Perl 和 Oracle。您还可以创建自己的语言环境。

Live Connection (现场连接). 一个 Net.Data 组件, 由连接管理器和多个 client 组成。现场连接用于管理数据库和 Java 虚拟机连接的再使用。

LOB. Large object 的缩写, 大型对象。

middleware (中件). 一种介于应用程序与网络之间的软件。它管理客户应用程序和服务器之间通过网络进行的交互。

NSAPI. Netscape API 的缩写。

null (空值). 表示信息异常的一个特殊值。

path (路径). 用于查找文件的搜索路径。

path name (路径名). 告诉系统如何找到一个对象。路径名的表示方法是: 目录名, 后面跟对象的名称。单独的目录和对象名之间用斜杠 (/) 或反斜杠 (\) 字符分隔。

Perl. 一种解释性编程语言。

persistence (持续性). 使指定的值在整个事务中得以保持的状态, 这里的事务可以跨越多个 Net.Data 调用。只有变量是可以持久性的。另外, 在完成一个显式的提交或撤消操作之前, 或者在事务结束之前, 对受确认控制影响的资源的操作将保持活动。

port (端口). 一个 16 位数, 用于在 TCP/IP 和高级协议或应用程序之间进行通信。

registry (注册表). 一个可以存储和检索字符串的“仓库”。

relative path name (相对路径名). 不以最高级目录(或“根”目录)开始的路径名。系统假定路径名从进程的当前工作目录开始。

TCP/IP. Transmission Control Protocol / Internet Protocol 的缩写, 传输控制协议/网际协议。

transaction (事务). 一个 Net.Data 调用。如果使用持久性的 Net.Data, 则一个事务可能跨越多个 Net.Data 调用。

Transmission Control Protocol / Internet Protocol (传输控制协议/网际协议). 一组通信协议, 同时支持局域网和广域网中的点对点连接功能。

URL. Uniform resource locator 的缩写, 统一资源定位器。

uniform resource locator (统一资源定位器). 一个用于命名 HTTP 服务器和(可选的)目录及文件名的地址, 例如: <http://www.software.ibm.com/data/net.data/index.html>。

unit of work (工作单位). 作为一个原子操作的可恢复操作序列。工作单位内的所有操作都可以完成(提交)或取消(撤消), 就如同它们是一个操作。只有那些对受确认控制影响的资源进行的操作才可以提交或撤消。

Web server (Web 服务器). 一台运行 HTTP 服务器软件(例如 Internet Connection) 的计算机。

索引

本索引按汉语拼音, 数字, 英文字母和特殊字符顺序排列。

[A]

安全性

口令 87

注册 ID 85

安全性建议, FFI_PATH 195

[B]

包含文件 30

报表

覆盖 Net.Data 缺省 69

格式化 43

报表变量

说明 67

ALIGN 68

DTW_DEFAULT_REPORT 69

DTW_HTML_TABLE 70

RPT_MAX_ROWS 71

START_ROW_NUM 72

本地 DB2 子系统, ID 78

变量

报表 67

表格 56, 57

环境 53

可执行 53

列表 55

条件 52

隐藏 54

语言环境 74

杂项 91

Net.Data, 概述 51

变量名 4

变量引用 4

标题 30

表格

HTML 中的结果 70

Net.Data, 指定行数 71

表格变量

例子 56

说明 56

表格处理变量

说明 57

指定 SQL 语言环境 82

NLIST 59

NUM_COLUMNS 60

NUM_ROWS 61

表格处理变量 (续)

Nn 58

ROW_NUM 62

TOTAL_ROWS 63

VLIST 65

Vn 66

V_columnName 64

表格函数

DTW_TB_APPENDROW 167

DTW_TB_COLS 168

DTW_TB_DELETEROW 169

DTW_TB_DLIST 170

DTW_TB_DUMPH 172

DTW_TB_DUMPV 173

DTW_TB_GETN 174

DTW_TB_GETV 175

DTW_TB_HTML_ENCODE 176

DTW_TB_INPUT_CHECKBOX 177

DTW_TB_INPUT_RADIO 178

DTW_TB_INPUT_TEXT 179

DTW_TB_INSERTCOL 180

DTW_TB_INSERTROW 181

DTW_TB_LIST 182

DTW_TB_MAXROWS 183

DTW_TB_QUERYCOLNONJ 184

DTW_TB_ROWS 185

DTW_TB_SELECT 186

DTW_TB_SETCOLS 187

DTW_TB_SETN 188

DTW_TB_SETV 189

DTW_TB_TABLE 190

DTW_TB_TEXTAREA 192

[C]

参数, 传送 19

操作系统参考 236

持久性宏函数

DTW_ACCEPT 227

DTW_COMMIT 228

DTW_ROLLBACK 229

DTW_RTVHANDLE 230

DTW_STATIC 231

DTW_TERMINATE 232

传送参数, 系统语言环境 19

传送值组 56

词汇表 244

错误处理 39

[D]

大小写, 指定给 SQL 命令 76

大写, 指定 76

当前目录, 平面文件 194

调用

函数 21

外部程序 13

调用 FFI 语言环境 193

定界的值字符串 55

定界符, FFI 语言环境

ASCIITEXT 195

DELIMITED 195

[F]

访问平面文件 193

[G]

滚动, “下一页”和“前一页”按钮 72

[H]

函数

表格 166

持久性的 226

传送值组 56

命名约定 103

平面文件接口(FFI) 193

数学 131

说明 103

一般 104

字 158

字符串 142

Web 注册表 214

函数调用

处理表格行 45

格式化输出 43

说明 21

语法 21

行长限制, 宏文件 3

宏文件

格式 2

公用的语法成份 4

行长限制 3

全局语法 1

声明部分 1

示例 2

停止处理 112

语言结构 1

HTML 部分 1

环境变量

例子 53

环境变量 (续)

说明 53

ENVVAR 语句 12

[J]

计划, 连接至 DB2 子系统 77

脚注 30

绝对路径, 平面文件 194

[K]

可执行变量

带参数 54

例子 54

说明 53

作为一个变量引用 54

[L]

连接至一个数据库, DATABASE 变量 75

连接至 DB2 子系统

位置 84

子系统 ID 78

DB2 计划 77

列表变量

例子 55

说明 55

值分隔符 56

列出定界字符串 55

[N]

内部函数 103

[P]

配置 FFI 语言环境 194

平面文件

安全性建议 195

定界符 195

定义 193

访问 193

访问的建议 194

绝对路径 194

配置规则 194

权限要求 195

数据源 193

锁定文件 196

位置

当前目录 194

FFI_PATH 194

平面文件 (续)
 与 FFI_PATH 匹配 194
 在当前目录中创建 194
平台支持参考 236

[Q]

权限要求, FFI_PATH 195

[R]

日期变量 91

[S]

上限 47
声明部分, 宏文件 1
释放文件, FFI 函数 196
数据库一致性, 事务处理范围 90
数学函数
 DTW_ADD 132
 DTW_DIVIDE 133
 DTW_DIVREM 134
 DTW_FORMAT 135
 DTW_INTDIV 138
 DTW_MULTIPLY 139
 DTW_POWER 140
 DTW_SUBTRACT 141
锁定文件, FFI 函数 196

[T]

替换文本, Web 浏览器 79
条件变量
 变量引用 52
 例子 55
 说明 52
 LIST 语句 52
条件字符串处理 25, 48

[W]

位置, 连接至 DB2 子系统 84
位置, 平面文件 194
文件位置变量 91

[X]

系统语言环境, 传送参数 19
限定数据库访问 85, 87
消息, 缺省文本 94
小写, 指定 76

性能, DTW_EXIT 112
循环 48

[Y]

一般函数 104
 DTW_ADDQUOTE 106
 DTW_CACHE_PAGE 108
 DTW_DATE 111
 DTW_EXIT 112
 DTW_GETCOOKIE 113
 DTW_GETENV 115
 DTW_GETINIDATA 116
 DTW_HTMLENCODER 117
 DTW_QHTMLENCODER 119
 DTW_SENDMAIL 120
 DTW_SETCOOKIE 123
 DTW_SETENV 126
 DTW_TIME 127
 DTW_URLESCSEQ 129

隐藏变量
 步骤 54
 例子, HTML 格式 54
 说明 54

隐藏变量名 54

语言环境变量
 说明 74
 DATABASE 75
 DB2PLAN 77
 DB2SSID 78
 DB_CASE 76
 DTW_APPLET_ALTTEXT 79
 DTW_EDIT_CODES 80
 DTW_MBMODE 81
 DTW_SAVE_TABLE_IN 82
 DTW_SET_TOTAL_ROWS 83
 LOCATION 84
 LOGIN 85
 NULL_RPT_FIELD 86
 PASSWORD 87
 SHOWSQL 88
 SQL_STATE 89
 TRANSACTION_SCOPE 90

语言结构
 变量名 4
 变量引用 4
 公用的语法成份 4
 函数调用 21
 宏文件
 说明 5
 语法 1
 字符串 4
 COMMENT 块 7

语言结构 (续)

- DB2 WWW Connection 235
- DEFINE 块或语句 9
- ENVVAR 语句 12
- EXEC 块或者语句 13
- FUNCTION 块 15
- HTML 块 23
- IF 块 25
- INCLUDE 语句 30
- INCLUDE_URL 语句 32
- LIST 语句 34
- MACRO_FUNCTION 块 36
- MESSAGE 块 39
- REPORT 块 43
- ROW 块 45
- TABLE 语句 47
- WHILE 块 48

远程 DB2 子系统, 位置 84

[Z]

杂项变量

- 说明 91
- DTW_CURRENT_FILENAME 92
- DTW_CURRENT_LAST_MODIFIED 93
- DTW_DEFAULT_MESSAGE 94
- DTW_MACRO_LAST_MODIFIED 97
- DTW_MP_PATH 98
- DTW_MP_VERSION 99
- DTW_PRINT_HEADER 100
- DTW_REMOVE_WS 101
- RETURN_CODE 102

支持的功能部件表格 236

注意事项 243

子系统 ID, 连接至 DB2 子系统 78

字符串函数

- DTW_DELWORD 159
- DTW_SUBWORD 160
- DTW_WORD 161
- DTW_WORDINDEX 162
- DTW_WORDLENGTH 163
- DTW_WORDPOS 164
- DTW_WORDS 165
- MBCS 支持 158

字符串

- 数值型比较 25, 48
- 说明 4
- 条件处理 25, 48
- 值, 定界 55

字符串的数值型比较 25, 48

字符串函数

- DTW_ASSIGN 143
- DTW_CONCAT 144

字符串函数 (续)

- DTW_DELSTR 145
- DTW_INSERT 146
- DTW_LASTPOS 148
- DTW_LENGTH 149
- DTW_LOWERCASE 150
- DTW_POS 151
- DTW_REVERSE 152
- DTW_STRIP 153
- DTW_SUBSTR 154
- DTW_TRANSLATE 155
- DTW_UPPERCASE 157
- MBCS 支持 142
- “前一页”, RPT_MAX_ROWS 72
- “下一页”按钮, RPT_MAX_ROWS 72

A

ALIGN 68

APPLET 标记, 替换文本 79

C

COMMENT 块

- 说明 7
- 语法 7

cookie

- 发送 100
- DTW_GETCOOKIE 113
- DTW_PRINT_HEADER 100
- DTW_SETCOOKIE 123

D

DATABASE 75

DB2 WWW Connection, 语言结构 235

DB2PLAN 77

DB2SSID 78

DB_CASE 76

DEFINE 块

- 说明 9
- 语法 9

DEFINE 语句

- 说明 9
- 语法 9

DTWF_APPEND 196

DTWF_CLOSE 196, 199

DTWF_DELETE 200

DTWF_INSERT 202

DTWF_OPEN 196, 204

DTWF_READ 205

DTWF_REMOVE 207

DTWF_SEARCH	208	DTW_PRINT_HEADER	100
DTWF_UPDATE	210	DTW_QHTMLENCODE	119
DTWF_WRITE	212	DTW_REMOVE_WS	101
DTWR_ADDENTRY	214	DTW_REVERSE	152
DTWR_CLEARREG	216	DTW_ROLLBACK	229
DTWR_CLOSEREG	217	DTW_RTVHANDLE	230
DTWR_CREATEREG	218	DTW_SAVE_TABLE_IN	82
DTWR_DELENTY	219	DTW_SENMAIL	120
DTWR_DELREG	220	DTW_SETCOOKIE	123
DTWR_LISTREG	221	DTW_SETENV	126
DTWR_LISTSUB	222	DTW_SET_TOTAL_ROWS	83
DTWR_OPENREG	223	DTW_STATIC	231
DTWR_RTVENTRY	224	DTW_STRIP	153
DTWR_UPDATEENTRY	225	DTW_SUBSTR	154
DTW_ACCEPT	227	DTW_SUBTRACT	141
DTW_ADD	131	DTW_SUBWORD	160
DTW_ADDQUOTE	106	DTW_TB_APPENDROW	167
DTW_APPLET_ALTTEXT	79	DTW_TB_COLS	168
DTW_ASSIGN	58, 142, 143	DTW_TB_DELETEROW	169
DTW_CACHE_PAGE	108	DTW_TB_DLIST	170
DTW_COMMIT	228	DTW_TB_DUMP	172
DTW_CONCAT	144	DTW_TB_DUMPV	173
DTW_CURRENT_FILENAME	92	DTW_TB_GETN	174
DTW_CURRENT_LAST_MODIFIED	93	DTW_TB_GETV	175
DTW_DATE	111	DTW_TB_HTMLLENODE	176
DTW_DEFAULT_MESSAGE	94	DTW_TB_INPUT_CHECKBOX	177
DTW_DEFAULT_REPORT	69	DTW_TB_INPUT_RADIO	178
DTW_DELSTR	145	DTW_TB_INPUT_TEXT	179
DTW_DELWORD	158	DTW_TB_INSERTCOL	180
DTW_DIVIDE	133	DTW_TB_INSERTROW	181
DTW_DIVREM	134	DTW_TB_LIST	180
DTW_EDIT_CODES	80	DTW_TB_MAXROWS	183
DTW_FORMAT	135	DTW_TB_QUERYCOLNONJ	184
DTW_GETCOOKIE	113	DTW_TB_ROWS	185
DTW_GETENV	115	DTW_TB_SELECT	186
DTW_GETINIDATA	116	DTW_TB_SETCOLS	187
DTW_HTMLLENODE	117	DTW_TB_SETN	188
DTW_HTML_TABLE	70	DTW_TB_SETV	189
DTW_INSERT	146	DTW_TB_TABLE	190
DTW_INTDIV	138	DTW_TB_TEXTAREA	192
DTW_LASTPOS	148	DTW_TERMINATE	232
DTW_LENGTH	149	DTW_TIME	127
DTW_LOG_LEVEL	95	DTW_TRANSLATE	155
DTW_LOWERCASE	150	DTW_UPPERCASE	157
DTW_MACRO_FILENAME	96	DTW_URLESCSEQ	129
DTW_MACRO_LAST_MODIFIED	97	DTW_WORD	161
DTW_MBMODE	81	DTW_WORDINDEX	162
DTW_MP_PATH	98	DTW_WORDLENGTH	163
DTW_MP_VERSION	99	DTW_WORDPOS	164
DTW_MULTIPLY	139	DTW_WORDS	165
DTW_POS	151		
DTW_POWER	140		

E

ENVVAR 语句 53
 说明 12
 语法 12
EXEC 块
 说明 13
 语法 13
EXEC 语句 53
 说明 13
 语法 13
EXEC_PATH 13
EXEC_SQL 235

F

FFI 函数
 释放文件 196
 锁定文件 196
 DTWF_APPEND 197
 DTWF_CLOSE 199
 DTWF_DELETE 200
 DTWF_INSERT 202
 DTWF_OPEN 204
 DTWF_READ 205
 DTWF_REMOVE 207
 DTWF_SEARCH 208
 DTWF_UPDATE 210
 DTWF_WRITE 212
FFI 语言环境
 安全性建议 195
 当前目录 194
 定界符 195
 访问文件 193
 配置规则 194
 权限要求 195
 文件位置 194
FFI_PATH
 安全性建议 195
 访问平面文件 193
 配置规则 194
 平面文件位置 194
 与 *filename* 参数匹配的路径 194
 语法 193
FUNCTION 块
 说明 15
 语法 16

H

HTML
 格式, 输入口令 87
 格式, 输入用户标识符 85

HTML (续)
 显示表格结果 70
 隐藏变量名 54
HTML 部分, 宏文件 1
HTML 块
 说明 23
 语法 23
HTML_INPUT 模块 235
HTML_REPORT 模块 235

I

IF 块
 说明 25
 语法 25
IN 关键字 16, 36, 103
INCLUDE 语句
 说明 30
 语法 30
INCLUDE_PATH 30
INCLUDE_URL 语句
 说明 32
 语法 32
INOUT 关键字 16, 36, 103

L

LIST 语句
 说明 34
 语法 34
LOCATION 84
LOGIN 85

M

MACRO_FUNCTION 块
 说明 36
 语法 36
MBCS 对函数的支持
 字处理函数 158
 字符串函数 142
MESSAGE 块
 说明 39
 语法 39

N

Net.Data 表格
 定义 47
 上限 47
NLIST 59
NULL_RPT_FIELD 86

NUM_COLUMNS 60
NUM_ROWS 61
Nn 58

O

OUT 关键字 16, 36, 103

P

PASSWORD 87

R

REPORT 块
 表格变量 56
 说明 43
 语法 43
 ALIGN 68
 DTW_DEFAULT_REPORT 69
 DTW_HTML_TABLE 70
 NLIST 59
 NUM_COLUMNS 60
 NUM_ROWS 61
 Nn 58
 RPT_MAX_ROWS 71
 START_ROW_NUM 72
 TOTAL_ROWS 63
RETURNS 关键字 17
RETURN_CODE 102
ROW 块
 说明 45
 语法 45
 NLIST 59
 NUM_COLUMNS 60
 NUM_ROWS 61
 Nn 58
 ROW_NUM 62
 TOTAL_ROWS 63
 Vn 65, 66
 V_columnName 64
ROW_NUM 62
RPT_MAX_ROWS 71

S

SHOWSQL 88
SQL
 命令, 指定大小写 76
 隐藏或者显示 88
SQL 模块 235
SQL state, 显示 89

SQL_CODE 236
SQL_MESSAGE 模块 236
SQL_REPORT 模块 236
SQL_STATE 89
START_ROW_NUM 72

T

TABLE 语句 56
 说明 47
 语法 47
TOTAL_ROWS 63
TRANSACTION_SCOPE 90

V

VLIST 65
Vn 66
V_columnName 64

W

Web 注册表函数
 DTWR_ADDENTRY 215
 DTWR_CLEARREG 216
 DTWR_CLOSEREG 217
 DTWR_CREATEREG 218
 DTWR_DELENTY 219
 DTWR_DELREG 220
 DTWR_LISTREG 221
 DTWR_LISTSUB 222
 DTWR_OPENREG 223
 DTWR_RTVENTRY 224
 DTWR_UPDATEENTRY 225
WHILE 块 48
 说明 48
 语法 48



Printed in China