



Net.Data 참조서

주의

이 정보와 지원하는 제품을 사용하기 전에 241페이지의 『부록C. 주의사항』의 정보를 읽어 보십시오.

서문	vii
Net.Data 정보	vii
이 책에 대하여	vii
이 책의 독자	viii
이 책의 예제에 대한 정보	viii
구문 도표를 읽는 방법	viii
제1장 Net.Data 매크로 언어 구성	1
Net.Data 매크로 파일 구문	1
공통 구문 요소	4
변수명	4
변수 참조	4
문자열	5
매크로 언어 구성	5
주석 블록	7
DEFINE 블록 또는 명령문	9
ENVVAR 명령문	13
EXEC 블록 또는 명령문	14
FUNCTION 블록	16
함수 호출 (@).	23
HTML 블록	26
IF 블록	29
INCLUDE 명령문	38
INCLUDE_URL 명령문	40
LIST 명령문	42
MACRO_FUNCTION 블록	44
MESSAGE 블록	48
REPORT 블록	53
ROW 블록	56
TABLE 명령문	59
WHILE 블록	61
제2장 변수	65
사용자 정의 변수	66
조건 변수	66
환경 변수	68
실행 변수	68
숨겨진 변수	69
List 변수	70
Table 변수	71
Net.Data 기타 변수	72
DTW_CURRENT_FILENAME	73
DTW_CURRENT_LAST_MODIFIED	74
DTW_DEFAULT_MESSAGE	75
DTW_LOG_LEVEL	76
DTW_MACRO_FILENAME	77
DTW_MACRO_LAST_MODIFIED	78
DTW_MP_PATH	79
DTW_MP_VERSION	80

DTW_PRINT_HEADER.	81
DTW_REMOVE_WS.	82
RETURN_CODE	83
Net.Data 테이블 처리 변수	84
Nn	85
NLIST.	86
NUM_COLUMNS.	87
NUM_ROWS	88
ROW_NUM	89
TOTAL_ROWS	90
V_columnName.	91
VLIST.	92
Vn	93
Net.Data 보고서 변수	94
ALIGN	95
DTW_DEFAULT_REPORT	96
DTW_HTML_TABLE	97
RPT_MAX_ROWS	98
START_ROW_NUM.	99
Net.Data 언어 환경 변수	102
DATABASE.	103
DB_CASE	105
DB2PLAN	106
DB2SSID.	107
DTW_APPLET_ALTTEXT.	108
DTW_EDIT_CODES.	109
DTW_MBMODE	110
DTW_SAVE_TABLE_IN	111
DTW_SET_TOTAL_ROWS	112
LOCATION	114
LOGIN	115
NULL_REPORT_FIELD	116
PASSWORD.	117
SHOWSQL	118
SQL_STATE	119
TRANSACTION_SCOPE	120
제3장 Net.Data 내장 함수	123
함수명.	123
입출력 매개변수	123
함수 결과 형식 지정	124
함수 매개변수 규칙.	124
일반 함수	125
DTW_ADDQUOTE	126
DTW_CACHE_PAGE	128
DTW_DATE.	132
DTW_EXIT	134
DTW_GETENV	135
DTW_GETINIDATA.	136
DTW_HTMLLENCODE	137
DTW_QHTMLLENCODE	139
DTW_SETENV.	140

DTW_TIME	141
DTW_URLESCSEQ	143
수학 함수	145
DTW_ADD	146
DTW_DIVIDE	147
DTW_DIVREM	148
DTW_FORMAT	150
DTW_INTDIV	153
DTW_MULTIPLY	154
DTW_POWER	155
DTW_SUBTRACT	156
문자열 함수	158
DTW_ASSIGN	159
DTW_CONCAT	160
DTW_DELSTR	161
DTW_INSERT	162
DTW_LASTPOS	164
DTW_LENGTH	165
DTW_LOWERCASE	166
DTW_POS	167
DTW_REVERSE	168
DTW_STRIP	169
DTW_SUBSTR	170
DTW_TRANSLATE	172
DTW_UPPERCASE	174
단어 함수	175
DTW_DELWORD	176
DTW_SUBWORD	177
DTW_WORD	179
DTW_WORDINDEX	180
DTW_WORDLENGTH	181
DTW_WORDPOS	182
DTW_WORDS	184
테이블 함수	185
DTW_TB_DLIST	186
DTW_TB_DUMPH	188
DTW_TB_DUMPV	189
DTW_TB_HTMLENCOD	190
DTW_TB_INPUT_CHECKBOX	191
DTW_TB_INPUT_RADIO	192
DTW_TB_INPUT_TEXT	193
DTW_TB_LIST	195
DTW_TB_SELECT	197
DTW_TB_TABLE	198
DTW_TB_TEXTAREA	200
플랫 파일 인터페이스 함수	201
플랫 파일 인터페이스 분리문자	201
플랫 파일 인터페이스 함수	201
DTWF_APPEND	203
DTWF_CLOSE	205
DTWF_DELETE	206
DTWF_INSERT	208

DTWF_OPEN	210
DTWF_READ	212
DTWF_REMOVE	214
DTWF_SEARCH	215
DTWF_UPDATE	217
DTWF_WRITE	219
웹 레지스트리 함수	221
DTWR_ADDENTRY	222
DTWR_CLEARREG	223
DTWR_CREATEREG	224
DTWR_DELENTY	225
DTWR_DELREG	226
DTWR_LISTREG	227
DTWR_LISTSUB	228
DTWR_RTVENTRY	229
DTWR_UPDATEENTRY	230
 부록A. DB2 WWW 연결	 231
EXEC_SQL	231
HTML_INPUT	231
HTML_REPORT	231
SQL	231
SQL_MESSAGE	232
SQL_REPORT	232
SQL_CODE	233
 부록B. Net.Data 운용 시스템 참조	 235
 부록C. 주의사항	 241
등록상표	242
 용어	 243
 색인	 245

서문

동적 웹 페이지를 작성하기 위해 IBM에서 개발한 툴인 Net.Data 버전 2를 선택해 주셔서 감사합니다. Net.Data를 사용하면 다양한 자료 소스의 자료를 통합하고 이미 알고 있는 프로그래밍 언어의 장점을 활용하여 동적 내용을 통해 웹 페이지를 빠르게 개발할 수 있습니다.

Net.Data 버전 2는 인터넷 업무 솔루션을 구축하고 설치하는 능력을 제공하는 새로운 기능과 함께 크게 향상된 성능을 제공합니다.

Net.Data 정보

IBM의 Net.Data 제품을 사용하면 DB2, IMS 및 ODBC 작동 가능 데이터베이스를 포함하는 관계형 및 비관계형 데이터베이스 관리 시스템(DBMS) 양쪽의 자료를 사용하고 프로그래밍 언어(예: Java, JavaScript, Perl, C, C++ 및 REXX)로 작성된 응용 프로그램을 사용하여 동적 웹 페이지를 작성할 수 있습니다.

Net.Data를 웹 서버 상의 미들웨어로 실행되는 매크로 프로세서로 간주할 수 있습니다. Net.Data가 사용자의 입력, 데이터베이스의 현재 상태, 기존 업무 로직 및 매크로로 디자인한 기타 요소를 기초로 하여 사용자 정의한 내용으로 동적 웹 페이지를 작성하기 위해 해석하는 매크로라는 Net.Data 응용 프로그램을 작성할 수 있습니다.

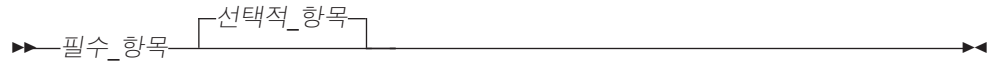
URL(uniform resource locator) 형태의 요청은 브라우저(예: Netscape 또는 Internet Explorer)로부터 실행을 위해 Net.Data로 요청을 전송하는 웹 서버로 전달됩니다. Net.Data는 매크로를 찾아 실행하고 사용자가 작성한 함수를 기초로 사용자 정의한 웹 페이지를 구축합니다. 이들 함수는 다음을 수행할 수 있습니다.

- Perl 스크립트, C 및 C++ 응용 프로그램 또는 REXX 프로그램 내에 업무 로직을 캡슐화합니다.
- DB2와 같은 데이터베이스에 액세스합니다.

Net.Data는 산업 표준 게이트웨이(예: HTTP(HyperText Transfer Protocol) 및 CGI(Common Gateway Interface). HTTP는 브라우저와 웹 서버 사이에서 사용되며 CGI는 웹 서버와 Net.Data 사이에서 사용됩니다. 따라서 Net.Data와 함께 사용하기 위해 선호하는 브라우저나 웹 서버를 선택할 수 있습니다. 또한 Net.Data는 여러 운영체제에서 FastCGI와 주요 웹 서버 API를 지원합니다.

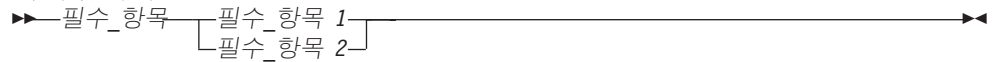
이 책에 대하여

이 책에서는 Net.Data 언어 구성, 변수 및 일반적인 함수의 구문과 사용법에 대해 설명합니다.



- 2개 이상의 항목을 선택할 수 있는 경우에는 항목이 수직으로 스택을 이루어 표시됩니다.

항목 중 하나를 선택해야 하는 경우라면 주 경로에 스택 항목 중 하나가 나타납니다.



항목 중에 하나를 선택하는 것이 선택적이면, 스택 전체가 주 경로 아래에 표시됩니다.



항목 중의 하나가 생략시 값이면, 이 항목은 주 경로의 위에 나타나고 나머지는 아래에 표시됩니다.



- 주 경로위의 왼쪽으로 되돌아 오는 화살표는 반복될 수 있는 항목을 나타냅니다.



반복 화살표내에 구두점이 들어 있으면, 반복되는 항목을 지정된 구두점으로 분리해야 합니다.



스택 위의 반복 화살표는 스택 내에서 항목을 반복할 수 있음을 나타냅니다.

- 키워드는 대문자로 표시됩니다(예: FROM). Net.Data에서, 키워드는 대소문자 모두를 사용할 수 있습니다. 키워드가 아닌 용어는 소문자로 표시됩니다(예: column-name). 이는 사용자가 제공한 이름이나 값을 나타냅니다.
- 구두점, 괄호, 산술 연산자 또는 그밖의 기호가 표시될 경우에는 이를 구문의 일부로 입력하십시오.

제1장 Net.Data 매크로 언어 구성

이 장에서는 Net.Data 매크로 파일에서 사용되는 Net.Data 매크로 구문과 언어 구성에 대해 설명합니다. 언어 구성은 키워드 및 명령문, Net.Data내의 블록으로 구성되며, 다른 변수 유형을 지정하고 파일 포함(include)과 같은 특수 작업을 수행합니다.

이 장에서는 다음에 대해 설명합니다.

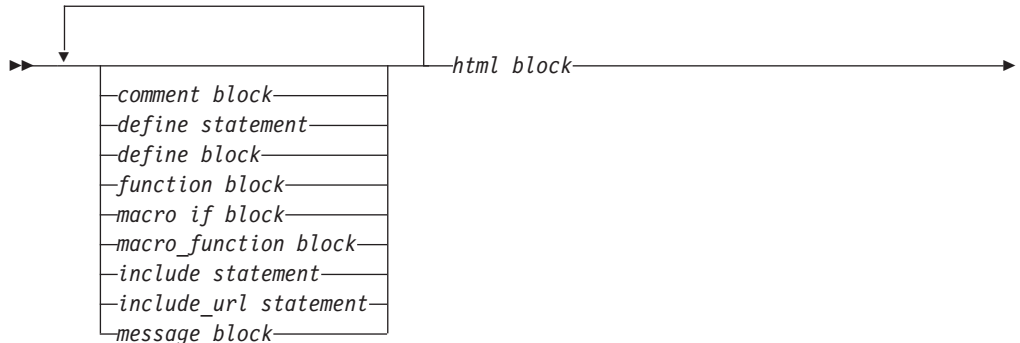
- 1페이지의 『Net.Data 매크로 파일 구문』
- 4페이지의 『공통 구문 요소』
- 5페이지의 『매크로 언어 구성』

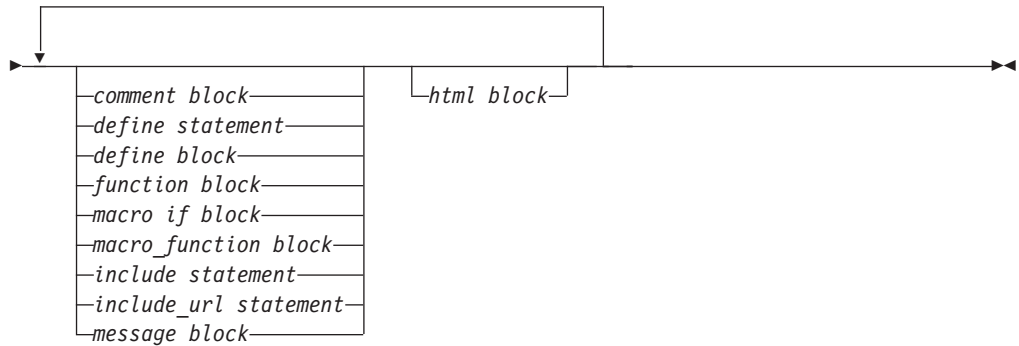
Net.Data 매크로 파일 구문

Net.Data 매크로는 다음을 구성하는 일련의 Net.Data 매크로 언어를 구성하는 일반적인 텍스트 파일입니다.

- 웹 페이지의 배치를 지정합니다.
- 변수 및 함수를 정의합니다.
- 매크로 파일에 정의되어 있으며 Net.Data가 처리를 위해 언어 환경에 제공하는 함수를 호출합니다.
- 처리 출력 형식을 HTML로 지정하고 이를 웹 브라우저로 리턴합니다.

각 명령문은 하나 이상의 언어 구성으로 이루어지며, 이는 키워드, 특수 문자, 문자열, 이름 및 변수의 차례로 구성됩니다. 다음 도표는 문법적으로 유효한 Net.Data 매크로의 전체 구조를 설명한 것입니다. 전역 구조에서 각 요소에 대한 자세한 구문을 보려면 1페이지의 『제1장 Net.Data 매크로 언어 구성』을 참고하십시오.





Net.Data 매크로에는 설명 부분과 HTML 부분의 두 부분이 들어 있습니다. 이 두 부분을 임의의 순서대로 반복해서 사용할 수 있습니다.

- 선언 부분에는 매크로 파일의 변수 및 함수 정의가 들어 있습니다.
- HTML 부분에는 웹 페이지의 배치를 지정하는 HTML문을 포함하는 HTML 블록이 들어 있습니다. 이 부분에는 보고서 절이 포함됩니다.

2페이지의 그림 1에는 매크로 파일의 선언과 HTML 부분이 나옵니다.

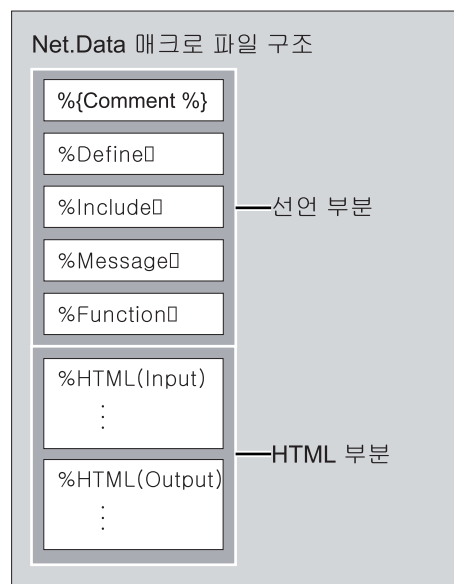


그림 1. 매크로 파일 구조

선언 또는 HTML 부분에 사용되는 변수 및 함수는 변수 참조 또는 함수 호출에서 사용하기 전에 먼저 정의해야 합니다.

3페이지의 그림 2에서는 매크로 파일의 여러 부분들에 대한 예를 제공합니다. 선언 부분에는 DEFINE 및 FUNCTION 정의 블록이 포함됩니다. HTML 블록은 입출력 블록으로 작동합니다.

```

%{ ***** 정의 블록 *****%}
%DEFINE {
    page_title="Net.Data 매크로 템플릿"
%}

%{ ***** 함수 정의 블록 *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
    { %EXEC{ompsamp.cmd %}
%}

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

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

<FORM METHOD="post" ACTION="output">
Type some data to pass to a REXX program:
<INPUT NAME="input_data" TYPE="text" SIZE="30">
<p>
<INPUT TYPE="submit" VALUE="Enter">

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

%{ ***** HTML 블록: 출력 *****%}
%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>
%}

```

그림 2. 매크로 파일 템플릿 형식

Net.Data 매크로 언어는 형식이 정해져 있지 않은 언어로, 매크로 작성시 유연성을 부여합니다. 특별히 명시되지 않은 한, 기타 공백 문자는 무시됩니다. 각 Net.Data 매크로 언어 구성은 구성을 정의하는 데 사용되는 여러 가지 기타 요소와 함께 다음 절에서 설명됩니다. Net.Data 매크로 언어는 이전 버전과의 호환을 위해 DB2 WWW 연결 언어 요소를 지원합니다. 이러

한 언어 요소가 231페이지의 『부록A. DB2 WWW 연결』에서 다루어지고
있어도 Net.Data 언어 구성을 사용하는 것이 좋습니다.

예제에 언어 구성, 변수, 함수 및 매크로 파일내의 기타 요소를 사용할 수
있는 몇가지 방법이 나와 있습니다. Net.Data 웹 페이지에서 좀더 자세한 예
제 및 데모를 다운로드할 수 있습니다.

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

공통 구문 요소

다음 구문 요소는 언어 구성 설명에 자주 사용됩니다.

- 4페이지의 『변수명』
- 4페이지의 『변수 참조』
- 5페이지의 『문자열』

변수명

목적:

하나 이상의 이름을 나타내며 각 이름은 점(.)으로 연결됩니다. 이름은 영문
자나 밑줄로 시작하는 영숫자 문자열이며, 영문자, 숫자 또는 밑줄 등을 임
의로 조합하여 사용할 수 있습니다.

인용 부호(『』)로 묶인 문자열에는 개행 문자를 제외한 모든 문자가 포함
될 수 있습니다. 문자열이 괄호({ %})로 묶여 있으면 개행 문자를 포함한
문자가 포함될 수 있습니다.

변수명은 문자나 밑줄(_)로 시작해야 하며, 모든 영숫자 문자 및 밑줄을 포
함할 수 있습니다. 모든 변수명은 *N_columnName* 과 *V_columnName*을 제외
하고 대소문자를 구분합니다(이러한 두 가지 예외에 대한 자세한 내용은 84
페이지의 『Net.Data 테이블 처리 변수』를 참고하십시오.).

구문:



변수 참조

목적:

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if
VAR = 'abc', \$(VAR) returns the value 'abc'. 변수 참조는 수행시에 값이 평

가됩니다. 변수가 EXEC문이나 블록에 대해 정의되어 있으면 Net.Data는 변수 참조를 읽을 때 지정된 조치를 수행합니다.

참조되는 변수는 참조되기 전에 Net.Data 매크로에 정의되어 있어야 합니다. 변수가 정의되어 있지 않으면, 빈 문자열이 리턴됩니다.

구문:

▶▶\$—(—variable_name—)————▶▶

문자열

연속된 영문자와 숫자 및 구두점. 문자열이 큰 따옴표로 묶여 있으면 개행 문자는 허용되지 않습니다. 언어 구성에서 사용될 때 제한 사항에 대해서는 각 언어 구성의 문자열 매개변수 설명을 참고하십시오.

HTML 명령문은 문자열입니다.

매크로 언어 구성

이 절에서는 Net.Data 매크로 파일에서 사용되는 언어 구성에 대해 설명합니다.

각 언어 구성 설명에는 다음 정보가 들어 있습니다.

목적 Net.Data 매크로에 해당 언어 구성을 사용하는 이유를 정의합니다.

구문 언어 구성의 논리 구조의 도표를 제공합니다.

매개변수

구문 도표의 모든 요소를 정의하고 다른 언어 구성 구문 및 예제로의 상호 참조를 제공합니다.

문맥 Net.Data 매크로 구조에서 언어 구성이 사용될 수 있는 위치에 대해 설명합니다.

제한사항

어떤 요소를 포함할 수 있는지를 정의하고 모든 사용상의 제한사항을 지정합니다.

예제 Net.Data 매크로내에서 키워드 명령문 및 블록 사용에 대한 간단한 예제 및 설명을 제공합니다.

다음 구성이 매크로에서 사용됩니다. 구문 및 예제는 각 구성 설명을 참고하십시오.

- 7페이지의 『주석 블록』
- 9페이지의 『DEFINE 블록 또는 명령문』
- 13페이지의 『ENVVAR 명령문』
- 14페이지의 『EXEC 블록 또는 명령문』

- 16페이지의 『FUNCTION 블록』
- 23페이지의 『함수 호출 (@)』
- 26페이지의 『HTML 블록』
- 29페이지의 『IF 블록』
- 38페이지의 『INCLUDE 명령문』
- 40페이지의 『INCLUDE_URL 명령문』
- 42페이지의 『LIST 명령문』
- 44페이지의 『MACRO_FUNCTION 블록』
- 48페이지의 『MESSAGE 블록』
- 53페이지의 『REPORT 블록』
- 56페이지의 『ROW 블록』
- 59페이지의 『TABLE 명령문』
- 61페이지의 『WHILE 블록』

1
1
1

Net.Data 매크로 함수를 문서로 작성합니다. COMMENT 블록을 매크로 파일의 모든 위치에서 사용할 수 있으므로 다른 구문 도표에 문서로 나타나지는 않습니다.

►► `%{text%}` ◄◄

text 한 행 이상의 문자열. Net.Data는 모든 주석의 내용을 무시합니다.

11

주석은 Net.Data 매크로의 Net.Data 언어 구성 사이의 어느 위치에도 놓일 수 있습니다.

모든 텍스트나 문자가 허용되나 주석 블록은 중첩될 수 없습니다.

예제 1: 기본 주석 블록

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

—

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

—

```
%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>
%}

```

예제 4: DEFINE 블록의 주석

```

%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 블록 또는 명령문

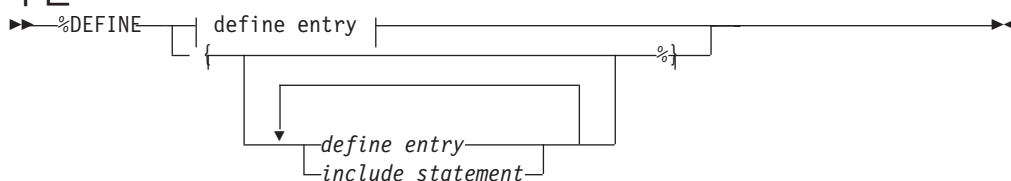
목적

DEFINE 절은 매크로의 선언 부분에 변수명을 정의하며 명령문이나 블록이 될 수 있습니다.

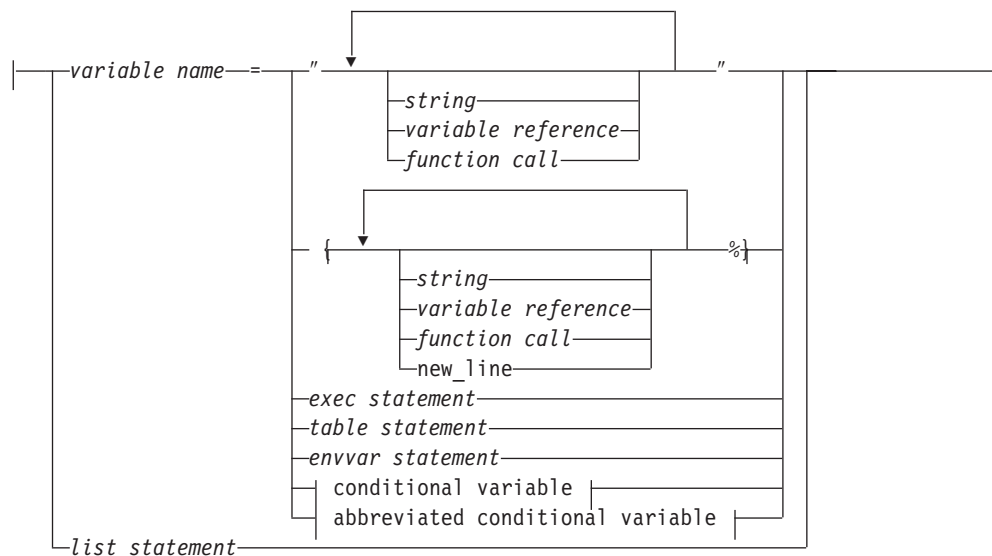
- 한번에 한 변수를 정의하려면 명령문을 사용하십시오.
- 여러 변수를 정의하려면 블록을 사용하십시오.

변수는 큰 따옴표(" ")를 사용하여 한 행에 정의할 수도 있고, 중괄호와 퍼센트 기호({ %})를 사용하여 여러 줄에 정의할 수도 있습니다. 변수가 정의 되면 매크로의 어떤 위치에서도 이를 참조할 수 있습니다.

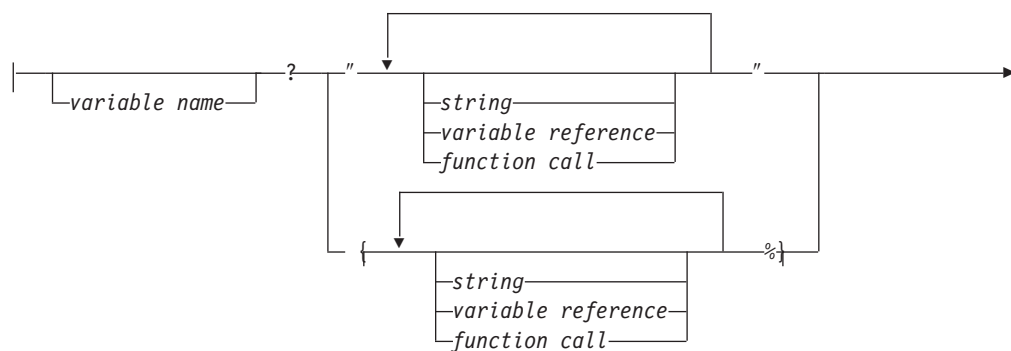
구문

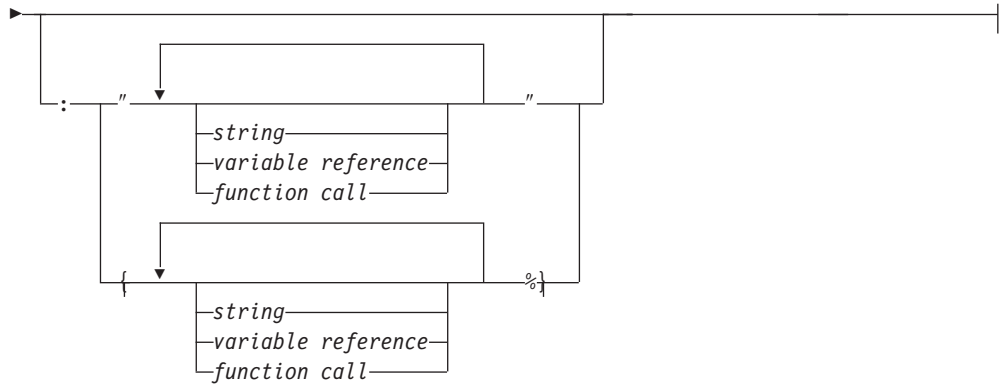


define entry

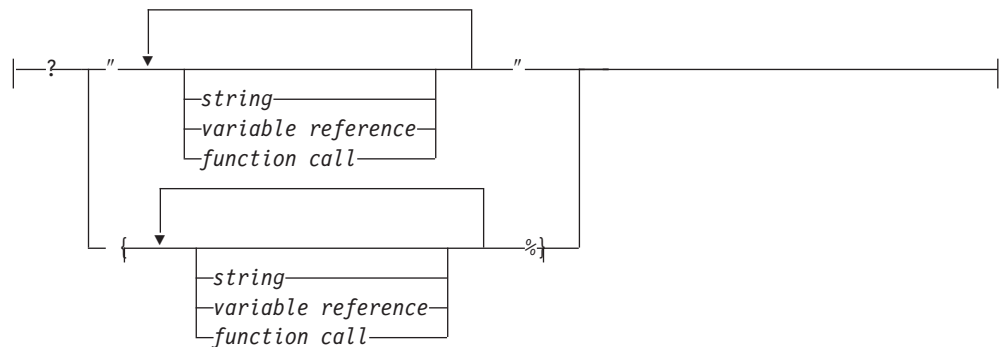


conditional variable





abbreviated conditional variable



값

%DEFINE

변수를 정의하는 키워드.

define entry:

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문 정보를 보려면 4페이지의 『변수명』을 참고하십시오.

string

연속된 영문자와 숫자 및 구두점. 문자열이 큰 따옴표로 묶여 있으면 개행 문자는 허용되지 않습니다.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다.
예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

exec statement

EXEC 명령문. 변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름. 구문 및 예제는 14페이지의 『EXEC 블록 또는 명령문』을 참고하십시오.

table statement

TABLE 명령문. 동일한 레코드 배열, 행 및 각 행의 필드를 설명하는 컬럼 이름의 배열이 포함된 관련 자료의 집합을 정의합니다. 구문 및 예제에 대해서는 59페이지의 『TABLE 명령문』을 참고하십시오.

envvar statement

ENVVAR 명령문. 환경 변수를 참조합니다. 구문 및 예제는 13페이지의 『ENVVAR 명령문』을 참고하십시오.

conditional variable

다른 변수나 문자열의 값에 따라 변수의 값을 설정합니다.

abbreviated conditional variable

다른 변수나 문자열의 값에 따라 변수의 값을 설정합니다. 더 짧은 형태의 조건 변수.

list statement

LIST 명령문. 분리된 값 목록을 구성하는 데 사용되는 값을 정의합니다. 구문 및 예제는 42페이지의 『LIST 명령문』을 참고하십시오.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제에 대해서는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

문맥

DEFINE 블록 또는 명령문은 Net.Data 매크로에서 선언 부분의 IF 블록 내에 또는 다른 모든 블록 밖에 있어야 합니다.

제한

- 다음과 같은 요소들을 포함할 수 있습니다.
 - 주식 블록
 - 조건 변수
 - LIST 명령문
 - TABLE 명령문
 - 변수 참조
 - INCLUDE 명령문
 - EXEC 명령문
 - 함수 호출
 - ENVVAR 명령문

- 조건 변수는 널(NULL) 결과값을 가질 수 없습니다. 자세한 내용은 예제 5를 참고하십시오.
- 자체의 정의에서는 변수를 사용할 수 없습니다. 예를 들어 다음 변수 정의는 허용되지 않습니다.

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

예

예제 1: 단순 변수 정의

```
%DEFINE var1 = "orders"
%DEFINE var2 = "${var1}.html"
```

런타임 중에 변수 참조 `$(var2)` 는 `orders.html`로 지정됩니다.

예제 2: 문자열 내의 인용 부호

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

변수 `hi`가 표시될 때는 값 `say "hello"`가 나타납니다. 변수 `empty`는 널(NULL)이 됩니다.

예제 3: 복수 변수의 정의

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

예제 4: 변수의 복수 행 정의

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

예제 5: 다음의 조건 변수 예제는 변수 `var`이 결과 값에 널(NULL) 값이 없을 때 인용 부호 (『』)로 묶인 결과 값을 취하는 방법을 보여줍니다. 아래 예제에서 `$(V)`와 `MyFunc`는 모두 널(NULL) 값을 결과로 가질 수 없습니다.

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

ENVVAR 명령문

목적

변수를 DEFINE 블록의 환경 변수로 정의합니다. ENVVAR 변수가 참조되면 Net.Data는 같은 이름으로 환경 변수의 현재 값을 리턴합니다. 이 방법을 사용하여 환경 변수를 참조하는 것이 DTW_GETENV를 사용하는 것보다 훨씬 효과적입니다. 자세한 내용은 135페이지의 『DTW_GETENV』를 참고하십시오.

구문

▶▶%ENVVAR◀◀

문맥

ENVVAR문은 DEFINE 블록이나 명령문 내에 있을 수 있습니다.

값

%ENVVAR

DEFINE 블록에서 변수를 환경 변수로 정의하기 위한 키워드. 이 변수는 매크로 파일 내의 모든 위치에서 환경 변수의 값을 가져옵니다.

제한

ENVVAR문은 다른 요소를 포함할 수 없습니다.

예

예제 1: 이 예제에서 %ENVVAR는 참조될 때 웹 서버의 이름인 환경 변수 SERVER_SOFTWARE에 대한 현재 값을 리턴하는 변수를 정의합니다.

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

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

EXEC 블록 또는 명령문

목적

변수를 참조하거나 함수를 호출할 때 실행할 외부 프로그램을 지정합니다.

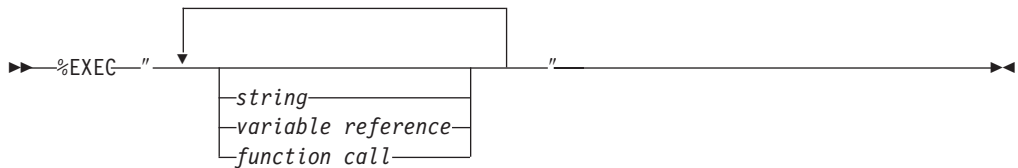
변수가 참조되거나 함수가 호출될 때, Net.Data는 먼저 Net.Data 초기화 파일의 EXEC_PATH 변수에 지정된 디렉토리를 검색하고 이 디렉토리에서 찾지 못하면 시스템 셸로 실행 파일의 이름을 제공합니다.

권한 정보: 웹 서버가 EXEC문이나 블록에서 참조하는 파일에 대한 액세스 권한을 가지는지 확인하십시오. 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장에서 Net.Data 파일에 대한 웹 서버 액세스 권한 지정에 관한 절을 참고하십시오.

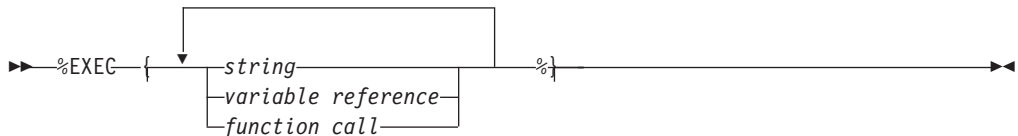
EXEC 명령문과 블록은 사용되는 위치에 따라 두 개의 다른 문맥에 사용되고 서로 다른 구문을 가집니다. DEFINE 블록에서는 EXEC문을 사용하고 FUNCTION 블록에서는 EXEC 블록을 사용하십시오.

구문

DEFINE 블록에서 사용될 때 EXEC문 구문:



FUNCTION 블록에서 사용될 때 EXEC 블록 구문:



값

%EXEC

변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름을 지정하는 키워드. Net.Data가 EXEC 명령문에 정의되어 있는 변수 참조를 만나면, Net.Data는 EXEC 명령문이 변수에 대해 선언한 내용을 처리합니다.

string

연속된 영문자와 숫자 및 구두점. 문자열이 큰 따옴표로 묶여 있으면 개행 문자는 허용되지 않습니다.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예:

if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4 페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

문맥

EXEC 블록 또는 명령문은 다음 문맥에 들어 있을 수 있습니다.

- DEFINE 블록
- FUNCTION 블록

제한

EXEC 블록 또는 명령문은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- 문자열
- 변수 참조
- 함수 호출

예

예제 1: 변수에서 참조하는 실행 파일

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

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

이 예제는 변수, mycall를 참조할 때마다 MYEXEC.EXE를 실행합니다.

예제 2: 함수에서 참조하는 실행 파일

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

이 예제는 함수 my_rexx_pgm이 호출될 때 mypgm.cmd를 실행합니다.

FUNCTION 블록

목적

Net.Data가 매크로 파일로부터 호출하는 서브루틴을 정의합니다. FUNCTION 블록의 실행 명령문은 언어 환경에서 직접 해석되는 인라인 명령문일 수 있으며 또는 외부 프로그램에 대한 호출을 나타낼 수 있습니다.

FUNCTION 블록에서 EXEC 블록을 사용하면 이 블록은 FUNCTION 블록에서 유일한 실행 명령문이어야 합니다. 언어 환경에 실행 명령문을 제공하기 전에 Net.Data는 초기화 파일의 EXEC_PATH 구성 명령문에 의해 결정되는 경로명으로 EXEC 블록 내의 프로그램의 파일명을 첨부합니다. 결과 문자열은 실행될 언어 환경으로 전송됩니다.

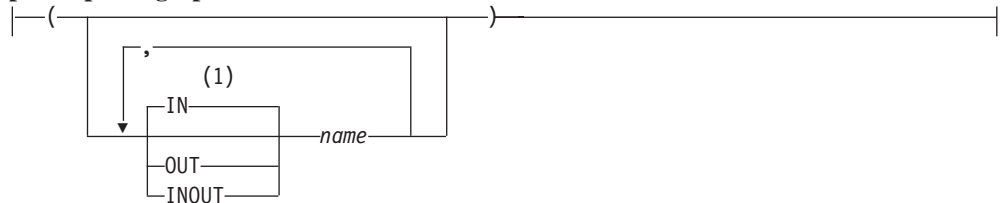
언어 환경이 EXEC 블록을 처리하는 데 사용하는 방법은 특정 언어 환경에 좌우됩니다. REXX, System 및 Perl Net.Data 제공 언어 환경에서만 EXEC 블록을 지원합니다.

구문

```
►►%FUNCTION—(—lang_env—)—function_name—| parm passing spec |—————►
```

```
◄◄| ;—————| returns spec | { | function body | } |—————►
```

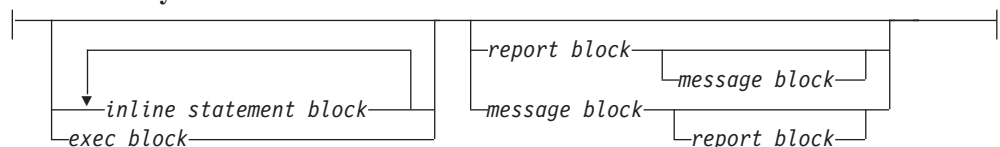
parm passing spec



return spec



function body



주:

1. 생략시 매개변수 유형인 IN은 매개변수 목록 맨 앞에 지정된 매개변수 유형이 없을 때 적용됩니다. 매개변수 유형이 없는 매개변수는 매개변수 목록에 가장 최근에 지정된 유형을 사용하거나 지정된 유형이 없을 때는 유형 IN을 사용합니다. 예를 들어 매개변수 목록(*parm1*, INOUT

parm2, parm3, OUT parm4, parm5)에서 매개변수 *parm1, parm3* 및 *parm5*는 매개변수 유형을 가지지 않습니다. 매개변수 *parm1*은 초기 매개변수 유형이 지정되지 않았으므로 유형 IN을 가집니다. 매개변수 *parm3*은 가장 최근에 지정된 매개변수 유형인 INOUT을 가집니다. 이와 마찬가지로 매개변수 *parm5*는 유형 OUT이 매개변수 목록에서 가장 최근에 지정된 유형이므로 이 유형을 가집니다.

값

%FUNCTION

Net.Data는 매크로 파일에서 호출하는 서브루틴을 지정하는 키워드.

lang_env

함수 내용을 처리하는 언어 환경. 자세한 내용은 *Net.Data Language Environment Reference*를 참고하십시오.

function_name

영문자나 밑줄로 시작하며 영문자, 숫자 또는 밑줄 등이 조합되어 포함되는 영문자나 숫자 문자열이 될 수 있는 함수 이름.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

매개변수 제공 스펙:

IN Net.Data가 언어 환경으로 입력 자료를 제공하도록 지정합니다. IN이 생략시 값입니다.

OUT

언어 환경이 Net.Data로 출력 자료를 리턴하도록 지정합니다.

INOUT

Net.Data가 언어 환경에 입력 자료를 제공하고 언어 환경은 Net.Data로 출력 자료를 리턴하도록 지정합니다.

리턴 스펙:

RETURNS

함수가 완료된 후에 언어 환경에 의해 지정된 함수 값을 포함하는 변수를 선언합니다.

함수 내용:

인라인 명령문 블록

함수 정의에 지정된 언어 환경(예: REXX, SQL 또는 Perl)에서 가져온 구문이 유효한 명령문. 사용 중인 언어 환경에 대한 설명은 *Net.Data Language Environment Reference*를 참고하십시오. 구문 및 사용에 대해서는 프로그래밍 언어의 프로그래밍 참조서를 참고하십시오. 인라인 명령문 블록을 나타내는 문자열은 인라인 명령문 블록(프로그램)의 실행 이전에 진행되는 Net.Data 변수 참조 및 함수 호

출을 포함할 수 있습니다. 제한사항: Net.Data 변수 참조 또는 함수 호출을 포함하지 않는 가장 긴 연속 인라인 명령문 블록 문자열은 다음 길이로 제한됩니다.

- OS/2 및 NT의 경우: 64KB
- AIX의 경우: 256KB
- OS/390의 경우: 256KB
- OS/400의 경우: 256KB

exec block

EXEC 블록. 변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름. 구문 및 예제는 14페이지의 『EXEC 블록 또는 명령문』을 참고하십시오.

report block

REPORT 블록. 함수 호출의 출력에 대한 포매팅 지침. 보고서에 머리말 및 꼬리말 정보를 사용할 수 있습니다. 구문 및 예제는 53페이지의 『REPORT 블록』을 참고하십시오.

message block

MESSAGE 블록. 리턴 코드 세트, 연관된 메시지, 함수 호출이 리턴될 때 Net.Data가 취하는 조치. 구문 및 예제는 48페이지의 『MESSAGE 블록』을 참고하십시오.

문맥

FUNCTION 블록은 다음 문맥에 들어 있을 수 있습니다.

- IF 블록
- Net.Data 매크로의 선언 부분에 있는 블록이나 명령문의 외부.

제한

FUNCTION 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- EXEC 블록
- MESSAGE 블록
- REPORT 블록
- 인라인 명령문 블록

REXX, System 및 Perl Net.Data 제공 언어 환경에서만 EXEC 명령문을 지원합니다.

예

다음 예제는 범용 예제로, 모든 언어 환경에 적용되지는 않습니다. 특정 언어 환경에서 FUNCTION 블록을 사용하는 것에 대한 자세한 내용은 *Net.Data Language Environment Reference*를 참고하십시오.

예제 1: REXX 부속 문자열 함수

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

*a*가 평가되면 @substring 함수 호출이 발견되고 부속 문자열 FUNCTION 블록이 실행됩니다. FUNCTION 블록의 실행 명령문에서 변수가 대체되면 텍스트 문자열 *s* = substr("longstring", 1, 4)가 REXX 해석기로 제공되어 실행됩니다. RETURNS 절이 지정되었으므로 *a*의 평가에서 @substring 함수 호출 값은 『long』, 값 *s*로 대체됩니다.

예제 2: 외부 REXX 프로그램 호출

- Net.Data 매크로:

```
%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)
}%
```

변수 *w*와 *x*는 함수에서 INOUT 매개변수 *a*와 *b*에 해당합니다. 이들의 값과 IN 매개변수 *c*에 해당하는 값 *y*는 이미 HTML 형식 입력이나 DEFINE 문으로부터 정의되어 있어야 합니다. 변수 *a*와 *b*는 매개변수 *a*와 *b*가 값을 리턴할 때 새로운 값으로 지정됩니다. 변수 *z*는 OUT 매개변수 *d*가 값을 리턴할 때 정의됩니다.

- REXX 프로그램 mypgm.cmd:

```
/* Sample REXX Program for
예제 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.cmd으로부터의 출력:

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

EXEC문은 REXX 해석기가 외부 REXX 프로그램 mypgm.cmd를 실행하도록 REXX 언어 환경에 지시합니다. REXX 언어 환경은 REXX 프로그램과 직접 Net.Data 변수를 공유할 수 있으므로 이 언어 환경은 mypgm.cmd을 실행하기 전에 REXX 변수 *a*, *b* 및 *c*를 Net.Data 변수값인 *w*, *x* 및 *y*로 지정합니다. mypgm.cmd는 REXX 명령문에서 변수 *a*, *b* 및 *c*를 직접 사용할 수

있습니다. 이 프로그램이 종료되면 REXX 변수 *a*, *b* 및 *d*는 REXX 프로그램으로부터 검색되며 그 값들은 Net.Data 변수 *w*, *x* 및 *z*로 지정됩니다. RETURNS 절이 my_rexx_pgm FUNCTION 블록의 정의에서 사용되지 않으므로 @my_rexx_pgm 함수 호출의 값은 널(NULL) 문자열이 되거나 (리턴 코드가 0이 아닌 경우) REXX 프로그램 리턴 코드 값이 됩니다.

예제 3: SQL 조회 및 보고서

```
%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 함수 호출은 SELECT문에서 \$(x)를 IBM으로, \$(y)를 12345로 대체합니다. SQL 함수 정의 query_1이 출력 테이블 변수를 식별하지 못하므로 생략시 테이블이 사용됩니다 (자세한 내용은 TABLE 변수 블록을 참고하십시오). REPORT 블록에서 참조되는 NLIST 및 Vi 변수는 생략시 테이블 정의에 의해 정의됩니다. REPORT 블록에서 생성한 보고서는 query_1 함수가 호출되는 출력 HTML에 배치됩니다.

예제 4: Perl 스크립트를 실행하기 위한 시스템 호출.

- Net.Data 매크로:

```
%FUNCTION(DTW_SYSTEM) today() RETURNS(result) {
  %exec{ perl "today.pr1" %}
%}
%HTML(INPUT) {
  @today()
%}
```

- Perl 프로그램 today.pr1:

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

System 언어 환경은 FUNCTION 블록내의 실행 명령문을 C 언어 system() 함수 호출을 통해 운영체제로 전달하여 해석합니다. 이 방법은 Net.Data 변수

를 REXX 언어 환경이 하는 것처럼 실행 명령문으로 직접 전달하거나 검색하지 않도록 합니다. 따라서, System 언어 환경은 다음과 같은 방식으로 변수를 전달 및 검색합니다.

- 입력 매개변수는 putenv() 함수를 통해 시스템 환경 변수로 전달되며, 실행 프로그램에 의해 검색될 수 있습니다. 다른 언어 환경에서는 다른 방법으로 변수를 참조합니다. UNIX cshell 스크립트는 \$x와 같이 환경 변수명 앞에 '\$'를 붙여 환경 변수를 참조합니다. Perl 언어 스크립트는 %ENV{'x'}와 같이 연관 배열 %ENV를 참조하여 환경 변수를 참조합니다. DOS 배치(.BAT) 파일은 %x%와 같이 퍼센트 기호로 묶어 변수 이름을 참조합니다.
- 출력 매개변수는 출력 매개변수가 시스템 환경 변수로서 언어 환경으로 다시 전달되는 OS/400 플랫폼의 경우를 제외하고 이름이 환경 변수 DTWPIPE에서 전달되는 파이프에 기록되어 언어 환경으로 다시 전달됩니다. 명명된 파이프에 기록된 자료는 DEFINE문에서처럼 name="value" 형식을 가집니다. 출력 매개변수에 해당하는 변수 이름이 이런 방법으로 기록되면, 새로운 값이 현재 값을 대체합니다. 출력 매개변수에 해당하지 않는 변수 이름이 기록되면, 그 변수 이름은 무시됩니다.

@today 함수 호출이 발견되면 Net.Data는 실행 명령문에서 변수 대체를 수행합니다. 이 예제에서는 실행 명령문에 Net.Data 변수가 없으므로, 어떠한 변수 대체도 수행되지 않습니다. 실행 명령문과 매개변수가 System 언어 환경으로 전달되면, 언어 환경은 명명된 파이프를 작성하고 환경 변수 DTWPIPE를 파이프 이름으로 설정합니다.

그런 다음, 외부 프로그램이 C system() 함수 호출을 사용하여 호출됩니다. 외부 프로그램은 파이프를 쓰기 전용으로 열고 표준 스트림 파일인 것처럼 출력 매개변수의 값을 파이프에 기록합니다. 외부 프로그램은 STDOUT에 기록하여 HTML 출력을 생성합니다. 이 예제에서, 시스템 날짜 프로그램의 출력은 FUNCTION 블록의 RETURNS 절에서 식별되는 변수인 result에 지정됩니다. 이 결과 변수 값은 HTML 블록에서 @today() 함수 호출을 대신합니다.

예제 5: Perl 언어 환경

```
%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()
}%
```

EXEC 블록이 사용되는 방식을 보려면 예제 4와 이 예제를 비교해 보십시오. 예제 4에서, 시스템 언어 환경은 Perl 프로그램 해석 방법을 알지 못하지만 외부 프로그램을 호출하는 방법을 알고 있습니다. EXEC 블록은 perl이라는 프로그램을 외부 프로그램으로 호출하도록 지시합니다. 실제 Perl 언어 명령문은 외부 Perl 프로그램에 의해 해석됩니다. 예제 5는 Perl 언어 환경이 Perl 언어 명령문을 직접 해석할 수 있으므로 EXEC 블록을 포함하지

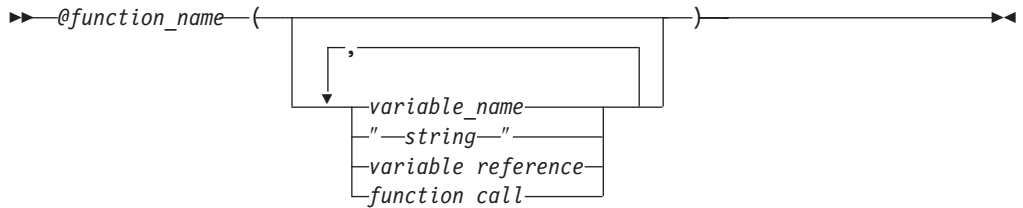
않습니다.

함수 호출 (@)

목적

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 내장 함수를 호출합니다. 함수가 내장 함수가 아니면 함수 호출을 지정하기 전에 이를 Net.Data 매크로에 미리 정의해야 합니다.

구문



값

@function_name

임의의 기존 함수 이름. 영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문 정보를 보려면 4페이지의 『변수명』을 참고하십시오.

string

개행 문자를 제외한 연속된 영문자, 숫자 및 구두점.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다.

문맥

함수 호출은 다음 문맥에 들어 있을 수 있습니다.

- HTML 블록
- REPORT 블록
- ROW 블록
- DEFINE 블록
- IF 블록
- MACRO_FUNCTION 블록

- MESSAGE 블록
- WHILE 블록
- 함수 호출 명령문
- Net.Data 매크로 선언부에서 모든 블록의 외부

제한

- 함수 호출은 다음 요소를 포함할 수 있습니다.
 - 주석 블록
 - 문자열
 - 함수 호출
 - 변수 참조
- 함수 호출은 함수 정의의 OUT 또는 INOUT 매개변수에 대해 정의된 변수 참조 및 함수 호출을 포함할 수 없습니다.

예

예제 1: SQL 함수 formQuery 호출

```
%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()
%}
```

예제 2: 입출력 매개변수를 사용한 REXX 함수 호출

```
%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)
%}
```

예제 3: 변수 참조와 함수 호출을 사용하여 입력 매개변수를 통한 REXX 함수 호출

```

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

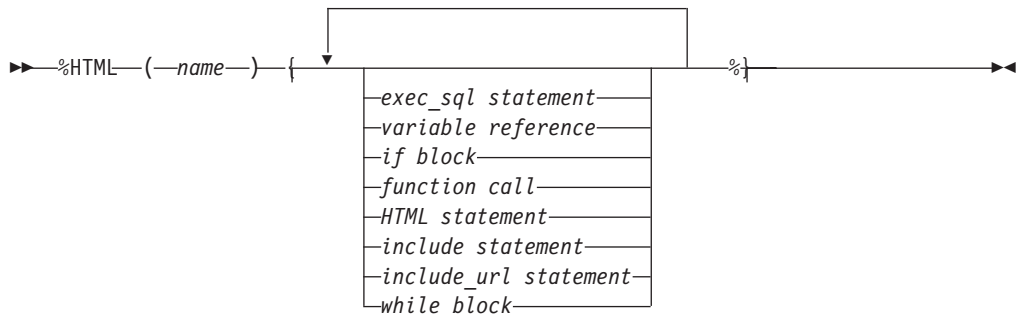
```

HTML 블록

목적

HTML을 이해하는 클라이언트의 웹 브라우저나 툴에 의해 처리될 HTML 태그나 텍스트를 포함합니다. 또한 HTML 블록은 수행 시에 평가되고 실행되는 대부분의 Net.Data 매크로 언어 명령문을 포함할 수 있습니다. Net.Data는 Net.Data 매크로 명령문을 찾아 실행합니다. Net.Data는 다른 모든 텍스트를 HTML로 가정하고 이를 웹 브라우저로 보냅니다.

구문



값

%HTML

클라이언트의 웹 브라우저에 표시될 HTML 태그 및 원문이 들어 있는 블록을 지정하는 키워드.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 231페이지의 『부록A. DB2 WWW 연결』 또는 DB2 월드 와이드 웹 릴리스 1 설명서를 참고하십시오.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내는 문자열이며 시작 또는 끝 공백이 없는 경우 비교하기 위해 숫자로 취급합니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다. 구문 및 예제는 29페이지의 『IF 블록』을 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나

지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

HTML statements

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data Web 매크로에 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예제는 40페이지의 『INCLUDE_URL 명령문』을 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

문맥

HTML 블록은 다음 문맥에 들어 있을 수 있습니다.

- IF 블록
- Net.Data 매크로 선언부에서 모든 블록의 외부

제한

HTML 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- EXEC_SQL 명령문
- IF 블록
- HTML 명령문
- INCLUDE 명령문
- INCLUDE_URL 명령문
- WHILE 블록
- 변수 참조
- 함수 호출

예

예제 1: 머리말 및 꼬리말에 대한 포함 파일을 가지는 HTML 블록

```
%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:
```

```
| @xmp1()  
| %INCLUDE"footer.html"  
| %}
```

IF 블록

목적

조건부 문자열 처리를 수행합니다. IF 블록은 하나 이상의 조건을 테스트하고 조건 테스트의 출력을 기초로 명령문 블록을 수행할 기능을 제공합니다. Net.Data 매크로, HTML 블록, MACRO_FUNCTION 블록, REPORT 블록, WHILE 블록 및 ROW 블록의 선언 부분에 IF 블록을 사용할 수 있으며 이를 다른 IF 블록에 중첩할 수도 있습니다.

조건 목록의 문자열 값은 정수를 나타내는 문자열이며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급합니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다.

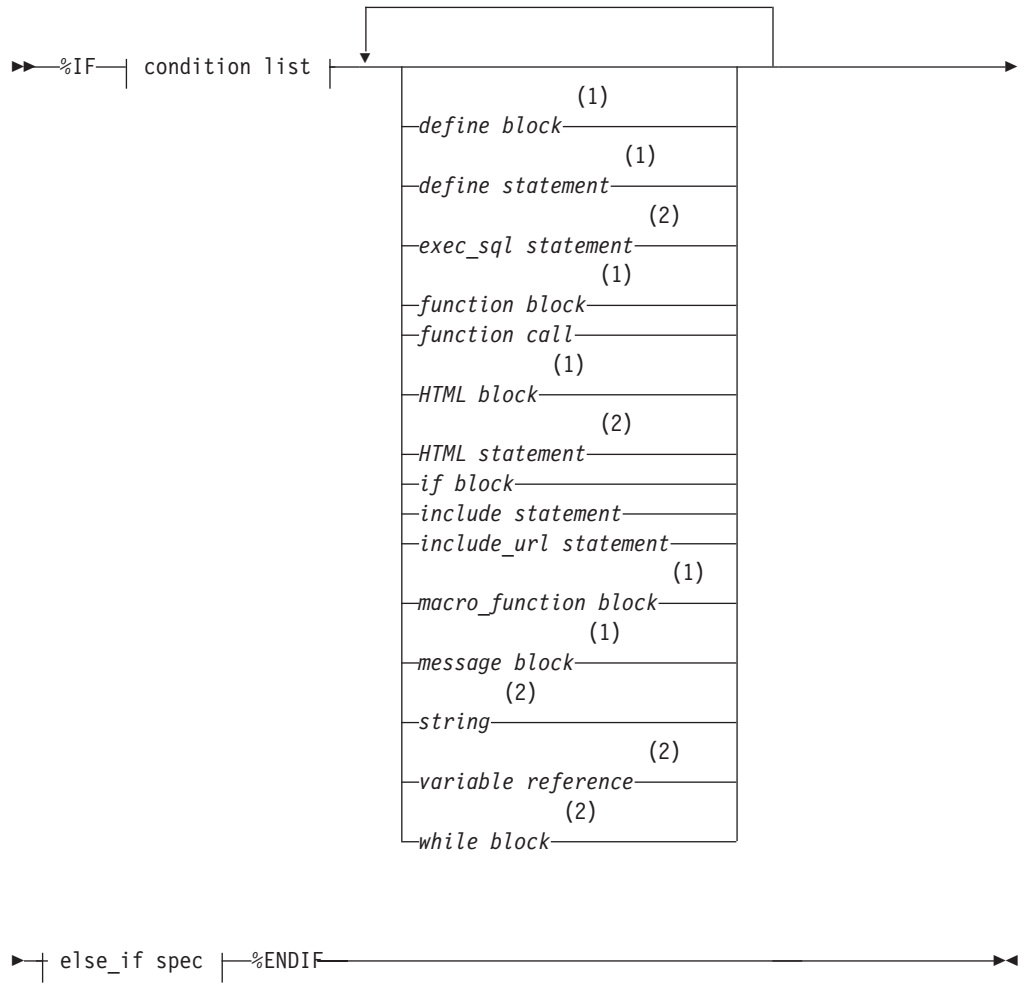
중첩된 IF 블록: IF 블록 구문에 대한 규칙은 매크로 파일에서 이 블록의 위치에 의해 좌우됩니다. IF 블록이 선언 부분의 다른 블록 외부에 있는 IF 블록 내에 중첩되면 이 블록은 외부 블록이 사용할 수 있는 모든 요소를 사용할 수 있습니다. IF 블록이 IF 블록에 있지 않은 다른 블록 내에 중첩되면 이 블록이 중첩된 블록에 대한 구문 규칙을 취합니다.

다음 예제에서 중첩된 IF 블록은 HTML 블록내에 있을 때 사용되는 규칙을 따라야 합니다.

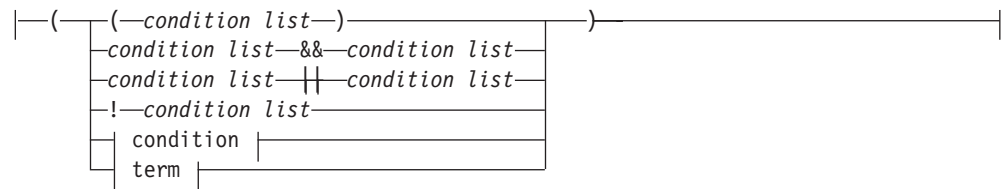
```
%IF block
...
%HTML block
...
%IF block
```

이 절 뒷부분에 나열된 제한사항을 참고하십시오.

구문



condition list



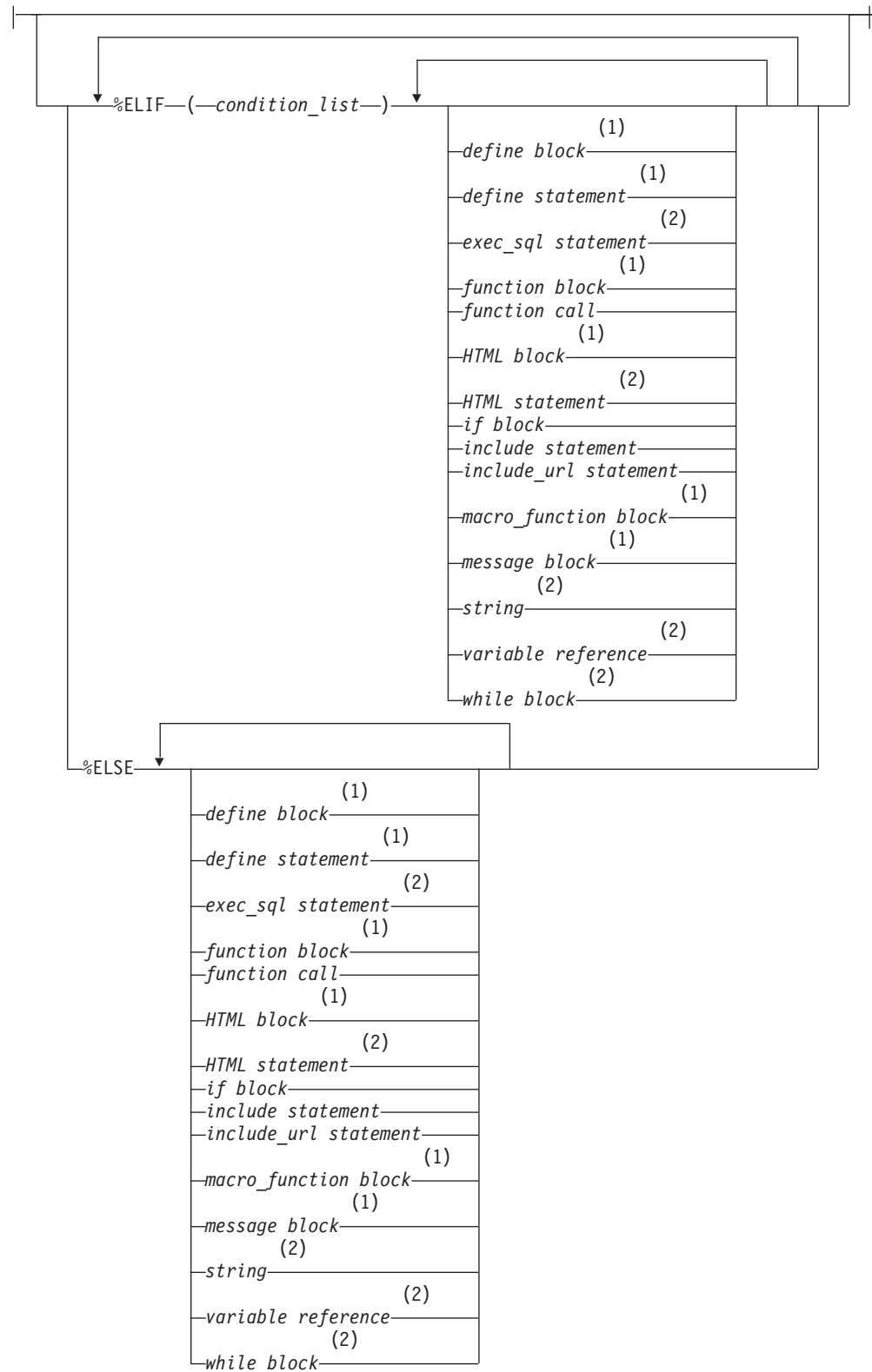
condition



term



else_if spec



주:

1. 이 언어 구성은 IF 블록이 매크로의 선언 부분에 있는 다른 블록 외부에 있을 때 유효합니다.

2. 이 언어 구성은 IF 블록이 HTML 블록, MACRO_FUNCTION 블록, REPORT 블록 또는 WHILE 블록에 있을 때 유효합니다.

값

%IF

조건부 문자열 처리를 지정하는 키워드.

condition list

조건과 규정의 값을 비교합니다. 조건 목록은 부울 연산자를 사용하여 연결할 수 있습니다. 조건 목록은 다른 조건 목록 내에 중첩될 수 있습니다.

define statement

DEFINE 블록 또는 명령문. 변수를 정의하고 구성 변수를 설정합니다. 변수명은 문자나 밑줄(_)로 시작해야 하며, 모든 영숫자 문자 및 밑줄을 포함할 수 있습니다. 구문 및 예제는 9페이지의 『DEFINE 블록 또는 명령문』을 참고하십시오.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 231페이지의 『부록A. DB2 WWW 연결』 또는 DB2 월드 와이드 웹 릴리스 1 설명서를 참고하십시오.

function block

Net.Data 매크로에서 호출할 수 있는 서브루틴을 지정하는 키워드. FUNCTION 블록의 실행 명령문은 언어 환경에서 직접 해석되는 언어 명령문을 포함할 수 있으며 또는 외부 프로그램에 대한 호출을 나타낼 수 있습니다. 구문 및 예제는 16페이지의 『FUNCTION 블록』을 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

HTML block

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

HTML statement

클라이언트 브라우저용으로 형식화될 모든 영문자, 숫자 및 HTML 태그를 포함합니다.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내는 문자열이며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급합니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data Web 매크로에 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예제는 40페이지의 『INCLUDE_URL 명령문』을 참고하십시오.

macro_function block

Net.Data 매크로로부터 호출될 수 있는 서브루틴을 지정하는 키워드. MACRO_FUNCTION 블록의 실행 명령문은 Net.Data 매크로 언어 소스 명령문을 포함할 수 있습니다. 구문 및 예제는 44페이지의 『MACRO_FUNCTION 블록』을 참고하십시오.

message block

MESSAGE 블록. 리턴 코드 세트, 연관된 메세지, 함수 호출이 리턴될 때 Net.Data가 취하는 조치. 구문 및 예제는 48페이지의 『MESSAGE 블록』을 참고하십시오.

string

연속된 영문자와 숫자 및 구두점. 문자열이 조건 목록의 규정에 있으면 개행 문자를 제외한 모든 문자를 포함할 수 있습니다. 문자열이 실행 코드 블록에 있으면 개행 문자를 포함하는 모든 문자를 포함할 수 있습니다.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

condition

비교 연산자를 사용하여 두 용어를 비교하는 것. IF 조건은 다음 두 조건이 모두 참일 때 숫자 비교로 취급됩니다.

- 조건 연산자가 <,<=,>,>=,==,!= 중 하나입니다.
- 규정과 문자열이 모두 유효한 정수를 나타내는 문자열입니다. 여기에 유효한 정수는 경우에 따라 더하기 (+) 또는 빼기 (-) 기호가 앞에 붙으며 공백은 앞에 나오지 않는 일련의 숫자를 나타냅니다.

두 조건 중 하나가 참이 아니면 일반적인 문자열 비교가 수행됩니다.

term

변수명, 문자열, 변수 참조 또는 함수 호출.

%ELIF

대체 처리 경로를 시작하며 조 목록과 대부분의 Net.Data 매크로 명령문을 포함할 수 있는 키워드.

%ENDIF

%IF 블록을 닫는 키워드.

%ELSE

기타 모든 조건 목록이 만족하지 않을 경우에 연관된 명령문을 실행하는 키워드.

문맥

IF 블록은 다음 문맥에 들어 있을 수 있습니다.

- Net.Data 매크로 선언 부분에 있는 다른 블록의 외부
- HTML 블록
- IF 블록
- MACRO_FUNCTION 블록
- REPORT 블록
- ROW 블록
- WHILE 블록

제한

IF 블록은 Net.Data 매크로의 선언 부분에 있는 다른 블록 외부에 위치할 때 다음 요소를 포함할 수 있습니다.

- 주석 블록
- DEFINE 블록
- DEFINE 명령문
- FUNCTION 블록
- 함수 호출
- HTML 블록
- IF 블록
- INCLUDE 명령문
- INCLUDE_URL 명령문
- MACRO_FUNCTION 블록
- MESSAGE 블록
- 변수 참조

IF 블록은 Net.Data 매크로의 HTML 블록, MACRO_FUNCTION 블록, REPORT 블록, ROW 블록 또는 WHILE 블록에 위치할 때 다음 요소를 포함할 수 있습니다.

- 주석 블록
- EXEC_SQL 명령문

- 함수 호출
- IF 블록
- INCLUDE 명령문
- INCLUDE_URL 명령문
- HTML 명령문
- 문자열
- 변수 참조
- WHILE 블록

예

예제 1: Net.Data 매크로의 선언 부분에 있는 IF 블록

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

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

예제 2: HTML 블록 내의 IF 블록

```
%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
%}
```

예제 3: 숫자 비교

```
%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
```

내재적 테이블 변수인 ROW_NUM이 항상 정수값을 리턴하며 비교되는 값도 정수이므로 숫자 비교가 수행됩니다.

예제 4: 중첩 IF 블록

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

INCLUDE 명령문

목적

파일을 읽어 명령문이 지정된 Net.Data 매크로에 통합시킵니다.

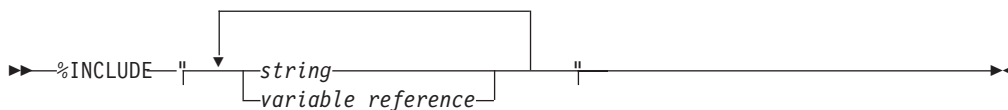
Net.Data는 초기화 파일의 INCLUDE_PATH 명령문에 지정된 디렉토리에서 포함 파일을 찾습니다.

대부분의 고급 언어에서처럼 포함 파일을 사용할 수 있습니다. 공통 머리말과 꼬리말을 삽입하거나, 공통 변수 세트를 정의하거나, FUNCTION 블록 정의의 공통 서브루틴 라이브러리를 Net.Data 매크로에 통합시킬 수 있습니다.

권한 정보: 웹 서버가 INCLUDE문에서 참조하는 파일에 대한 액세스 권한을 가지는지 확인하십시오. 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장에서 Net.Data 파일에 대한 웹 서버 액세스 권한 지정에 관한 절을 참고하십시오.

팁: 로컬 웹 서버에서 가져온 HTML 파일을 포함시키려면 INCLUDE_URL에 대해 예제 3에서 나타나는 것처럼 INCLUDE_URL 구성을 사용하십시오. 예제 구문을 사용하면 웹 서버가 이미 알고 있는 디렉토리를 지정하기 위해 Net.Data 초기화 파일에서 INCLUDE_PATH를 지정할 필요가 없습니다.

구문



값

%INCLUDE

파일을 지정하는 키워드를 읽은 후 Net.Data 매크로로 통합됩니다.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

string

개행 문자를 제외한 연속된 영문자, 숫자 및 구두점.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

문맥

INCLUDE문은 다음 문맥에 들어 있을 수 있습니다.

- DEFINE 블록

- HTML 블록
- REPORT 블록
- ROW 블록
- IF 블록
- MESSAGE 블록
- MACRO_FUNCTION 블록
- WHILE 블록
- Net.Data 매크로 선언부에서 모든 블록의 외부

제한

INCLUDE문은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- 문자열
- 변수 참조

예

예제 1: HTML 블록의 INCLUDE문

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

예제 2: REPORT 블록의 INCLUDE문

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

예제 3: INCLUDE문의 변수 참조

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

#include "$ (library)$ (filename)"
```

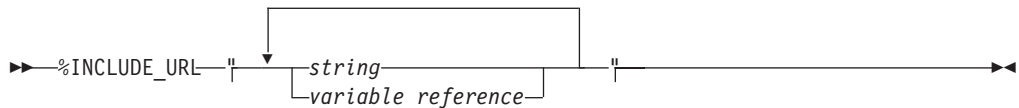
INCLUDE_URL 명령문

목적

다른 파일을 읽어 명령문이 지정된 Net.Data 생성 출력으로 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다.

INCLUDE_URL문을 사용하면 응용 프로그램 사용자가 제출 버튼을 선택하지 않아도 다른 매크로에서 매크로를 호출할 수 있습니다.

구문



값

%INCLUDE_URL

국지 또는 원격 서버로부터 파일을 읽어 Net.Data 매크로로 통합되도록 지정하는 키워드.

string

개행 문자를 제외한 연속된 영문자, 숫자 및 구두점.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

문맥

INCLUDE_URL문은 다음 문맥에 들어 있을 수 있습니다.

- HTML 블록
- REPORT 블록
- ROW 블록
- WHILE 블록
- MACRO_FUNCTION 블록
- Net.Data 매크로 선언부에서 모든 블록의 외부

제한

INCLUDE_URL문은 다음 요소들을 포함할 수 있습니다.

- 주석 블록
- 문자열
- 변수 참조

INCLUDE_URL 파일은 다음의 파일 크기 제한을 가집니다.

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

INCLUDE_URL는 OS/400 환경에서는 지원되지 않습니다.

예

예제 1: 다른 서버로부터의 HTML 파일 포함

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

예제 2: 서버명을 호출하여 원격 서버로부터의 HTML 파일 포함

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

여기에서 myserver는 서버명입니다.

예제 3: 국지 웹 서버로부터의 HTML 파일 포함

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

팁: 이 메서드를 사용하면 웹 서버가 이미 알고 있는 디렉토리를 지정하기 위해 Net.Data 구성 파일에서 INCLUDE_URL 경로를 갱신할 필요가 없습니다. 문자열이 슬래시로 시작되지 않으면 Net.Data는 이 문자열을 서버명으로 가정하고 해당 이름을 가진 서버로부터 파일을 검색하려 합니다.

예제 4: 원격 서버로부터의 다른 Net.Data 매크로 포함

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

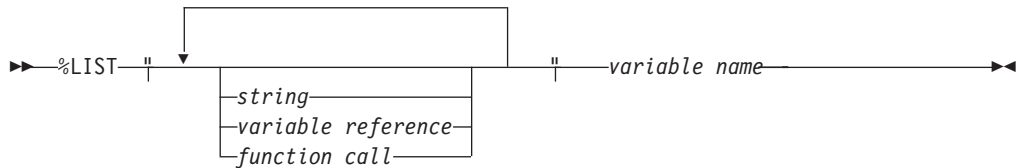
이 예제에서는 매크로 파일 hotpic.mac가 호출되고 custno가 변수로 송신됩니다. 문자열이 슬래시로 시작되면 Net.Data는 국지 웹 서버로부터 INCLUDE 파일을 검색합니다.

LIST 명령문

목적

값의 구분 목록을 작성합니다. 일부 WHERE 또는 HAVING 절에서 찾을 수 있는 것과 같이 여러 항목이 있는 SQL 조회를 구성할 때 LIST문을 사용할 수 있습니다.

구문



값

%LIST

값의 구분 목록을 작성하는 데 사용되는 변수를 지정하는 키워드.

string

개행 문자를 제외한 연속된 영문자, 숫자 및 구두점.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문 정보를 보려면 4페이지의 『변수명』을 참고하십시오.

문맥

LIST문은 다음 문맥에서 찾을 수 있습니다.

- DEFINE 명령문

제한

LIST문은 다음 요소를 포함할 수 있습니다.

- 주식 블록
- 변수 참조
- 함수 호출
- 문자열

예

예제 1: 변수 목록

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

MACRO_FUNCTION 블록

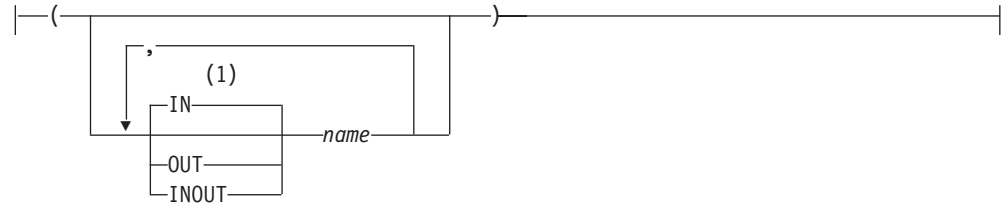
목적

Net.Data 매크로에서 호출할 수 있는 서브루틴을 정의합니다. MACRO_FUNCTION 블록의 실행 명령문은 Net.Data 매크로 언어 소스 명령문이어야 합니다.

구문

```
►► %MACRO_FUNCTION—function_name | parm passing spec |  
|  
| { function body | report block } % |  
◄◄
```

parm passing spec



function body



주:

1. 생략시 매개변수 유형인 IN은 매개변수 목록 맨 앞에 지정된 매개변수 유형이 없을 때 적용됩니다. 매개변수 유형이 없는 매개변수는 매개변수 목록에 가장 최근에 지정된 유형을 사용하거나 지정된 유형이 없을 때는 유형 IN을 사용합니다. 예를 들어 매개변수 목록(*parm1*, INOUT *parm2*, *parm3*, OUT *parm4*, *parm5*)에서 매개변수 *parm1*, *parm3* 및 *parm5*는 매개변수 유형을 가지지 않습니다. 매개변수 *parm1*은 초기 매개변수 유형이 지정되지 않았으므로 유형 IN을 가집니다. 매개변수 *parm3*은 가장 최근에 지정된 매개변수 유형인 INOUT을 가집니다. 이와 마찬가지로 매개변수 *parm5*는 유형 OUT이 매개변수 목록에서 가장 최근에 지정된 유형이므로 이 유형을 가집니다.

값

%macro_function

Net.Data 매크로에서 호출할 수 있는 서브루틴을 지정하는 키워드. MACRO_FUNCTION 블록의 실행 명령문에는 Net.Data가 직접 해석하는 언어 명령문이 포함되어야 합니다.

function_name

정의할 함수의 이름. 영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

매개변수 제공 스펙:

IN Net.Data가 언어 환경으로 입력 자료를 전달하도록 지정합니다. IN이 생략시 값입니다.

OUT

언어 환경이 Net.Data로 출력 자료를 리턴하도록 지정합니다.

INOUT

Net.Data가 언어 환경에 입력 자료를 전달하고 언어 환경은 Net.Data로 출력 자료를 리턴하도록 지정합니다.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

함수 내용:

exec_sql

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 231페이지의 『부록A. DB2 WWW 연결』 또는 DB2 월드 와이드 웹 릴리스 1 설명서를 참고하십시오.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급합니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

HTML statement

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제에 대해서는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로로 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예제는 40페이지의 『INCLUDE_URL 명령문』을 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

문맥

MACRO_FUNCTION 블록은 다음 문맥에서 찾을 수 있습니다.

- IF 블록
- Net.Data 매크로 선언부내의 모든 블록의 외부

제한

이 구성은 OS/390 운영체제에서는 사용할 수 없습니다.

MACRO_FUNCTION 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- EXEC_SQL 명령문
- IF 블록
- HTML 명령문
- INCLUDE 명령문
- WHILE 블록
- 변수 참조
- 함수 호출

예

예제 1: 메시지 처리를 지정하는 매크로 함수

```
%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
%}

```

예제 2: 헤더 정보를 지정하는 매크로 함수

```

%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 **블록**

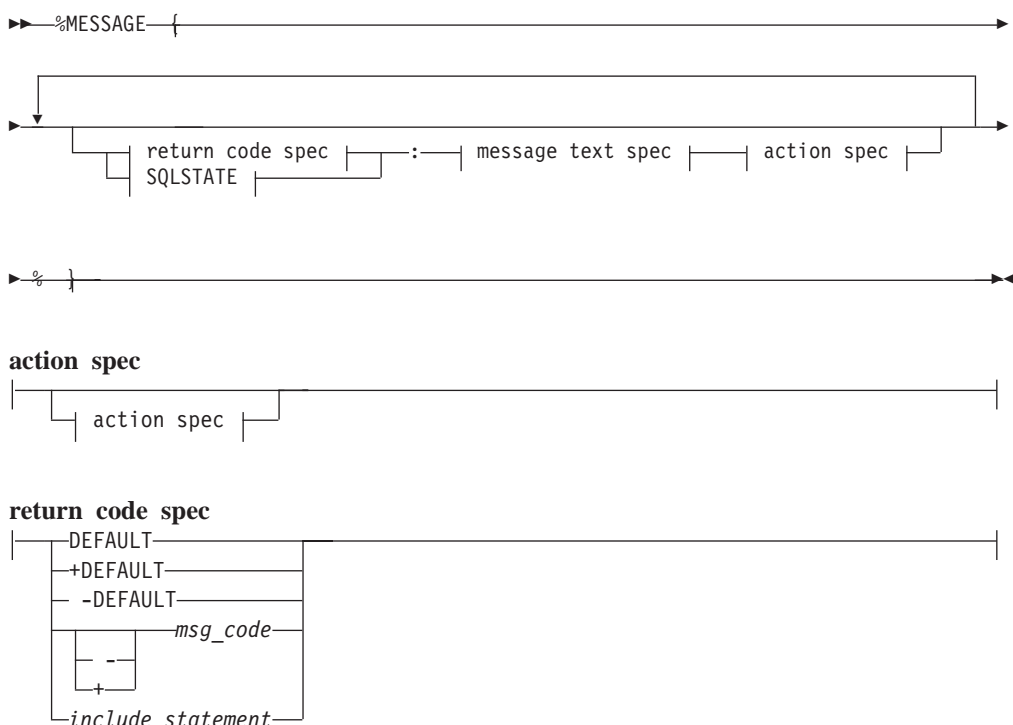
목적

함수로부터의 리턴 코드를 기초로 표시할 메세지와 취할 조치를 지정합니다.

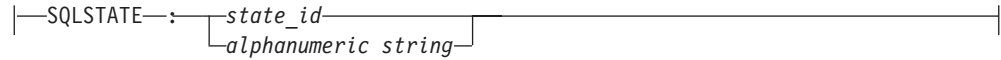
MESSAGE 블록에서 해당 메시지와 조치와 함께 리턴 코드 집합을 정의합니다. 함수 호출이 완료되면, Net.Data는 여기서 리턴된 리턴 코드를 MESSAGE 블록에 정의되어 있는 리턴 코드와 비교합니다. 함수의 리턴 코드가 MESSAGE 블록에 정의되어 있는 리턴 코드와 일치하면 Net.Data는 처리를 계속할지 또는 Net.Data 매크로를 종료할지의 여부를 판별하기 위해 메시지를 표시하고 권장 조치를 평가합니다.

MESSAGE 블록은 범위내에서 전역이거나 하나의 FUNCTION 블록에 대해 국지가 될 수 있습니다. MESSAGE 블록이 가장 바깥의 매크로 계층에 정의되어 있으면, 이는 범위내에서는 전역으로 간주됩니다. 복수의 전역 MESSAGE 블록이 정의되어 있으면, 마지막으로 처리된 블록만 사용 중인 것으로 간주됩니다. MESSAGE 블록이 FUNCTION 블록 내부에 정의되어 있으면, 블록이 정의된 FUNCTION 블록에 대한 범위내에서는 국지입니다. 리턴 코드 처리 규칙에 대해서는 *Net.Data Administration and Programming Guide*에서 MESSAGE 블록 절을 참고하십시오.

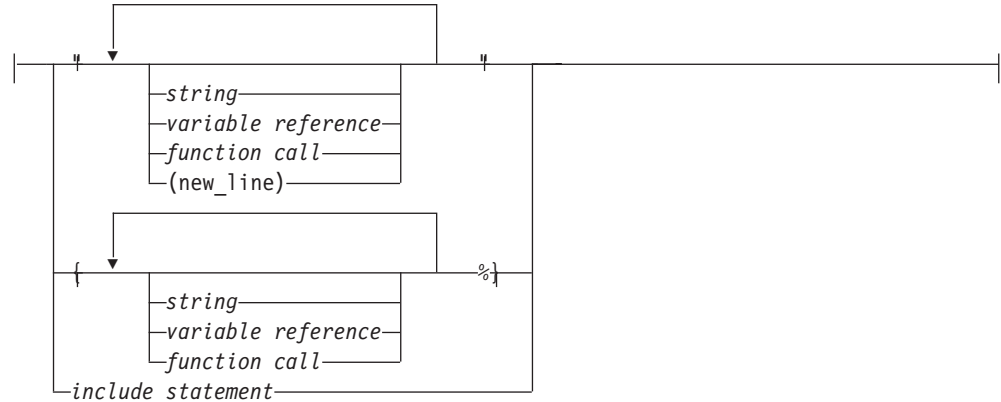
구문



SQLSTATE



message text spec



action spec



값

%MESSAGE

리턴 코드 세트, 연관된 메세지, 함수 호출이 리턴될 때 Net.Data가 취할 조치를 정의하는 블록에 대한 키워드.

리턴 코드 스펙

양의 정수 또는 음의 정수. Net.Data RETURN_CODE 변수 값이 리턴 코드 스펙 값과 일치하면 메세지 명령문의 나머지 정보는 함수 호출을 처리하는 데 사용됩니다. 또한 MESSAGE 블록에 특별하게 입력되어 있지 않은 리턴 코드에 대한 메세지를 지정할 수 있습니다.

+DEFAULT

양수의 생략시 메세지 코드를 지정하는 데 사용되는 키워드. Net.Data는 RETURN_CODE가 0보다 크고 정확한 일치 사항이 지정되지 않은 경우 이 메세지 명령문의 정보를 사용하여 함수 호출을 처리합니다.

-DEFAULT

음수의 생략시 메세지 코드를 지정하는 데 사용되는 키워드. Net.Data는 RETURN_CODE가 0보다 작고 정확한 일치 사항이 지정되지 않은 경우 이 메세지 명령문의 정보를 사용하여 함수 호출을 처리합니다.

DEFAULT

생략시 메시지 코드를 지정하는 데 사용되는 키워드. Net.Data는 다음 조건이 모두 만족될 때 이 메시지 명령문의 정보를 사용하여 함수 호출을 처리합니다.

- RETURN_CODE가 0보다 크거나 작지만 0이 아닌 경우.
- 리턴 코드에 대해 정확한 일치 사항이 지정되지 않은 경우.
- RETURN_CODE가 0보다 크거나 작을 때 +DEFAULT 또는 -DEFAULT 값이 지정되지 않은 경우.

msg_code

처리 중에 발생할 수 있는 오류나 경고를 지정하는 메시지 코드. 0에서 9자리의 숫자 문자열.

SQLSTATE

공통 오류 조건에 대한 공통 코드를 응용 프로그램에 제공하는 키워드. SQLSTATE 값은 SQL 표준에 포함된 SQLSTATE 스펙을 기초로 하며 코딩 기법은 모든 IBM SQL 구현에서 동일합니다. 제한사항: OS/400 플랫폼에서는 지원되지 않음.

state_id

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

영숫자 문자열

영문자 또는 숫자가 조합된 영문자 또는 숫자 문자열. 여기에는 구두점이 포함될 수 있습니다.

message text spec

RETURN_CODE가 현재 메시지 명령문의 *return_code* 값과 일치할 경우 웹 브라우저로 송신되는 문자열.

string

연속된 영문자와 숫자 및 구두점. 문자열이 큰 따옴표로 묶여 있으면 개행 문자는 허용되지 않습니다.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예 : if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

조치 스펙

RETURN_CODE가 현재 메시지 명령문의 *return_code* 값과 일치할 경우 Net.Data가 취하는 조치를 판별합니다.

EXIT

지정된 메세지 코드에 해당하는 오류나 경고가 발생했을 때 매크로가 즉시 종료되도록 지정하는 키워드. 이 값은 생략시 값입니다.

CONTINUE

지정된 메세지 코드에 해당하는 오류나 경고가 발생했을 때 처리를 계속하도록 지정하는 키워드.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. INCLUDE문은 MESSAGE의 모든 위치에서 나타날 수 있습니다. 구문 및 예제에 대해서는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

문맥

MESSAGE 블록은 다음 문맥에서 찾을 수 있습니다.

- FUNCTION 블록
- IF 블록
- Net.Data 매크로 선언부에서 모든 블록 및 명령문의 외부

제한

MESSAGE 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- 함수 호출
- 변수 참조
- HTML 명령문
- 문자열
- INCLUDE 명령문

SQLSTATE는 OS/400 플랫폼에서는 지원되지 않습니다.

예

예제 1: 국지 MESSAGE 블록

```
%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>"
%}
```

예제 2: 전역 MESSAGE 블록

```
%{ global message block %}
%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
anchors 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
anchors and forms, in this message. %}    : exit
    %}
}

```

예제 3: INCLUDE문을 포함하는 MESSAGE 블록.

```

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

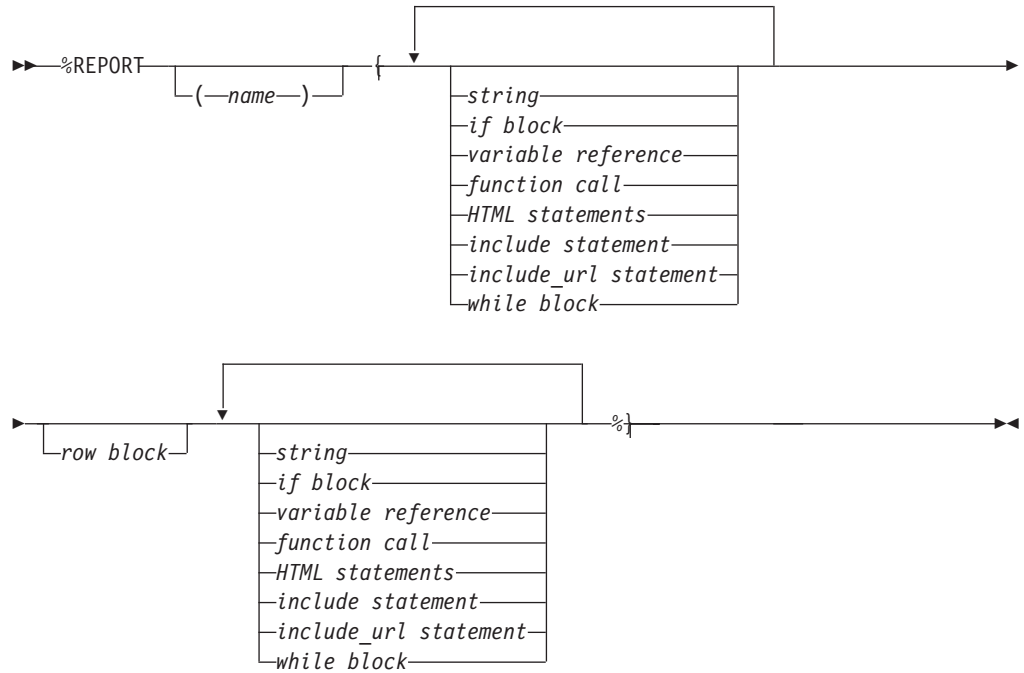
```

REPORT 블록

목적

함수 호출로부터의 출력 형식을 지정합니다. 보고서가 명명된 테이블의 자료를 사용하도록 지정하기 위해 테이블명 매개변수를 입력할 수 있습니다. 이 매개변수를 입력하지 않으면 함수 매개변수 목록에서 찾을 수 있는 첫 번째 출력 테이블을 사용하여 보고서가 생성되며 목록에 테이블명이 없으면 생략시 테이블 자료를 사용하여 보고서가 생성됩니다.

구문



값

%REPORT

함수 호출 출력에 대한 포매팅 지침을 지정하는 키워드. 보고서에 머리말 및 꼬리말 정보를 사용할 수 있습니다.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다.

string

연속된 영문자와 숫자 및 구두점.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 이들 값은 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다. 구문 및 예제는 29페이지의 『IF 블록』을 참고하십시오.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 Net.Data 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오. 제한사항: REPORT 블록은 OS/400 환경을 제외하고는 SQL 함수 호출을 포함할 수 없습니다.

HTML statements

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제에 대해서는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로로 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예제는 40페이지의 『INCLUDE_URL 명령문』을 참고하십시오.

row block

ROW 블록. 함수 호출로부터 리턴된 각 자료 행에 대해 한번씩 HTML 형식 자료를 표시합니다. 구문 및 예제는 56페이지의 『ROW 블록』을 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

문맥

REPORT 블록은 다음 문맥에서 찾을 수 있습니다.

- FUNCTION 명령문 또는 블록
- MACRO_FUNCTION 블록
- SQL 명령문 또는 블록

제한

REPORT 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- IF 블록
- INCLUDE 명령문

- INCLUDE_URL 명령문

- ROW 블록

- WHILE 블록

- 함수 호출

OS/390 플랫폼의 경우: SQL 함수는 SQL 함수 내부에서 호출할 수 없습니다.

- HTML 명령문

- 문자열

- 변수 참조

예

예제 1: 이름 및 위치 목록을 나타내는 2개 열로 이루어진 HTML 테이블

```
%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>
%}
```

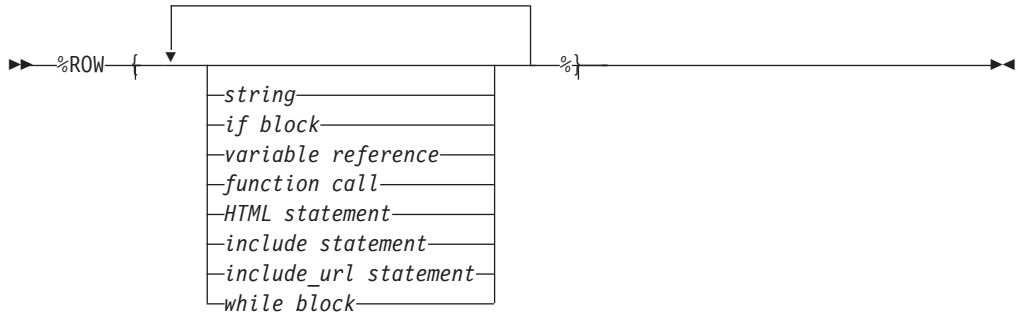
테이블의 이름을 선택하면 *name.mac* Net.Data 매크로의 *details* HTML 블록을 호출하고 URL의 일부인 두 값에 이를 송신합니다. 이 예제에서 이름에 대한 좀더 자세한 정보를 얻기 위해 *name.mac*에서 값을 사용할 수 있습니다.

ROW 블록

목적

함수 호출로부터 리턴된 각 테이블 행을 처리합니다. Net.Data는 각 행마다 한번씩 ROW 블록내의 명령문을 처리합니다.

구문



값

%ROW

함수 호출로부터 리턴된 각 자료 행에 대해 한번씩 표시될 HTML 포매팅 자료를 지정하는 키워드.

string

연속된 영문자와 숫자 및 구두점.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내는 문자열이며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 이들 값은 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다. 구문 및 예제는 29페이지의 『IF 블록』을 참고하십시오.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오. 제한사항: ROW는 OS/400 환경을 제외하고는 SQL 함수 호출에 해당하는 함수 호출을 포함할 수 없습니다.

HTML statements

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예제에 대해서는 38페이지의 『INCLUDE 명령문』을 참고하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로로 통합시킵니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예제는 40페이지의 『INCLUDE_URL 명령문』을 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

문맥

ROW 블록은 다음 문맥에서 찾을 수 있습니다.

- REPORT 블록

제한

ROW 블록은 다음 요소를 포함할 수 있습니다.

- 주식 블록
- IF 블록
- INCLUDE 명령문
- INCLUDE_URL 명령문
- WHILE 블록
- 함수 호출

OS/390 플랫폼의 경우: SQL 함수는 SQL 함수 내부에서 호출할 수 없습니다.

- 변수 참조
- HTML 명령문
- 문자열

예

예제 1: 이름 및 위치 목록을 나타내는 2개 열로 이루어진 HTML 테이블

```
%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>  
%}
```

테이블의 이름을 선택하면 *name.mac* Net.Data 매크로의 *details* HTML 블록을 호출하고 URL의 일부인 두 값에 이를 송신합니다. 이 예제에서 이름에 대한 자세한 정보를 얻기 위해 *name.mac*에서 값을 사용할 수 있습니다.

TABLE 명령문

목적

관련 자료의 집합에 해당하는 변수를 정의합니다. 여기에는 동일한 레코드 배열, 행 및 각 행의 필드를 설명하는 컬럼 이름의 배열이 포함됩니다. 테이블 명령문은 DEFINE문이나 블록 내에서만 정의할 수 있습니다.

구문

►►%TABLE | upper limit |

upper limit

| (number)
| ALL

값

%TABLE

동일한 레코드 배열, 행 및 각 행의 필드를 설명하는 컬럼 이름의 배열이 포함된 관련 자료 집합의 정의를 지정하는 키워드.

upper limit

테이블에 포함될 수 있는 행의 수.

number

0에서 9자리의 숫자 문자열. 테이블의 행 수를 제한하지 않는 0값이 허용됩니다.

ALL

테이블내의 행의 수를 제한하지 않는 키워드.

문맥

TABLE문은 다음 문맥에서 찾을 수 있습니다.

- DEFINE 명령문

제한

TABLE문은 다음 요소를 포함할 수 있습니다.

- 주식 블록
- 숫자

예

예제 1: 30개의 행 수 상한을 가지는 Net.Data 테이블

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

예제 2: 생략시 값이 모든 행을 사용하는 Net.Data 테이블

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

예제 3: 모든 행을 지정하는 Net.Data 테이블

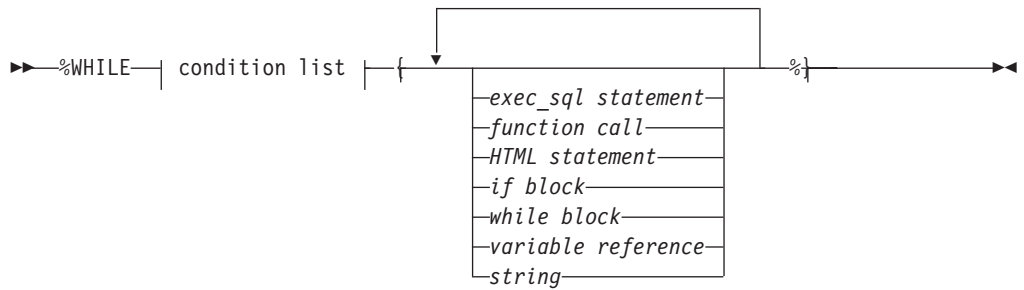
```
%DEFINE myTable3=%TABLE(ALL)
```

WHILE 블록

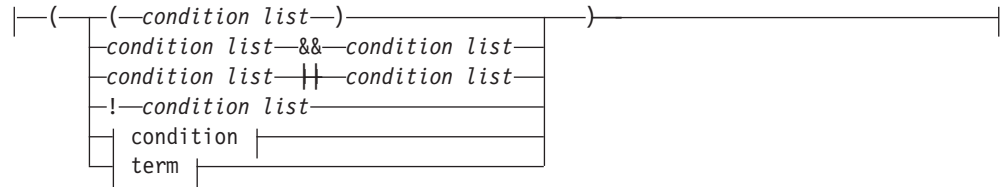
목적

조건부 문자열 처리를 기초로 루핑 구성을 제공합니다. HTML 블록, REPORT 블록, ROW 블록, IF 블록 및 MACRO_FUNCTION 블록에 WHILE 블록을 사용할 수 있습니다. 조건 목록의 문자열 값은 정수를 나타내는 문자열이며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급합니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다.

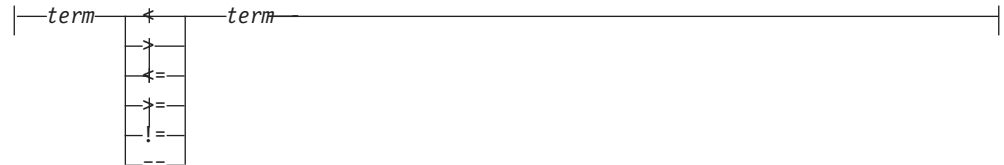
구문



condition list



condition



term



값

WHILE

루핑 처리를 지정하는 키워드.

condition list

조건과 규정의 값을 비교합니다. 조건 목록은 부울 연산자를 사용하여 연결할 수 있습니다. 조건 목록은 다른 조건 목록 내에 중첩될 수 있습니다.

condition

비교 연산자를 사용하여 두 용어를 비교하는 것. IF 조건은 다음 두 조건이 모두 참일 때 숫자 비교로 취급됩니다.

- 조건 연산자가 <,<=,>,>=,==,!= 중 하나입니다.
- 규정은 유효한 정수를 나타내는 문자열입니다. 여기에서 유효한 정수는 경우에 따라 더하기 (+) 또는 빼기 (-) 기호가 앞에 붙으며, 공백은 앞에 나오지 않는 일련의 숫자를 나타냅니다.

두 조건 중 하나가 참이 아니면 일반적인 문자열 비교가 수행됩니다.

term

함수 호출에 대한 변수명, 문자열, 변수 참조.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 231페이지의 『부록A. DB2 WWW 연결』 또는 DB2 월드 와이드 웹 릴리스 1 설명서를 참고하십시오.

function call

이전에 정의된 FUNCTION 또는 MACRO_FUNCTION 블록을 호출하거나 지정된 인수를 사용하여 내장 함수를 호출합니다. 구문 및 예제는 23페이지의 『함수 호출 (@)』을 참고하십시오.

HTML statement

클라이언트 브라우저용으로 형식이 지정될 HTML 태그와 함께 영문자 또는 숫자를 포함합니다.

IF block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 정수를 나타내며 시작 또는 끝 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 이들 값은 하나의 더하기 (+) 또는 빼기 (-) 부호로 시작될 수 있습니다. 구문 및 예제는 29페이지의 『IF 블록』을 참고하십시오.

while block

WHILE 블록. 조건부 문자열 처리에서 루핑을 수행합니다. 구문 및 예제는 61페이지의 『WHILE 블록』을 참고하십시오.

variable reference

이는 이전에 정의된 변수 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예: if VAR='abc' then \$(VAR) returns the value 'abc'. 구문 정보를 보려면 4페이지의 『변수 참조』를 참고하십시오.

string

연속된 영문자와 숫자 및 구두점. 조건 목록의 규정을 따르는 문자열은 개행 문자를 제외한 모든 문자를 포함할 수 있습니다.

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문 정보를 보려면 4페이지의 『변수명』을 참고하십시오.

문맥

WHILE 블록은 다음 문맥에서 찾을 수 있습니다.

- HTML 블록
- REPORT 블록
- ROW 블록
- MACRO_FUNCTION 블록
- IF 블록
- WHILE 블록

제한

WHILE 블록은 다음 요소를 포함할 수 있습니다.

- 주석 블록
- EXEC_SQL 명령문
- IF 블록
- WHILE 블록
- 문자열
- HTML 명령문
- 함수 호출
- 변수 참조
- INCLUDE 명령문

예

예제 1: 테이블에 행을 생성하는 WHILE 블록

```
%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)
```


제2장 변수

Net.Data는 사용자 정의 변수와 Net.Data 변수의 두 가지 유형의 변수를 제공합니다.

66페이지의 『사용자 정의 변수』

응용 프로그램에 대해 정의하는 변수. 다음 작업을 수행하는 변수를 정의할 수 있습니다.

- 66페이지의 『조건 변수』

다른 변수나 문자열의 값에 따라 변수 값을 지정하십시오.

- 68페이지의 『환경 변수』

ENVVAR 언어 구성을 사용하여 환경 변수를 참조합니다.

- 68페이지의 『실행 변수』

EXEC 언어 구성을 사용하여 변수 참조로부터 다른 프로그램을 호출하거나 실행 변수를 사용하여 함수를 호출합니다.

- 69페이지의 『숨겨진 변수』

HTML 소스로부터의 변수 참조를 숨깁니다.

- 70페이지의 『List 변수』

LIST 언어 구성을 사용하여 분리된 일련의 값들을 작성합니다.

- 71페이지의 『Table 변수』

함수로 값 배열을 제공하거나 함수로부터 값 배열을 제공받습니다. 보고서 출력에 사용할 수 있습니다.

Net.Data 변수

기타 처리, 파일 조작, 테이블 처리, 보고서 형식 지정 및 언어 환경에 대한 변수.

일부 변수는 사용자가 정의하거나 수정할 수 있는 값을 가지며 다른 값들은 Net.Data에 의해 정의됩니다. 값에 대한 설명에는 사용자가 값을 정의했는지 여부가 나타납니다. 값이 정의된 방식에 대해서는 변수의 설명을 참고하십시오.

Net.Data는 다음의 변수 유형을 제공합니다.

- 84페이지의 『Net.Data 테이블 처리 변수』

Net.Data 테이블을 처리할 수 있도록 Net.Data에 의해 정의됩니다. SQL 조회 및 함수 호출로부터 자료에 액세스하려면 이들 변수를 사용하십시오. 이들 변수가 지정되어 있지 않으면 REPORT 블록 내에서만 인식됩니다.

- 94페이지의 『Net.Data 보고서 변수』

함수로부터 보고서를 조정할 수 있게 합니다. 이 변수를 참조하기 전에 먼저 정의해야 합니다. 임의의 Net.Data 매크로 블록에서 보고서 변수를 정의하거나 참조할 수 있습니다.

- 102페이지의 『Net.Data 언어 환경 변수』

언어 환경을 사용하여 FUNCTION 블록의 처리 방법을 조정할 수 있게 합니다.

- 72페이지의 『Net.Data 기타 변수』

Net.Data 처리에 영향을 주고, 함수 호출의 상태를 알아내고, 데이터베이스 조회의 결과 세트에 대한 정보를 구하기 위해 Net.Data에 의해 정의됩니다. 몇몇 기타 변수가 Net.Data에 의해 설정되며 변경할 수 없습니다.

여러 Net.Data 변수에 대한 출력은 출력이 수행되는 운영체제에 따라 달라집니다.

Net.Data 매크로에서 최대 256KB까지 상수가 될 수 있습니다. 따라서 매크로 파일에서 변수를 초기화하거나 길이가 256KB보다 큰 생략시 값을 설정할 수 없습니다.

이 장에서 각 변수에 대한 운영체제 지원이 지정됩니다. 다음 목록에는 운영체제 약어가 정의되어 있습니다.

HP-UX

Hewlett Packard UNIX 운영체제

SCO Santa Cruz UNIX 운영체제

SUN Sun Solaris UNIX 운영체제

Win NT

Microsoft의 Windows NT 운영체제

사용자 정의 변수

이 절에서는 사용자 정의 변수에 대해 설명합니다. 매크로 파일에서 이들 변수를 정의해야 합니다.

조건 변수

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

조건 변수의 값은 다른 변수나 문자열의 값에 따라 조건적으로 설정됩니다. 이것을 3진 연산이라고도 합니다.

조건 변수의 구문은 다음과 같습니다.

```
test ? trueValue : falseValue
```

여기에서 다음이 적용됩니다.

test 테스트할 조건.

trueValue

테스트 결과가 참일 때 사용할 값.

falseValue

테스트 결과가 거짓일 때 사용할 값.

예제 1: 가능한 두 값을 사용하여 정의한 조건 변수

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

If varB exists, varA=value_1, otherwise varA=value_2.

예제 2: 변수 참조를 사용하여 정의한 조건 변수

```
varname = ? "${value_1}"
```

이 경우에, *value_1*이 널(null)이면 *varname*이 널(null)이고, 그렇지 않으면 *value_1*로 설정됩니다.

예제 3: LIST문과 WHERE절과 함께 사용되는 조건 변수

```
%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)
}%
```

조건 변수와 LIST 변수를 함께 사용할 때 가장 효과적입니다. 위의 예제는 DEFINE 블록에서 WHERE 절을 설정하는 방법을 보여줍니다. 변수 *cust_inp*와 *prod_inp*는 일반적으로 HTML 형식으로 웹 브라우저가 제공한 HTML 입력 변수입니다. 변수 *where_list*는 두 개의 조건 명령문으로 구성된 LIST 변수로 각 명령문은 웹 브라우저가 제공한 변수를 포함합니다.

웹 브라우저가 변수 *cust_inp* 및 *prod_inp*에 대한 값을 리턴한 경우(예: IBM 및 755C) *where_clause*은 다음과 같습니다.

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

변수 *cust_inp*나 *prod_inp*가 널(NULL)이거나 정의되어 있지 않으면 WHERE 절은 널(NULL) 값을 생략하도록 변경됩니다. 예를 들어 *prod_inp*가 널(NULL)이면 WHERE 절은 다음과 같습니다.

```
WHERE custid = IBM
```

두 값이 널(NULL)이거나 정의되어 있지 않으면 변수 *where_clause*은 널(NULL)이고 *\$(where_clause)*를 포함하는 SQL 조회에는 WHERE 절이 나타나지 않습니다.

환경 변수

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

환경 변수를 사용하면 Net.Data ENVVAR 언어 구성을 사용하여 Net.Data가 실행되고 있는 프로세스의 환경 변수를 참조합니다.

예제 1: 변수가 환경 변수 값으로 할당됩니다.

```
%define SERVER_NAME=%ENVVAR
```

```
...
```

```
The server is $(SERVER_NAME)
```

환경 변수 `SERVER_NAME`은 현재 서버명의 값을 가지며 이 예제에서는 `www.software.ibm.com`이 됩니다.

```
The server is www.software.ibm.com
```

ENVVAR문에 대한 자세한 내용은 13페이지의 『ENVVAR 명령문』을 참고하십시오.

실행 변수

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

실행 변수를 통해 실행 변수 기능을 사용하여 변수 참조로부터 다른 프로그램을 호출할 수 있습니다. 실행 변수는 EXEC 언어 요소를 사용하여 Net.Data 매크로에 정의되어 있습니다. EXEC 언어 요소에 대한 자세한 내용은 14페이지의 『EXEC 블록 또는 명령문』을 참고하십시오.

Net.Data가 매크로 파일에서 실행 변수를 만나면 다음 메서드를 사용하여 참조된 실행 프로그램을 찾습니다.

1. Net.Data 초기화 파일에서 EXEC_PATH를 탐색합니다. EXEC_PATH에 대한 자세한 내용은 *Net.Data Administration and Programming Guide*에서 구성 장을 참고하십시오.
2. Net.Data가 프로그램을 찾지 못하면 시스템 PATH 환경 변수에서 정의한 디렉토리를 탐색합니다. 실행 프로그램을 찾으면 Net.Data는 이 프로그램을 실행합니다.

예제 1: 실행 변수 정의

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

변수 `runit`가 실행 프로그램 `testProg`를 실행하도록 정의되어 있으므로 `runit`는 실행 변수가 됩니다.

Net.Data는 Net.Data 매크로에서 실행 변수 참조를 발견하면 실행 프로그램을 실행합니다. 예를 들어 프로그램 *testProg*는 Net.Data 매크로에서 변수 *runit*에 대한 실행 변수 참조가 수행될 때 실행됩니다.

단순 메서드는 다른 변수 정의로부터 실행 변수를 참조하는 것을 나타냅니다. 예제 2는 이 메서드를 보여줍니다. 변수 *date*는 실행 변수로 정의되며 *dateRpt*는 이 실행 변수를 포함하는 변수 참조로 정의됩니다.

예제 2: 변수 참조에 해당하는 실행 변수

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

Net.Data는 변수 참조 *\$(dateRpt)*를 분석할 때 실행 날짜를 탐색하고, 프로그램을 실행하고 다음을 리턴합니다.

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

실행 변수는 이 변수가 호출하는 실행 프로그램의 출력 값으로는 설정되지 않습니다. 이전 예제를 사용하면 날짜 값은 널(NULL)이 됩니다. 이 값을 DTW_ASSIGN 함수 호출시 사용하여 다른 변수에 할당하면 이 할당이 수행된 후에 새로운 변수의 값도 널(NULL)이 됩니다. 실행 변수의 유일한 목적은 정의하는 프로그램을 호출하는 것입니다.

또한 변수 정의에서 프로그램명과 함께 매개변수를 지정하여 실행할 프로그램에 매개변수를 제공할 수 있습니다.

예제 3: 매개변수를 포함하는 실행 변수

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

*distance*와 *time*의 값이 프로그램 *calcMPH*로 전달됩니다.

숨겨진 변수

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

숨겨진 변수를 통해 HTML 소스에 실제 변수 값이 숨겨져 있는 경우에도 변수를 참조할 수 있습니다. 숨겨진 변수를 사용하려면 다음과 같이 하십시오.

1. 숨기려는 각 문자열에 대해 변수를 정의하십시오.
2. 변수를 참조하려면, 변수가 참조되는 HTML 블록에서 하나의 달러 기호 대신 두 개의 달러 기호를 사용하십시오. 예를 들면, *\$(X)* 대신 *\$(X)*를 사용하십시오.

예제 1: HTML 형식으로 숨겨진 변수

```
%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
%}
.
.
.

```

HTML 형식이 웹 브라우저에 나타나면 `$(name)` 와 `$(addr)`가 각각 `$(name)` 와 `$(addr)`로 대체되어 실제 테이블 및 컬럼명이 HTML 형식으로 나타나지 않으므로 실제 변수명이 숨겨져 있다는 사실을 아무도 알 수 없습니다. 사용자가 양식을 제출하면, HTML(REPORT) 블록이 호출됩니다. @mySelect()가 FUNCTION 블록을 호출하면 `$(Field)`는 SQL문에서는 `customer.name`으로, SQL 조회에서는 `customer.addr`로 대체됩니다.

List 변수

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

list 변수를 사용하여 구분된 값 문자열을 작성할 수 있습니다. 이들 중 일부는 WHERE 또는 HAVING 절에서 발견된 것과 같은 복수의 항목을 사용하여 SQL 조회를 구성하는 데 유용합니다.

공백은 중요합니다. 일반적으로 값의 양끝에 공백을 두어 값을 분리합니다. 대부분의 조회에서는 부울 또는 산술 연산자(예: AND, OR 및 >)를 사용합니다. 구문 및 자세한 정보를 보려면 42페이지의 『LIST 명령문』을 참고하십시오.

예제 1: 조건 변수, 숨겨진 변수 및 list 변수의 사용

```

%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()
%}

```

HTML 형식에서 선택된 박스가 없으면 조건이 널(NULL)이므로 조회에서 whereClause도 널(NULL)이 됩니다. 그렇지 않으면 whereClause는 부울 연산자 OR로 분리된 선택된 값을 가집니다. 예를 들어, 세 도시가 모두 선택 되었으면 SQL 조회는 다음과 같습니다.

```

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

```

예제 2: 값 분리자

```

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

```

이 예제에서 내재적 변수 VLIST는 두 개의 큰 따옴표와 하나의 OR 막대 " | "를 값 분리자로 사용합니다. 문자열 값은 따옴표로 분리됩니다.

Table 변수

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

Table 변수에는 값의 배열과 연관된 컬럼명이 들어 있습니다. 배열의 각 요소는 행을 나타냅니다. 그룹 값을 함수에 전달하려면 테이블 변수를 사용하십시오. 함수의 REPORT 블록에서 테이블의 개별 요소(행)를 참조할 수 있습니다. Table 변수는 SQL 함수의 출력과 보고서 입력에 사용되지만 이들 변수를 IN, OUT 또는 INOUT 매개변수로 비SQL 함수에 제공할 수도 있습니다. 테이블은 SQL 함수에 OUT 매개변수로 전달할 수도 있습니다. 구문 및 자세한 정보를 보려면 59페이지의 『TABLE 명령문』을 참고하십시오.

예제 1: REXX 프로그램에 제공된 SQL 결과 세트

```

%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
end
%}

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

```

HTML REPORT 블록은 SQL 조회를 호출하고, table 변수에 그 결과를 저장한 다음 이 변수를 REXX 함수에 제공합니다.

Net.Data 기타 변수

이들 변수는 파일 위치 및 날짜에 대한 정보를 판별할 뿐 아니라 Net.Data 처리에 영향을 주고, 함수 호출의 상태를 알아내고, 데이터베이스 조회의 결과 세트에 대한 정보를 얻는 데 사용할 수 있는 Net.Data 정의 변수입니다. Net.Data 매크로를 테스트할 때 함수에 이 변수를 쓰거나 사용할 경우에 이 변수가 유용하다는 것을 알 수 있습니다.

- 73페이지의 『DTW_CURRENT_FILENAME』
- 74페이지의 『DTW_CURRENT_LAST_MODIFIED』
- 75페이지의 『DTW_DEFAULT_MESSAGE』
- 76페이지의 『DTW_LOG_LEVEL』
- 77페이지의 『DTW_MACRO_FILENAME』
- 78페이지의 『DTW_MACRO_LAST_MODIFIED』
- 79페이지의 『DTW_MP_PATH』
- 80페이지의 『DTW_MP_VERSION』
- 81페이지의 『DTW_PRINT_HEADER』
- 82페이지의 『DTW_REMOVE_WS』
- 83페이지의 『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

목적

현재 입력 파일의 이름 및 확장자. 입력 파일은 INCLUDE문에 지정된 Net.Data 매크로나 파일입니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

```
<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

목적

현재 파일이 마지막으로 수정된 날짜 및 시간. 현재 파일은 Net.Data 매크로 파일이나 INCLUDE 명령문에 지정된 파일일 수 있습니다. 출력 형식은 Net.Data가 수행되는 시스템에 따라 다릅니다.

이 변수는 사전 정의된 변수이며 그 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

```
<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			

목적

오류가 발생할 때 내장 함수나 언어 환경 호출로부터 리턴된 메세지 텍스트를 포함합니다.

Net.Data 매크로 파일의 일부분에 DTW_DEFAULT_MESSAGE 변수를 사용할 수 있습니다.

이 변수는 사전 정의 변수로 값을 수정하지 않는 것이 좋습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: 함수가 성공적으로 완료되었는지를 알려주는 메세지

```
@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
```

예제 2: 함수가 0이 아닌 리턴 코드를 리턴할 때의 생략시 텍스트

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

함수가 0이 아닌 리턴 코드를 리턴할 경우 생략시 오류 메세지가 나타납니다.

DTW_LOG_LEVEL

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

목적

Net.Data가 로그 파일에 기록한 메시지 레벨.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정할 수 있습니다.

요구사항: Net.Data 초기화 파일에 DTW_LOG_DIR을 정의하여 로깅을 초기화하십시오. 그렇지 않으면 Net.Data는 사용자가 매크로 파일에 DTW_LOG_LEVEL 변수를 지정할 때 메시지를 기록하지 않습니다.

값

DTW_LOG_LEVEL="level"

표 1. DTW_LOG_LEVEL 값

값	설명
OFF	Net.Data는 오류를 기록하지 않습니다. OFF가 생략시 값입니다.
ERROR	Net.Data는 오류 메시지를 기록합니다.
WARNING	Net.Data는 오류 메시지와 함께 경고를 기록합니다.

예

```
%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

목적

현재 Net.Data 매크로 파일의 이름 및 확장자.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

<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

목적

Net.Data 매크로가 마지막으로 수정된 날짜와 시간. 출력 형식은 Net.Data가 수행되는 시스템에 따라 다릅니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

<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

목적

Net.Data 실행 파일의 경로와 이름. 시스템에 따라 출력은 다음의 샘플 경로 및 이름처럼 나타납니다.

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

이 변수는 사전 정의된 변수이며 그 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

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

목적

서버에서 실행 중인 Net.Data의 버전 및 릴리스. 출력은 다음 형식을 가집니다.

Net.Data Version 2.1

이 변수는 사전 정의된 변수이며 그 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

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

목적

파일 헤더에 대한 텍스트를 지정합니다.

Net.Data는 텍스트를 표시하기 전에 한 번 이 변수를 읽고 다시 보지 않으므로 Net.Data가 웹 브라우저로 송신된 텍스트를 처리하기 전에 이 변수를 설정해야 합니다. DTW_PRINT_HEADER 변수를 변경해도 Net.Data가 브라우저로 송신된 후에는 무시됩니다.

OS/390 사용자: DTW_PRINT_HEADER를 사용하여 자체의 헤더(DTW_PRINT_HEADER="NO")를 생성하는 경우, DTW_REMOVE_WS="NO"를 설정해야 합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_PRINT_HEADER="YES"|"NO"

표 2. DTW_PRINT_HEADER 값

값	설명
YES	Net.Data는 HTTP 헤더에 대해 텍스트 Content-type: text/html을 인쇄합니다. YES가 생략시 값입니다.
NO	Net.Data는 HTTP 헤더를 인쇄하지 않습니다. 사용자 정의 HTTP 헤더 정보를 생성할 수 있습니다.

예

이 변수를 사용하는 가장 일반적인 방법 중의 하나는 Net.Data 매크로가 쿠키(cookies)를 보낼 수 있도록 하는 것입니다. 쿠키를 설정하려면 DTW_PRINT_HEADER 변수가 NO로 설정되며 처음 세 행이 내용 유형 헤더, Set-Cookie 명령문 및 공백 행이어야 합니다.

예제 1: 쿠키를 송신하도록 Net.Data 설정

```
%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

목적

도표 작성기, 공백 및 개행 문자로 인해 생성되는 여분의 공백을 줄여 동적으로 생성된 웹 페이지의 크기를 줄입니다.

DEFINE 블록에 이 변수의 값을 지정하십시오.

<PRE></PRE> 태그 사용: 이 변수를 YES로 설정하면 인쇄된 공백의 양과 유형이 달라집니다. 이 변수가 YES로 설정되면 <PRE></PRE> 태그를 사용하는 HTML 페이지의 부분이 예상대로 표시되지 않을 수 있습니다.

OS/390 사용자:

1. DTW_PRINT_HEADER를 사용하여 자체의 헤더(DTW_PRINT_HEADER="NO")를 생성하는 경우, DTW_REMOVE_WS="NO"를 설정해야 합니다.
2. Net.Data 초기화 파일에 이 변수를 설정하여 모든 매크로에 대해 값을 지정하십시오. 값을 매크로 파일에 지정하여 교체할 수 있습니다. DTW_REMOVE_WS가 매크로 파일에 정의되어 있지 않으면 초기화 파일의 값을 사용합니다.

값

DTW_REMOVE_WS="YES"|"NO"

표 3. DTW_REMOVE_WS 값

값	설명
YES	Net.Data는 둘 이상의 공백을 하나의 개행 문자로 압축하고 더 짧아진 HTML 결과 페이지를 생성합니다.
NO	Net.Data는 공백을 압축하지 못합니다. NO가 생략시 값입니다.

예

예제 1: 공백 압축

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

목적

내장 함수 호출이나 언어 환경 호출에 의해 리턴된 리턴 코드. Net.Data는 이 값을 사용하여 MESSAGE 블록을 처리합니다. 이 변수를 사용하여 함수 호출이 성공했는지 실패했는지를 판별할 수 있습니다. 값이 0이면, 함수 호출이 성공적으로 완료되었음을 나타냅니다.

Net.Data 매크로 파일의 일부분에서 RETURN_CODE 변수를 참조할 수 있습니다.

이 값은 사전 정의된 값으로 수정하지 않는 것이 좋습니다. 이 값을 변수 참조로 사용하십시오.

예

예제 1: 함수가 성공적으로 완료되었는지를 알려주는 메시지

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

예제 2: 리턴 코드가 0이 아닐 때의 생략시 메시지

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

함수가 0이 아닌 리턴 코드를 리턴하면 생략시 메시지가 표시됩니다.

Net.Data 테이블 처리 변수

Net.Data는 다른 언급이 없는 경우 REPORT 및 ROW 블록에서 사용하도록 이들 변수를 정의합니다. 조회가 리턴하는 값을 참조하려면 이 변수를 사용하십시오.

제한사항: DEFINE 절에는 이러한 변수 값을 정의하지 않도록 하십시오.

- 85페이지의 『*Nn*』
- 86페이지의 『*NLIST*』
- 87페이지의 『*NUM_COLUMNS*』
- 88페이지의 『*NUM_ROWS*』
- 89페이지의 『*ROW_NUM*』
- 90페이지의 『*TOTAL_ROWS*』
- 91페이지의 『*V_columnName*』
- 93페이지의 『*Vn*』
- 92페이지의 『*VLIST*』

Nn

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

목적

함수 호출이나 컬럼 n에 대한 조회에 의해 리턴된 컬럼명. Nn은 REPORT 및 ROW 블록에서 유효합니다.

Net.Data는 테이블의 각 컬럼에 대한 변수를 지정합니다. 따라서 이 변수를 변수 참조에 사용하고 참조하려는 컬럼의 이름을 지정하십시오.

예

예제 1: 컬럼명에 대한 변수 참조

The name of column 2 is \$(N2).

예제 2: DTW_ASSIGN을 사용하여 REPORT 블록 외부에서 사용할 컬럼명의 값을 저장합니다.

```
%define col1=""
...
%function (DTW_SQL) myfunc() {
    select * from atable
    %report {
        @dtw_assign(col1, N1)
        %row{ %}
    %}
%}

%html(report) {
    @myfunc()
    The column name for for the first column is $(col1)
%}
```

이 예제는 DTW_ASSIGN을 사용하여 REPORT 블록 외부에서 해당 변수를 사용할 수 있는 방법을 보여줍니다. 자세한 내용은 159페이지의 『DTW_ASSIGN』을 참고하십시오.

예제 3: 컬럼명을 정의하기 위한 HTML 테이블 내의 Nn

```
%REPORT{
<H2>Product directory</H2>
<TABLE BORDER=1 CELLPADDING=3>
<TR><TD>$(N1)</TD><TD>$(N2)</TD><TD>$(N5)</TD>
%ROW{
<TR><TD>$(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

목적

함수 호출 또는 조회의 결과로 리턴된 모든 컬럼명 목록을 포함합니다. 생략시 분리자는 공백입니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: ALIGN을 사용한 컬럼명 목록

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

컬럼명 목록은 ALIGN이 YES로 설정된 컬럼명 사이에 공백을 사용합니다.

예제 2: 분리자를 " | "로 변경하는 %LIST 변수

```
%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

목적

Net.Data가 보고서 블록에서 처리하고 있는 테이블 컬럼의 번호. 이들 컬럼은 함수 호출이나 조회에 의해 리턴됩니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: NLIST와 함께 변수 참조로 사용되는 NUM_COLUMNS

```
%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			

목적

Net.Data가 REPORT 블록에서 처리하고 있는 테이블의 행 수. 행의 수는 자료를 보유하고 있는 Net.Data 테이블에 대해 정의된 *upper limit* 매개변수의 값에 따라 달라집니다. 예를 들어 *upper limit*가 30으로 설정되거나 SELECT문이 1000개의 행을 리턴하면 NUM_ROWS 값은 30이 됩니다. 또한 *upper limit*가 30으로 설정되고 SELECT문이 20개의 행을 리턴하고 NUM_ROWS는 20이 됩니다. TABLE문 및 *upper limit* 매개변수에 대한 자세한 내용은 59페이지의 『TABLE 명령문』을 참고하십시오.

NUM_ROWS는 START_ROW_NUM이 언어 환경에 제공되지 않는 한 값 START_ROW_NUM에 의해 영향받지 않습니다. 예를 들어 START_ROW_NUM이 5(웹 페이지에 표시된 테이블이 행 5로부터 채워져야 함을 나타냄)로 설정되고 SELECT문이 25개의 행을 리턴하면 NUM_ROWS는 21이 아닌 25로 설정됩니다. 처음 네 개의 행이 테이블에서 삭제되거나 값 NUM_ROWS에 포함됩니다. 그러나 START_ROW_NUM이 언어 환경에 제공되면 START_ROW_NUM에서 지정하는 행에서 시작하는 행 수만 NUM_ROWS에 포함됩니다. 위의 예제에서 NUM_ROWS는 21로 설정됩니다.

NUM_ROWS는 REPORT 및 ROW 블록에서 유효합니다.

이 변수는 사전 정의된 변수이며 그 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: REPORT 블록에서 처리 중인 이름의 수를 표시합니다.

```
%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

목적

Net.Data 테이블에서 행이 처리될 때마다 Net.Data가 그 값을 증가시키는 테이블 변수. 이 변수는 카운터로 작동하며 그 값은 처리 중인 현재 행의 번호를 나타냅니다.

이 변수는 사전 정의된 변수이며 그 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

RPT_MAX_ROWS는 값 ROW_NUM에 영향을 미칠 수 있습니다. 예를 들어 테이블에 행이 100개 있고 RPT_MAX_ROWS가 20으로 설정되면 행 20이 마지막으로 처리된 행이므로 ROW_NUM의 마지막 값은 20이 됩니다.

ROW_NUM은 ROW 블록에서만 유효합니다.

예

예제 1: ROW_NUM을 사용하여 테이블의 각 행에 레이블을 붙여서 HTML 출력으로 컬럼을 채웁니다.

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

REPORT 블록은 아래에 나타나는 것과 같은 테이블을 생성합니다.

행 번호	고객
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

목적

TABLE 언어 구성에 대한 값 *upper_limit*에 관계없이 조회가 리턴하는 행의 총 수. 예를 들어 RPT_MAX_ROWS가 최대 20행을 표시하도록 설정되어 있으나 조회가 100개의 행을 리턴하면 이 변수는 ROW 처리 후에 100으로 설정됩니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

운영체제의 차이점:

- OS/400 운용 시스템에서 이 변수는 REPORT 또는 ROW 블록의 모든 위치에서 참조할 수 있습니다.
- OS/2, Windows NT 및 UNIX 운용 시스템에서는 이 변수를 REPORT 꼬리말에서만 참조할 수 있습니다.

필수: 이 변수를 사용하려면 DTW_SET_TOTAL_ROWS를 YES로 설정해야 합니다. 자세한 내용은 112페이지의 『DTW_SET_TOTAL_ROWS』를 참고하십시오.

예

예제 1: 발견된 총 이름의 수를 표시합니다.

```
%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

목적

현재 행에 대해 지정된 컬럼명에 대한 값. 이 변수는 정의되지 않은 컬럼명에 대해서는 설정되지 않습니다. 동일한 이름을 가진 두 개의 컬럼명이 들어 있는 조회에서는 예상치 않은 결과가 나타납니다. 중복된 컬럼명의 이름을 변경하려면 SQL에 AS 절을 사용하십시오. ROW 블록에서는 V_columnName만 유효합니다.

이 변수의 값을 변수 참조로 사용하여 지정하고 컬럼의 실제 이름을 대체하십시오.

값

V_columnName

표 4. V_columnName 값

값	설명
columnName	데이터베이스 테이블의 현재 행의 컬럼명.

예

예제 1: V_columnName을 변수 참조로 사용

You have selected \$(V_destcity).

VLIST

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

목적

ROW 블록에서 처리 중인 현재 행에 대한 모든 필드 값 목록. VLIST는 ROW 블록에서만 유효합니다. 생략시 분리자는 공백입니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: 목록 태그를 사용하여 조회 결과 표시

```
%DEFINE ALIGN="YES"

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

예제 2: list 변수를 사용하여 분리자를 <P>로 변경

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

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

Vn

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

목적

함수 호출이나 1에서 n 까지의 필드에 대한 SQL 조회에 의해 리턴된 각 행에 대한 필드값. Vn 은 ROW 블록에서만 인식됩니다.

Net.Data는 테이블의 각 필드에 대한 변수를 지정합니다. 따라서 이 변수를 변수 참조에 사용하고 참조할 컬럼의 이름을 지정하십시오. 블록 외부에서 이 변수를 사용하려면 값 Vn 을 이전에 지정된 전역 변수나 OUT 또는 INOUT 함수 매개변수에 지정하십시오.

예

예제 1: HTML 테이블을 표시하는 보고서

```
%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.
%}
```

두 번째 컬럼은 전자 메일 주소를 나타냅니다. 링크를 클릭하여 사람에게 메세지를 송신할 수 있습니다.

Net.Data 보고서 변수

이 변수를 사용하여 보고서를 조정할 수 있습니다. 이 변수를 사용하기에 앞서 이를 정의해야 합니다.

- 95페이지의 『ALIGN』
- 96페이지의 『DTW_DEFAULT_REPORT』
- 97페이지의 『DTW_HTML_TABLE』
- 98페이지의 『RPT_MAX_ROWS』
- 99페이지의 『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

목적

테이블 처리 변수 **NLIST** 및 **VLIST**에서 사용되는 맨 앞 및 맨 뒤 공백을 제어합니다. 이를 **YES**로 설정하면 **ALIGN**은 화면에 맞게 테이블 처리 변수를 정렬하도록 공백을 채웁니다. **HTML** 형식으로 조회 결과를 포함시키거나 조치 형식을 지정하려면 **Net.Data**가 맨 앞 및 맨 뒤 공백을 **report** 변수에 사용하지 못하도록 생략시 값 **NO**를 사용하십시오.

DEFINE문을 사용하거나 **@DTW_ASSIGN()** 함수를 통해 이 변수의 값을 지정하십시오.

값

ALIGN="YES" | "NO"

표 5. **ALIGN** 값

값	설명
YES	Net.Data 는 화면에 맞게 변수가 정렬되도록 맨 앞과 맨 뒤에 공백을 추가하여 변수를 보고합니다.
NO	Net.Data 는 맨 앞이나 맨 뒤에 공백을 추가하지 않습니다. NO 가 생략시 값입니다.

예

예제 1: **ALIGN** 변수를 사용하여 공백으로 각 컬럼 구분

```
%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

목적

NO로 설정될 때, REPORT 블록이 없으며 브라우저에 결과를 표시하지 않는 함수에 대해 Net.Data가 생성하는 생략시 보고서를 무시합니다. 예를 들어 테이블 변수로 함수 호출의 결과를 수신하고 처리를 위해 이 결과를 다른 함수로 전달하는 경우 생략시 보고서를 무시하는 것이 좋습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_DEFAULT_REPORT="YES"|"NO"

표 6. DTW_DEFAULT_REPORT 값

값	설명
YES	Net.Data는 REPORT 블록 없이 함수에 대한 생략시 보고서를 생성하고 브라우저에 그 결과를 표시합니다. YES가 생략시 값입니다.
NO	Net.Data는 REPORT 블록 없이 함수에 대한 생략시 보고서를 무시합니다.

예

예제 1: Net.Data에 의해 생성된 생략시 보고서 무시

```
%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

목적

텍스트 유형 형식에 테이블을 표시하는 대신 HTML 테이블에 결과를 표시합니다(즉, PRE 태그 대신 TABLE 태그 사용).

생성된 TABLE 태그에는 경계 및 셀 패딩 스펙이 포함됩니다.

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

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_HTML_TABLE="YES"|"NO"

표 7. DTW_HTML_TABLE 값

값	설명
YES	HTML 테이블 태그를 사용하여 테이블 자료를 표시합니다.
NO	PRE 태그를 사용하여 텍스트 형식으로 테이블 자료를 표시합니다. NO가 생략시 값입니다.

예

예제 1: HTML 태그를 사용하여 SQL 함수의 결과를 표시합니다.

```
%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

목적

함수 REPORT 블록에 의해 생성된 테이블에 표시되는 행의 수를 지정합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

OS/400, Windows NT, OS/2 및 UNIX 사용자: 이 변수를 사용하려면 이 변수가 사용 중인 데이터베이스 언어 환경에 대한 ENVIRONMENT문에 IN 변수로 초기화 파일에 포함되어 있는지 확인하십시오. 데이터베이스 언어 환경 명령문에 대한 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장을 참고하십시오.

값

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

표 8. RPT_MAX_ROWS 값

값	설명
ALL	함수 호출에 의해 생성된 테이블에 표시될 행 수에 제한이 없음을 지정합니다. 모든 행이 표시됩니다.
0	테이블의 모든 행이 표시되도록 지정합니다. 이 값은 ALL을 지정하는 것과 같습니다.
number	함수 호출에 의해 생성된 테이블에 표시될 최대 행 수를 나타내는 양의 정수. FUNCTION 블록에 REPORT 및 ROW 블록이 들어 있으면 이 수는 ROW 블록이 실행되는 횟수를 지정합니다.

예

예제 1: DEFINE문에 RPT_MAX_ROWS는 정의합니다.

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

위의 메서드는 함수가 리턴하는 행의 수를 20개로 제한합니다.

예제 2: HTML 입력을 사용하여 HTML 형식으로 변수를 정의합니다.

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

위 예제의 행들은 FORM 태그에 놓일 수 있으므로 응용 프로그램 사용자는 조회로부터 리턴될 행의 수를 설정할 수 있습니다.

START_ROW_NUM

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

목적

보고서에 Net.Data 테이블의 결과를 표시하기 시작할 행 번호를 지정합니다. 이 변수를 RPT_MAX_ROWS와 함께 사용하여 큰 결과 세트를 가진 조회를 더 작은 테이블로 분리하고 다음 버튼을 사용하여 결과 테이블 간에 이동할 수 있습니다.

제한사항: 성능을 위해 Net.Data는 데이터베이스 언어 환경이 전체 결과 세트를 Net.Data로 리턴하지 못하도록 이 언어 환경에 START_ROW_NUM을 제공합니다. 이 변수를 자동으로 전달하려면 초기화 파일의 데이터베이스 언어 환경 ENVIRONMENT문에 IN 변수로 포함시키십시오. 이 변수가 ENVIRONMENT문에서 생략되면 검색이 시작될 행 번호는 결과 세트에서 처음 행으로 가정됩니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

START_ROW_NUM="number"

표 9. START_ROW_NUM 값

값	설명
<i>number</i>	<p>보고서 표시를 시작할 행 번호를 나타내는 양의 정수.</p> <p>초기화 파일의 데이터베이스 언어 환경의 환경 명령문에 START_ROW_NUM 이 지정되면 이 번호는 데이터베이스 언어 환경이 처리한 결과 세트의 해당 행 번호를 지정합니다.</p> <p>START_ROW_NUM을 언어 환경에 제공하지 않으면 이 수는 보고서를 표시하는 데 사용된 Net.Data 테이블의 행 번호를 지정합니다.</p>

예

예제 1: HTML 형식의 다음 및 이전 버튼을 사용하여 화면 이동

```
%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_ADD(START_ROW_NUM, RPT_MAX_ROWS) > totalSize)
            @DTW_ASSIGN(includeNext, "NO")
            @DTW_ASSIGN(includeLast, "NO")
        %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
</form>
</center>
%}

```

Net.Data 언어 환경 변수

함수에서 이들 변수를 사용하여 언어 환경에 의해 FUNCTION 블록이 처리되는 방식을 조정할 수 있습니다. 이들 변수는 참조하기 전에 정의해야 합니다.

- 103페이지의 『DATABASE』
- 105페이지의 『DB_CASE』
- 106페이지의 『DB2PLAN』
- 107페이지의 『DB2SSID』
- 108페이지의 『DTW_APPLET_ALTTEXT』
- 109페이지의 『DTW_EDIT_CODES』
- 110페이지의 『DTW_MBMODE』
- 111페이지의 『DTW_SAVE_TABLE_IN』
- 112페이지의 『DTW_SET_TOTAL_ROWS』
- 114페이지의 『LOCATION』
- 115페이지의 『LOGIN』
- 116페이지의 『NULL_REPORT_FIELD』
- 117페이지의 『PASSWORD』
- 118페이지의 『SHOWSQL』
- 119페이지의 『SQL_STATE』
- 120페이지의 『TRANSACTION_SCOPE』

DATABASE

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

목적

데이타베이스 함수를 호출할 때 액세스할 데이타베이스나 ODBC 자료 소스를 지정합니다. 이 변수는 복수의 데이타베이스나 ODBC 자료 소스에 액세스하기 위해 매크로에서 여러 번 변경할 수 있습니다.

OS/400 운용 시스템: 이 변수는 선택적입니다. 생략시에 Net.Data는 DATABASE=『*LOCAL』을 지정합니다. DTW_SQL 언어 환경은 국지 관계형 데이타베이스 디렉토리 항목을 사용합니다.

Windows NT, OS/2 및 UNIX 운영체제: DTW_ORA (Oracle) 언어 환경을 사용할 때를 제외하고 데이타베이스 함수를 호출하기 전에 이 변수를 정의하십시오. 또한 같은 HTML 블록에서와 같은 언어 환경을 통해 복수의 데이타베이스에 액세스할 때는 Live Connection을 사용해야 합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DATABASE="dbname"

표 10. DATABASE 값

값	설명
dbname	Net.Data가 연결하는 데이타베이스의 이름.

예

예제 1: SQL 연산을 위해 CELDIAL 데이타베이스로 연결하도록 지정합니다.

```
%DEFINE DATABASE="CELDIAL"
```

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

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

함수 getRpt를 호출할 때 데이타베이스 CELDIAL에 액세스할 수 있습니다.

예제 2: DTW_ASSIGN으로 이전 DATABASE 정의 교체

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

HTML 블록은 이전 DATABASE 값에 관계없이 데이터베이스 DB2D1을 조회합니다.

DB_CASE

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

목적

SQL 명령에 사용할 대소문자를 지정하고 모든 문자를 대문자나 소문자로 변환합니다. 이 변수가 정의되어 있지 않은 경우, 생략시 조치는 SQL 명령 문자를 변환하지 않는 것입니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DB_CASE="UPPER"|"LOWER"

표 11. DB_CASE 값

값	설명
UPPER	모든 SQL 명령 문자를 대문자로 변환합니다.
LOWER	모든 SQL 명령 문자를 소문자로 변환합니다.

예

예제 1: 모든 SQL 명령에 대문자를 지정합니다.

```
%DEFINE DB_CASE="UPPER"
```

DB2PLAN

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

목적

국지 DB2 서브시스템으로 연결 계획을 할당합니다. 이 변수는 Net.Data가 액세스할 국지 DB2 서브시스템에서 Net.Data SQL 언어 환경에 대한 계획의 이름을 지정합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

요구사항: 이 변수는 DTW_SQL ENVIRONMENT 명령문의 Net.Data 초기화 파일이나 경우에 따라 매크로 파일에 지정되어야 합니다. 이 변수가 OS/390 초기화 파일에 대한 Net.Data 또는 매크로에 지정되어 있지 않고 이 초기화 파일에 없으면 SQL 함수를 실행하려 할 때 오류가 발생합니다.

값

DB2PLAN="*plan_name*"

표 12. DB2PLAN 값

값	설명
<i>plan_name</i>	DB2 계획의 이름. 이 이름은 8문자 이하가 될 수 있습니다.

예

예제 1: DEFINE문에 계획을 지정합니다.

```
%DEFINE DB2PLAN="DTWGAV21"
```

DB2SSID

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

목적

국지 DB2 서브시스템으로의 연결을 설정합니다. 변수는 Net.Data가 액세스 할 국지 DB2 서브시스템의 서브시스템 ID를 지정합니다. 각 매크로당 하나의 국지 데이터베이스 연결만 허용됩니다.

요구사항: 이 변수는 Net.Data 초기화 파일이나 경우에 따라 매크로 파일에 지정되어야 합니다. 이 변수가 OS/390용 Net.Data 초기화 파일에도 지정되어 있지 않고 매크로에도 정의되어 있지 않은 경우에, 매크로가 SQL 함수를 실행하려 하면 오류가 발생합니다.

값

DB2PLAN="*subsystem_id*"

표 13. DB2SSID 값

값	설명
<i>subsystem_id</i>	DB2 서브시스템의 이름. 이 이름은 8문자 이하가 될 수 있습니다.

예

예제 1: DEFINE문에 서브시스템 ID를 지정합니다.

```
%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

목적

APPLET 태그를 인식하지 않으며 애플릿 언어 환경에서 사용되는 브라우저에 HTML 태그 및 텍스트를 표시합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_APPLET_ALTTEXT="*HTML_text_and_tags*"

표 14. DTW_APPLET_ALTTEXT 값

값	설명
<i>HTML_text_and_tags</i>	APPLET 태그를 인식하지 않는 브라우저에 대한 HTML 태그 및 텍스트.

예

예제 1: 웹 브라우저 제한 사항을 나타내는 대체 텍스트.

```
%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			

목적

DTW_SQL 언어 환경에 대한 SQL 조작의 결과로 리턴된 NUMERIC, DECIMAL, INTEGER 및 SMALLINT 자료 유형을 변환합니다. 변수 DTW_EDIT_CODES는 DTW_SQL LE가 작성할 테이블의 결과 컬럼에 해당하는 문자열입니다. 예를 들어 DTW_EDIT_CODES의 다섯번째 문자는 이 컬럼이 지원되는 유형을 가질 때 결과 세트의 다섯번째 컬럼에 적용됩니다. 이 단일 문자는 *Data Description Specification Reference*에 정의된 지원 시스템 제공 편집 코드 중 하나가 됩니다.

예를 들어 DECIMAL(6,0) 필드는 보통 문자열 '112698'로 표시됩니다. 변수 DTW_EDIT_CODES의 해당 컬럼에 대해 편집 코드 'Y'를 지정하면 'Y'가 날짜 '11/26/98'을 나타내는 문자열로 표시됩니다.

팁: 숫자가 아닌 문자(예: 쉼표나 통화 기호)를 가진 문자열을 가져오는 사용자 제공 편집 코드를 컬럼에 적용하면 이 문자열이 Net.Data 매크로에서 나중에 처리되도록 서버로 다시 송신될 때 구문 오류가 발생할 수 있습니다. 예를 들어 비슷자 컬럼값이 이후에 DTW_SQL 함수 호출에서 숫자 비교에 사용되면 구문 오류가 발생할 수 있습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_EDIT_CODES="edit_code"

표 15. DTW_EDIT_CODES 값

값	설명
edit_code	SQL 언어 환경이 작성하는 테이블의 결과 컬럼에 해당하는 문자열을 지정합니다.

예

예제 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

목적

생략시 언어 환경에서 사용하는 문자열 및 단어 함수에 대한 MBCS(복수 바이트 문자 세트) 지원을 제공합니다. 이 변수를 Net.Data 초기화 파일에 설정할 수 있으나 매크로 파일에 이를 사용하면 현재 설정을 덮어쓸 수 있습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

OS/400 사용자: OS/400용 Net.Data는 MBCS 지원 기능을 자동으로 작동 가능하게 할 수 있으며 이 변수를 필요로 하지 않습니다. OS/400용 Net.Data는 OS/400 운용 시스템으로 이주되는 매크로 파일에서는 이 변수를 무시합니다.

값

DTW_MBMODE="YES"|"NO"

표 16. DTW_MBMODE 값

값	설명
YES	문자열 및 단어 함수에 대한 MBCS 지원을 지정합니다.
NO	문자열 및 단어 함수가 MBCS 지원을 가지지 않도록 지정합니다. NO가 생략시 값입니다.

예

<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

목적

SQL 언어 환경이 조회로부터 리턴된 테이블 자료를 저장하는 데 사용하는 table 변수를 식별합니다. 이 테이블은 나중에 테이블 자료를 분석하는 REXX 프로그램에서 사용될 수 있습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

DTW_SAVE_TABLE_IN="table_name_var"

표 17. DTW_SAVE_TABLE_IN 값

값	설명
table_name_var	SQL 언어 환경이 조회로부터 리턴된 테이블 자료를 저장하는 데 사용하는 테이블의 이름.

예

예제 1: REXX 호출에서 사용되는 사전 정의 테이블 변수

```
%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 블록은 테이블의 자료를 분석하는 데 table 변수 theTable 을 사용하는 REXX 프로그램 anzTbl.cmd를 호출합니다. 변수 theTable은 이전 SQL 함수 호출로부터 리턴됩니다.

DTW_SET_TOTAL_ROWS

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

목적

조회에 대한 결과 세트에서 전체 행 수가 TOTAL_ROWS로 지정되도록 데이터베이스 언어 환경에 지정합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

이 변수를 자동으로 전달하려면 Net.Data 초기화 파일의 데이터베이스 언어 환경 명령문에 IN 변수로 포함시키십시오. 데이터베이스 언어 환경 명령문에 대한 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장을 참고하십시오.

값

DTW_SET_TOTAL_ROWS="YES"|"NO"

표 18. DTW_SET_TOTAL_ROWS 값

값	설명
YES	전체 행 수를 TOTAL_ROWS 변수로 지정합니다. 중요: 조회로부터 리턴된 행의 수를 판별하기 위해 변수 TOTAL_ROWS를 참조하려는 경우 이 값을 설정해야 합니다.
NO	Net.Data는 TOTAL_ROWS 변수를 설정하지 않으면 TOTAL_ROWS는 매크로 파일에서 참조될 수 없습니다. NO가 생략시 값입니다.

성능 팁: DTW_SET_TOTAL_ROWS를 YES로 설정하면 데이터베이스 언어 환경은 전체 행을 결정하기 위해 모든 행이 검색되도록 요구하므로 성능에 영향을 주게 됩니다.

예

예제 1: TOTAL_ROWS를 사용하기 위한 DTW_SET_TOTAL_ROWS를 정의합니다.

```
%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				

목적

원격 데이터베이스 서버로의 연결을 설정합니다. 이 변수는 국지 DB2 서비스 시스템이 국지 서버를 인식하는 데 사용하는 이름을 지정합니다. LOCATION의 값은 통신 데이터베이스(CDB)의 SYSIBM.SYSLOCATIONS 테이블 내에 정의되어 있어야 합니다. 이 변수가 매크로 내에 정의되어 있지 않으면 매크로가 생성한 SQL 조회는 국지 DB2 서비스 시스템에서 실행됩니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

LOCATION="remote_dbase_name"

표 19. LOCATION 값

값	설명
remote_dbase_name	CDB의 SYSIBM.SYSLOCATIONS 테이블에 정의된 유효한 원격 데이터베이스 서버의 이름. 이 이름은 8문자 이하가 될 수 있습니다.

예

예제 1: DEFINE문에서 원격 데이터베이스 위치를 정의합니다.

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

LOGIN

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

목적

데이터베이스 언어 환경에 사용자 ID를 제공하여 보호 자료에 대한 액세스 권한을 제공합니다. DB2의 보안 알고리즘을 합치려면 이 변수를 PASSWORD와 함께 사용하십시오.

보안 팁: Net.Data 매크로에 이 값을 코딩할 수 있으나 응용 프로그램 사용자가 사용자 ID를 HTML 형식으로 입력하게 하는 것이 좋습니다. 또한 웹 서버 ID의 생략시 값을 사용하면 보안 요구를 만족시키지 못하는 액세스 레벨이 제공될 수 있습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

LOGIN="database_user_id"

표 20. LOGIN 값

값	설명
database_user_id	유효한 데이터베이스 사용자 ID. 생략시 값은 웹 서버를 시작한 사용자 ID를 사용하는 것입니다.

예

예제 1: 액세스를 사용자 ID, DB2USER로 제한

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

예제 2: HTML 형식 입력 행 사용

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

이 예제에서는 응용 프로그램 사용자들이 사용자 ID를 입력할 때 HTML 형식의 일부로 포함시킬 수 있는 행을 나타냅니다.

NULL_REPORT_FIELD

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

목적

사용자가 SQL 결과 세트에 리턴된 널(NULL) 값을 나타내기 위해 DTW_SQL 언어 환경에 제공할 수 있는 문자열을 지정합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

NULL_REPORT_FIELD="null_char"

표 21. NULL_REPORT_FIELD 값

값	설명
null_char	SQL 결과 세트에 리턴된 널(NULL) 값을 나타내기 위한 문자를 지정합니다. 생략시 값은 빈 문자열입니다.

예

예제 1: SQL 언어 환경에서 널(NULL) 값을 나타내는 문자열을 지정합니다.

```
%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

목적

데이터베이스 언어 환경에 암호를 제공하여 보호 자료에 대한 액세스 권한을 제공합니다. DB2의 보안 알고리즘을 합치려면 이 변수를 LOGIN과 함께 사용하십시오.

보안 팁: Net.Data 매크로에 이 값을 코딩할 수 있으나 응용 프로그램 사용자가 암호를 HTML 형식으로 입력하도록 하는 것이 좋습니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

PASSWORD="password"

표 22. PASSWORD 값

값	설명
<i>password</i>	데이터베이스 언어 환경에 대한 자동 액세스를 제공하는 유효한 암호를 지정합니다.

예

예제 1: 액세스를 암호 NETDATA를 가지는 응용 프로그램 사용자로 제한

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

예제 2: HTML 형식 입력 행

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

이 예제에서는 응용 프로그램 사용자들이 암호를 입력할 때 HTML 형식의 일부로 포함시킬 수 있는 행을 나타냅니다.

SHOWSQL

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

목적

웹 브라우저에 사용되는 SQL 조회를 숨기거나 표시합니다. 테스트 중에 SQL을 표시하면 Net.Data 매크로를 디버그할 때 특히 유용합니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

값

SHOWSQL="YES"|"NO"

표 23. SHOW_SQL 값

값	설명
YES	데이터베이스에 송신된 조회의 SQL을 표시합니다.
NO	데이터베이스에 송신된 조회의 SQL을 숨깁니다. NO가 생략시 값입니다.

예

예제 1: 모든 SQL 조회를 표시합니다.

```
%DEFINE SHOWSQL="YES"
```

예제 2: HTML 형식 입력을 사용하여 SQL을 표시할지 여부 지정

```
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

목적

데이터베이스로부터 리턴된 SQL 상태 값에 액세스하거나 이 값을 표시합니다.

이 변수는 사전 정의된 변수이며 그 값은 수정할 수 없습니다. 이 변수를 변수 참조로 사용하십시오.

예

예제 1: REPORT 블록에서 SQL 상태를 표시합니다.

```
%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

목적

SQL 명령에 대한 트랜잭션 영역을 지정하고, HTML 블록에서 각 SQL 명령이나 모든 SQL 명령이 성공적으로 완료된 후에 Net.Data가 COMMIT를 수행할지 여부를 결정합니다. 확약 이전에 모든 SQL 명령이 성공적으로 완료되도록 지정하면 성공적이지 못한 SQL 명령으로 인해 이 블록 내의 동일한 데이터베이스에 대해 이전에 실행된 모든 SQL이 구간 복원됩니다.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 통해 이 변수의 값을 지정하십시오.

일관성 고려사항: OS/400 및 OS/390 이외의 운용 시스템에서는, 다음 조건이 모두 참일 때 같은 HTML 블록에서 액세스한 다른 데이터베이스에 대한 갱신이 확약될 수 있는 반면 성공적이지 못한 응답을 수신하는 데이터베이스에 대한 갱신은 구간 복원될 수 있습니다.

- TRANSACTION_SCOPE = "MULTIPLE"이 지정됩니다.
- 여러 데이터베이스를 한 HTML 블록내에서 액세스합니다 (이는 라이브 연결을 사용할 경우에 가능합니다).
- SQL 요청시 실패한 응답이 수신되었습니다

IBM의 DataJoiner를 사용하여 Net.Data로부터 여러 데이터베이스에 액세스할 경우, Net.Data로부터 갱신을 수행할 때 여러 데이터베이스 갱신 조정 및 일치 내용을 보관할 수 있습니다.

OS/400 및 OS/390에서, TRANSACTION_SCOPE = "MULTIPLE"를 지정하면 단일 HTML 블록으로부터 수행된 모든 IBM 데이터베이스 갱신은 함께 확약되거나 구간복원됩니다.

OS/400이외의 운용 시스템에서, REXX, Perl 및 Java 언어 환경은 자체의 별도 운용 시스템 프로세스로 수행됩니다. 따라서, 이러한 언어 환경에서 수행한 모든 데이터베이스 갱신은 Net.Data TRANSACTION_SCOPE 값에 관계없이 Net.Data 매크로 파일에서 발행된 데이터베이스 갱신으로부터 별도로 확약되거나 구간복원됩니다.

값

TRANSACTION_SCOPE="SINGLE" | "MULTIPLE"

표 24. TRANSACTION_SCOPE 값

값	설명
SINGLE	HTML 블록의 각 SQL 명령이 성공적으로 완료된 후에야 Net.Data는 COMMIT를 수행합니다.

표 24. TRANSACTION_SCOPE 값 (계속)

값	설명
MULTIPLE	HTML 블록의 모든 SQL 명령이 성공적으로 완료된 후에만 Net.Data가 COMMIT를 수행하도록 지정합니다. MULTIPLE이 생략시 값입니다.

예

예제 1: 각 트랜잭션 후에 COMMIT를 수행하도록 지정합니다.

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

제3장 Net.Data 내장 함수

Net.Data는 자체의 FUNCTION 블록을 작성하지 않고 사용할 수 있는 다양한 함수를 제공합니다. Net.Data 내장 함수는 다음 범주로 구분됩니다.

- 일반 함수는 Net.Data를 사용하여 웹 페이지를 개발하는 데 도움을 주며 다른 범주에는 맞지 않습니다. 125페이지의 『일반 함수』를 참고하십시오.
- 수학 함수는 산술 연산을 수행합니다. 145페이지의 『수학 함수』를 참고하십시오.
- 문자열 조작 함수는 문자열과 문자를 수정합니다. 158페이지의 『문자열 함수』를 참고하십시오.
- 단어 조작 함수는 단어 또는 단어 세트를 수정합니다. 175페이지의 『단어 함수』를 참고하십시오.
- 테이블 조작 함수는 테이블 자료로부터 양식 및 보고서를 생성하는 데 도움을 줍니다. 185페이지의 『테이블 함수』를 참고하십시오.
- 플랫폼 파일 인터페이스 함수 파일 입출력을 수정합니다. 201페이지의 『플랫폼 파일 인터페이스 함수』를 참고하십시오.
- 웹 레지스트리 함수는 웹 레지스트리에 대한 조작을 수행합니다. 221페이지의 『웹 레지스트리 함수』를 참고하십시오.

다음 설명에서, 함수 매개변수는 유형 문자열, 정수, 부동 소수점 및 테이블에 따라 설명됩니다. 모든 Net.Data 변수는 문자열 유형을 가지나 용어 정수 및 부동 소수점은 각각 정수 또는 부동 소수점 값을 나타내는 문자열을 표시하는 데 사용됩니다.

함수명

Net.Data 내장 함수는 예약된 접두어인 DTW_로 시작됩니다. 사용자 정의 함수는 이 접두어를 사용할 수 없습니다.

내장 함수명은 대소문자를 구분하지 않습니다.

입출력 매개변수

함수는 Net.Data가 입력, 출력 또는 입출력 모두에 대한 매개변수를 사용할 것인지 결정하는 매개변수 제공 스펙을 가질 수 있습니다. 이러한 매개변수 제공 스펙은 다음 키워드로 지정됩니다.

IN 매개변수가 Net.Data로부터 언어 환경에 입력 자료를 제공하도록 지정합니다.

OUT 매개변수가 언어 환경으로부터 Net.Data로 출력 자료를 리턴하도록 지정합니다.

INOUT

매개변수가 입력 자료를 언어 환경으로 제공하고 언어 환경에서 Net.Data로 출력 자료를 리턴하도록 지정합니다.

함수 결과 형식 지정

대부분의 함수는 다음은 형식을 갖습니다.

- DTW_r, DTWF_r, DTWR_r 등으로 시작하는 함수는 그 결과를 함수 호출로 리턴하므로, 출력 매개변수가 없습니다. 다음 예는 서버 시간을 나타냅니다.

```
Current local time is @DTW_rTIME().
```

- DTW_m으로 시작하는 함수는 여러 매개변수에서 함수를 수행합니다. 각 매개변수는 입력 매개변수와 출력 매개변수로 기능합니다. 함수가 매개변수에 대해 수행되며 결과가 매개변수에 리턴됩니다. 다음 예는 세 가지 입력 매개변수가 일관성 있게 표시되도록 모두 대문자로 변환하게 됩니다.

```
@DTW_mUPPERCASE(model, style, shipNo)  
Shipment $(shipNo) contains $(quantity) of model $(model) $(style).
```

- DTW_, DTWF_ 및 DTWR_ return으로 시작하는 다른 함수 결과는 출력 매개변수로 리턴됩니다. 출력 매개변수를 지정해야 합니다. 다음 예는 서버 시간을 나타냅니다.

```
@DTW_TIME(nowTime)  
Current local time is $(nowTime).
```

함수 매개변수 규칙

함수 매개변수를 올바른 순서로 배치하십시오. 마지막 입력 매개변수가 지정되기 전에 먼저 모든 입력 매개변수를 지정하거나 널(NULL)(『』)을 지정하여 생략시 값을 그대로 사용하십시오. 예를 들어 다음 예제에서처럼 DTW_TB_INPUT_TEXT를 호출할 수 있습니다.

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

위의 예제에서 네번째 및 다섯번째 매개변수는 생략시 값을 사용합니다. 생성된 HTML에서 『32』가 MAXLENGTH의 값을 나타내려면 이들 매개변수를 널(NULL) 값으로 포함해야 합니다. 마지막 매개변수가 지정되지 않았으므로 생략시 값이 사용됩니다. MAXLENGTH와 앞의 두 매개변수에 대해 생략시 값을 그대로 사용하려면 다음과 같이 생략하십시오.

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

널(NULL)이 아닌 입력 매개변수가 또 있으면 입력 매개변수에 대한 매개변수 목록에 중간 널(NULL) 값을 지정해야 합니다. 마지막 출력 매개변수를 지정하기 전에는 중간 널(NULL) 입력 매개변수를 지정할 필요가 없습니다.

일반 함수

일반 함수는 Net.Data를 사용하여 웹 페이지를 개발하는 데 도움을 주며 다른 범주에는 맞지 않습니다. 일반 함수는 다음과 같습니다.

- 126페이지의 『DTW_ADDQUOTE』
- 128페이지의 『DTW_CACHE_PAGE』
- 132페이지의 『DTW_DATE』
- 134페이지의 『DTW_EXIT』
- 135페이지의 『DTW_GETENV』
- 136페이지의 『DTW_GETINIDATA』
- 137페이지의 『DTW_HTMLENCODER』
- 139페이지의 『DTW_QHTMLENCODER』
- 140페이지의 『DTW_SETENV』
- 141페이지의 『DTW_TIME』
- 143페이지의 『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

목적

입력 문자열의 작은 따옴표 하나를 작은 따옴표 두 개로 교체합니다. 문자열에 작은 따옴표가 포함될 때 SQL문이 올바르게 처리될 수 있도록 교체하는 과정이 필요합니다.

모든 SQL INPUT 명령문에 이 함수를 사용해 보십시오. 예를 들어, 다음 예에서와 같이 성을 O'Brien으로 입력하면 작은 따옴표로 인해 오류가 생길 수 있습니다.

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

다음과 같이 DTW_ADDQUOTE 함수를 사용하면 SQL문이 변경되어 오류가 생기지 않습니다.

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

형식

@DTW_ADDQUOTE(stringIn, stringOut)

@DTW_rADDQUOTE(stringIn)

@DTW_mADDQUOTE(stringMult, stringMult2, ..., stringMultn)

값

표 25. DTW_ADDQUOTE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열. DTW_mADDQUOTE에는 여러 개의 입력 문자열이 있을 수 있습니다.
문자열	<i>stringOut</i>	OUT	수정된 <i>stringIn</i> 유형이 들어 있는 변수.
문자열	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"> 입력: 문자열이 들어 있는 변수. 출력: 각 작은 따옴표(')가 두 개의 작은 따옴표로 교체된 입력 문자열이 들어 있는 변수.

예

예제 1: OUT 매개변수에 작은 따옴표를 추가합니다.

```
@DTW_ADDQUOTE(string1,string2)
```

- 입력: string1="John's Web page"
- 리턴: string2="John''s Web page"

예제 2: 함수 호출로부터 리턴된 값에 작은 따옴표를 추가합니다.

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

- 리턴: "The title of the article is ''Once upon a time''"

예제 3: 함수 호출의 각 INOUT 매개변수에 작은 따옴표를 추가합니다.

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

- 입력: string1="Joe's bag", string2="'to be or not to be'"
- 리턴: string1="Joe''s bag", string2="''to be or not to be''"

예제 4: DB2 테이블에 삽입되는 자료에 작은 따옴표를 삽입합니다.

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

- 입력: lastname="O'Brien", firstname="Patrick"
- 리턴: "O''Brien", "Patrick"

DTW_CACHE_PAGE

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

목적

매크로 파일에서 함수 다음에 나오는 모든 HTML 출력의 캐시 작업을 시작합니다. 이 함수가 호출되면 캐시로부터 지정된 페이지를 검색하고 이 페이지가 매크로로부터 생성된 출력 페이지인 것처럼 웹 브라우저로 송신하려 합니다. 이 페이지를 찾았으며 아직 만기되지 않은 경우 Net.Data는 매크로 처리를 중단하고 매크로 파일로부터 빠져 나간 다음 캐시 페이지를 웹 브라우저로 송신합니다.

요청된 페이지가 캐시에 없거나 기존 캐시 페이지가 *age* 값보다 더 오래된 경우 Net.Data는 새로운 출력 페이지를 생성합니다. 매크로가 성공적으로 완료되면 Net.Data는 브라우저로 새로운 페이지를 송신하고 이 페이지를 캐시 처리합니다.

매크로 파일에서 **DTW_CACHE_PAGE** 함수의 위치 판별:

- 대부분의 캐시 응용 프로그램에서 매크로 맨 위에 **DTW_CACHE_PAGE**를 지정하여 모든 HTML 출력을 캐시 처리하십시오. 이 기술을 통해 새로운 보고서 블록이 추가될 때 매크로 파일을 유지보수하기가 더 쉬워집니다. 예를 들어 이 함수가 매크로 중간에 있으면 HTML 보고서 절이 매크로에서 더 먼저 추가될 때 이를 알지 못할 수 있습니다. Net.Data는 새로운 보고서 출력을 캐시 처리하지 않습니다. 또한 이 방법은 Net.Data가 페이지가 캐시되었음을 알게 될 때 모든 후속 처리를 중단하므로 성능을 향상시킵니다.
- 고급 캐시 응용 프로그램에서는 매크로 파일의 처음이 아니라, 처리 중의 특정 시점에서 캐시를 수행하도록 결정해야 할 때 HTML 출력 섹션에 이 함수를 배치할 수 있습니다. 예를 들면 조회나 함수 호출로부터 리턴되는 행의 수에 따라 캐시하도록 결정할 수 있습니다.

형식

@DTW_CACHE_PAGE(cacheid, url, age, status)

값

표 26. DTW_CACHE_PAGE 매개변수

매개변수	사용	설명
<i>cache_id</i>	IN	페이지를 넣을 캐시를 식별하는 문자열 변수.
<i>cached_page_ID</i>	IN	후속 DTW_CACHE_PAGE 캐시 요청에서 캐시 처리된 페이지를 찾는 데 사용되는 식별자가 들어 있는 문자열 변수. 이 문자열은 URL일 수 있습니다.

표 26. DTW_CACHE_PAGE 매개변수 (계속)

매개변수	사용	설명
<i>age</i>	IN	<p>초 단위의 시간을 포함하는 문자열 변수. 이 매개변수는 페이지가 만기되었는지 여부를 결정합니다. 이 페이지가 <i>age</i> 보다 더 오래된 경우 브라우저로 송신되지 않습니다.</p> <p><i>age</i>가 -1로 지정되고 해당 페이지가 캐시에 있으면 Net.Data는 그 사용 기간에 관계없이 캐시에서 웹 브라우저로 이 페이지를 직접 송신합니다. Net.Data는 캐시의 페이지는 바꾸지 않습니다.</p>
<i>status</i>	OUT	<p>캐시 페이지의 상태를 나타내는 문자열 변수. 다음과 같은 소문자로 된 값을 사용할 수 있습니다.</p> <ul style="list-style-type: none"> • ok: 매크로 실행이 종료될 때 출력 페이지가 캐시됩니다. • new: 페이지는 캐시에 없습니다. • renew: 페이지가 캐시에 있으나 만기되었습니다. • no_cache: 지정된 캐시 식별자가 없습니다. 이 식별자는 캐시 구성 파일에 정의되어야 합니다. 페이지 캐시 없이 매크로를 계속 실행할 수 없습니다. • inactive: 지정한 캐시가 비활성으로 표시되어 있습니다. 페이지 캐시 없이 매크로를 계속 실행할 수 없습니다. • busy: 매크로가 이 실행 이전에 DTW_CACHE_PAGE 내장 함수를 실행했습니다. 매크로를 계속 실행할 수 없습니다. • error: 캐시와 통신하는 도중 오류가 발생했습니다.

예

예제 1: 모든 HTML 출력을 캡처하도록 매크로 파일 맨 앞에 DTW_CACHE_PAGE 함수를 넣습니다.

```
%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>

%}

```

예제 2: 캐시 결정은 HTML 출력의 예상 크기에 따라 달라지므로 함수를 HTML 캐시에 넣습니다.

```

% 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>
%}

```

이 예제에서 페이지는 HTML 출력의 예상 크기를 기초로 캐시되거나 검색됩니다. HTML 출력 페이지는 데이터베이스 테이블에 100개가 넘는 행이 들어 있을 때만 캐시할 가치가 있는 것으로 간주됩니다. Net.Data는 OUTPUT 블록, This is the customer list의 텍스트를 항상 매크로를 실행한 후에 브라우저로 송신하므로 이 텍스트는 캐시 처리되지 않습니다. 이 함수 호출 다음 행, @count_rows()은 IF 블록의 조건이 만족될 때 캐시 처리되거나 검색됩니다. 두 부분이 함께 완성된 Net.Data 페이지를 형성합니다.

예제 3: 캐시 ID와 캐시 페이지 ID를 동적으로 검색합니다.

```

%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

목적

지정된 형식으로 현재 시스템 날짜를 리턴합니다.

형식

@DTW_DATE(format, stringOut)

@DTW_DATE(stringOut)

@DTW_rDATE(format)

@DTW_rDATE()

값

표 27. DTW_DATE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>format</i>	IN	<p>자료 형식을 지정하는 리터럴 문자열 또는 변수. 유효한 형식은 다음과 같습니다.</p> <p>D - 연 중 일 (001-366)</p> <p>E - 유럽 날짜 형식 (dd/mm/yy)</p> <p>N - 일반 날짜 형식 (dd mon yyyy)</p> <p>O - 순서화된 날짜 형식 (yy/mm/dd)</p> <p>S - 표준 날짜 형식 (yyyymmdd)</p> <p>U - 미국 날짜 형식 (mm/dd/yy)</p> <p>생략시 값은 N입니다.</p>
문자열	<i>stringOut</i>	OUT	지정된 형식으로 된 날짜가 들어있는 변수.

예

예제 1: 일반 날짜 형식

```
@DTW_DATE(results)
```

- 리턴: results = "25 Apr 1997"

예제 2: 유럽 날짜 형식

```
@DTW_DATE("E", results)
```

- 리턴: results="25/04/97"

예제 3: 미국 날짜 형식

```
%HTML(report){
<P>This report created on @DTW_rDATE("U").
```

- 리턴: 04/25/97

DTW_EXIT

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

목적

매크로를 즉시 나가도록 지정합니다. Net.Data는 지금까지 매크로가 작성한 페이지를 브라우저로 송신하도록 합니다.

성능 팁: Net.Data가 전체 파일을 처리해야 하는 시간을 저장하기 위해 출력이 생성되었을 때 매크로 파일의 처리를 중단하려면 DTW_EXIT를 사용하십시오.

형식

@DTW_EXIT()

예

예제 1: 매크로 나감

```
%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_GETENV

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

목적

지정된 환경 변수의 값을 리턴합니다. ENVVAR를 사용하여 환경 변수 값을 참조할 수도 있습니다. 자세한 내용은 13페이지의 『ENVVAR 명령문』을 참고하십시오.

형식

@DTW_GETENV(envVarName, envVarValue)

@DTW_rGETENV(envVarName)

값

표 28. DTW_GETENV 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>envVarName</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>envVarValue</i>	OUT	<i>envVarName</i> 에 지정된 환경 변수의 값. 이 값이 없으면 널(NULL) 문자열이 리턴됩니다.

예

예제 1: OUT 매개변수에 대해 PATH문에 대한 값을 리턴합니다.

```
@DTW_GETENV(myEnvVarName, myEnvVarValue)
```

- 입력: myEnvVarName = "PATH"
- 리턴: myEnvVarValue = "/usr/path"

예제 2: PATH문에 대한 값을 리턴합니다.

```
@DTW_rGETENV(myPath)
```

- 입력: myPath = "PATH"
- 리턴: "/usr/path"

예제 3: 서버의 이름에 대한 값을 리턴합니다.

```
The server is @DTW_rGETENV("SERVER_NAME").
```

- 리턴: "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

목적

지정된 구성 변수의 값을 리턴합니다. 값이 없으면 널(NULL) 문자열이 리턴됩니다.

제한사항: 비 OS/400 운영체제의 경우, ENVIRONMENT문 뿐 아니라 구성 경로 변수(MACRO_PATH, EXEC_PATH 및 INCLUDE_PATH)는 이 호출로부터 검색할 수 없습니다. OS/400 운용 시스템에서는, 이러한 제한이 ENVIRONMENT문에만 적용됩니다.

형식

@DTW_GETINIDATA(iniVarName, iniVarValue)

@DTW_rGETINIDATA(iniVarName)

값

표 29. DTW_GETINIDATA 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>iniVarName</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>iniVarValue</i>	OUT	<i>iniVarName</i> 에 지정된 구성 변수의 값.

예

예제 1: Net.Data 경로 변수 값을 리턴합니다.

```
@DTW_GETINIDATA(myEnvVarName, myEnvVarValue)
```

- 입력: myEnvVarName = "FFI_PATH"
- 리턴: myEnvVarValue = "D:\FFI"

DTW_HTMLENCODER

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

목적

표준 HTML 십진 escape 코드를 사용하여 문자를 코드화하는 데, 모든 문자를 코드화하는 것은 아닙니다. 이 함수를 사용하여 웹 브라우저가 HTML로 해석하지 않게 하려는 자료를 코드화할 수 있습니다. 예를 들어 적절한 escape 문자를 사용하여, 보통 HTML 태그로 예약되어 있는 보다 작음(<) 및 보다 큼(>) 기호를 표시할 수 있습니다.

다른 예제에서, HTML로 된 다음 문자열은 각 번호 사이에 하나의 공백만 나타냅니다.

1 2 3

DTW_HTMLENCODER를 사용하여 공백의 수를 맞게 표시하십시오.

137페이지의 표 30에는 DTW_HTMLENCODER 함수에 의해 코드화된 문자가 나타납니다.

표 30. HTML 십진 escape 문자

문자	이름	코드
SPACE	공간	
"	큰 따옴표	"
#	숫자 기호	#
%	퍼센트	%
&	앰퍼샌드	&
\	역슬래쉬	\
:	콜론	:
;	세미콜론	;
<	보다 작음	<
=	같음	=
>	보다 큼	>
?	물음표	?
@	위치 기호	@
[여는 대괄호	[
/	슬래쉬	/
]	닫는 대괄호]
^	캐럿	^
{	여는 중괄호	{
	직선	|

표 30. HTML 십진 *escape* 문자 (계속)

}	닫는 중괄호	}
~	틸드	~

형식

@DTW_HTMLENCODE(stringIn, stringOut)

@DTW_rHTMLENCODE(stringIn)

값

표 31. DTW_HTMLENCODE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	특정 문자가 코드화된 HTML escape 문자로 대체된 수정된 입력 문자열이 들어 있는 변수.

예

예제 1: 공백 문자를 코드화합니다.

@DTW_HTMLENCODE(string1,string2)

- 입력: string1 = "Jim's dog"
- 리턴: string2 = "Jim's dog"

예제 2: 공백, 보다 작음 기호 및 같음 기호를 코드화합니다.

@DTW_rHTMLENCODE("X <= 10")

- 리턴: "X <= 10"

DTW_QHTMLENCODE

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

목적

@DTW_QHTMLENCODE와 같은 함수를 수행하고 작은 따옴표(')를 '로 코드화합니다. DTW_QHTMLENCODE가 사용하는 HTML 십진 escape 문자는 137페이지의 표 30에 나옵니다.

형식

@DTW_QHTMLENCODE(stringIn, stringOut)

@DTW_rQHTMLENCODE(stringIn)

값

표 32. DTW_QHTMLENCODE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	특정 문자가 코드화된 HTML escape 문자로 대체된 수정된 양식의 <i>stringIn</i> 을 포함하는 변수.

예

예제 1: 어포스트로피와 공백을 코드화합니다.

```
@DTW_QHTMLENCODE(string1,string2)
```

- 입력: string1 = "Jim's dog"
- 리턴: string2 = "Jim's dog"

예제 2: 어포스트로피, 공백 및 앰퍼샌드를 코드화합니다.

```
@DTW_rQHTMLENCODE("John's & Jane's")
```

- 리턴: "John's & Jane's"

DTW_SETENV

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

목적

지정된 값으로 환경 변수를 지정하고 이전 값을 리턴합니다. 이전 값이 없는 경우 널(NULL) 문자열이 리턴됩니다.

형식

@DTW_SETENV(envVarName, envVarValue, prevValue)

@DTW_rSETENV(envVarName, envVarValue)

값

표 33. DTW_SETENV 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>envVarName</i>	IN	환경 변수를 나타내는 리터럴 문자열 또는 변수.
문자열	<i>envVarValue</i>	OUT	환경 변수가 지정된 값을 포함하는 변수 또는 리터널 문자열.
문자열	<i>prevValue</i>	OUT	환경 변수의 이전 값이 들어 있는 변수.

예

예제 1: 이전 경로에 대한 값을 리턴합니다.

```
@DTW_SETENV("PATH", "myPath", prevValue)
```

- 입력: myPath = "myPath"
- 리턴: prevValue = "myPreviousPath"

예제 2: 이전 경로에 대한 값을 리턴하고 PATH문에 대한 값을 지정합니다.

```
@DTW_rSETENV("PATH", "myPath")
```

- 입력: myPath = "myPath"
- 리턴: "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

목적

지정된 형식으로 현재 시스템 시간을 리턴합니다.

형식

@DTW_TIME(stringIn, stringOut)

@DTW_TIME(stringOut)

@DTW_rTIME(stringIn)

@DTW_rTIME()

값

표 34. DTW_TIME 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	시간 형식을 지정하는 리터럴 문자열 또는 변수. 유효한 형식은 다음과 같습니다. C - 상용 시간 (12시간 시계를 사용하는 hh:mmAM/PM) L - 현지 시간 (hh:mm:ss) N - 일반 시간 (24시간 시계를 사용하는 hh:mm:ss). 생략시 값임 H - 자정 이후 시간 M - 자정 이후 분 S - 자정 이후 초
문자열	<i>stringOut</i>	OUT	지정된 형식으로 된 시간이 들어 있는 변수.

예

예제 1: 24시간 형식

@DTW_TIME(results)

• 리턴: results = "10:30:53"

예제 2: 상용 시간 형식

@DTW_TIME("C", results)

• 리턴: results = "10:30AM"

예제 3: 함수 호출을 사용하여 자정 이후의 분 수를 리턴합니다.

@DTW_rTIME("M")

• 결과: "630"

예제 4: 함수 호출을 사용하여 생략시 시간 및 자료 형식을 리턴합니다.

```
%REPORT{  
<P>This report was created at @DTW_rTIME(), @DTW_rDATE().  
%}
```

- 결과: 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

목적

URL에서 허용되지 않는 문자를 URL 코드화 값으로도 알려진 escape 값으로 대체합니다. 143페이지의 표 35에 표시된 문자를 다른 매크로 파일이나 HTML 블록으로 전달하려면 이 함수를 사용해야 합니다.

표 35. URL에서 허용되지 않는 문자

문자	이름	코드
SPACE	공간	%20
"	큰 따옴표	%22
#	숫자 기호	%23
%	퍼센트	%25
&	앰퍼샌드	%26
\	역슬래쉬	%2F
:	콜론	%3A
;	세미콜론	%3B
<	보다 작음	%3C
=	같음	%3D
>	보다 큼	%3E
?	물음표	%3F
@	위치 기호	%40
[여는 대괄호	%5B
/	슬래쉬	%5C
]	닫는 대괄호	%5D
^	캐럿	%5E
{	여는 중괄호	%7B
	직선	%7C
}	닫는 중괄호	%7D
~	틸드	%7E

형식

@DTW_URLESCSEQ(stringIn, stringOut)

@DTW_rURLESCSEQ(stringIn)

값

표 36. DTW_URLESCSEQ 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	URL에서 허용되지 않아 16진 escape 값으로 교체된 문자가 있는 입력 문자열이 포함된 변수.

예

예제 1: *string1*의 공백 및 앰퍼샌드를 URL escape 코드로 대체하고 결과를 *string2*로 지정합니다.

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

- 입력: *string1* = "Guys & Dolls"
- 결과: *string2* = "Guys%20%26%20Dolls"

예제 2: 공백 및 앰퍼샌드를 URL 코드화 형식으로 변환합니다.

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

- 결과: "Guys%20%26%20Dolls"

예제 3: ROW 블록에서 DTW_rURLESCSEQ를 사용하고 공백과 at 기호를 URL 코드화 형식으로 변환합니다.

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

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

- 결과:

```
<P><a href="fullrpt.mac/input?name=Patrick%20'O'Brien&email="obrien%40ibm.com">
Patrick O'Brien</a>
```

응용 프로그램 사용자가 이름을 클릭할 때 이름과 전자우편 주소가 변수 *name* 및 *email*로 코드화된 값과 함께 Net.Data 매크로 fullrpt.mac의 입력 블록으로 송신됩니다.

수학 함수

이 함수를 통해 수학적 계산을 수행할 수 있습니다.

UNIX, Windows NT 및 OS/2에 대한 성능 팁: Net.Data 초기화 파일이나 매크로 파일에서 DTW_OPTIMIZE_MATH 구성 값을 YES로 설정하여 이 구성 값으로 수학 함수의 성능을 최적화할 수 있습니다.

- 이 값을 YES로 설정하면 Net.Data는 C 수학 형식과 함수를 더 빨리 수행할 수 있으나 이 변수가 없을 때와 그 출력 형식이 달라집니다. 소수점 뒤의 0은 표시되지 않습니다.
- DTW_OPTIMIZE_MATH가 NO로 설정되면 Net.Data는 REXX 수학 형식을 사용합니다. 함수는 더 느리게 수행되지만 이전 버전의 Net.Data가 생성한 출력과 일관된 출력 형식을 제공합니다. 생략시 값은 NO입니다.

이 변수 구성 방법에 대해서는 *Net.Data Administration and Programming Guide*의 구성 변수 절을 참고하십시오.

- 146페이지의 『DTW_ADD』
- 147페이지의 『DTW_DIVIDE』
- 148페이지의 『DTW_DIVREM』
- 150페이지의 『DTW_FORMAT』
- 153페이지의 『DTW_INTDIV』
- 154페이지의 『DTW_MULTIPLY』
- 155페이지의 『DTW_POWER』
- 156페이지의 『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

목적

두 매개변수의 값을 추가합니다.

형식

@DTW_ADD(number1, number2, precision, result)

@DTW_rADD(number1, number2, precision)

값

표 37. DTW_ADD 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 과 <i>number2</i> 의 합계가 들어 있는 변수.

예

예제 1:

```
@DTW_ADD(NUM1, NUM2, "2", result)
```

- 입력: NUM1 = "105", NUM2 = "3"
- 결과: result = "1.1E+2"

예제 2:

```
@DTW_rADD("12", NUM2,  
"5")
```

- 입력: NUM2 = "7.00"
- 결과: "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

목적

첫번째 매개변수 값을 두번째 매개변수 값으로 나눕니다.

형식

@DTW_DIVIDE(number1, number2, precision, result)

@DTW_rDIVIDE(number1, number2, precision)

값

표 38. DTW_DIVIDE 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 을 <i>number2</i> 로 나눈 결과가 들어 있는 변수.

예

예제 1:

```
@DTW_DIVIDE("8.0", NUM2, result)
```

- 입력: NUM2 = "2"
- 결과: result = "4"

예제 2:

```
@DTW_rDIVIDE("1", NUM2, "5")
```

- 입력: "1", NUM2 = "3"
- 결과: "0.33333"

예제 3:

```
@DTW_rDIVIDE(NUM1, "2", "5")
```

- 입력: NUM1 = "5"
- 결과: "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

목적

첫번째 매개변수를 두번째 매개변수로 나눈 나머지를 리턴합니다. 나머지가 0이 아닌 경우, 나머지의 부호는 첫번째 매개변수의 부호와 같습니다.

형식

@DTW_DIVREM(number1, number2, precision, result)

@DTW_rDIVREM(number1, number2, precision)

값

표 39. DTW_DIVREM 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 을 <i>number2</i> 로 나눈 나머지가 들어 있는 변수.

예

예제 1:

```
@DTW_DIVREM(NUM1, NUM2,
result)
```

- 입력: NUM1 = "2.1", NUM2 = "3"
- 결과: result = "2.1"

예제 2:

```
@DTW_rDIVREM("10",
NUM2)
```

- 입력: NUM2 = "0.3"
- 결과: "0.1"

예제 3:

```
@DTW_rDIVREM("3.6", "1.3")
```

- 결과: "1.0"

예제 4:

```
@DTW_rDIVREM("-10", "3")
```

- 결과: "-1"

DTW_FORMAT

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

목적

숫자의 형식을 조정합니다. *number* 매개변수만 지정되면 결과는 @DTW_rADD(*number*, 『0』)가 수행된 것처럼 형식화됩니다. 다른 옵션이 지정되는 경우, 다음 규칙에 따라 숫자가 형식화됩니다.

- *before* 및 *after* 매개변수는 각각, *result* 매개변수의 정수 부분과 소수 부분에 사용되는 문자의 수를 설명합니다. 이들 매개변수 중 하나를 생략하거나 두 매개변수를 모두 생략하는 경우 해당 부분에 사용되는 문자의 수는 필요한 만큼 사용됩니다.
- *before* 매개변수가 숫자의 정수 부분(음수의 경우 부호까지 포함)을 포함할 만큼 충분히 크지 않으면 오류가 발생합니다. *before* 매개변수가 해당 부분에 필요한 것보다 더 크면 *number* 매개변수 값의 왼쪽이 공백으로 채워집니다. *after* 매개변수가 *number* 매개변수의 소수 부분과 크기가 같지 않으면 크기에 맞도록 반올림됩니다 (또는 0으로 채워짐). 0을 지정하면 숫자는 정수로 반올림됩니다.
- 또한 *expp* 및 *expt* 매개변수는 결과의 지수 부분을 제어합니다. *expp* 매개변수는 지수 부분의 자리 수를 설정합니다. 생략시 값은 필요한 만큼 자리 수를 사용하는 것입니다 (0이 될 수 있음). *expt* 매개변수는 지수 표기를 사용하기 위해 트리거 포인트를 설정합니다. 생략시 값은 *precision* 매개변수의 생략시 값입니다.
- *expp*가 0이면 지수가 제공되지 않으며 숫자는 필요한 만큼 0이 추가된 단순 양식으로 표현됩니다. *expp*가 지수를 포함할 만큼 충분히 크지 않으면 오류가 발생합니다.
- 정수나 소수 부분에 필요한 자리수가 각각 *expt*나 두 배의 *expt*를 초과하면 지수 표기법을 사용하십시오. *expt*가 0이면 지수가 0이 아닌 경우에 항상 지수 표기법이 사용됩니다. (*expp*가 0이면 이것은 *expt*의 0값을 대체합니다.) 0이 아닌 *expp*가 지정될 때 지수가 0이면 *expp*+2개의 공백이 결과의 지수 부분에 제공됩니다. 지수가 0이고 *expp*가 지정되지 않으면 단순 양식이 사용됩니다.

형식

@DTW_FORMAT(*number*, *before*, *after*, *expp*, *expt*, *precision*, *result*)

@DTW_rFORMAT(*number*, *before*, *after*, *expp*, *expt*, *precision*)

값

표 40. DTW_FORMAT 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>before</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 이것은 선택적 매개변수입니다. 추가 매개변수를 사용하려면 널 (NULL) 문자열("")을 입력해야 합니다.
정수	<i>after</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 이것은 선택적 매개변수입니다. 추가 매개변수를 사용하려면 널 (NULL) 문자열("")을 입력해야 합니다.
정수	<i>expp</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 추가 매개변수를 사용하려면 널 (NULL) 문자열("")을 지정해야 합니다.
정수	<i>expt</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 추가 매개변수를 사용하려면 널 (NULL) 문자열("")을 입력해야 합니다.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	지정된 반올림 형식으로 된 숫자가 들어있는 변수.

예

예제 1:

```
@DTW_FORMAT(NUM, BEFORE, result)
```

- 입력: NUM = "3", BEFORE = "4"
- 결과: result = " 3"

예제 2:

```
@DTW_FORMAT("1.73", "4", "0", result)
```

- 결과: result = " 2"

예제 3:

```
@DTW_FORMAT("1.73", "4", "3", result)
```

- 결과: result = " 1.730"

예제 4:

```
@DTW_FORMAT(" - 12.73", "", "4", result)
```

- 결과: result = "-12.7300"

예제 5:

```
@DTW_FORMAT("12345.73", "", "", "2", "2", result)
```

- 결과: result = "1.234573E+04"

예제 6:

```
@DTW_FORMAT("1.234573", "", "3", "", "0", result)
```

- 결과: result = "1.235"

예제 7:

```
@DTW_rFORMAT(" - 12.73")
```

- 결과: " - 12.73"

예제 8:

```
@DTW_rFORMAT("0.000")
```

- 결과: "0"

예제 9:

```
@DTW_rFORMAT("12345.73", "", "", "3", "6")
```

- 결과: "12345.73"

예제 10:

```
@DTW_rFORMAT("1234567e5", "", "3", "0")
```

- 결과: "123456700000.000"

예제 11:

```
@DTW_rFORMAT("12345.73", "", "3", "", "0")
```

- 결과: "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

목적

첫번째 매개변수를 두번째 매개변수로 나눈 몫의 정수 부분을 리턴합니다.

형식

@DTW_INTDIV(number1, number2, precision, result)

@DTW_rINTDIV(number1, number2, precision)

값

표 41. DTW_INTDIV 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 을 <i>number2</i> 로 나눈 몫의 정수 부분이 들어 있는 변수.

예

예제 1:

```
@DTW_INTDIV(NUM1, NUM2,
result)
```

- 입력: NUM1 = "10", NUM2 = "3"
- 결과: result = "3"

예제 2:

```
@DTW_rINTDIV("2",
NUM2)
```

- 입력: NUM2 = "3"
- 결과: "0"

DTW_MULTIPLY

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

목적

두 매개변수를 곱하여 결과를 리턴합니다.

형식

@DTW_MULTIPLY(number1, number2, precision, result)

@DTW_rMULTIPLY(number1, number2, precision)

값

표 42. DTW_MULTIPLY 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 과 <i>number2</i> 를 곱한 값이 들어 있는 변수.

예

예제 1:

```
@DTW_MULTIPLY(NUM1, NUM2, result)
```

- 입력: NUM1 = "4", NUM2 = "5"
- 결과: result = "20"

예제 2:

```
@DTW_rMULTIPLY("0.9",  
NUM2)
```

- 입력: NUM2 = "0.8"
- 결과: "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

목적

두번째 매개변수의 첫번째 매개변수 승을 구하고 결과를 리턴합니다.

형식

@DTW_POWER(number1, number2, precision, result)

@DTW_rPOWER(number1, number2, precision)

값

표 43. DTW_POWER 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number2</i> 의 <i>number1</i> 승을 포함하는 변수.

예

예제 1:

```
@DTW_POWER(NUM1, NUM2,
result)
```

- 입력: NUM1 = "2", NUM2 = "-3"
- 결과: result = "0.125"

예제 2:

```
@DTW_rPOWER("1.7", NUM2, precision)
```

- 입력: NUM2 = "8", precision = "5"
- 결과: "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

목적

첫번째 매개변수 값에서 두번째 매개변수 값을 빼고 결과를 리턴합니다.

형식

@DTW_SUBTRACT(number1, number2, precision, result)

@DTW_rSUBTRACT(number1, number2, precision)

값

표 44. DTW_SUBTRACT 매개변수

자료 유형	매개변수	사용	설명
부동	<i>number1</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
부동	<i>number2</i>	IN	숫자를 나타내는 리터럴 문자열 또는 변수.
정수	<i>precision</i>	IN	결과의 정밀도를 지정하는 전체 양수를 나타내는 리터럴 문자열 또는 변수. 생략시 값은 9입니다.
부동	<i>result</i>	OUT	<i>number1</i> 과 <i>number2</i> 의 차가 들어 있는 변수.

예

예제 1:

```
@DTW_SUBTRACT(NUM1, NUM2, comp)
%IF(comp > "0")
<P>$(NUM1) is larger than $(NUM2).
%ENDIF
```

- 입력: NUM2 = "2.07"
- 결과: "-0.77"

이 예제에는 Net.Data에 있는 문자열에 해당하는 숫자 값을 비교하는 방법이 제공됩니다.

예제 2:

```
@DTW_SUBTRACT(NUM1, NUM2, result)
• 입력: NUM1 = "1.3", NUM2 = "1.07"
• 결과: result = "0.23"
```

예제 3:

```
@DTW_rSUBTRACT("1.3",
NUM2)
```

- 입력: NUM2 = "2.07"
- 결과: "-0.77"

문자열 함수

다음 함수는 Net.Data가 지원하는 표준 문자열 함수 세트입니다.

- 159페이지의 『DTW_ASSIGN』
- 160페이지의 『DTW_CONCAT』
- 161페이지의 『DTW_DELSTR』
- 162페이지의 『DTW_INSERT』
- 164페이지의 『DTW_LASTPOS』
- 165페이지의 『DTW_LENGTH』
- 166페이지의 『DTW_LOWERCASE』
- 167페이지의 『DTW_POS』
- 168페이지의 『DTW_REVERSE』
- 169페이지의 『DTW_STRIP』
- 170페이지의 『DTW_SUBSTR』
- 172페이지의 『DTW_TRANSLATE』
- 174페이지의 『DTW_UPPERCASE』

OS/390, OS/2, Windows NT 및 UNIX에 대한 MBCS 지원: DTW_MBMODE 구성 값을 사용하여 단어 및 문자열 함수에 대한 MBCS(다중 바이트 문자 세트) 지원을 지정할 수 있습니다. Net.Data 초기화 파일에서 이 값을 지정하십시오. 생략시, 지원되지 않습니다. Net.Data 매크로 파일에서 DTW_MBMODE 변수를 설정하여 초기화 파일의 해당 값을 교체할 수 있습니다. 자세한 내용은 *Net.Data Administration and Programming Guide*와 110페이지의 『DTW_MBMODE』를 참고하십시오.

OS/400에 대한 MBCS 지원: DBCS 지원은 자동으로 제공되며 이 변수를 요구하지 않습니다.

DTW_ASSIGN

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

목적

입력 변수 값을 출력 변수에 할당합니다. $\$(Vn)$ (여기에서 n 은 숫자임)가 ROW 블록 외부에서는 인식되지 않으므로 ROW 블록 외부에서 이 값을 참조하려면 이 함수를 사용하여 값을 다른 변수로 할당할 수 있습니다.

이 함수를 사용하여 매크로에서 변수를 변경할 수도 있습니다. 예를 들어, HTML 블록에 대해 DATABASE를 변경할 수 있습니다. (예제는 103페이지의 『DATABASE』를 참고하십시오.)

형식

@DTW_ASSIGN(stringOut, stringIn)

값

표 45. DTW_ASSIGN 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 과 같은 리터럴 문자열이 들어 있는 변수.
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.

예

예제 1:

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

- RC를 "0"으로 설정합니다.

예제 2:

```
@DTW_ASSIGN(string1, string2)
```

- *string1*을 *string2*의 값으로 설정합니다.

DTW_CONCAT

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

목적

두 문자열을 연결합니다.

형식

@DTW_CONCAT(stringIn1, stringIn2, stringOut)

@DTW_rCONCAT(stringIn1, stringIn2)

값

표 46. DTW_CONCAT 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn1</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringIn2</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	문자열 ' <i>stringIn1stringIn2</i> '를 포함하는 변수로, 여기에서 <i>string1</i> 은 <i>string2</i> 와 연결되어 있습니다.

예

예제 1:

```
@DTW_CONCAT("This", " is a test.", result)
```

- 결과: result = "This is a test."

예제 2:

```
@DTW_CONCAT(string1, "1-2-3", result)
```

- 입력: string1 = "Testing "
- 결과: result = "Testing 1-2-3"

예제 3:

```
@DTW_rCONCAT("This", " is a test.")
```

- 결과: "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

목적

`length` 문자에 대해 n 번째 문자에서 지정된 문자열의 부속 문자열을 삭제합니다.

형식

@DTW_DELSTR(stringIn, n, length, stringOut)

@DTW_DELSTR(stringIn, n, stringOut)

@DTW_rDELSTR(stringIn, n, length)

@DTW_rDELSTR(stringIn, n)

값

표 47. DTW_DELSTR 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	삭제할 부속 문자열이 시작하는 문자의 위치. n 이 <i>stringIn</i> 의 길이보다 더 길면 <i>stringOut</i> 이 <i>stringIn</i> 값으로 설정됩니다.
정수	<i>length</i>	OUT	삭제할 부속 문자열의 길이. 생략시 값은 <i>stringIn</i> 의 끝까지 모든 문자를 삭제하는 것입니다.
문자열	<i>stringOut</i>	OUT	수정된 <i>stringIn</i> 유형이 들어 있는 변수.

예

예제 1:

```
@DTW_DELSTR("abcde", "3", "2", result)
```

- 결과: result = "abe"

예제 2:

```
@DTW_rDELSTR("abcde", "4", "1")
```

- 결과: "abc"

DTW_INSERT

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

목적

n 번째 문자 다음에 시작하는 다른 문자열로 문자열을 삽입합니다.

형식

```
@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)
```

값

표 48. DTW_INSERT 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn1</i>	IN	<i>stringIn2</i> 에 삽입할 변수 또는 리터럴 문자열.
문자열	<i>stringIn2</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	뒤에 <i>stringIn1</i> 이 삽입된 <i>stringIn1</i> 내의 문자 위치. n 이 <i>stringIn2</i> 의 길이보다 크면 충분한 문자 수가 될 때까지 채움 문자, <i>pad</i> 로 채워집니다. 생략시 값은 <i>stringIn2</i> 시작 부분에 삽입하는 것입니다.
정수	<i>length</i>	IN	삽입한 <i>stringIn1</i> 의 문자 수. 이 매개변수가 <i>stringIn1</i> 의 길이보다 더 길면 문자열은 채움 문자, <i>pad</i> 로 채워집니다. 생략시 값은 <i>stringIn1</i> 의 길이입니다.
정수	<i>pad</i>	IN	n 및 <i>length</i> 에 사용되는 채움 문자. 생략시 채움 문자는 공백입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn1</i> 의 일부나 전체를 삽입함으로써 수정된 <i>stringIn2</i> 를 포함하는 변수.

예

예제 1:

```
@DTW_INSERT("123", "abc", result)
```

- 결과: result = "123abc"

예제 2:

```
@DTW_INSERT("123", "abc", "5", result)
```

- 결과: result = "abc 123"

예제 3:

```
@DTW_INSERT("123", "abc", "5", "6", result)
```

- 결과: result = "abc 123"

예제 4:

```
@DTW_INSERT("123", "abc", "5", "6", "/", result)
```

- 결과: result = "abc//123//"

예제 5:

```
@DTW_rINSERT("123", "abc", "5", "6", "+")
```

- 결과: "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

목적

n 번째 문자에서 시작하여 거꾸로 가면서 (오른쪽에서 왼쪽으로) 두번째 문자열에서 첫번째 문자열이 나타나는 마지막 경우에 그 위치를 리턴합니다.

형식

@DTW_LASTPOS(stringIn1, stringIn2, n, position)

@DTW_LASTPOS(stringIn1, stringIn2, position)

@DTW_rLASTPOS(stringIn1, stringIn2, n)

@DTW_rLASTPOS(stringIn1, stringIn2)

값

표 49. DTW_LASTPOS 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn1</i>	IN	<i>stringIn2</i> 에서 탐색되는 변수 또는 리터럴 문자열.
문자열	<i>stringIn2</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	<i>stringIn1</i> 탐색을 시작할 <i>stringIn2</i> 내의 문자 위치. 생략시 값은 마지막 문자에서 탐색을 시작하여 역 방향으로 (오른쪽에서 왼쪽으로) 스캔하는 것입니다.
정수	<i>position</i>	OUT	<i>stringIn2</i> 에서 <i>stringIn1</i> 이 나오는 마지막 위치. 발견되지 않는 경우, 0이 리턴됩니다.

예

예제 1:

```
@DTW_LASTPOS(" ", "abc def ghi", result)
```

- 결과: result = "8"

예제 2:

```
@DTW_LASTPOS(" ", "abc def ghi", "10", result)
```

- 결과: result = "8"

예제 3:

```
@DTW_rLASTPOS(" ", "abc def ghi", "7")
```

- 결과: "4"

DTW_LENGTH

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

목적

문자열의 길이를 리턴합니다.

형식

@DTW_LENGTH(stringIn, length)

@DTW_rLENGTH(stringIn)

값

표 50. DTW_LENGTH 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>length</i>	OUT	<i>stringIn</i> 의 문자 수를 포함하는 기호.

예

예제 1:

```
@DTW_LENGTH("abcdefgh",
result)
```

- 결과: result = "8"

예제 2:

```
@DTW_rLENGTH("")
```

- 결과: "0"

DTW_LOWERCASE

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

목적

모두 소문자로 된 문자열을 리턴합니다.

형식

@DTW_LOWERCASE(stringIn, stringOut)

@DTW_rLOWERCASE(stringIn)

@DTW_mLOWERCASE(stringMult1, stringMult2, ..., stringMultn)

값

표 51. DTW_LOWERCASE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	대소문자에 관계없이 문자가 있는 리터럴 문자열 또는 변수.
문자열	<i>stringOut</i>	OUT	모든 문자가 소문자로 지정된 <i>stringIn</i> 을 포함하는 변수.
문자열	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"> 입력: 문자열이 들어 있는 변수. 출력: 소문자로 변환된 입력 문자열이 들어 있는 변수.

예

예제 1:

@DTW_LOWERCASE("This", stringOut)

- 결과: stringOut = "this"

예제 2:

@DTW_rLOWERCASE(string1)

- 입력: string1 = "Hello"
- 결과: "hello"

예제 3:

@DTW_mLOWERCASE(string1, string2, string3)

- 입력: string1 = "THIS", string2 = "IS", string3 = "LOWERCASE"
- 결과: 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

목적

정방향 탐색 패턴을 사용하여, 다른 문자열에서 한 문자열이 나타나는 첫 번째 위치를 리턴합니다.

형식

@DTW_POS(stringIn1, stringIn2, n, nOut)

@DTW_POS(stringIn1, stringIn2, nOut)

@DTW_rPOS(stringIn1, stringIn2, n)

@DTW_rPOS(stringIn1, stringIn2)

값

표 52. DTW_POS 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn1</i>	IN	탐색할 리터럴 문자열 또는 변수.
문자열	<i>stringIn2</i>	IN	탐색할 리터럴 문자열 또는 변수.
정수	<i>n</i>	IN	탐색을 시작할 <i>stringIn2</i> 의 문자 위치. 생략시 값은 <i>stringIn2</i> 의 첫 번째 문자에서 탐색을 시작하는 것입니다.
정수	<i>nOut</i>	OUT	<i>stringIn2</i> 에서 <i>stringIn1</i> 이 첫 번째로 나타나는 경우에 그 위치를 포함하는 변수. 발생이 발견되지 않는 경우, 0이 리턴됩니다.

예

예 1:

```
@DTW_POS("day", "Saturday", result)
```

- 결과: result = "6"

예제 2:

```
@DTW_POS("a", "Saturday", "3", result)
```

- 결과: result = "7"

예제 3:

```
@DTW_rPOS(" ", "abc def ghi", "5")
```

- 결과: "8"

DTW_REVERSE

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

목적

입력 문자열을 역순으로 합니다.

형식

@DTW_REVERSE(stringIn, stringOut)

@DTW_rREVERSE(stringIn)

값

표 53. DTW_REVERSE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	역순으로 할 리터럴 문자열 또는 변수.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 역순 형태가 들어 있는 변수.

예

예제 1:

```
@DTW_REVERSE("This is it.", result)
```

- 결과: result = ".ti si sihT"

예제 2:

```
@DTW_rREVERSE(string1)
```

- 입력: string1 = "reversed"
- 결과: "desrever"

DTW_STRIP

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

목적

선행 공백이나 후행 공백, 또는 입력 문자열의 선행 공백을 모두 제거합니다.

형식

@DTW_STRIP(stringIn, option, stringOut)

@DTW_STRIP(stringIn, stringOut)

@DTW_rSTRIP(stringIn, option)

@DTW_rSTRIP(stringIn)

값

표 54. DTW_STRIP 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>option</i>	IN	<i>stringIn</i> 에서 제거할 공백을 지정합니다. 생략시 값은 B입니다. B 또는 b - 맨 앞 및 맨 뒤 공백을 모두 제거 L 또는 l - 맨 앞 공백만 제거 T 또는 t - 맨 뒤 공백만 제거
문자열	<i>stringOut</i>	OUT	옵션에 의해 지정된 대로 공백이 제거된 <i>stringIn</i> 을 포함하는 변수.

예

예제 1:

```
@DTW_STRIP(" day ",
result)
```

- 결과: result = "day"

예제 2:

```
@DTW_STRIP(" day ", "T", result)
```

- 결과: result = "day"

예제 3:

```
@DTW_rSTRIP(" a day ", "L")
```

- 결과: "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

목적

입력 문자열의 부속 문자열을 리턴합니다. 채움 문자가 포함될 수도 있습니다.

형식

@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)

값

표 55. DTW_SUBSTR 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	탐색할 리터럴 문자열 또는 변수.
정수	<i>n</i>	IN	부속 문자열의 첫번째 문자 위치. 생략시 값은 <i>stringIn</i> 시작 부분에서 시작하는 것입니다.
정수	<i>length</i>	IN	부속 문자열의 문자 수. 생략시 값은 나머지 문자열입니다.
문자열	<i>pad</i>	IN	<i>n</i> 이 <i>stringIn</i> 의 길이보다 더 길거나 <i>length</i> 가 <i>stringIn</i> 보다 더 길 경우 사용되는 채움 문자. 생략시 값은 공백입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 부속 문자열이 들어 있는 변수.

예

예제 1:

```
@DTW_SUBSTR("abc", "2",
result)
```

- 결과: result = "bc"

예제 2:

```
@DTW_SUBSTR("abc", "2", "4", result)
```

- 결과: result = "bc"

예제 3:

```
@DTW_SUBSTR("abc", "2", "4", ".", result )
```

- 결과: result = "bc.."

예제 4:

```
@DTW_rSUBSTR("abc", "2", "6", ".")
```

- 결과: "bc...."

DTW_TRANSLATE

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

목적

입력 및 출력 변환 테이블, *tableI* 및 *tableO*을 사용하여 입력 문자열에 있는 문자를 변환합니다. 매개변수 목록에 *tableI*, *tableO* 및 생략시 문자가 없으면 *stringIn* 매개변수가 대문자로 변환됩니다. *tableI* 및 *tableO*이 목록에 있으면 입력 문자열의 각 문자는 *tableI*에서 탐색되고 *tableO*의 해당 문자로 변환됩니다. *tableI*의 문자가 *tableO*에 해당 문자를 가지고 있지 않으면 생략시 문자가 대신 사용됩니다.

형식

@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)

값

표 56. DTW_TRANSLATE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>tableO</i>	IN	변환 테이블로 사용되는 리터럴 문자열 또는 변수. <i>tableI</i> 또는 생략시 값을 지정하려면 널(NULL)("")을 사용하십시오. 그렇지 않은 경우 이 매개변수는 선택적입니다.
문자열	<i>tableI</i>	IN	<i>stringIn</i> 에서 탐색되는 변수 또는 리터럴 문자열. 생략시 값을 지정하려면 널(NULL)("")을 사용하십시오. 그렇지 않은 경우 이 매개변수는 선택적입니다.
문자열	생략시 값	IN	사용할 생략시 문자. 생략시 값은 공백입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 변환된 결과가 들어 있는 변수.

예

예제 1:

```
@DTW_TRANSLATE("abbc",  
result)
```

- 결과: result = "ABBC"

예제 2:

```
@DTW_TRANSLATE("abbc", "R", "bc", result)
```

- 결과: result = "aRR "

예제 3:

```
@DTW_rTRANSLATE("abcdef", "12", "abcd", ".")
```

- 결과: "12..ef"

예제 4:

```
@DTW_rTRANSLATE("abbc", "", "", "")
```

- 결과: "abbc"

DTW_UPPERCASE

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

목적

문자열을 대문자로 리턴합니다.

형식

@DTW_UPPERCASE(stringIn, stringOut)

@DTW_rUPPERCASE(stringIn)

@DTW_mUPPERCASE(stringMult1, stringMult2, ..., stringMultn)

값

표 57. DTW_UPPERCASE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	대소문자에 관계없이 문자가 있는 리터럴 문자열 또는 변수.
문자열	<i>stringOut</i>	OUT	모든 문자가 대문자로 지정된 <i>stringIn</i> 을 포함하는 변수.
문자열	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"> 입력: 문자열이 들어 있는 변수. 출력: 대문자로 변환된 입력 문자열이 들어있는 변수.

예

예제 1:

```
@DTW_UPPERCASE("Test",
result)
```

- 결과: result = "TEST"

예제 2:

```
@DTW_rUPPERCASE(string1)
```

- 입력: string1 = "Web pages"
- 결과: "WEB PAGES"

예제 3:

```
@DTW_mUPPERCASE(string1, string2, string3)
```

- 입력: string1 = "This", string2 = "is", string3 = "uppercase"
- 결과: string1 = "THIS", string2 = "IS", string3 = "UPPERCASE"

단어 함수

이 함수는 단어나 단어 세트를 수정하여 문자열 함수를 보충합니다. Net.Data 는 단어를 공백으로 분리되는 문자열로 해석하거나, 양쪽에 공백이 있는 문자열로 해석합니다. 몇가지 예는 다음과 같습니다.

문자열 값	단어 수
one two three	3
one , two , three	5
Part 2: Internet Sales Grow	5

OS/390, OS/2, Windows NT 및 UNIX에 대한 MBCS 지원: DTW_MBMODE 구성 값을 사용하여 단어 및 문자열 함수에 대한 MBCS(다중 바이트 문자 세트) 지원을 지정할 수 있습니다. Net.Data 초기화 파일에서 이 값을 지정하십시오. 생략시 값은 지원하지 않는 것입니다. Net.Data 매크로 파일에서 DTW_MBMODE 변수를 설정하여 초기화 파일의 해당 값을 교체할 수 있습니다. 자세한 내용은 *Net.Data Administration and Programming Guide* 와 110 페이지의 『DTW_MBMODE』를 참고하십시오.

OS/400에 대한 MBCS 지원: DBCS 지원은 자동으로 제공되며 이 변수를 요구하지 않습니다.

Net.Data가 지원하는 단어 함수는 다음과 같습니다.

- 176페이지의 『DTW_DELWORD』
- 177페이지의 『DTW_SUBWORD』
- 179페이지의 『DTW_WORD』
- 180페이지의 『DTW_WORDINDEX』
- 181페이지의 『DTW_WORDLENGTH』
- 182페이지의 『DTW_WORDPOS』
- 184페이지의 『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

목적

입력 문자열의 부속 문자열을 리턴합니다. *n* 단어에서 시작하여 *length*에 의해 지정된 단어수 만큼 단어가 삭제됩니다.

형식

@DTW_DELWORD(stringIn, n, length, stringOut)

@DTW_DELWORD(stringIn, n, stringOut)

@DTW_rDELWORD(stringIn, n, length)

@DTW_rDELWORD(stringIn, n)

값

표 58. DTW_DELWORD 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	삭제될 첫번째 단어의 단어 위치.
정수	<i>length</i>	IN	삭제할 단어 수. 생략시 값은 <i>n</i> 에서 <i>stringIn</i> 끝까지 모든 단어를 삭제하는 것입니다. 선택 가능 매개변수.
문자열	<i>stringOut</i>	OUT	수정된 <i>stringIn</i> 유형이 들어 있는 변수.

예

예제 1:

```
@DTW_DELWORD("Now is the time", "5", result)
```

- 결과: result = "Now is the time"

예제 2:

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

- 결과: result = "Now"

예제 3:

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

- 결과: result = "Now time"

예제 4:

```
@DTW_rDELWORD("Now is the time.", "3")
```

- 결과: "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

목적

입력 문자열의 부속 문자열을 리턴합니다. 부속 문자열은 단어 *n*에서 시작되고 *length*에서 지정하는 단어의 수만큼 계속됩니다.

형식

@DTW_SUBWORD(stringIn, n, length, stringOut)

@DTW_SUBWORD(stringIn, n, stringOut)

@DTW_rSUBWORD(stringIn, n, length)

@DTW_rSUBWORD(stringIn, n)

값

표 59. DTW_SUBWORD 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	부속 문자열 첫번째 단어의 단어 위치. 이 값이 <i>stringIn</i> 의 단어 수보다 더 크면 널(NULL)이 리턴됩니다.
정수	<i>length</i>	IN	부속 문자열에 있는 단어 수. 이 값이 <i>n</i> 에서 <i>stringIn</i> 끝까지의 단어 수보다 더 크면 <i>stringIn</i> 끝까지 모든 단어가 리턴됩니다. 생략시 값은 <i>n</i> 에서 <i>stringIn</i> 끝까지 모든 단어를 리턴하는 것입니다.
문자열	<i>stringOut</i>	OUT	<i>n</i> 과 <i>length</i> 에 의해 지정된 <i>stringIn</i> 의 부속 문자열이 포함된 변수.

예

예제 1:

```
@DTW_SUBWORD("Now is the time", "5", result)
```

- 결과: result = ""

예제 2:

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

- 결과: result = "is the time"

예제 3:

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

- 결과: result = "is the"

예제 4:

```
@DTW_rSUBWORD("Now is the time", "3")
```

- 결과: "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

목적

입력 문자열의 지정된 위치로부터 단어 하나를 리턴합니다.

형식

@DTW_WORD(stringIn, n, stringOut)

@DTW_rWORD(stringIn, n)

값

표 60. DTW_WORD 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	리턴할 단어의 단어 위치. 이 값이 <i>stringIn</i> 에 있는 단어 수보다 더 크면 널(NULL)값이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	단어 위치 <i>n</i> 에 단어가 들어 있는 변수.

예

예제 1:

```
@DTW_WORD("Now is the time", "3", result)
```

- 결과: result = "the"

예제 2:

```
@DTW_WORD("Now is the time", "5", result)
```

- 결과: result = ""

예제 3:

```
@DTW_rWORD("Now is the time", "4")
```

- 결과: "time"

DTW_WORDINDEX

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

목적

입력 문자열에서 n 번째 단어의 첫번째 문자의 위치를 리턴합니다.

형식

@DTW_WORDINDEX(stringIn, n, stringOut)

@DTW_rWORDINDEX(stringIn, n)

값

표 61. DTW_WORDINDEX 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	색인으로 만들 단어의 위치. 이 값이 입력 문자열의 단어 수보다 큰 경우 0이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 n 번째 단어의 문자 위치를 포함하는 변수.

예

예제 1:

```
@DTW_WORDINDEX("Now is the time", "3", result)
```

- 결과: result = "8"

예제 2:

```
@DTW_WORDINDEX("Now is the time", "6", result)
```

- 결과: result = "0"

예제 3:

```
@DTW_rWORDINDEX("Now is the time", "2")
```

- 결과: "5"

DTW_WORDLENGTH

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

목적

입력 문자열에서 n 번째 단어의 길이를 리턴합니다.

형식

@DTW_WORDLENGTH(stringIn, n, stringOut)

@DTW_rWORDLENGTH(stringIn, n)

값

표 62. DTW_WORDLENGTH 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	<i>n</i>	IN	길이를 알고자 하는 단어의 위치. 이 값이 입력 문자열의 단어 수보다 큰 경우 0이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 n 번째 단어의 길이를 포함하는 변수.

예

예제 1:

```
@DTW_WORDLENGTH("Now is the time", "1", result)
```

- 결과: result = "3"

예제 2:

```
@DTW_rWORDLENGTH("Now is the time", "6")
```

- 결과: "0"

DTW_WORDPOS

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

목적

다른 문자열에서 어떤 문자열이 처음 나올 때 그 단어 수를 리턴합니다. 여러 공백은 비교를 위해 단일 공백으로 취급됩니다. 비교시 대소문자를 구분합니다.

형식

@DTW_WORDPOS(stringIn1, stringIn2, n, stringOut)

@DTW_WORDPOS(stringIn1, stringIn2, stringOut)

@DTW_rWORDPOS(stringIn1, stringIn2, n)

@DTW_rWORDPOS(stringIn1, stringIn2)

값

표 63. DTW_WORDPOS 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn1</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringIn2</i>	IN	탐색할 리터럴 문자열 또는 변수.
정수	<i>n</i>	IN	탐색을 시작할 <i>stringIn2</i> 내의 단어 위치. 이 값이 <i>stringIn2</i> 에 있는 단어 수보다 더 크면 0이 리턴됩니다. 생략시 값은 <i>stringIn2</i> 의 시작 부분에서 탐색을 시작하는 것입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn2</i> 에서 <i>stringIn1</i> 의 단어 위치.

예

예제 1:

```
@DTW_WORDPOS("the", "Now is the time", result)
```

- 결과: result = "3"

예제 2:

```
@DTW_WORDPOS("The", "Now is the time", result)
```

- 결과: result = "0"

예제 3:

```
@DTW_WORDPOS("The", "Now is the time", "5", result)
```

- 결과: result = "0"

예제 4:

```
@DTW_WORDPOS("is the", "Now is the time", result)
```


- 결과: result = " 2"

예제 5:

```
@DTW_rWORDPOS("be", "To be or not to be", "3")
```

- 결과: "6"

DTW_WORDS

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

목적

문자열에 있는 단어 수를 리턴합니다.

형식

@DTW_WORDS(stringIn, stringOut)

@DTW_rWORDS(stringIn)

값

표 64. DTW_WORDS 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 에 있는 단어 수를 포함하는 변수.

예

예제 1:

```
@DTW_WORDS("Now is the time", result)
```

• 결과:

```
result = "4"
```

예제 2:

```
@DTW_rWORDS(" ")
```

• 결과: "0"

테이블 함수

이 함수들은 Net.Data 테이블에 대한 작업을 단순화하고, REXX, C 또는 Perl 을 사용하여 사용자 자신의 함수를 작성하는 것보다 더 효율적입니다.

- 186페이지의 『DTW_TB_DLIST』
- 188페이지의 『DTW_TB_DUMPV』
- 189페이지의 『DTW_TB_DUMPV』
- 190페이지의 『DTW_TB_HTML_ENCODE』
- 191페이지의 『DTW_TB_INPUT_CHECKBOX』
- 192페이지의 『DTW_TB_INPUT_RADIO』
- 193페이지의 『DTW_TB_INPUT_TEXT』
- 195페이지의 『DTW_TB_LIST』
- 197페이지의 『DTW_TB_SELECT』
- 198페이지의 『DTW_TB_TABLE』
- 200페이지의 『DTW_TB_TEXTAREA』

DTW_TB_DLIST

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

목적

매크로 테이블로부터 HTML 정의 목록을 리턴합니다.

형식

@DTW_TB_DLIST(table, term, def, termstyle, defstyle, anchor, anchor_u, image, image_u)

값

표 65. DTW_TB_DLIST 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	매크로 테이블 변수를 HTML 정의 목록으로 표시하도록 지정하는 기호.
정수	<i>term</i>	IN	용어 이름 값(<DT> 태그 뒤에 오는 텍스트)을 포함하는 테이블 내의 컬럼 번호. 생략시 값은 첫번째 컬럼을 사용하는 것입니다.
정수	<i>def</i>	IN	용어 정의 값(<DD> 태그 뒤에 오는 텍스트)을 포함하는 테이블 내의 컬럼 번호. 생략시 값은 두 번째 컬럼을 사용하는 것입니다.
문자열	<i>termstyle</i>	IN	용어 이름 값에 대한 HTML 요소 목록을 포함하는 변수 또는 리터럴 문자열. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
문자열	<i>defstyle</i>	IN	용어 정의 값에 대한 HTML 요소 목록을 포함하는 변수 또는 리터럴 문자열. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
문자열	<i>anchor</i>	IN	앵커 참조가 작성될 HTML 요소를 지정합니다. 유효값은 DT와 DD입니다. 생략시 값은 앵커 참조를 작성하지 않는 것입니다.
정수	<i>anchor_u</i>	IN	앵커 참조에 대한 URL을 포함하는 테이블 내의 컬럼 번호. 생략시 값은 앵커 참조를 작성하지 않는 것입니다.
문자열	<i>이미지</i>	IN	인라인 이미지를 작성할 HTML 요소를 지정합니다. 유효값은 DT와 DD입니다. 생략시, 인라인 이미지(DT)를 생성하지 않습니다.

표 65. DTW_TB_DLIST 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>image_u</i>	IN	인라인 이미지에 대한 URL을 포함하는 테이블 내의 컬럼 번호. 생략시, 인라인 이미지를 생성하지 않습니다.

예

예제 1: 테이블 자료에 따라 아래에 나타난 HTML을 생성하는 정의 목록을 작성합니다.

```
@DTW_TB_DLIST(Mytable,"3","4","b i","strong","DD","2","DT","1")
```

결과:

```
<DL>
<DT>
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b><i>image1text</i></b>
<DD>
<A HREF="http://www.mycompany.com/anchor1.html"><strong>anchor1text</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/anchor2.html"><strong>anchor2text</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/anchor3.html"><strong>anchor3text</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/anchor4.html"><strong>anchor4text</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

목적

매크로 테이블 변수의 목차를 리턴합니다. 각 테이블 행이 서로 다른 라인에 표시됩니다. 전체 테이블은 <PRE></PRE> 태그로 묶어 표시됩니다.

형식

@DTW_TB_DUMP(table)

값

표 66. DTW_TB_DUMP 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	표시할 매크로 테이블 변수를 지정하는 기호.

예

예제 1:

@DTW_TB_DUMP(Mytable)

이 예에서 생성되는 HTML은 다음과 같습니다.

```
<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

목적

매크로 테이블 변수의 목차를 리턴합니다. 각 테이블 값은 서로 다른 행에 있습니다. 전체 테이블은 <PRE></PRE> 태그로 묶어 표시됩니다.

형식

@DTW_TB_DUMPV(table)

값

표 67. DTW_TB_DUMPV 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	표시할 매크로 테이블 변수를 지정하는 기호.

예

예제 1:

@DTW_TB_DUMPV(Mytable)

이 예의 HTML은 다음과 같습니다.

```
<PRE>
http://www.mycompany.com/images/image1.gif
http://www.mycompany.com/anchor1.html
image1text
anchor1text
http://www.mycompany.com/images/image2.gif
http://www.mycompany.com/anchor2.html
image2text
anchor2text
http://www.mycompany.com/images/image3.gif
http://www.mycompany.com/anchor3.html
image3text
anchor3text
http://www.mycompany.com/images/image4.gif
http://www.mycompany.com/anchor4.html
image4text
anchor4text
</PRE>
```

DTW_TB_HTML ENCODE

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

목적

다음 HTML 문자로 코드화된 입력 매크로 테이블을 리턴합니다.

이름	문자	코드
앰퍼샌드	&	&
큰 따옴표	"	"
보다 큼	>	>
보다 작음	<	<

형식

@DTW_TB_HTML ENCODE(table, collist)

값

표 68. DTW_TB_HTML ENCODE 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	수정할 매크로 테이블 변수.
문자열	<i>collist</i>	IN	코드화할 테이블 내의 컬럼 수. 생략 시 값은 모든 컬럼을 코드화하는 것입니다.

예

예제 1:

@DTW_TB_HTML ENCODE(Mytable, "3 4")

지정된 테이블의 컬럼 3과 4에 있는 특수 문자들은 코드화 형식으로 교체됩니다.

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

목적

매크로 테이블 변수로부터 하나 이상의 HTML 체크 박스 입력 태그를 리턴합니다.

형식

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol, valuecol, rows, checkedrows)

값

표 69. DTW_TB_INPUT_CHECKBOX 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	체크박스 입력 태그로 표시할 매크로 테이블 변수.
문자열	<i>프롬프트</i>	IN	체크 박스 옆에 표시된 텍스트를 포함하는 문자열이나 테이블 내의 컬럼 번호. 이 매개변수는 필수이나 널(NULL)("") 값을 가질 수 있습니다. 프롬프트가 널(NULL)이면 사용되는 값은 <i>namecol</i> 에 대해 정의된 값입니다.
문자열	<i>namecol</i>	IN	입력 필드명이 포함된 문자열이나 테이블 내의 컬럼 번호.
문자열	<i>valuecol</i>	IN	입력 필드 값이 포함된 문자열 또는 테이블 내의 컬럼 번호. 생략시 값은 1입니다.
정수	<i>rows</i>	IN	입력 필드 생성에 사용되는 테이블의 행 목록. 생략시 값은 모든 행을 사용하는 것입니다.
정수	<i>checkedrows</i>	IN	선택할 테이블의 행을 지정하는 행 목록. 생략시 값은 필드를 선택하지 않는 것입니다.

예

예제 1: 세 개의 체크 박스 입력 태그에 대해 HTML을 생성합니다.

```
@DTW_TB_INPUT_CHECKBOX(Mytable,"3","4","","2 3 4","1 3 4")
```

결과:

```
<INPUT TYPE="CHECKBOX" NAME="anchor2text" VALUE="1">image2text<BR>
<INPUT TYPE="CHECKBOX" NAME="anchor3text" VALUE="1" CHECKED>image3text<BR>
<INPUT TYPE="CHECKBOX" NAME="anchor4text" 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

목적

매크로 테이블 변수로부터 하나 이상의 HTML 단일 선택 버튼 입력 태그를 리턴합니다.

형식

@DTW_TB_INPUT_RADIO(table, prompt, namecol, valuecol, rows, checkedrows)

값

표 70. DTW_TB_INPUT_RADIO 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	단일 선택 버튼 입력 태그로 표시할 매크로 테이블 변수.
문자열	<i>프롬프트</i>	IN	단일 선택 버튼 옆에 표시할 텍스트를 포함하는 문자열이나 테이블 내의 컬럼 번호. 이 매개변수는 필수이지만 널(NULL)("") 값을 포함할 수 있습니다. 프롬프트가 널(NULL)이면 <i>valuecol</i> 값을 사용합니다.
문자열	<i>namecol</i>	IN	입력 필드명이 포함된 문자열이나 테이블 내의 컬럼 번호.
문자열	<i>valuecol</i>	IN	입력 필드 값이 포함된 문자열 또는 테이블 내의 컬럼 번호.
문자열	<i>rows</i>	IN	입력 필드 생성에 사용되는 테이블의 행 목록. 생략시, 모든 행을 사용합니다.
정수	<i>checkedrows</i>	IN	해당 단일 선택 버튼을 선택된 상태로 표시할 테이블 내의 행 번호. 값은 하나만 허용됩니다.

예

예제 1: 세 개의 단일 선택 버튼 입력 태그에 대해 HTML을 생성합니다.

```
@DTW_TB_INPUT_RADIO(Mytable,"3","Radio4","4","2 3 4","4")
```

결과:

```
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="anchor2text">image2text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="anchor3text">image3text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="anchor4text" 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

목적

Net.Data 테이블에 지정된 행에 대한 HTML <INPUT> 태그를 리턴합니다. 예를 들어 Net.Data는 다음과 같이 VALUE, SIZE 및 MAXLENGTH 속성을 사용하여 HTML <INPUT> 태그를 생성할 수 있습니다.

```
<INPUT TYPE="TEXT" NAME="someName" VALUE="someValue" SIZE="20" MAXLENGTH="40">
```

형식

@DTW_TB_INPUT_TEXT(table, prompt, namecol, valuecol, size, maxlen, rows)

값

표 71. DTW_TB_INPUT_TEXT 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	텍스트 입력 태그로 표시할 매크로 테이블 변수.
문자열	프롬프트	IN	입력 필드 옆에 표시할 텍스트를 포함하는 문자열 또는 테이블 내의 컬럼 번호. 프롬프트가 없으면 텍스트는 표시되지 않습니다.
문자열	<i>namecol</i>	IN	입력 필드명이 포함된 문자열이나 테이블 내의 컬럼 번호.
문자열	<i>valuecol</i>	IN	생략시 입력 필드 값을 포함하는 문자열이나 테이블 내의 컬럼 번호로 INPUT 태그에서 VALUE 속성에 대해 지정됩니다. 생략시, VALUE 속성 값을 생성하지 않습니다.
정수	<i>size</i>	IN	입력 필드의 문자 수로 INPUT 태그의 SIZE 속성에 대해 지정됩니다. 생략시 값은 가장 긴 생략시 입력 값의 길이이며 생략시 입력 값이 없는 경우에는 10이 됩니다.
정수	<i>maxlen</i>	IN	입력 문자열의 최대 길이로, INPUT 태그의 MAXLENGTH 속성에 대해 지정됩니다. 생략시 값은 MAXLENGTH 속성 값을 생성하지 않는 것입니다.
정수	<i>rows</i>	IN	입력 필드 생성에 사용되는 테이블의 행 목록. 생략시 값은 모든 행을 사용하는 것입니다.

예

예제 1: 세 개의 HTML <INPUT> 태그를 리턴합니다.

```
@DTW_TB_INPUT_TEXT(Mytable,"3","3","4","35","40","1 2 3")
```

결과:

```
<P>image1text  
<INPUT TYPE="TEXT" NAME="image1text" VALUE="anchor1text" SIZE="35" MAXLENGTH="40">  
<P>image2text  
<INPUT TYPE="TEXT" NAME="image2text" VALUE="anchor2text" SIZE="35" MAXLENGTH="40">  
<P>image3text  
<INPUT TYPE="TEXT" NAME="image3text" VALUE="anchor3text" SIZE="35" MAXLENGTH="40">
```

DTW_TB_LIST

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

목적

HTML 목록을 리턴합니다.

형식

@DTW_TB_LIST(table, listtype, listitem, itemstyle, anchor_u, image_u)

값

표 72. DTW_TB_LIST 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	매크로 테이블 변수를 HTML 목록으로 표시하도록 지정하는 기호.
문자열	<i>listtype</i>	IN	생성할 목록 유형. 사용 가능한 값들은 다음과 같습니다. DIR MENU OL UL
정수	<i>listitem</i>	IN	목록 값(태그 뒤에 오는 텍스트)을 포함하는 테이블 내의 컬럼 번호. 생략시, 첫번째 컬럼을 사용합니다.
문자열	<i>itemstyle</i>	IN	용어 이름 값에 대한 HTML 요소 목록이 포함된 리터럴 문자열 또는 변수. 생략시, 스타일 태그를 사용하지 않습니다.
정수	<i>anchor_u</i>	IN	앵커 참조에 대한 URL을 포함하는 테이블 내의 컬럼 번호. 이 값이 지정되지 않는 경우, 앵커 참조가 생성되지 않습니다.
정수	<i>image_u</i>	IN	인라인 이미지에 대한 URL을 포함하는 테이블 내의 컬럼 번호. 이 값이 지정되지 않는 경우, 인라인 이미지가 생성되지 않습니다.

예

예제 1: 정렬된 목록에 대해 HTML 태그를 생성합니다.

```
@DTW_TB_LIST(Mytable,"OL","4","TT U","2","1")
```

결과:

```
<TT><U>
<OL>
<LI><A HREF="http://www.mycompany.com/anchor1.html">
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT="">anchor1text</A>
<LI><A HREF="http://www.mycompany.com/anchor2.html">
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT="">anchor2text</A>
<LI><A HREF="http://www.mycompany.com/anchor3.html">
<IMG SRC="http://www.mycompany.com/images/image3.gif" ALT="">anchor3text</A>
<LI><A HREF="http://www.mycompany.com/anchor4.html">
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT="">anchor4txt</A>
</OL>
</U></TT>
```

DTW_TB_SELECT

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

목적

HTML SELECT 메뉴를 리턴합니다.

형식

@DTW_TB_SELECT(table, name, optioncol, size, multiple, rows, selectrows)

값

표 73. DTW_TB_SELECT 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	SELECT 필드로 표시할 매크로 테이블 변수.
문자열	<i>name</i>	IN	SELECT 필드의 NAME 속성 값.
정수	<i>optioncol</i>	IN	SELECT 필드의 OPTION 태그에 사용할 값을 포함하는 테이블 내의 컬럼 번호. 생략시, 첫번째 컬럼을 사용합니다.
정수	<i>size</i>	IN	SELECT 필드의 OPTION 태그에 사용할 테이블 내의 행 수. 생략시, 모든 행을 사용합니다.
문자열	<i>multiple</i>	IN	복수 선택이 허용되는지를 지정합니다. 생략시, 복수 선택을 허용하지 않는 N입니다.
문자열	<i>selectedrows</i>	IN	SELECT 필드에 사용할 테이블의 행 번호. 생략시, 모든 행을 사용합니다.
문자열	<i>rows</i>	IN	OPTION 태그가 체크표시되는 테이블의 행 목록. 하나 이상의 행을 지정하려면, 여러 매개변수가 Y로 설정되어 있어야 합니다. 생략시, 첫번째 항목을 선택합니다.

예

복수의 선택을 통해 HTML SELECT 메뉴를 생성합니다.

```
@DTW_TB_SELECT(Mytable,"URL6","3","","y","1 2 4","1 4")
```

결과:

```
<SELECT NAME="URL6" SIZE="3" MULTIPLE>
<OPTION SELECTED>image1text
<OPTION>image2text
<OPTION SELECTED>image4text
</SELECT>
```

DTW_TB_TABLE

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

목적

매크로 테이블 변수로부터 HTML 테이블을 리턴합니다.

형식

@DTW_TB_TABLE(table, options, collist, cellstyle, anchor_u, image_u, url_text, url_style)

값

표 74. DTW_TB_TABLE 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	HTML 테이블로 표시할 매크로 테이블 변수.
문자열	<i>option</i>	IN	TABLE 태그 안에 있는 테이블 속성. 생략시, 속성을 사용하지 않습니다. 유효한 값은 다음과 같습니다. • BORDER • CELSPACING • WIDTH
문자열	<i>collist</i>	IN	HTML 테이블에서 사용할 테이블 내의 컬럼 수. 생략시, 모든 컬럼을 사용합니다.
문자열	<i>cellstyle</i>	IN	각 TD 태그에 있는 텍스트 주위에 오게 될, B와 I 같은 HTML 스타일 요소 목록. 생략시, 스타일 태그를 사용하지 않습니다.
정수	<i>anchor_u</i>	IN	앵커 참조 작성에 사용되는 URL을 포함하는 테이블 내의 컬럼 번호. <i>collist</i> 에도 컬럼을 지정해야 합니다. 생략시, 앵커 참조 태그를 생성하지 않습니다.
정수	<i>image_u</i>	IN	인라인 이미지 작성에 사용되는 URL을 포함하는 테이블 내의 컬럼 번호. <i>collist</i> 에도 컬럼을 지정해야 합니다. 생략시, 이미지 태그를 생성하지 않습니다.
정수	<i>url_text</i>	IN	앵커 참조나 인라인 이미지에 표시할 텍스트를 포함하는 테이블 내의 컬럼 번호. 생략시, URL 자체를 사용합니다.

표 74. DTW_TB_TABLE 매개변수 (계속)

자료 유형	매개변수	사용	설명
문자열	<i>url_style</i>	IN	<i>url_text</i> 에 지정된 텍스트에 대한 HTML 스타일 요소 목록. 생략시, 스타일 태그를 생성하지 않습니다.

예

예제 1: 경계가 있으며 B(굵은체)와 I(이탤릭체) 태그를 사용하여 테이블에 대한 HTML 태그를 생성합니다.

```
@DTW_TB_TABLE(Mytable,"BORDER","4 2 1","i","2","1","4","b")
```

결과:

```
<TABLE BORDER>
<TR>
<TH>TITLE
<TH>ANCHORURL
<TH>IMAGEURL
<TR>
<TD><i>anchor1text</i>
<TD><A HREF="http://www.mycompany.com/anchor1.html"><b>anchor1text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b>anchor1text</b>
<TR>
<TD><i>anchor2text</i>
<TD><A HREF="http://www.mycompany.com/anchor2.html"><b>anchor2text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b>anchor2text</b>
<TR>
<TD><i>anchor3text</i>
<TD><A HREF="http://www.mycompany.com/anchor3.html"><b>anchor3text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b>anchor3text</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

목적

매크로 테이블 변수로부터 HTML TEXTAREA 태그를 리턴합니다.

형식

@DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol, rows)

값

표 75. DTW_TB_TEXTAREA 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	TEXTAREA 태그로 표시할 매크로 테이블 변수.
문자열	<i>name</i>	IN	텍스트 영역의 이름.
정수	<i>numrows</i>	IN	행에 지정된 텍스트 영역의 높이. 생략시 값은 테이블에 있는 행의 수입니다.
정수	<i>numcols</i>	IN	컬럼에 지정된 텍스트 영역의 너비. 생략시 값은 테이블에서 가장 긴 행의 길이입니다.
정수	<i>valuecol</i>	IN	그 값이 텍스트 영역에 표시되는 테이블 내의 컬럼 번호. 생략시 값은 첫번째 컬럼입니다.
문자열	<i>rows</i>	IN	TEXTAREA 태그를 생성하는 데 사용되는 테이블 내의 행 목록. 생략시, 모든 행을 사용합니다.

예

예제 1: HTML TEXTAREA 태그를 생성하고 포함시킬 열을 지정합니다.

```
@DTW_TB_TEXTAREA(Mytable,"textarea5","3","70","4","1 3 4")
```

결과:

```
<TEXTAREA NAME="textarea5" ROWS="3" COLS="70">
anchor1text
anchor3text
anchor4text
<TEXTAREA>
```

플랫 파일 인터페이스 함수

FFI(플랫 파일 인터페이스)를 사용하면 플랫 파일에 자료를 저장할 수 있을 뿐 아니라 플랫 파일 소스(텍스트 파일)로부터 자료를 열고, 읽고, 조작할 수 있습니다.

플랫 파일 인터페이스 분리문자

성능을 향상시키기 위해, 일련의 SQL 요청에 대한 Net.Data를 표 형식의 출력을 플랫 파일에 보관할 수 있습니다. SQL 요청을 다시 수행하는 대신, 이후의 요청시 플랫 파일을 검색할 수 있습니다.

Net.Data 플랫 파일은 Net.Data 테이블로부터 작성할 수 있고, Net.Data 테이블은 플랫 파일로부터 작성할 수 있습니다. 테이블과 플랫 파일간 변환을 하려면, 테이블의 컬럼과 플랫 파일의 레코드 사이에 대응을 정의해야 합니다. 분리 문자는 플랫 파일에서 레코드의 일부 구분하고 테이블의 컬럼에 대응되는 방법 및 테이블의 컬럼이 플랫 파일에 있는 레코드로 대응되는 방법을 정의하는 메서드를 제공합니다.

다음과 같은 두 가지 분리문자가 있습니다.

개행 문자(ASCITEXT)

사용자 테이블이 한 컬럼으로 구성되어 있는 경우 이 변환을 사용하십시오. Net.Data는 해당 플랫 파일의 각 레코드를 테이블내 단일 행으로 대응시킵니다. 이 경우에 플랫 파일의 레코드들을 분리하는 정규 개행 문자가 분리 문자로 유일하게 사용됩니다.

개행 문자 및 분리문자 문자열(DELIMITED)

사용자 테이블이 여러 컬럼으로 구성되어 있는 경우 이 변환을 사용하십시오. Net.Data가 테이블의 행으로부터 플랫 파일 레코드를 작성하는 경우, 항목간 분리자로서 분리문자 문자열을 둡니다. Net.Data가 플랫 파일로부터 테이블을 재작성하는 경우, 분리문자 문자열을 사용하여 테이블내 컬럼에 둘 각 행의 문자열을 결정합니다. 이런 경우 정규 개행 문자는 테이블 내 행에 해당하는 플랫 파일 내 레코드들을 분리하고 분리문자 문자열은 단일 레코드 내 항목들을 분리합니다.

DTWF_SEARCH 함수를 사용하여 Net.Data 테이블로부터 작성된 플랫 파일에 보유되어 있는 특정 레코드들을 검색할 수 있습니다. Net.Data 테이블에 있는 행으로 플랫 파일에 문자열이 들어 있는 모든 레코드를 리턴하려면 DTWF_SEARCH에 문자열을 지정하십시오.

플랫 파일 인터페이스 함수

- 203페이지의 『DTWF_APPEND』
- 205페이지의 『DTWF_CLOSE』
- 206페이지의 『DTWF_DELETE』

- 206페이지의 『DTWF_DELETE』
- 210페이지의 『DTWF_OPEN』
- 212페이지의 『DTWF_READ』
- 214페이지의 『DTWF_REMOVE』
- 215페이지의 『DTWF_SEARCH』
- 217페이지의 『DTWF_UPDATE』
- 219페이지의 『DTWF_WRITE』

DTWF_APPEND

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

목적

테이블 변수 목차를 파일 끝에 기록합니다.

형식

@DTWF_APPEND(filename, transform, delimiter, table, retry, rows)

값

표 76. DTWF_APPEND 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	변수의 내용이 추가되는 파일의 이름. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	레코드를 읽어 들이는 테이블 변수.
정수	재시도	IN	파일을 즉시 첨부할 수 없는 경우, 재시도 횟수. 생략시, 재시도를 하지 않습니다.
정수	<i>rows</i>	IN	첨부할 테이블의 최대 행 수. 생략시, 모든 행을 첨부합니다. 0을 지정하면 모든 행을 첨부합니다.

예

예제 1:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
}%
@DTWF_APPEND(myFile, "DELIMITED", " ;", myTable)
```

예제 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCII TEXT", " ;", myTable)
```

예제 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCII TEXT", " ;", 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

목적

DTWF_OPEN로 열린 파일을 닫습니다.

형식

@DTWF_CLOSE(filename, retry)

값

표 77. DTWF_CLOSE 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	닫을 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
정수	재시도	IN	파일을 즉시 닫을 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

예

예제 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

목적

파일에서 레코드를 삭제합니다. (빈 파일은 삭제하지 않습니다.)

형식

@DTWF_DELETE(filename, transform, delimiter, retry, rows, startrow)

값

표 78. DTW_DELETE 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	레코드를 삭제할 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
정수	재시도	IN	레코드를 즉시 삭제할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.
정수	<i>rows</i>	IN	삭제할 최대 행 수. 생략시 값은 모든 행을 삭제하는 것입니다. 0을 지정하면 모든 행이 삭제됩니다.
정수	<i>startrow</i>	INOUT	삭제를 시작할 행 번호. 값 1은 첫번째 행에서 삭제가 시작됨을 의미합니다. 이 값이 파일의 전체 행 수보다 큰 경우, 마지막 레코드로 값이 변경되고 오류가 발생합니다. 생략시, 1에서 시작합니다.

예

예제 1:


```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "5000"
    myRows = "2"
}%
@DTWF_DELETE(myFile, "Delimited", "|", myWait, myRows)
```

예제 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myStart = "1"
    myRows = "2"
}%
@DTWF_DELETE(myFile, "Asciiertext", "|", "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

목적

레코드를 파일에 삽입합니다.

형식

@DTWF_INSERT(filename, transform, delimiter, table, retry, rows, startrow)

값

표 79. DTWF_INSERT 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	레코드를 삽입할 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	테이블에 레코드를 삽입할 테이블 변수.
정수	재시도	IN	파일에 즉시 기록할 수 없는 경우 재시도되는 횟수. 생략시, 재시도를 하지 않습니다.
정수	<i>rows</i>	IN	테이블로부터 삽입할 최대 행 수. 생략시, 모든 행을 삽입합니다. 값 0은 모든 행을 삽입합니다.
정수	<i>startrow</i>	INOUT	삽입을 시작할 행 번호. 1을 지정하면 첫번째 행에서 삽입이 시작됩니다. 이 값이 파일의 전체 행 수보다 큰 경우, 마지막 레코드로 값이 변경되고 오류가 발생합니다. 생략시, 1에서 시작합니다.

예

예제 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "3000"
}%
@DTWF_INSERT(myFile, "Delimited", "|", myTable, myWait)
```

예제 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

목적

파일을 명시적으로 엽니다. DTWF_OPEN는 파일을 열린 상태로 유지하며 그렇지 않은 경우 각 플랫폼 파일 조작 후에 파일을 닫습니다.

성능 팁: 파일을 여는 횟수를 줄이려면 DTWF_OPEN을 사용하십시오.

파일은 DTWF_CLOSE를 사용하여 닫거나 매크로 처리가 완료될 때까지 열린 상태를 유지합니다.

형식

@DTWF_OPEN(filename, mode, retry)

값

표 80. DTWF_OPEN 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	열 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	모드	IN	요청되는 액세스 유형은 다음과 같습니다. <ul style="list-style-type: none"> • r - 읽기 위해 기존의 파일을 엽니다. • w - 쓰기 위해 파일을 작성합니다. (같은 이름의 파일이 있는 경우 기존의 파일은 없어집니다.) • a - 첨부하기 위해 파일을 엽니다. 해당 파일이 없는 경우 Net.Data는 파일을 작성합니다. • r+ - 읽기 및 쓰기를 위해 기존의 파일을 엽니다. • w+ - 읽기 및 쓰기를 위해 파일을 작성합니다. (같은 이름의 파일이 있는 경우 기존의 파일은 없어집니다.) • a+ - 읽기 및 첨부을 위해 첨부 모드로 파일을 엽니다. 해당 파일이 없는 경우 Net.Data는 파일을 작성합니다.
정수	재시도	IN	파일을 즉시 열 수 없는 경우 재시도 횟수. 생략시, 재시도를 하지 않습니다.

예

예제 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

목적

파일에서 테이블 변수로 레코드를 읽어옵니다.

형식

@DTWF_READ(filename, transform, delimiter, table, retry, rows, startrow, columns)

값

표 81. DTWF_READ 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	테이블 변수로 레코드를 읽을 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	OUT	파일 레코드를 읽어 넣을 테이블 변수.
정수	재시도	IN	파일을 즉시 읽을 수 없는 경우, 재시도 횟수. 생략시, 재시도를 하지 않습니다.
정수	<i>rows</i>	INOUT	테이블로 읽어들이 파일 레코드의 최대 수. 생략시, 모든 레코드를 읽거나, 테이블이 꽉 찰때까지 읽습니다. 값 0은 파일 끝까지 읽는 것을 의미합니다. 결과 테이블의 행 수가 리턴됩니다.
정수	<i>startrow</i>	IN	읽기를 시작할 파일의 레코드. 생략시, 첫번째 레코드에서 읽기 시작합니다.
정수	컬럼	OUT	테이블의 컬럼 수를 리턴합니다.

예

예제 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1000"  
%}  
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait)
```

예제 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "0"  
    myRows = "0"  
    myStartrow = "1"  
    myColumns = ""  
%}  
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait, myRows,  
            myStartrow, myColumns)
```

예제 3:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_READ(myFile, "ASCITEXT", ";", myTable, myColumns)  
@DTW_TB_TABLE(myTable, "BORDER", "")
```

DTWF_REMOVE

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

목적

전체 파일을 삭제합니다.

형식

@DTWF_REMOVE(filename, retry)

값

표 82. DTW_REMOVE 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	삭제할 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
정수	재시도	IN	파일을 즉시 삭제할 수 없는 경우 재시도 횟수. 생략시, 재시도를 하지 않습니다.

예

예제 1:

```
%DEFINE myFile = "c:/private/myfile"
@DTWF_REMOVE(myFile)
```

예제 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

목적

테이블 변수로 문자열 탐색 결과를 리턴합니다.

형식

@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows, startrow)

값

표 83. DTWF_SEARCH 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	탐색할 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	OUT	탐색 결과가 위치할 테이블 변수. <i>transform</i> 이 DELIMITED인 경우 다음의 세 컬럼이 리턴됩니다. <ul style="list-style-type: none"> • 일치가 발견된 행. • 일치가 발견된 컬럼. • 파일로부터의 일치 컬럼.
문자열	<i>searchFor</i>	IN	탐색할 문자들의 문자열.
정수	재시도	IN	파일을 즉시 탐색할 수 없는 경우 재시도 횟수. 생략시, 재시도를 하지 않습니다.

표 83. DTWF_SEARCH 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>rows</i>	INOUT	테이블로 읽어들이 최대 행 수. 생략시, 모든 행을 읽거나 테이블 이 끝 찰때까지 읽습니다. 0을 지정하면 파 일 끝까지 읽습니다. 이 매개변수에 의해 결과 테이블의 행 수가 리턴됩니 다.
정수	<i>startrow</i>	IN	탐색을 시작할 파일의 레코드. 생략시 값은 1입니다. 즉, 첫번째 레코드에서 탐색을 시작합니다.

예

예제 1:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "1000"
  mySearch = "0123456789abcdef"
@DTWF_SEARCH(myFile, "DELIMITED", ";",
              myTable, mySearch, myWait)
```

예제 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

목적

테이블 변수로부터 파일 레코드를 갱신합니다.

형식

@DTWF_UPDATE(filename, transform, delimiter, table, retry, rows, startrow)

값

표 84. DTWF_UPDATE 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	테이블 변수로부터 레코드가 갱신되는 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	파일 레코드가 갱신될 해당 테이블 변수.
정수	재시도	IN	파일에 즉시 기록할 수 없는 경우 재시도 횟수. 생략시, 재시도를 하지 않습니다.
정수	<i>rows</i>	IN	테이블에서 갱신할 최대 레코드 수. 생략시, 모든 레코드를 갱신합니다. 값 0은 파일의 모든 행을 갱신하는 것을 의미합니다.
정수	<i>startrow</i>	INOUT	갱신할 첫번째 파일 레코드. 생략시 값은 1입니다. 즉, 파일의 시작 부분에서 갱신이 시작됩니다. 이 값이 파일의 전체 레코드 수보다 큰 경우, 파일의 마지막 레코드 번호를 표시하도록 값이 변경되고 오류가 발생합니다.

예

예제 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1500"  
    myRows = "2"  
%}  
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

예제 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

목적

테이블 변수 목차를 파일에 기록합니다.

형식

@DTWF_WRITE(filename, transform, delimiter, table, retry, rows, startrow)

값

표 85. DTWF_WRITE 매개변수

자료 유형	매개변수	사용	설명
문자열	파일명	INOUT	테이블 변수의 레코드를 기록할 파일명. 성공적인 호출 완료 후에 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이의 개행 문자로 테이블을 파일에 쓰고 분리문자 매개변수를 무시합니다. • DELIMITED - 분리문자 매개변수에 지정된 분리문자로 테이블을 파일에 씁니다.
문자열	분리문자	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	행을 파일로 반출하는 데 사용되는 테이블 변수.
정수	재시도	IN	파일에 즉시 기록할 수 없는 경우 재시도 횟수. 생략시, 재시도를 하지 않습니다.
정수	<i>rows</i>	IN	기록할 파일 레코드의 최대 수. 생략시, 전체 테이블을 씁니다. 값 0은 파일 끝까지 모든 레코드를 쓰는 것을 의미합니다.
정수	<i>startrow</i>	INOUT	파일 내에서 쓰기를 시작할 레코드 번호. 생략시 값은 1입니다. 즉, 첫번째 레코드에서 시작합니다. 파일 범위를 넘어서는 값이 지정되는 경우, 파일의 마지막 행에 오류가 발생합니다.

예

예제 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "DELIMITED", ";", myTable)
```

예제 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000")
```

예제 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000", "10", "50")
```

웹 레지스트리 함수

웹 레지스트리란, 항목을 쉽게 추가, 검색, 삭제하기 위해 Net.Data에 의해 관리되는 키가 들어 있는 파일입니다. 한 시스템에 여러 개의 Net.Data 웹 레지스트리를 작성할 수 있습니다. 각 레지스트리에는 이름이 있고, 여러 개의 항목이 포함될 수 있습니다. Net.Data는 레지스트리와 그 안에 포함된 항목들을 관리하기 위한 함수를 제공합니다.

- 222페이지의 『DTWR_ADDENTRY』
- 223페이지의 『DTWR_CLEARREG』
- 224페이지의 『DTWR_CREATEREG』
- 225페이지의 『DTWR_DELENTY』
- 226페이지의 『DTWR_DELREG』
- 227페이지의 『DTWR_LISTREG』
- 228페이지의 『DTWR_LISTSUB』
- 229페이지의 『DTWR_RTVENTRY』
- 230페이지의 『DTWR_UPDATEENTRY』

제한사항: OS/2를 사용할 경우 *registry*, *registryVariable*, *registryData* 매개변수에 대해서는 별표(*)를 사용하지 않도록 하십시오.

DTWR_ADDENTRY

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

목적

항목을 웹 레지스트리에 추가합니다.

형식

@DTWR_ADDENTRY(registry, registryVariable, registryData, index)

값

표 86. DTWR_ADDENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	레지스트리	IN	항목을 추가할 레지스트리명.
문자열	registryVariable	IN	추가할 레지스트리 항목의 registryVariable 문자열 부분의 값.
문자열	registryData	IN	추가할 레지스트리 항목의 registryData 문자열 부분의 값.
문자열	색인	IN	추가할 색인 항목에 있는 registryVariable 문자열 색인 부분의 값. 이 매개변수는 선택적입니다. 색인 항목이 지정되면, 지정된 레지스트리에 추가됩니다.

예

예제 1:

```
@DTWR_ADDENTRY("Myregistry", "Jones", "http://Advantis.com/~Jones/webproj")
```

예제 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

목적

웹 레지스트리에서 항목을 제거합니다.

형식

@DTWR_CLEARREG(registry)

값

표 87. DTWR_CLEARREG 매개변수

자료 유형	매개변수	사용	설명
문자열	레지스트리	IN	제거할 레지스트리명.

예

예제 1:

@DTWR_CLEARREG("Myregistry")

DTWR_CREATEREG

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

목적

새로 웹 레지스트리를 작성합니다.

형식

@DTWR_CREATEREG(registry, security)

값

표 88. DTWR_CREATEREG 매개변수

자료 유형	매개변수	사용	설명
문자열	레지스트리	IN	작성할 레지스트리명.
문자열	보안	IN	레지스트리 작성 시 사용하는 보안 유형. UNIX 운용 시스템에서, 생략시 보안은 레지스트리가 작성된 디렉토리와 같습니다. 사용자, 그룹 및 공용의 세 가지 보안 그룹에 대해 보안을 지정하십시오. R은 읽기 권한을, W는 쓰기 권한을, X는 실행 권한을 부여합니다. 예를 들어 세 그룹 모두에 전체 권한을 부여하려면 이 매개변수에 대해 *RWX, *RWX, *RWX를 지정하십시오.

예

예제 1:

```
@DTWR_CREATEREG("myRegistry")
```

예제 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

목적

웹 레지스트리에서 항목을 삭제합니다.

형식

@DTWR_DELENTY(registry, registryVariable, index)

값

표 89. DTWR_DELENTY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	항목을 제거할 레지스트리명.
문자열	<i>registryVariable</i>	IN	제거할 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>index</i>	IN	색인 항목에 있는 <i>registryVariable</i> 문자열 색인 부분의 값. 이것은 선택적 매개변수입니다. 색인 항목이 지정된 경우, 레지스트리에서 제거됩니다.

예

예제 1:

```
@DTWR_DELENTY("Myregistry", "Jones")
```

예제 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

목적

웹 레지스트리 삭제

형식

@DTWR_DELREG(registry)

값

표 90. DTWR_DELREG 매개변수

자료 유형	매개변수	사용	설명
문자열	레지스트리	IN	삭제할 레지스트리명.

예

예제 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

목적

전체 웹 레지스트리를 나열합니다. DTWR_LISTREG는 사용자가 제공한 OUT 테이블 변수의 레지스트리 항목에 대한 정보를 리턴합니다. 테이블 변수는, 매개변수로서 LISTREG 레지스트리 연산에 대한 FUNCTION 블록으로 전달 되기 전에 사용자 매크로에 정의됩니다.

사용자가 테이블의 최대 행 수에 대해 ALL 옵션을 사용하여 테이블 변수를 정의한 경우, 이 조작을 수행하면 각 테이블 행에 하나씩 테이블에 사용 가능한 모든 레지스트리 항목이 나열됩니다. 한편, 사용자가 테이블 행의 최대 수로 X 값을 지정했는 데 지정된 레지스트리에 X 항목보다 많은 항목이 있는 경우, 첫번째 X 항목들만 표시되고, 오류 코드가 전송됩니다. 이 코드는 사용 가능한 테이블 행이 충분치 못해 추가 항목을 표시할 수 없어 부분적인 목록만 표시되었음을 나타냅니다. X 값이 지정된 레지스트리에서 사용 가능한 항목 수를 초과하는 경우, 모든 레지스트리 항목이 표시됩니다.

항상 테이블에는 2개 컬럼이 있습니다. 테이블에 대한 컬럼 표제는 웹 레지스트리 언어 환경에 의해 "REGISTRY_VARIABLE"과 "REGISTRY_DATA"로 설정됩니다.

형식

@DTWR_LISTREG(registry, registryTable)

값

표 91. DTWR_LISTREG 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	표시할 레지스트리명.
문자열	<i>registryTable</i>	OUT	레지스트리 항목이 있는 테이블 변수의 이름.

예

예제 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

목적

웹 레지스트리에 직속 부속키 항목이 나열됩니다. DTWR_LISTSUB는 사용자가 제공한 OUT 테이블 매개변수의 레지스트리 항목에 대한 정보를 리턴합니다. 테이블 변수는, 매개변수로서 LISTSUB 레지스트리 연산으로 전달되기 전에 매크로에 정의됩니다.

사용자가 테이블의 최대 행 수에 대해 ALL 옵션을 사용하여 테이블 변수를 정의한 경우, 이 조작을 수행하면 각 테이블 행에 하나씩, 테이블에 사용 가능한 모든 레지스트리 항목이 나열됩니다. 한편, 사용자가 테이블 행의 최대 수로 X 값을 지정했는 데 지정된 레지스트리에 X 항목보다 많은 항목이 있는 경우, 첫번째 X 항목들만 표시되고, 오류 코드가 전송됩니다. 이 코드는 사용 가능한 테이블 행이 충분치 못해 추가 항목을 표시할 수 없어 부분적인 목록만 표시되었음을 나타냅니다. X 값이 지정된 레지스트리에서 사용 가능한 항목 수를 초과하는 경우, 모든 레지스트리 항목이 표시됩니다. 테이블에 있는 컬럼의 수는 항상 1입니다.

테이블에 대한 컬럼 표제는 "REGISTRY_SUBKEY"로 설정됩니다.

이 함수는 Windows95 시스템 레지스트리와 호환되는 운용 시스템에서만 유효합니다.

형식

@DTWR_LISTSUB(registry, registryTable)

값

표 92. DTWR_LISTSUB 매개변수

자료 유형	매개변수	사용	설명
문자열	레지스트리	IN	표시할 레지스트리명.
문자열	registryTable	OUT	레지스트리 항목이 놓인 테이블 변수의 이름.

예

예제 1:

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

```
@DTWR_LISTSUB("URLLIST", RegistryTable)
```

DTWR_RTVENTRY

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

목적

웹 레지스트리 항목에서 registryData 문자열을 검색합니다.

형식

@DTWR_RTVENTRY(registry, registryVariable, registryData, index)

@DTWR_rRTVENTRY(registry, registryVariable, index)

값

표 93. DTWR_RTVENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	검색할 항목이 들어 있는 레지스트리 명.
문자열	<i>registryVariable</i>	IN	registryData 문자열이 검색되는 레지스트리 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>registryData</i>	OUT	<i>registryVariable</i> 와 일치하는 레지스트리 항목의 <i>registryData</i> 문자열 부분의 값을 리턴합니다.
문자열	<i>index</i>	IN	<i>registryData</i> 문자열이 리턴되는 색인 항목에 있는 <i>registryVariable</i> 문자열의 색인 부분의 값. 이것은 선택적 매개변수입니다. 이 값이 지정된 경우 색인 항목의 <i>registryData</i> 문자열이 리턴됩니다.

예

예제 1:

```
%DEFINE RegistryData = ""
@DTWR_RTVENTRY("Myregistry", "Jones", RegistryData)
```

예제 2:

```
@DTWR_RTVENTRY("URLLIST", "SMITH", RegistryData, "WORK_URL")
```

예제 3:

```
@DTWR_rRTVENTRY("Myregistry", "Jones")
```

예제 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

목적

지정된 레지스트리 항목에 대한 기존의 *registryData* 문자열 값을 호출자가 지정한 새 값으로 대체합니다. *registryVariable* 문자열은 변경할 수 없습니다.

형식

@DTWR_UPDATEENTRY(registry, registryVariable, newData, index)

값

표 94. DTWR_UPDATEENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	갱신할 항목이 들어 있는 레지스트리 명.
문자열	<i>registryVariable</i>	IN	갱신할 레지스트리 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>newData</i>	IN	갱신할 레지스트리 항목의 <i>registryData</i> 문자열 부분에 대한 새로운 값.
문자열	<i>index</i>	IN	갱신할 색인 항목에 있는 <i>registryVariable</i> 문자열 색인 부분의 값. 이것은 선택적 매개변수입니다. 색인 항목이 지정된 경우, 갱신됩니다.

예

예제 1:

```
@DTWR_UPDATEENTRY("Myregistry", "Jones", "http://advantis.com/~Jones/personal")
```

예제 2:

```
@DTWR_UPDATEENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/personal", "WORK_URL")
```

부록A. DB2 WWW 연결

DB2 WWW 연결이 설치되어 있으면 Net.Data를 통해 기존의 응용 프로그램을 실행할 수 있습니다. Net.Data 버전 2 기능을 활용하여 응용 프로그램을 갱신하는 것이 좋습니다.

다음과 같은 DB2 WWW 언어 구성이 있습니다.

- 231페이지의 『EXEC_SQL』
- 231페이지의 『HTML_INPUT』
- 231페이지의 『HTML_REPORT』
- 231페이지의 『SQL』
- 232페이지의 『SQL_MESSAGE』
- 232페이지의 『SQL_REPORT』
- 233페이지의 『SQL_CODE』

EXEC_SQL

이 언어 구성은 SQL 블록을 호출합니다. 함수 대신 SQL문을 호출하는 것이 좋습니다. 자세한 내용은 16페이지의 『FUNCTION 블록』을 참고하십시오.

HTML_INPUT

이 언어 구성은 HTML 블록 INPUT과 동일합니다. 자세한 내용은 26페이지의 『HTML 블록』을 참고하십시오.

HTML_REPORT

이 언어 구성은 HTML 블록 REPORT와 동일합니다. 자세한 내용은 26페이지의 『HTML 블록』을 참고하십시오.

SQL

이 언어 구성은 Net.Data에서 FUNCTION(DTW_SQL)을 통해 호출된 함수와 동일합니다.

이는 SQL_REPORT와 SQL_MESSAGE문을 포함할 수 있으며, 이 명령문들도 DB2 WWW 연결에 들어 있습니다. DB2 WWW 연결은 nsamed %SQL 블록을 지원하지 않습니다.

예제:

예제 1: DB2 WWW 연결 매크로

```
%SQL{
UPDATE $(dbtbl) SET URL='$(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>
%}
```

예제 1: Net.Data 매크로와 같음

```
%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

이 언어 구성은 Net.Data MESSAGE문과 동일합니다. 예제는 48페이지의 『MESSAGE 블록』을 참고하십시오.

SQL_REPORT

이 언어 구성은 Net.Data REPORT문과 동일합니다. 예제는 53페이지의 『REPORT 블록』을 참고하십시오.

SQL_CODE

이 언어 구성은 DB2 WWW 연결로부터 가져온 것이며 호환을 위해 Net.Data에 의해 지원됩니다. 이것은 83페이지의 『RETURN_CODE』와 동일합니다.

부록B. Net.Data 운용 시스템 참조

각 운용 시스템에서 모든 Net.Data 기능이 지원되는 것은 아닙니다. 이 절에는 운용 시스템에 대해 지원되는 기능이 나옵니다. **X**가 있으면 해당 기능이 지원됩니다.

여기에 나열된 일부 기능은 아직 일반적인 용도로는 사용할 수 없습니다. 따라서 Windows NT, OS/2 및 UNIX 운용 시스템의 Net.Data 버전 2는 아직 베타 버전이나 베타 이전 버전일 수 있습니다.

표 95. Net.Data 언어 환경

언어 환경	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
생략시 값	X	X	X	X	X	X	X	X
플랫 파일 인터페이스	X	X	X	X	X	X	X	X
IMS 웹				X				X
자바 애플릿	X	X	X	X		X	X	X
자바 응용 프로그램	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
웹 레지스트리	X	X	X		X	X	X	X

표 96. Net.Data 저장 프로시듀어 자료 유형

자료 유형	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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

표 96. Net.Data 저장 프로시듀어 자료 유형 (계속)

자료 유형	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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
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
VARCHAR	X	X	X	X	X	X	X	X
VARGRAPHIC	X	X	X	X	X	X	X	X

표 97. Net.Data 구성 변수

구성 변수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
CACHE_MACHINE	X							
CACHE_PORT	X							
DefaultDBCp				X				
DB2INSTANCE	X	X	X			X	X	X
DB2MSGs				X				
DB2PLAN				X				
DB2SSID				X				
DSNAOINI				X				
DTW_DEFAULT_EDITMASK					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_SQL_ISOLATION					X			
DTW_SQL_NAMING_MODE					X			
DTWR_CLOSE_REGISTRIES					X			
LOGIN	X	X	X		X	X	X	X
PASSWORD	X	X	X		X	X	X	X

표 98. Net.Data 변수

변수	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
DB2PLAN				X				
DB2SSID				X				
DB_CASE	X	X	X	X	X	X	X	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
DTW_MACRO_FILENAME	X	X	X	X	X	X	X	X
DTW_MACRO_LAST_MODIFIED	X	X	X	X	X	X	X	X
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
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
N_columnName	X	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

표 98. Net.Data 변수 (계속)

변수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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

표 99. Net.Data 함수

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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							
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
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
DTW_FORMAT	X	X	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_HTML_ENCODE	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

표 99. Net.Data 함수 (계속)

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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_SETENV	X	X	X	X	X	X	X	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_DLIST	X	X	X	X	X	X	X	X
DTW_TB_DUMPH	X	X	X	X	X	X	X	X
DTW_TB_DUMPV	X	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
DTW_TB_LIST	X	X	X	X	X	X	X	X
DTW_TB_SELECT	X	X	X	X	X	X	X	X
DTW_TB_TABLE	X	X	X	X	X	X	X	X
DTW_TB_TEXTAREA	X	X	X	X	X	X	X	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

표 99. Net.Data 함수 (계속)

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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_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
DTWR_RTVENTRY	X	X	X		X	X	X	X
DTWR_UPDATEENTRY	X	X	X		X	X	X	X

표 100. Net.Data 인터페이스

인터페이스 유형	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
Live Connection	X	X	X				X	X
Lotus Domino Go Web Server (GWAPI)	X		X	X				X
Netscape API (NSAPI)	X						X	X
Servlets	X							X

표 101. Net.Data 툴

툴	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
관리 툴	X		X					X
NetObjects Fusion Plug-ins								X
마법사	X	X	X			X	X	X

부록C. 주의사항

이 정보는 미국에서 제공하는 제품 및 서비스에 대해 개발된 것입니다. IBM은 다른 국가에서는 이 문서에서 다루는 제품, 서비스 또는 기능을 제공하지 않을 수 있습니다. 현재 각 지역에서 사용할 수 있는 제품 및 서비스에 대해 자세히 알려면, 각 나라의 IBM 대표부에 문의하십시오. 이 책에서 IBM의 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM의 제품, 프로그램 또는 서비스가 사용되어야 함을 의미하지는 않습니다. IBM의 지적재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램, 서비스가 대신 사용될 수 있습니다. 그러나 IBM 이외의 제품, 프로그램 또는 서비스의 조작을 평가하고 검증하는 것은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대한 특허를 보유하거나 현재 출원 중일 수 있습니다. 이 책을 부여한다고 해서 이 특허에 대한 사용권을 부여하는 것은 아닙니다. 특허 사용권에 대한 문의는 한국 IBM 지적재산권부(02-781-6028)로 하시기 바랍니다.

2 바이트 (DBCS) 정보에 대한 사용권을 조회하려면 각 국가의 IBM 지적 소유권 부서에 문의하십시오.

이 정보에는 기술적 부정확성이나 입력 오류가 있을 수 있습니다. 다음에 나오는 정보는 주기적으로 갱신되며 이러한 갱신 사항은 새로운 갱신판에 통합됩니다. IBM에서는 항상 특별한 통보없이 이 책자에서 설명되는 제품 및 프로그램을 갱신하거나 변경할 수 있습니다.

(1) 개별적으로 작성된 프로그램과 다른 프로그램(이 프로그램을 포함하여)과의 정보 교환 및 (2) 교환 가능한 정보와 공동 사용이 가능하도록 하기 위한 정보를 원하는 이 프로그램의 사용권자는 소프트웨어 사업본부 (02-781-7777)로 문의하십시오.

이러한 정보는 간혹 사용권료 지불을 포함하여 계약 조건에 따라 사용이 가능합니다.

이 정보에서 다루는 사용권 프로그램과 이 정보에 대해 사용할 수 있는 모든 사용권 정보는 IBM 고객 협의의 규정이나 쌍방 간의 동등한 협의에 따라 IBM에서 제공합니다.

IBM 이외의 제품에 대한 정보는 제품, 발표문 또는 기타 공개적으로 사용할 수 있는 소스의 공급자로부터 구할 수 있습니다. IBM은 이러한 제품을 테스트하지 않으며 IBM 이외의 제품과 관련된 성능 정확도, 호환성 또는 다른 주장을 확약할 수 없습니다. IBM 이외의 제품 기능에 대한 의문은 이러한 제품 제공업체로 보내야 합니다.

저작권 사용권:

본 정보에는 소스 언어로 만들어진 샘플 응용 프로그램이 포함되어 있으며 다양한 작동 플랫폼에서 프로그램을 만드는 기술을 보여줍니다. 샘플 프로

그램이 만들어진 작동 플랫폼용 응용 프로그램 인터페이스를 따르는 응용 프로그램을 개발, 사용, 판매 또는 배포할 목적인 경우, 귀하는 IBM에 비용을 지불하지 않고 본 샘플 프로그램을 어떠한 형태로든 복사, 수정 또는 배포할 수 있습니다. 본 샘플 프로그램을 모든 조건하에서 철저히 테스트하지는 않았습니다. 따라서 IBM은 본 프로그램에 대해 어떠한 보증도 하지 않으며 본 프로그램의 기능, 신뢰성 또는 서비스에 대해 묵시적으로 보증하지 않습니다. IBM의 응용 프로그램 인터페이스를 따르는 응용 프로그램을 개발, 사용, 판매 또는 배포할 목적인 경우, 귀하는 IBM에 비용을 지불하지 않고 본 샘플 프로그램을 어떠한 형태로든 복사, 수정 또는 배포할 수 있습니다.

등록상표

다음 용어는 미국이나 다른 나라 또는 모두에서 사용되는 IBM사의 등록상표입니다.

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

다음 용어는 다른 회사의 등록상표입니다.

Java 및 HotJava는 Sun Microsystems, Inc의 상표입니다.

Microsoft, Windows, Windows NT 및 Windows 95 로고는 Microsoft Corporation의 등록상표입니다.

UNIX는 미국과 다른 나라에서 사용되는 X/Open Company Limited에 독점권이 있는 등록상표입니다.

그 밖에 회사, 제품 및 서비스 이름은 다른 회사의 상표이거나 서비스 표시입니다.

용어

가

경로(path). 파일을 찾는데 사용되는 탐색 경로.

공통 게이트웨이 인터페이스(Common Gateway Interface). Web 서버가 응용 프로그램에 제어를 전달하고 다시 자료를 수신할 때 사용하는 동기화 방법.

나

널(NULL). 정보가 없음을 나타내는 특수한 값.

다

데이터베이스 관리 시스템(database management system (DBMS)). 데이터베이스의 작성, 구성 및 수정과 그 안에 저장된 자료에 대한 액세스를 제어하는 소프트웨어 시스템.

데이터베이스(database). 테이블의 집합, 테이블 공간 및 색인 공간의 집합.

마

미들웨어. 응용 프로그램과 네트워크 간을 중재하는 소프트웨어. 이것은 이질적인 컴퓨팅 운용 시스템에서 별도의 응용 프로그램 간의 상호작용을 관리합니다. 컴퓨팅에 대한 무료 온라인 사전

바

방화벽(firewall). 권한없는 외부 액세스로부터 내부 네트워크를 보호하는 소프트웨어를 가지는 컴퓨터.

아

애플릿(applet). HTML 페이지에 포함된 자바 프로그램. 애플릿은 Netscape와 같은 자바 사용가능 브라우저와 함께 작동하며, HTML 페이지가 로드될 때 로드됩니다.

언어 환경(language environment). Net.Data 매크로에서 DB2와 같은 외부 자료 소스나 Perl과 같은 프로그래밍 언어로의 액세스를 제공하는 모듈. Net.Data에는 REXX, Perl 및 Oracle과 같은 일부 언어 환경이 제공됩니다. 자체의 언어 환경을 작성할 수도 있습니다.

연결 관리 프로그램. Live 연결을 지원하는 데 필요한 Net.Data의 실행 파일, dtwcm.

웹 서버(Web server). 인터넷 연결과 같은 http 서버 소프트웨어를 수행하는 컴퓨터.

응용 프로그램 인터페이스(application programming interface (API)). 기능 인터페이스는 운용 시스템 또는 사용권 프로그램의 특정 자료나 기능을 사용하기 위해 고급 언어로 쓰여진 응용 프로그램을 허용하는 운용 시스템 또는 별도로 주문가능한 사용권 프로그램에 의해 제공됩니다. Net.Data는 CGI 프로세스에 대해 향상된 성능을 제공하기 위해 웹 서버 API인 ICAP, GWAPI, ISAPI 및 NSAPI를 지원합니다.

인터넷 연결 보안 서버(Internet Connection Secure Server). IBM의 보안 웹 서버.

인터넷 연결 서버(Internet Connection Server). IBM의 비보안 웹 서버.

인터넷(Internet). 국제 공용 TCP/IP 컴퓨터 네트워크.

인터넷라넷. 회사 방어벽 내부의 TCP/IP 네트워크.

일반 파일 인터페이스(flat file interface). 일반 텍스트 파일에 자료를 읽고 쓸 수 있게 하는 Net.Data 내장 함수 세트.

자

자료 유형(data type). 컬럼 및 리터럴의 속성.

자바(Java). 인터넷 응용 프로그램에 특히 유용한 운용 시스템 독립형 오브젝트 지향 프로그래밍 언어.

카

캐시. 최근 액세스한 자료를 포함한 메모리 유형으로 나중에 같은 자료에 액세스할 때 그 속도를 높이기 위해 고안되었습니다. 캐시는 네트워크를 통해 액세스할 수 있는 자주 사용되는 자료의 로컬 사본을 보유하는데 사용되는 경우도 있습니다.

캐시 관리 프로그램. 한 머신의 캐시를 관리하는 프로그램. 이 프로그램은 복수의 캐시를 관리할 수 있습니다.

캐시 처리. 정보를 갱신할 때가 될 때까지, 빠른 검색을 위해 로컬로 요청으로부터의 자주 사용되는 결과를 웹 서버에 저장하는 프로세스.

쿠키. HTTP 서버에 의해 웹 브라우저로 송신된 다음 브라우저가 이 서버에 액세스할 때마다 브라우저에 의해 다시 송신된 정보의 패킷. 쿠키는 서버가 선택한 임의의

| 정보를 포함할 수 있으며 그렇지 않으면 일정한 HTTP
| 트랜잭션 간의 상태를 유지하는 데 사용됩니다. 컴퓨팅
| 에 대한 무료 온라인 사전

파

포트(port). TCP/IP와 고급 프로토콜 또는 응용 프로그램 간의 통신에 사용되는 16비트 수.

하

하이퍼텍스트 마크업 언어(hypertext markup language). 웹 문서를 작성하는 데 사용되는 태그 언어.

하이퍼텍스트 전송 프로토콜. 웹 서버와 브라우저간에 사용되는 통신 프로토콜.

A

API. 응용 프로그램 인터페이스.

B

BLOB. 2진 대형 오브젝트(Binary large object).

C

CGI. 공통 게이트웨이 인터페이스.

cliette. 웹 서버로부터의 요청을 처리하는 장시간 수행 프로세스. 연결 관리 프로그램은 이러한 요구를 처리하기 위해 cliette 프로세스를 계획합니다.

CLOB. 문자 대형 오브젝트(Character large object).

D

DBMS. 데이터베이스 관리 시스템.

H

HTML. 하이퍼텍스트 마크업 언어

HTTP. 하이퍼텍스트 전송 프로토콜.

I

ICAPI. 인터넷 연결 API.

ICS. 인터넷 연결 서버.

ICSS. 인터넷 연결 보안 서버.

ISAPI. Microsoft의 인터넷 서버 API.

L

Live 연결. 연결 관리 프로그램과 웹 서버 API에서 작동하는 Net.Data 구성. Live 연결을 사용하면 데이터베이스 연결을 재사용할 수 있습니다.

LOB. 대형 오브젝트.

N

NSAPI. Netscape API.

P

Perl. 해석된 프로그래밍 언어.

T

TCP/IP. TCP/IP(Transmission Control Protocol / Internet Protocol).

TCP/IP(Transmission Control Protocol / Internet Protocol). 국지 및 광역 네트워크 둘다에 대해 피어 투 피어(Peer-to-peer) 연결성 기능을 지원하는 통신 프로토콜 세트.

U

URL. URL(Uniform resource locator).

URL(uniform resource locator). HTTP 서버 및 경우에 따라 디렉토리 및 파일명을 지정하는 주소. 예: <http://www.software.ibm.com/data/net.data/index.html>.

색인

[가]

값 그룹 제공 71
계획, DB2 서브시스템에 연결 106
구분된 값 문자열 70
구분된 문자열 일람 70
국지 DB2 서브시스템, ID 107
기타 변수
 설명 72
 DTW_CURRENT_FILENAME 73
 DTW_CURRENT_LAST_MODIFIED 74
 DTW_DEFAULT_MESSAGE 75
 DTW_MACRO_LAST_MODIFIED 78
 DTW_MP_PATH 79
 DTW_MP_VERSION 80
 DTW_PRINT_HEADER 81
 DTW_REMOVE_WS 82
 RETURN_CODE 83
꼬리말 38

[나]

내장 함수 123
내포된 변수
 설명 84
 NLIST 86
 NUM_COLUMNS 87
 NUM_ROWS 88
 Nn 85
 ROW_NUM 89
 TOTAL_ROWS 90
 VLIST 92
 Vn 93
 V_columnName 91

[다]

다음 버튼, RPT_MAX_ROWS 99
단어 함수
 DTW_DELWORD 176
 DTW_SUBWORD 177
 DTW_WORD 179
 DTW_WORDINDEX 180
 DTW_WORDLENGTH 181
 DTW_WORDPOS 182
 DTW_WORDS 184
 MBCS 지원 175
대문자, 지정 105
대소문자, SQL명령에 대한 대소문자 지정 105
대체 텍스트, 웹 브라우저 108

데이터베이스 액세스 제한 115, 117
데이터베이스 일관성, 트랜잭션 영역 120
데이터베이스에 연결, DATABASE 변수 103

[라]

루핑 61

[마]

매개변수 제공, 시스템 언어 환경 20
매개변수, 제공 20
매크로 파일
 공통 구문 요소 4
 샘플 2
 선언 부분 2
 언어 구성 1
 전역 구문 1
 처리 중단 134
 행 길이 한계 3
 형식 2
 HTML 부분 2
머리말 38
메세지, 생략시 텍스트 75
문자열
 값, 구분 70
 설명 5
 숫자 비교 29, 61
 조건부 처리 29, 61
문자열 함수
 MBCS 지원 158
문자열의 숫자 비교 29, 61

[바]

변수
 기타 72
 내포 84
 보고서 94
 숨겨진 69
 실행 가능 68
 언어 환경 102
 조건 66
 테이블 71
 환경 68
 list 70
 Net.Data, 개요 65
변수명 숨기기 69
보고서
 형식 지정 53

보고서 (계속)

Net.Data 생략시 값 무시 96

보고서 변수

설명 94

ALIGN 95

DTW_DEFAULT_REPORT 96

DTW_HTML_TABLE 97

RPT_MAX_ROWS 98

START_ROW_NUM 99

보안

로그인 ID 115

암호 117

[사]

상한 59

서브시스템 ID, DB2 서브시스템에 연결 107

선언 부분, 매크로 파일 2

성능, DTW_EXIT 134

소문자, 지정 105

수학 함수

DTW_ADD 146

DTW_CONCAT 160

DTW_DELSTR 161

DTW_DIVIDE 147

DTW_DIVREM 148

DTW_FORMAT 150

DTW_INSERT 162

DTW_INTDIV 153

DTW_LASTPOS 164

DTW_LENGTH 165

DTW_LOWERCASE 166

DTW_MULTIPLY 154

DTW_POS 167

DTW_POWER 155

DTW_REVERSE 168

DTW_STRIP 169

DTW_SUBSTR 170

DTW_SUBTRACT 156

DTW_TRANSLATE 172

DTW_UPPERCASE 174

숨겨진 변수

단계 69

설명 69

예제, HTML 형식 69

시스템 언어 환경, 매개변수 제공 20

실행 변수

매개변수 포함 69

변수 참조 69

설명 68

예제 68

[아]

언어 구성

공통 구문 요소 4

매크로 파일

구문 1

설명 5

문자열 5

함수 호출 23

COMMENT 블록 7

DB2 WWW 연결 231

DEFINE 블록 또는 명령문 9

ENVVAR 명령문 13

EXEC 블록 또는 명령문 14

FUNCTION 블록 16

HTML 블록 26

IF 블록 29

INCLUDE 명령문 38

INCLUDE_URL 명령문 40

LIST 명령문 42

MACRO_FUNCTION 블록 44

MESSAGE 블록 48

REPORT 블록 53

ROW 블록 56

TABLE 명령문 59

variable name 4

variable reference 4

WHILE 블록 61

언어 환경 변수

설명 102

DATABASE 103

DB2PLAN 106

DB2SSID 107

DB_CASE 105

DTW_APPLET_ALTTEXT 108

DTW_EDIT_CODES 109

DTW_MBMODE 110

DTW_SAVE_TABLE_IN 111

DTW_SET_TOTAL_ROWS 112

LOCATION 114

LOGIN 115

NULL_REPORT_FIELD 116

PASSWORD 117

SHOWSQL 118

SQL_STATE 119

TRANSACTION_SCOPE 120

오류 처리 48

용어집 242

운용 시스템 참조 235

원격 DB2 서브시스템, 위치 114

웹 레지스트리 함수 221

DTWR_ADDENTRY 222

웹 레지스트리 함수 (계속)

DTWR_CLEARREG 223
DTWR_CREATEREG 224
DTWR_DELENTY 225
DTWR_DELREG 226
DTWR_LISTREG 227
DTWR_LISTSUB 228
DTWR_RTVENTRY 229
DTWR_UPDATEENTRY 230

위치, DB2 서브시스템에 연결 114

이전 버튼, RPT_MAX_ROWS 99

일반 함수 125

DTW_ADDQUOTE 126
DTW_CACHE_PAGE 128
DTW_DATE 132
DTW_EXIT 134
DTW_GETENV 135
DTW_GETINIDATA 136
DTW_HTMLLENCODE 137
DTW_QHTMLLENCODE 139
DTW_SETENV 140
DTW_TIME 141
DTW_URLESCSEQ 143

[자]

조건 변수

변수 참조 포함 67

설명 66

예제 70

LIST문 사용 67

조건부 문자열 처리 29, 61

주의사항 241

지원 기능 테이블 235

[카]

쿠키

송신 81

DTW_PRINT_HEADER 81

[타]

테이블

HTML로 결과 표시 97

Net.Data, 행 수 지정 98

테이블 함수

DTW_TB_DLIST 186
DTW_TB_DUMPH 188
DTW_TB_DUMPV 189
DTW_TB_HTMLLENCODE 190
DTW_TB_INPUT_CHECKBOX 191

테이블 함수 (계속)

DTW_TB_INPUT_RADIO 192
DTW_TB_INPUT_TEXT 193
DTW_TB_LIST 195
DTW_TB_SELECT 197
DTW_TB_TABLE 198
DTW_TB_TEXTAREA 200

[파]

포함 파일 38

플랫폼 지원 참조 235

[하]

함수

값 그룹 제공 71

단어 175

명명 규약 123

문자열 158

설명 123

수학 145

웹 레지스트리 221

일반 125

테이블 185

FFI(플랫 파일 인터페이스) 201

함수 호출

구문 23

설명 23

출력 형식 지정 53

테이블 행 처리 56

함수에 대한 MBCS 지원

단어 함수 175

문자열 함수 158

행 길이 한계, 매크로 파일 3

호출

외부 프로그램 14

함수 23

화면 이동, 다음 및 이전 버튼 사용 99

환경 변수

설명 68

예제 68

ENVVAR 명령문 13

A

ALIGN 95

APPLET 태그, 대체 텍스트 108

C

COMMENT 블록

구문 7

COMMENT 블록 (계속)

설명 7

D

DATABASE 103

date 변수 72

DB2 WWW 연결, 언어 구성 231

DB2 서브시스템에 연결

서브시스템 ID 107

위치 114

DB2 계획 106

DB2PLAN 106

DB2SSID 107

DB_CASE 105

DEFINE 명령문

구문 9

설명 9

DEFINE 블록

구문 9

설명 9

DTWF_APPEND 202

DTWF_CLOSE 205

DTWF_DELETE 206

DTWF_INSERT 208

DTWF_OPEN 210

DTWF_READ 212

DTWF_REMOVE 214

DTWF_SEARCH 215

DTWF_UPDATE 217

DTWF_WRITE 219

DTWR_ADDENTRY 221

DTWR_CLEARREG 223

DTWR_CREATEREG 224

DTWR_DELENTY 225

DTWR_DELREG 226

DTWR_LISTREG 227

DTWR_LISTSUB 228

DTWR_TRVENTRY 229

DTWR_UPDATEENTRY 230

DTW_ADD 145

DTW_ADDQUOTE 126

DTW_APPLET_ALTTEXT 108

DTW_ASSIGN 85, 158

DTW_CACHE_PAGE 128

DTW_CONCAT 160

DTW_CURRENT_FILENAME 73

DTW_CURRENT_LAST_MODIFIED 74

DTW_DATE 132

DTW_DEFAULT_MESSAGE 75

DTW_DEFAULT_REPORT 96

DTW_DELSTR 161

DTW_DELWORD 175

DTW_DIVIDE 147

DTW_DIVREM 148

DTW_EDIT_CODES 109

DTW_EXIT 134

DTW_FORMAT 150

DTW_GETENV 135

DTW_GETINIDATA 136

DTW_HTML_ENCODE 137

DTW_HTML_TABLE 97

DTW_INSERT 162

DTW_INTDIV 153

DTW_LASTPOS 164

DTW_LENGTH 165

DTW_LOG_LEVEL 76

DTW_LOWERCASE 166

DTW_MACRO_FILENAME 77

DTW_MACRO_LAST_MODIFIED 78

DTW_MBMODE 110

DTW_MP_PATH 79

DTW_MP_VERSION 80

DTW_MULTIPLY 154

DTW_POS 167

DTW_POWER 155

DTW_PRINT_HEADER 81

DTW_QHTMLENCODE 139

DTW_REMOVE_WS 82

DTW_REVERSE 168

DTW_SAVE_TABLE_IN 111

DTW_SETENV 140

DTW_SET_TOTAL_ROWS 112

DTW_STRIP 169

DTW_SUBSTR 170

DTW_SUBTRACT 156

DTW_SUBWORD 177

DTW_TB_DLIST 185

DTW_TB_DUMPH 188

DTW_TB_DUMPV 189

DTW_TB_HTML_ENCODE 190

DTW_TB_INPUT_CHECKBOX 191

DTW_TB_INPUT_RADIO 192

DTW_TB_INPUT_TEXT 193

DTW_TB_LIST 195

DTW_TB_SELECT 197

DTW_TB_TABLE 198

DTW_TB_TEXTAREA 200

DTW_TIME 141

DTW_TRANSLATE 172

DTW_UPPERCASE 174

DTW_URLESCSEQ 143

DTW_WORD 179

DTW_WORDINDEX 180

DTW_WORDLENGTH 181
DTW_WORDPOS 182
DTW_WORDS 184

E

ENVVAR 명령문 68
구문 13
설명 13
EXEC 명령문 68
구문 14
설명 14
EXEC 블록
구문 14
설명 14
EXEC_PATH 14
EXEC_SQL 231

F

FFI 함수
DTWF_APPEND 203
DTWF_CLOSE 205
DTWF_DELETE 206
DTWF_INSERT 208
DTWF_OPEN 210
DTWF_READ 212
DTWF_REMOVE 214
DTWF_SEARCH 215
DTWF_UPDATE 217
DTWF_WRITE 219
file location 변수 72
FUNCTION 블록
구문 16
설명 16

H

HTML
변수명 숨기기 69
테이블 결과 표시 97
형식, 사용자 ID 입력 115
형식, 암호 입력 117
HTML 부분, 매크로 파일 2
HTML 블록
구문 26
설명 26
HTML_INPUT block 231
HTML_REPORT 블록 231

I

IF 블록
구문 29

IF 블록 (계속)
설명 29
IN 키워드 17, 45, 123
INCLUDE 명령문
구문 38
설명 38
INCLUDE_PATH 38
INCLUDE_URL 명령문
구문 40
설명 40
INOUT 키워드 17, 45, 123

L

LIST 명령문
구문 42
설명 42
list 변수
값 분리자 71
설명 70
예제 70
LOCATION 114
LOGIN 115

M

MACRO_FUNCTION 블록
구문 44
설명 44
MESSAGE 블록
구문 48
설명 48

N

Net.Data 테이블
상한 59
정의 59
NLIST 86
NULL_REPORT_FIELD 116
NUM_COLUMNS 87
NUM_ROWS 88
Nn 85

O

OUT 키워드 17, 45, 123

P

PASSWORD 117

R

REPORT 블록
구문 53

REPORT 블록 (계속)
 설명 53
 ALIGN 95
 DTW_DEFAULT_REPORT 96
 DTW_HTML_TABLE 97
 NLIST 86
 NUM_COLUMNS 87
 NUM_ROWS 88
 Nn 85
 RPT_MAX_ROWS 98
 START_ROW_NUM 99
 table 변수 71
 TOTAL_ROWS 90
 RETURNS 키워드 17
 RETURN_CODE 83
 ROW 블록
 구문 56
 설명 56
 NLIST 86
 NUM_COLUMNS 87
 NUM_ROWS 88
 Nn 85
 ROW_NUM 89
 TOTAL_ROWS 90
 Vn 92, 93
 V_columnName 91
 ROW_NUM 89
 RPT_MAX_ROWS 98

S

SHOWSQL 118
 SQL
 명령, 대소문자 지정 105
 숨기기 또는 표시 118

SQL 블록 231
 SQL 상태, 표시 119
 SQL_CODE 233
 SQL_MESSAGE 블록 232
 SQL_REPORT 블록 232
 SQL_STATE 119
 START_ROW_NUM 99

T

TABLE 명령문 71
 구문 59
 설명 59
 table 변수
 설명 71
 예제 71
 SQL 언어 환경에 대해 table 변수 지정 111
 TOTAL_ROWS 90
 TRANSACTION_SCOPE 120

V

variable name 4
 variable reference 4
 VLIST 92
 Vn 93
 V_columnName 91

W

WHILE 블록 61
 구문 61
 설명 61