

Net.Data



참조서

버전 2 릴리스 2

Net.Data



참조서

버전 2 릴리스 2

주

이 책자와 이 책이 지원하는 제품을 사용하기 전에 291 페이지의 『부록D. 주의사항』의 내용을 반드시 읽으십시오.

— 목차

머리말	ix
Net.Data에 대하여	ix
이 책에 대하여	x
이 책의 독자	x
이 책에 실린 예제에 대한 정보	x
구문 도표를 읽는 방법	x
 제1장 Net.Data 매크로 언어 구문	1
Net.Data 매크로 구문	1
명령 구문 요소	5
변수명	5
변수 참조	5
문자열	6
매크로 언어 구문	7
주석 블록	9
DEFINE 블록 또는 명령문	11
ENVVAR 명령문	16
EXEC 블록 또는 명령문	17
FUNCTION 블록	20
함수 호출 (@)	28
HTML 블록	32
IF 블록	35
INCLUDE 명령문	42
INCLUDE_URL 명령문	45
LIST 명령문	48
MACRO_FUNCTION 블록	50
MESSAGE 블록	54
REPORT 블록	59
ROW 블록	62
TABLE 명령문	65
WHILE 블록	67
 제2장 변수	71
사용자 정의 변수	72
조건 변수	72
환경 변수	74
실행가능 변수	74
숨겨진 변수	75
목록 변수	76
테이블 변수	77
Net.Data 테이블 처리 변수	79
Nn	80
NLIST	81
NUM_COLUMNS	82

NUM_ROWS	83
ROW_NUM	84
TOTAL_ROWS	85
V_columnName	86
VLIST	87
Vn	88
Net.Data 보고서 변수	89
ALIGN	90
DTW_DEFAULT_REPORT	91
DTW_HTML_TABLE	92
RPT_MAX_ROWS	93
START_ROW_NUM	95
Net.Data 언어 환경 변수	98
DATABASE	99
DB_CASE	101
DB2PLAN	102
DB2SSID	103
DTW_APPLET_ALTTEXT	104
DTW_EDIT_CODES	105
DTW_MBMODE	106
DTW_SAVE_TABLE_IN	107
DTW_SET_TOTAL_ROWS	108
LOCATION	110
LOGIN	111
NULL_RPT_FIELD	112
PASSWORD	113
SHOWSQL	114
SQL_STATE	115
TRANSACTION_SCOPE	116
Net.Data 기타 변수	118
DTW_CURRENT_FILENAME	119
DTW_CURRENT_LAST_MODIFIED	120
DTW_DEFAULT_MESSAGE	121
DTW_LOG_LEVEL	122
DTW_MACRO_FILENAME	123
DTW_MACRO_LAST_MODIFIED	124
DTW_MP_PATH	125
DTW_MP_VERSION	126
DTW_PRINT_HEADER	127
DTW_REMOVE_WS	128
RETURN_CODE	129
제3장 Net.Data 내장 함수	131
함수명	131
입력 및 출력 매개변수	131
함수 결과 포매팅	132
함수 매개변수 규칙	132

일반 함수	133
DTW_ADDQUOTE	134
DTW_CACHE_PAGE	136
DTW_DATE.	140
DTW_EXIT	142
DTW_GETCOOKIE	143
DTW_GETENV	145
DTW_GETINIDATA.	146
DTW_HTMLENCODER	147
DTW_QHTMLENCODER	149
DTW_SENDMAIL	150
DTW_SETCOOKIE	154
DTW_SETENV.	157
DTW_TIME.	158
DTW_URLESCSEQ	160
수학 함수	162
DTW_ADD	163
DTW_DIVIDE	164
DTW_DIVREM	165
DTW_FORMAT	167
DTW_INTDIV	170
DTW_MULTIPLY.	171
DTW_POWER	172
DTW_SUBTRACT	173
문자열 함수	175
DTW_ASSIGN.	176
DTW_CONCAT	177
DTW_DELSTR.	178
DTW_INSERT	179
DTW_LASTPOS	181
DTW_LENGTH	182
DTW_LOWERCASE.	183
DTW_POS	184
DTW_REVERSE	185
DTW_STRIP	186
DTW_SUBSTR.	187
DTW_TRANSLATE	189
DTW_UPPERCASE	191
단어 함수	192
DTW_DELWORD.	193
DTW_SUBWORD.	194
DTW_WORD	196
DTW_WORDINDEX.	197
DTW_WORDLENGTH	198
DTW_WORDPOS.	199
DTW_WORDS	201

테이블 함수	202
DTW_TB_APPENDROW	203
DTW_TB_COLS	204
DTW_TB_DELETEROW	205
DTW_TB_DELETECOL	206
DTW_TB_DLIST	207
DTW_TB_DUMPH	209
DTW_TB_DUMPV	210
DTW_TB_GETN	211
DTW_TB_GETV	212
DTW_TB_HTMLLENCODE	213
DTW_TB_INPUT_CHECKBOX	214
DTW_TB_INPUT_RADIO	215
DTW_TB_INPUT_TEXT	216
DTW_TB_INSERTCOL	218
DTW_TB_INSERTROW	220
DTW_TB_LIST	221
DTW_TB_MAXROWS	223
DTW_TB_QUERYCOLNONJ	224
DTW_TB_ROWS	225
DTW_TB_SELECT	226
DTW_TB_SETCOLS	228
DTW_TB_SETN	229
DTW_TB_SETV	230
DTW_TB_TABLE	231
DTW_TB_TEXTAREA	233
플랫 파일 인터페이스 함수	234
플랫 파일 자료 소스에 액세스	234
플랫 파일 인터페이스 분리문자	237
파일 잠금	238
DTWF_APPEND	239
DTWF_CLOSE	241
DTWF_DELETE	242
DTWF_INSERT	244
DTWF_OPEN	246
DTWF_READ	248
DTWF_REMOVE	250
DTWF_SEARCH	251
DTWF_UPDATE	254
DTWF_WRITE	256
웹 레지스트리 함수	258
DTWR_ADDENTRY	259
DTWR_CLEARREG	260
DTWR_CLOSEREG	261
DTWR_CREATEREG	262
DTWR_DELENTY	263

DTWR_DELREG	264
DTWR_LISTREG	265
DTWR_LISTSUB	266
DTWR_OPENREG	267
DTWR_RTVENTRY	268
DTWR_UPDATEENTRY	269
영속적 매크로 함수.	270
DTW_ACCEPT.	271
DTW_COMMIT	273
DTW_ROLLBACK	274
DTW_RTVHANDLE.	275
DTW_STATIC	276
DTW_TERMINATE	277
부록A. Net.Data Technical Library	279
부록B. DB2 WWW 연결	281
EXEC_SQL	281
HTML_INPUT	281
HTML_REPORT	281
SQL	281
SQL_MESSAGE	282
SQL_REPORT	283
SQL_CODE	283
부록C. Net.Data 운영 체제.	285
부록D. 주의사항.	291
등록상표	292
용어.	293
색인.	297

머리말

동적 웹 페이지 작성을 위한 IBM의 개발 툴인 Net.Data를 선택해 주셔서 감사합니다. Net.Data로 다양한 자료 소스의 자료를 통합하고 이미 알고 있는 프로그래밍 언어를 사용하여 동적 내용을 갖는 웹 페이지를 신속하게 개발할 수 있습니다.

Net.Data에 대하여

IBM의 Net.Data 제품에서는, DB2, IMS 및 ODBC 허용 데이터베이스 등 관계형 및 비관계형 데이터베이스 관리 시스템(DBMS)의 자료와 Java, JavaScript, Perl, C, C++ 및 REXX와 같은 프로그래밍 언어로 쓰여진 응용프로그램을 사용하여 동적 웹 페이지를 작성할 수 있습니다. Net.Data 제품군은 Windows NT, AIX, OS/2, OS/390, OS/400, HP-UX, Sun Solaris, Santa Cruz Operating System(SCO) 및 Linux 운영 체제를 실행하는 기계에서 유사한 기능을 제공합니다.

Net.Data는 웹 서버 기계 상에서 미들웨어로 실행되는 매크로 프로세서입니다. 매크로라고 하는 Net.Data 응용프로그램을 작성할 수 있으며, Net.Data는 이 프로그램을 해석하여, 사용자 입력, 현재의 데이터베이스 상태, 기존의 업무 로직 및 매크로에 도입하려는 기타 인자에 기초한 사용자 조정 내용을 사용하여 동적 웹 페이지를 작성합니다.

URL(uniform resource locator) 형태의 요청은 Netscape Navigator 또는 Internet Explorer와 같은 브라우저에서, 실행을 위해 Net.Data로 요청을 보내는 웹 서버로 이동합니다. Net.Data는 매크로를 찾아 실행하고 사용자가 작성한 함수에 따라 조정되는 웹 페이지를 구축합니다. 이들 함수로 다음을 수행할 수 있습니다.

- C, C++, RPG, COBOL, JAVA, Perl 또는 REXX 프로그래밍 언어로 작성되었지만 이에 국한되지 않은 응용프로그램 내에 업무 로직을 캡슐화하십시오.
- DB2와 같은 데이터베이스에 액세스합니다
- 플랫폼 파일과 같은 다른 자료 소스에 액세스합니다

Net.Data는 이 웹 페이지를 웹 서버로 전달하며 이 웹 서버는 브라우저에 표시할 수 있도록 이 페이지를 네트워크에 보냅니다.

Net.Data는 HTTP(HyperText Transfer Protocol) 및 CGI(Common Gateway Interface)와 같은 인터페이스를 사용하도록 구성된 서버 환경에서 사용할 수 있습니다. HTTP는 브라우저와 웹 서버 간의 대화를 위한 업계 표준 인터페이스이며 CGI는 Net.Data와 같은 게이트웨이 응용프로그램의 웹 서버 호출을 위한 업계 표준 인터페이스입니다. 이러한 인터페이스들을 사용하면, Net.Data와 함께 사용할 원하는 브라우저나 웹 서버를 선택할 수 있습니다.

성능 개선을 위해 Net.Data는 다양한 웹 서버 API를 지원합니다. 그외에 Net.Data는 Java 서브릿으로 시작할 수 있습니다.

이 책에 대하여

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

이 책에서는 발표는 되었으나 아직 시판되지는 않은 제품이나 기능에 대해 언급할 수도 있습니다.

좀 더 상세한 정보, 샘플 Net.Data 매크로, 데모 및 이 책의 최신판이 다음의 World Wide Web 사이트에 제공됩니다.

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

이 책의 독자

Net.Data 응용프로그램의 설계 및 작성에 관여하는 사용자는 이 책을 통해 Net.Data가 제공하는 언어 구조, 변수 및 함수에 대해 이해하게 될 것입니다.

이 책에서 논의되는 개념에 대해 이해하려면 웹 서버, 단순한 SQL문, HTML(HTML 양식 사용 등) 및 *Net.Data Administration and Programming Guide*의 정보에 대해 잘 알아야 합니다.

이 책에 실린 예제에 대한 정보

이 책에 사용된 예제는 특정 개념을 주지시키기 위해 간결하게 작성되었습니다. 이 예제는 Net.Data 구조를 사용할 수 있는 모든 방법을 보여주기 위한 것은 아닙니다. 마찬가지로, 일부 예제는 자체적으로 실행할 수 없는 코드 조각입니다.

구문 도표를 읽는 방법

이 책에 사용된 구문 도표에는 다음과 같은 규칙이 적용됩니다.

- 구문 도표는 왼쪽에서 오른쪽으로, 위에서 아래로, 선의 경로를 따라 읽으십시오.

▶— 기호는 명령문의 시작을 나타냅니다.

→ 기호는 명령문 구문이 다음 행에 계속됨을 의미합니다.

▶— 기호는 명령문이 이전 행에서 이어졌음을 의미합니다.

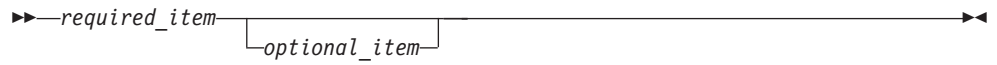
→◀ 기호는 명령문의 끝을 나타냅니다.

명령문의 끝이 아닌 구문 도표의 단위는 ▶—기호로 시작하여 → 기호로 끝납니다.

- 필수 항목은 수평선(주 경로)에 표시됩니다.

▶—required_item—▶

- 선택적 항목은 주 경로 아래에 나타납니다.

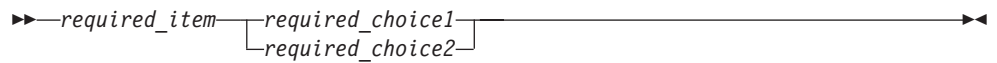


선택적 항목이 주 경로 위에 나타나면, 그 항목은 명령문의 실행에 아무 영향도 미치지 않으며 참조용으로만 사용됩니다.

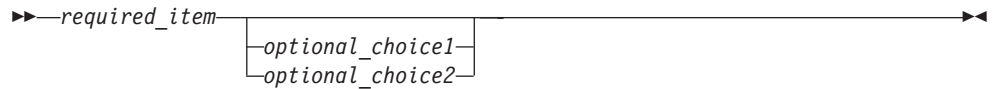


- 2개 이상의 항목을 선택할 수 있는 경우에는 항목이 수직으로 스택을 이루어 표시됩니다.

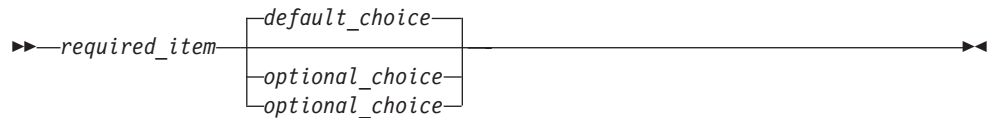
항목 중 하나를 반드시 선택해야 하는 경우, 스택의 한 항목이 주 경로에 나타납니다.



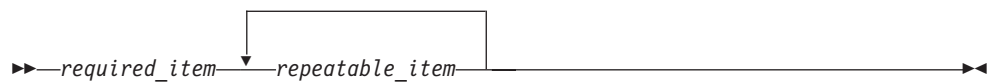
항목 중에 하나를 선택하는 것이 선택적이면, 스택 전체가 주 경로 아래에 표시됩니다.



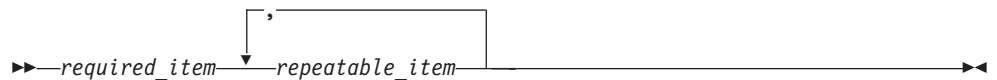
항목 중의 하나가 생략시 값이면, 이 항목은 주 경로의 위에 나타나고 나머지는 아래에 표시됩니다.



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



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



스택위의 반복 화살표는 스택내에서 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 구문

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

이 장에서는 다음에 대해 설명됩니다.

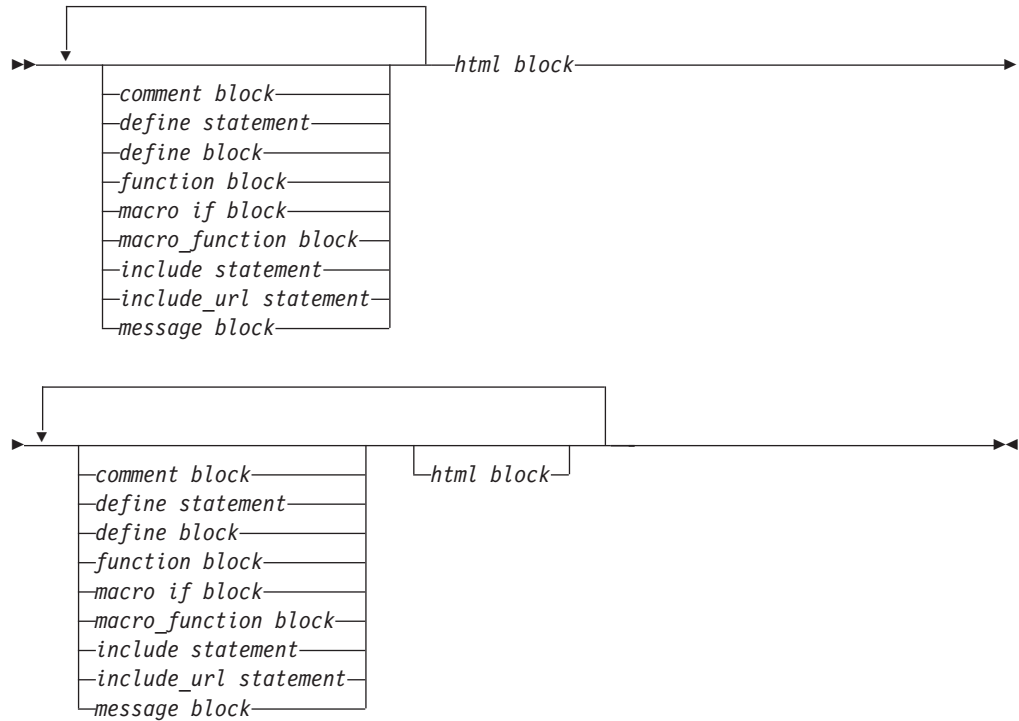
- 『Net.Data 매크로 구문』
- 5 페이지의 『명령 구문 요소』
- 7 페이지의 『매크로 언어 구문』

Net.Data 매크로 구문

Net.Data 매크로는 다음을 수행하는 일련의 Net.Data 매크로 언어 구조로 구성된 텍스트 파일입니다.

- 웹 페이지의 레이아웃을 지정합니다.
- 변수와 함수를 정의합니다.
- 매크로에 정의되었거나 처리를 위해 Net.Data에서 언어 환경에 전달한 함수를 호출합니다.
- 처리후의 출력을 HTML로 포맷하여 이를 웹 브라우저에 리턴합니다.

각 명령문은 하나 이상의 언어 구문으로 이루어지며, 이는 키워드, 특수 문자, 문자열, 이름 및 변수의 차례로 구성됩니다. 다음 도표는 문법적으로 유효한 Net.Data 매크로의 전체 구조를 설명한 것입니다. 전체 구조의 각 요소에 대한 자세한 구문을 알려면 7 페이지의 『매크로 언어 구문』을 참조하십시오.



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

- 선언 부분에는 매크로에 있는 변수와 함수의 정의가 들어 있습니다.
- 표시 부분에는 웹 페이지의 레이아웃을 지정하는 HTML 명령문이 포함된 HTML 블록이 들어 있습니다. 이 부분에 report 섹션이 들어 있습니다.

3 페이지의 그림1에서는 매크로의 선언 및 표시 부분을 나타냅니다.

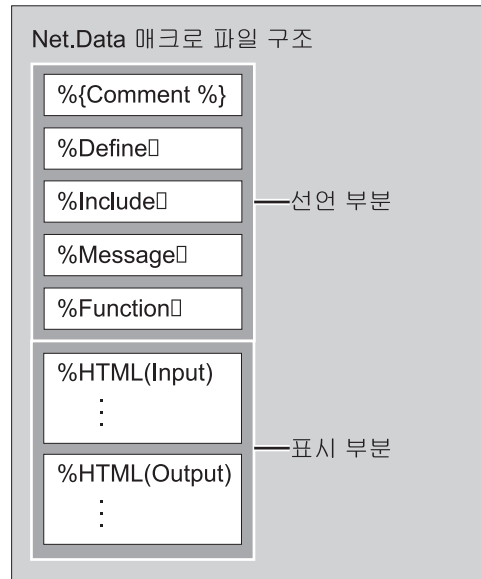


그림 1. 매크로 구조

선언 또는 표시 부분에 사용되는 변수와 함수는 변수 참조 또는 함수 호출에 사용하기 전에 정의해야 합니다.

4 페이지의 그림2에서는 매크로의 부분을 보여줍니다. 선언 부분에는 DEFINE 및 FUNCTION 정의 블록이 들어 있습니다. HTML 블록은 입력 및 출력 블록으로 작용합니다.

```

%{ ***** Define block *****}
%DEFINE {
    page_title="Net.Data macro Template"
%}

%{ ***** Function Definition block *****}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
    { %EXEC{ompsamp.cmd %}
%}

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

%{ ***** HTML Block: Input *****}
%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 Block: Output *****}
%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 템플릿 형식

Net.Data 매크로 언어는 형식이 정해져 있지 않은 언어로서, 매크로 작성시 유연성을 부여합니다. 특별히 명기되지 않은 한, 기타 공백 문자는 무시됩니다. 각각의 Net.Data 매크로 언어 구문은 이를 정의하는 데 사용되는 기타 여러 가지 요소와 함께 다음 절에서 설명됩니다. Net.Data 매크로 언어는 이전 버전과의 호환을 위해 DB2 WWW 연

결 언어 요소를 지원합니다. 이러한 언어 요소들이 281 페이지의 『부록B. DB2 WWW 연결』에 기술되어 있지는 않지만, Net.Data 언어 구문을 사용하도록 하십시오.

예제에 언어 구문, 변수, 함수 및 매크로내의 기타 요소를 사용할 수 있는 몇가지 방법이 나와 있습니다. Net.Data 웹 페이지에서 좀 더 광범위한 샘플 및 데모를 다운로드 할 수 있습니다.

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

명령 구문 요소

다음 구문 요소는 언어 구문 설명에 자주 사용됩니다.

- 『변수명』
- 『변수 참조』
- 6 페이지의 『문자열』

변수명

목적:

하나 이상의 이름을 식별합니다. 후속되는 이름들은 점(.)으로 연결됩니다. 이름은 영문자나 밑줄로 시작하는 영숫자 문자열이며, 영문자, 숫자 또는 밑줄등을 임의로 조합하여 사용할 수 있습니다.

큰 따옴표안의 문자열(『』)에는 개행 문자를 제외한 모든 문자를 포함시킬 수 있습니다. 문자열이 중괄호 안에 있을 경우({ %}), 개행 문자를 포함한 모든 문자를 포함시킬 수 있습니다.

변수명은 문자나 밑줄(_)로 시작해야 하며, 모든 영숫자 문자 및 밑줄을 포함할 수 있습니다. N_columnName과 V_columnName를 제외한 모든 변수명은 대소문자가 구분됩니다(이러한 두 예외에 대한 자세한 내용은 79 페이지의 『Net.Data 테이블 처리 변수』를 참조하십시오).

구문:

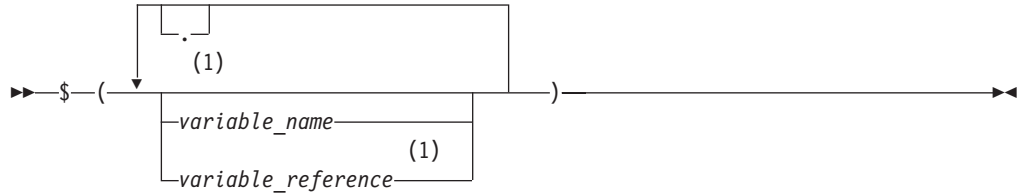


변수 참조

목적:

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR = 'abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 변수 참조는 수행시에 값이 평가됩니다. EXEC문이나 블록에 대해 변수가 정의된 경우, Net.Data는 변수 참조를 읽을 때 지정된 조치를 수행합니다.

구문:



주:

- 1 중첩된 변수 참조는 OS/2, Windows NT 및 UNIX에서 Net.Data에 의해 지원됩니다.

예 1: 변수 참조

변수 homeURL을 정의한 경우,

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

홈페이지를 \$(homeURL)로 나타내고 링크를 작성할 수 있습니다.

```
<a href=$(homeURL)"Home page">
```

예 2: 중첩된 변수 참조

중첩된 변수 참조를 사용하여 데이터베이스에서 리턴된 행에 필드 값을 동적으로 참조할 수 있습니다.

```
DTW_ASSIGN(INDEX, "2")
%ROW{
  $(V$(INDEX))
```

그러면, 중첩된 변수 참조는 행에서 두번째 필드인 V2의 필드 값을 얻어냅니다. 'V2'는 사전 정의된 Net.Data 변수이며 데이터베이스로부터 리턴된 행에서 두번째 필드를 의미합니다.

문자열

영문자, 숫자 및 구두점의 임의 조합. 문자열이 큰 따옴표안에 나타나면, 개행 문자를 사용할 수 없습니다. 언어 구문에 사용될 때의 제한사항에 대해서는 각 언어 구문의 문자열 매개변수 설명을 참조하십시오.

인용된 문자열에 큰 따옴표를 지정하려면, 두 쌍의 큰 따옴표를 사용하십시오. 비교 표현식에서 함수 인수나 용어로 사용되는 문자열에는 큰 따옴표를 포함시킬 수 있습니다. 예를 들어, 다음과 같이 문자열 값을 정의하는 경우

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

*result*의 값은 다음과 같습니다.

```
"Hello world!"
```

HTML 명령문은 문자열입니다.

함수 인수, 용어 및 변수 값으로 사용되는 문자열에는 변수 참조 및 함수 호출이 포함될 수 있습니다. 다음 예에서, 함수 호출 `myfunc2`은 변수 참조와 함수 호출을 포함하는 문자열 매개변수를 갖습니다.

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

`Net.Data`는 문자열을 함수 `myfunc2`에 전달하기 전에 변수 참조 `$(var1)` 및 함수 호출 `@myfunc()`를 문자 그대로 문자열의 일부로 해석하기 보다는 이들을 분석합니다.

매크로 언어 구문

이 절에서는 `Net.Data` 매크로에서 사용되는 언어 구조를 설명합니다.

각 언어 구문 설명에는 다음과 같은 정보가 들어 있습니다.

목적 `Net.Data` 매크로에 이 언어 구문을 사용하는 이유를 정의합니다.

구문 언어 구문의 논리 구조를 보여줍니다.

매개변수

구문 도표의 모든 요소를 정의하고 다른 언어 구문의 구문과 예제에 대한 상호 참조를 제공합니다.

컨텍스트

`Net.Data` 매크로 구조에서 언어 구문을 사용할 수 있는 위치를 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 구문은 매크로에서 사용됩니다. 구문과 예제에 대해서는 각 구문 설명을 참조하십시오.

- 9 페이지의 『주석 블록』
- 11 페이지의 『DEFINE 블록 또는 명령문』
- 16 페이지의 『ENVVAR 명령문』
- 17 페이지의 『EXEC 블록 또는 명령문』
- 20 페이지의 『FUNCTION 블록』
- 28 페이지의 『함수 호출 (@)』
- 32 페이지의 『HTML 블록』
- 35 페이지의 『IF 블록』
- 42 페이지의 『INCLUDE 명령문』
- 45 페이지의 『INCLUDE_URL 명령문』
- 48 페이지의 『LIST 명령문』
- 50 페이지의 『MACRO_FUNCTION 블록』
- 54 페이지의 『MESSAGE 블록』
- 59 페이지의 『REPORT 블록』
- 62 페이지의 『ROW 블록』
- 65 페이지의 『TABLE 명령문』
- 67 페이지의 『WHILE 블록』

주석 블록

목적

Net.Data 매크로의 함수를 문서화합니다. COMMENT 블록은 매크로의 아무 지점에 서나 사용할 수 있으므로 다른 구문 도표에서는 문서화되지 않습니다.

COMMENT 블록은 Net.Data 초기화 파일에서도 사용할 수 있습니다.

구문

▶▶—%{ —텍스트—%}—————▶▶

값

텍스트 한 행 이상의 임의의 문자열. Net.Data는 모든 주석의 내용을 무시합니다.

문맥

주석은 Net.Data 매크로나 Net.Data 초기화 파일에서 Net.Data 언어 구조들 사이의 임의의 지점에 배치할 수 있습니다.

제한

어떤 텍스트나 문자도 사용할 수 있습니다. 그러나 주석 블록은 중첩시킬 수 없습니다.

예

예 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.
%}
```

예 2: FUNCTION 블록의 주석

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

예 3: HTML 블록의 주석

```
%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>
%}

```

예 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 %}
%}

```

예 5: Net.Data 초기화 파일의 주석

```

%{ changes: removed RETURN_CODE parm and DTW_DEFAULT ENVIRONMENT statement %}
...
ENVIRONMENT (DTW_SQL) dtwsq1 (IN LOCATION, DB2SSID, DB2PLAN, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_ODBC) odbcd11 (IN LOCATION, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_PERL) perl111 ()
ENVIRONMENT (DTW_REXX) rexxd11 ()
ENVIRONMENT (DTW_FILE) filed11 ()
ENVIRONMENT (DTW_APPLET) appld11 ()
ENVIRONMENT (DTW_SYSTEM) sysd11 ()

```


DEFINE 블록 또는 명령문

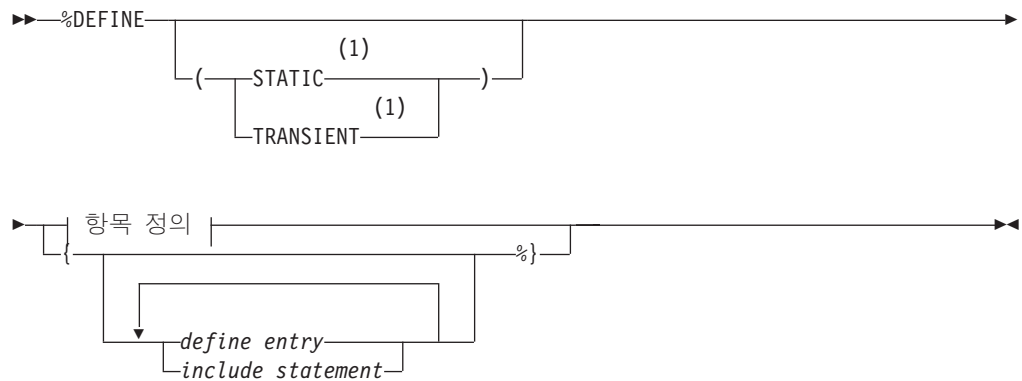
목적

DEFINE 섹션은 매크로의 선언 부분에 변수명을 정의하며 명령문이 되거나 블록이 될 수 있습니다.

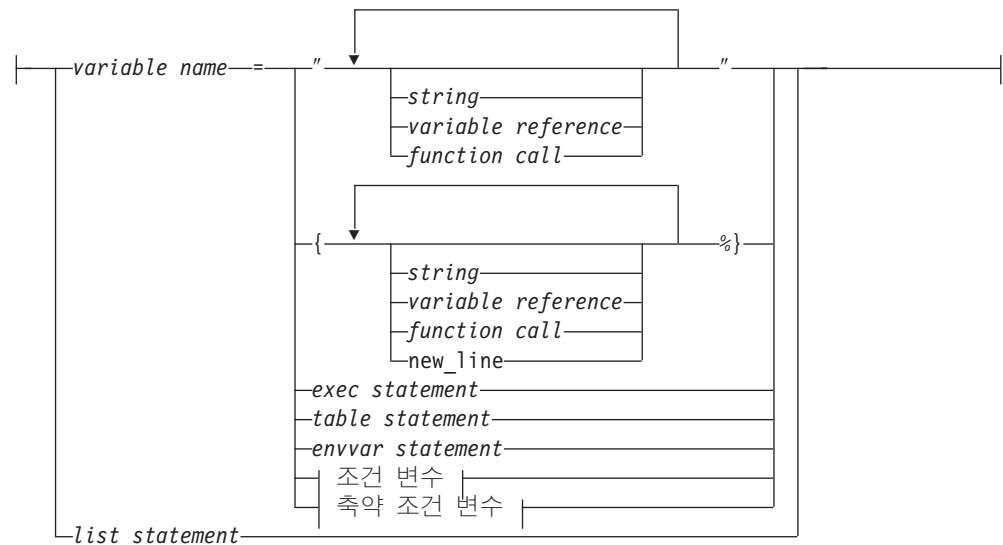
- 한번에 하나의 변수를 정의하려면 명령문을 사용하십시오.
- 여러 변수를 정의하려면 블록을 사용하십시오.

변수 정의는 큰 따옴표(")를 사용하여 한 행이 되게 할 수도 있고, 중괄호와 퍼센트 기호({ %})를 사용하여 여러 행이 되게 할 수도 있습니다. 변수가 정의되면 매크로의 임의의 지점에서 이를 참조할 수 있습니다.

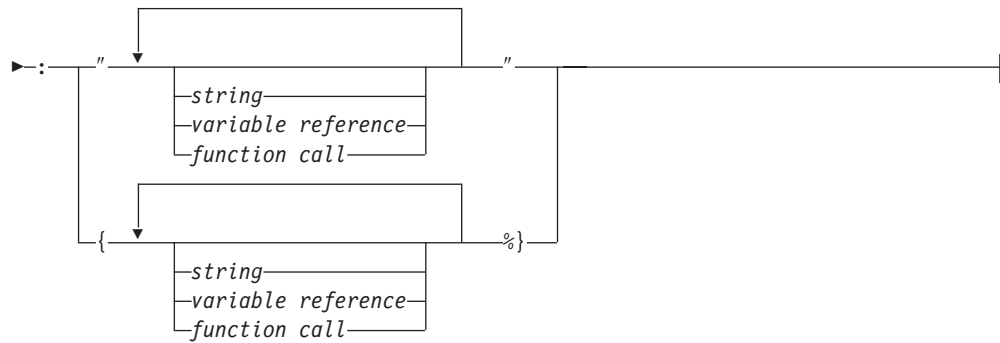
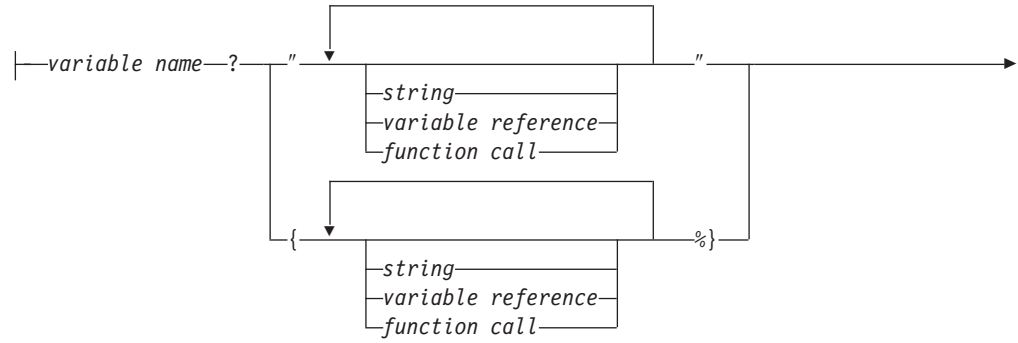
구문



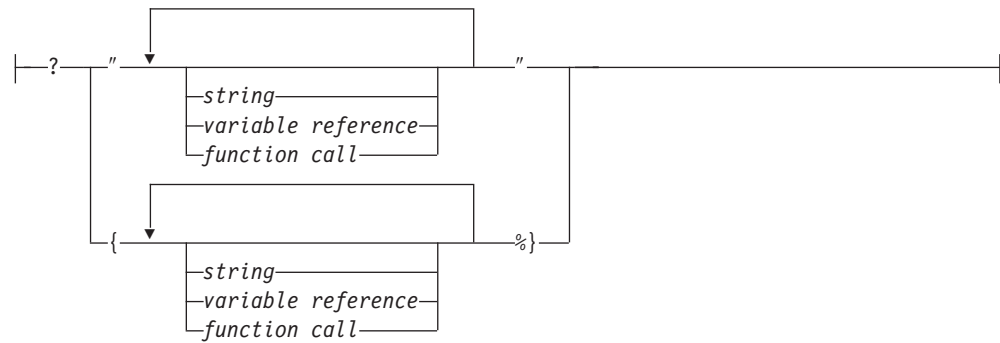
항목 정의:



조건 변수:



축약 조건 변수:



주:

- 1 STATIC 및 TRANSIENT은 영속적 매크로에 대한 키워드로, 현재 OS/400 운영 체제에서만 사용할 수 있습니다.

값

%DEFINE

변수를 정의하는 키워드.

STATIC

변수가 영속 트랜잭션내의 매크로 호출들 사이에서 이의 값을 유지하도록 지정하는 키워드. 영속 매크로의 경우 이것이 생략시 값입니다.

TRANSIENT

변수가 매크로 호출들 사이에서 이의 값을 유지하지 않도록 지정하는 키워드. 비영속 매크로의 경우 이것이 생략시 값입니다.

define entry:

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문에 대해서는 5 페이지의 『변수명』을 참조하십시오.

string

영문자, 숫자 및 구두점의 임의 조합. 문자열이 큰 따옴표안에 나타나면, 개행 문자를 사용할 수 없습니다.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

exec statement

EXEC 명령문. 변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름. 구문 및 예는 17 페이지의 『EXEC 블록 또는 명령문』을 참조하십시오.

table statement

TABLE 명령문. 동일한 레코드 배열, 행 및 각 행의 필드를 설명하는 컬럼 이름의 배열이 포함된 관련 자료의 집합을 정의합니다. 구문 및 예는 65 페이지의 『TABLE 명령문』을 참조하십시오.

envvar statement

ENVVAR 명령문. 환경 변수를 참조합니다. 구문 및 예는 16 페이지의 『ENVVAR 명령문』을 참조하십시오.

conditional variable

다른 변수나 문자열의 값에 따라 변수의 값을 설정합니다.

abbreviated conditional variable

다른 변수나 문자열의 값에 따라 변수의 값을 설정합니다. 조건 변수의 축약형.

list statement

LIST 명령문. 분리된 값 일람표를 구성하는 데 사용되는 값을 정의합니다. 구문 및 예는 48 페이지의 『LIST 명령문』을 참조하십시오.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

문맥

DEFINE 블록이나 명령문은 IF 블록의 내부 또는 Net.Data 매크로의 선언 부분에서 다른 모든 블록의 외부에 있어야 합니다.

제한

- 다음과 같은 요소들이 포함될 수 있습니다.
 - 주석 블록
 - 조건 변수
 - LIST 명령문
 - TABLE 명령문
 - 변수 참조
 - INCLUDE 명령문
 - EXEC 명령문
 - 함수 호출
 - ENVVAR 명령문
- 변수 정의에 변수를 사용할 수 없습니다. 예를 들어, 다음과 같은 변수 정의는 허용되지 않습니다.

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

예

예 1: 간단한 변수 정의

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

수행시 변수 참조 `$(var2)`는 `orders.html`로 평가됩니다.

예 2: 문자열 안의 인용 부호

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

표시될 때, 변수 `hi`는 `say "hello"`의 값을 갖습니다. 변수 `empty`는 널(NULL)이 됩니다.

예 3: 복수 변수의 정의

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

예 4: 변수의 복수 행 정의

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

예 5: 이 조건 변수 예에서는 결과 값에 널(NULL)이 포함되지 않은 경우 변수 var이 인용 부호(『』)안에 결과 값을 갖도록 하는 방법을 보여줍니다.

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

ENVVAR 명령문

목적

DEFINE 블록에서 변수를 환경 변수로 정의합니다. ENVVAR 변수가 참조되면, Net.Data는 환경 변수의 현재 값을 동일한 이름으로 리턴합니다.

구문

▶▶—%ENVVAR—▶▶

문맥

ENVVAR 명령문은 DEFINE 블록이나 명령문이 될 수 있습니다.

값

%ENVVAR

DEFINE 블록에서 변수를 환경 변수로 정의하기 위한 키워드. 이 변수는 매크로의 임의의 지점에서 환경 변수의 값을 가져옵니다.

제한

ENVVAR 명령문은 다른 요소를 포함할 수 없습니다.

예

예 1: 이 예에서 ENVVAR은 참조되었을 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

변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램을 지정합니다.

Net.Data는 매크로에서 실행가능 변수를 발견하면, 다음의 방법으로 참조된 실행가능 프로그램이 있는지 조회합니다.

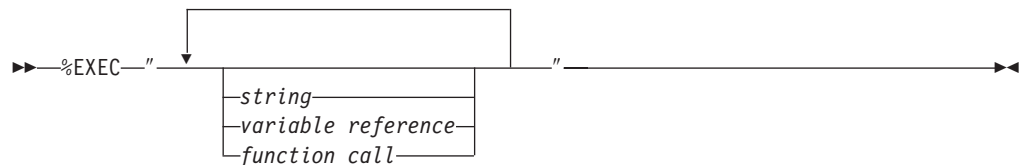
1. Net.Data 초기화 파일에서 EXEC_PATH를 탐색합니다. EXEC_PATH에 대한 자세한 내용은 운영 체제에 대한 *Net.Data* 관리 및 프로그래밍 안내서에서 구성 장을 참조하십시오.
2. 프로그램을 찾지 못할 경우 Net.Data는 시스템에서 정의한 디렉토리를 탐색합니다. 실행가능 프로그램을 찾을 경우, Net.Data는 그 프로그램을 수행합니다.

권한 부여 정보: Net.Data를 실행하는 사용자 ID는 EXEC문이나 블록에 참조된 모든 파일에 대한 액세스 권한을 가지고 있어야 합니다. 자세한 내용은 운영 체제에 대한 *Net.Data* 관리 및 프로그래밍 안내서의 구성 장에서 Net.Data 파일에 대한 웹 서버 액세스 권한 지정 절을 참조하십시오.

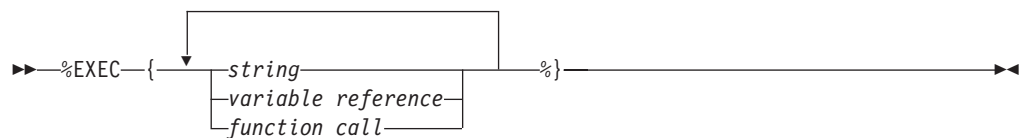
EXEC 명령문과 블록은 두개의 서로 다른 컨텍스트에 사용되며, 사용되는 컨텍스트에 따라 서로 다른 구문을 갖습니다. DEFINE 블록에서는 EXEC 명령문을 사용하고 FUNCTION 블록에서는 EXEC 블록을 사용하십시오.

구문

DEFINE 블록에서 사용되는 EXEC 명령문 구문



FUNCTION 블록에서 사용되는 EXEC 블록 구문



값

%EXEC

변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름을 지정하는 키

워드. Net.Data가 EXEC 명령문에 정의되어 있는 변수 참조를 만나면, Net.Data는 EXEC 명령문이 변수에 대해 선언한 내용을 처리합니다.

string

영문자, 숫자 및 구두점의 임의 조합. 문자열이 큰 따옴표안에 나타나면 개행 문자를 사용할 수 없습니다.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

문맥

EXEC 블록이나 명령문은 다음의 컨텍스트에서 찾을 수 있습니다.

- DEFINE 블록
- FUNCTION 블록

제한

EXEC 블록이나 명령문에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록
- 문자열
- 변수 참조
- 함수 호출

다음의 Net.Data 제공 언어 환경은 EXEC문을 지원합니다.

- REXX
- System
- Perl

예

예 1: 변수에 참조되는 실행가능 파일

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

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

이 예는 변수 mycall을 참조할 때마다 MYEXEC.EXE를 실행합니다.

예 2: 함수에 참조되는 실행가능 파일

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

이 예는 함수 my_rexx_pgm이 호출되면 mypgm.cmd를 실행합니다.

FUNCTION 블록

목적

Net.Data가 매크로에서 호출하는 서브루틴을 정의합니다. FUNCTION 블록의 실행가능 명령문은 언어 환경에서 직접 해석되는 인라인 명령문이 되거나 외부 프로그램에 대한 호출이 될 수 있습니다.

함수 블록내의 EXEC 블록: FUNCTION 블록내에 EXEC 블록을 사용하는 경우, 이것이 FUNCTION 블록에서의 유일한 실행가능 명령문이 되어야 합니다. 실행가능 명령문을 언어 환경에 전달하기 전에, Net.Data는 EXEC 블록에 있는 프로그램의 파일명을 초기화 파일의 EXEC_PATH 구성 명령문에 의해 정해진 경로명에 추가합니다. 이 결과 얻어진 문자열이 실행될 언어 환경으로 전송됩니다.

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

언어 명령문에 특수 문자 사용: Net.Data 언어 구문과 일치하는 문자가 함수 블록의 언어 명령문 섹션에서 구문상으로 유효한 내포 프로그램 코드(예: REXX 또는 Perl)의 일부로 사용되는 경우, 이들은 Net.Data 언어 구문으로 잘못 해석되어 매크로에서 오류나 예상치 못한 결과를 초래할 수 있습니다.

예를 들어, Perl 함수는 COMMENT 블록 분리문자 %{를 사용할 수 있습니다. 매크로가 수행되는 경우, %{ 문자는 COMMENT 블록의 시작으로 해석됩니다. 이 경우 Net.Data는 COMMENT 블록의 끝을 찾게 되고, 함수 블록의 끝을 찾게 되면 COMMENT 블록의 끝을 찾은 것으로 생각합니다. Net.Data는 다시 함수 블록의 끝을 찾게 되고 함수 블록의 끝을 찾을 수 없는 경우 오류를 발생시킵니다.

내포된 프로그램 코드를 Net.Data에서 특수 문자로 해석하지 않게 하면서 Net.Data 특수 문자를 내포된 프로그램 코드의 일부로 사용하려면 다음 방법 중 하나를 사용하십시오.

- 프로그램 코드를 호출하려면 코드를 인라인으로 만들기 보다는 EXEC 명령문을 사용하십시오.
- 특수 문자를 지정하려면 변수 참조를 사용하십시오.

예를 들어, 다음과 같은 Perl 함수는 COMMENT 블록 분리문자, %{를 나타내는 문자를 Perl 언어 명령문의 일부로 포함합니다.

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

Net.Data가 %{ 문자를 Net.Data COMMENT 블록 분리문자가 아닌 Perl 소스 코드로 해석하게 하려면 다음 중 한 방법으로 함수를 다시 쓰십시오.

- %EXEC 명령문 사용:

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

- 변수 참조를 사용하여 %{ 문자 지정:

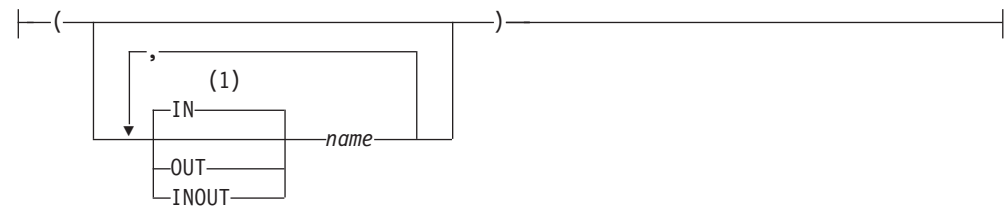
```
%define percent_openbrace = "%{"
%function(DTW_PERL) func() {
    ...
    for $num_words (sort bynumber keys $(percent_openbrace) $Rtitles{$num} )) {
        &make_links($Rtitles{$num}{$num_words});
    }
    ...
%}
```

구문

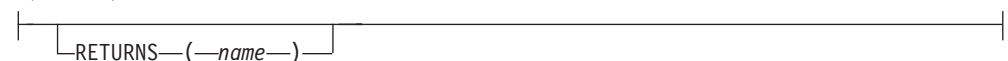
►► %FUNCTION—(—*lang_env*—)—*function_name*— | 매개변수 전달 스펙 |—————►

► | ;————— | 리턴 스펙 | {— | 함수 내용 | —} |—————►►

매개변수 전달 스펙:



리턴 스펙:



함수 내용:



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

OUT

언어 환경이 출력 자료를 Net.Data에 리턴하도록 지정합니다.

INOUT

Net.Data가 입력 자료를 언어 환경에 전달하고 언어 환경이 Net.Data에 출력 자료를 리턴하도록 지정합니다.

리턴 스펙:

RETURNS

함수가 완료된 후 언어 환경에 의해 할당된 함수 값이 들어 있는 변수를 선언합니다.

함수 내용:

inline statement block

함수 정의에 지정된 언어 환경(예: REXX, SQL 또는 Perl)의 구문상 유효한 명령문. 사용하고 있는 언어 환경에 대한 설명은 *Net.Data* 관리 및 프로그래밍 안내서에 나와 있습니다. 구문 및 사용법에 대해서는 프로그래밍 언어의 프로그래밍 참조서를 참조하십시오. 인라인 명령문 블록을 나타내는 문자열에는 Net.Data 변수 참조와 함수 호출을 포함시킬 수 있는 데, 이들은 인라인 명령문 블록(프로그램)을 실행하기 전에 평가됩니다.

exec block

EXEC 블록. 변수가 참조되거나 함수가 호출될 때 실행되는 외부 프로그램의 이름. 구문 및 예는 17 페이지의 『EXEC 블록 또는 명령문』을 참조하십시오.

report block

REPORT 블록. 함수 호출의 출력에 대한 포매팅 지침. 보고서에 머리말 및 꼬리말 정보를 사용할 수 있습니다. 구문 및 예는 59 페이지의 『REPORT 블록』을 참조하십시오.

message block

MESSAGE 블록. 리턴 코드 세트, 연관된 메시지, 함수 호출이 리턴될 때 Net.Data가 취하는 조치. 구문 및 예는 54 페이지의 『MESSAGE 블록』을 참조하십시오.

문맥

FUNCTION 블록은 다음 컨텍스트에 들어 있습니다.

- IF 블록
- Net.Data 매크로의 선언 부분에서 블록 또는 명령문의 외부.

제한

- FUNCTION 블록에는 다음의 요소가 포함될 수 있습니다.
 - 주석 블록
 - EXEC 블록
 - MESSAGE 블록
 - REPORT 블록
 - 인라인 명령문 블록
- Net.Data 변수 참조나 함수 호출이 없는 연속적인 인라인 명령문 블록 문자열의 최장 길이는 다음과 같이 제한됩니다.
 - OS/2 및 Windows NT의 경우: 64KB
 - AIX의 경우: 256KB
 - OS/390의 경우: 256KB
 - OS/400의 경우: 256KB
- 인라인 명령문 블록에 있는 SQL문의 길이는 다음과 같습니다. 각각의 데이터베이스에는 다른 제한사항이 있을 수 있으므로 해당 데이터베이스 문서를 참조하여 데이터베이스에 보다 적은 제한사항이 있는지 판별하십시오. IBM DB2 데이터베이스 제한사항은 Net.Data 한계와 다른 경우 아래에 나열되어 있습니다.
 - OS/2, Windows NT 및 UNIX의 경우: 64 KB
DB2에는 다음의 제한사항이 있습니다.
 - DB2 Universal Database V6 이상: 64 KB
 - DB2 Universal Database V5.2 이상: 32 KB
 - OS/390의 경우: 32 KB
 - OS/400의 경우: 32 KB

예

다음 예제는 범용 예제로써 모든 언어 환경에 적용되지는 않습니다. 특정 언어 환경에서 FUNCTION 블록을 사용하는 데 대한 자세한 내용은 *Net.Data Language Environment Reference*를 참조하십시오.

예 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 찾아지고 부속 문자열 FUNCTION 블록이 실행됩니다. 변수는 FUNCTION 블록내의 실행가능 명령문에서 대체되고, 텍스트 스트링

`s = substr("longstring", 1, 4)`이 REXX 해석기로 전달되어 실행됩니다. RETURNS 절이 지정되었기 때문에, *a*의 평가시 `@substring` 함수 호출의 값은 *s*의 값인 『long』으로 대체됩니다.

예 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)
  %}
```

함수의 INOUT 매개변수 *a* 및 *b*에 해당하는 변수 *w* 및 *x*. *y*의 값과 IN 매개변수 *c*에 해당하는 *y*의 값은 HTML 양식 입력이나 DEFINE 명령문에서 이미 정의했어야 합니다. 매개변수 *a*와 *b*가 값을 리턴하면 변수 *a* 및 *b*에 새로운 값이 지정됩니다. OUT 매개변수 *d*가 값을 리턴하면 변수 *z*가 정의됩니다.

- REXX 프로그램 mypgm.cmd:

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

- mypgm.cmd의 출력:

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

EXEC 명령문은 REXX 해석기가 외부 REXX 프로그램인 mypgm.cmd를 실행하도록 REXX 언어 환경에 지시합니다. REXX 언어 환경은 REXX 프로그램과 Net.Data 변수를 직접 공유할 수 있기 때문에, mypgm.cmd를 실행하기 전에 REXX 변수 *a*, *b* 및 *c*에 Net.Data 변수 *w*, *x* 및 *y*의 값을 지정합니다. mypgm.cmd는 REXX 명령문에 변수 *a*, *b* 및 *c*를 바로 사용할 수 있습니다. 프로그램이 종료되면, REXX 프로그램으로부터 REXX 변수 *a*, *b* 및 *d*가 검색되고, *y*의 값이 Net.Data 변수 *w*, *x* 및 *z*에 지정됩니다. my_rexx_pgm FUNCTION 블록의 정의에는 RETURNS 절이 사용되지 않으므로, @my_rexx_pgm 함수 호출의 값은 널(NULL) 문자열 『』이 되거나(리턴 코드가 0인 경우), REXX 프로그램 리턴 코드의 값이 됩니다(리턴 코드가 0이 아닌 경우).

예 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 query_1의 정의가 출력 테이블 변수를 식별하지 못하므로, 생략시 테이블이 사용됩니다(자세한 내용은 TABLE 변수 블록을 참조하십시오). REPORT 블록에서 참조되는 NLIST 및 Vi 변수는 생략시 테이블 정의에 의해 정의됩니다. REPORT 블록에 의해 작성된 보고서는 query_1 함수가 호출된 출력 HTML에 배치됩니다.

예 4: Perl 스크립트를 실행하기 위한 시스템 호출

- Net.Data 매크로:

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

- Perl 프로그램 today.pl:

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

System 언어 환경은 FUNCTION 블록내의 실행가능 명령문들을 C 언어 system() 함수 호출을 통해 운용 시스템으로 전달하여 해석합니다. 이 방법은 Net.Data 변수를 REXX 언어 환경에서처럼 실행가능 명령문으로 직접 전달하거나 검색하지 못하도록 합니다. 따라서 System 언어 환경은 다음과 같은 방식으로 변수를 전달 및 검색합니다.

- 입력 매개변수는 putenv() 함수를 통해 시스템 환경 변수로서 전달되며, 실행가능 프로그램에 의해 검색될 수 있습니다. 다른 언어 환경에서는 다른 방법으로 변수를 참

조합니다. UNIX cshell 스크립트는 \$x와 같이 환경 변수명 앞에 \$를 붙여서 환경 변수를 참조합니다. Perl 언어 스크립트는 %ENV{'x'}와 같이 연관 배열 %ENV를 사용하여 환경 변수를 참조합니다. DOS 배치(.BAT) 파일은 %x%와 같이 변수를 퍼센트 기호로 묶어서 변수를 참조합니다.

- 출력 매개변수는 환경 변수 DTWPIPE로 이름이 전달되는 파이프를 작성하여 환경 변수에 다시 전달됩니다. 단, OS/400 플랫폼의 경우는 예외인데, 여기서는 출력 매개변수가 시스템 환경 변수로 언어 환경에 전달됩니다. named pipe에 기록된 자료는 DEFINE 문에서와 마찬가지로 name="value" 형식을 갖습니다. 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 실행가능 명령문에서 변수 대체를 수행합니다. 이 예제에서는 실행가능 명령문에 Net.Data 변수가 없으므로, 어떠한 변수 대체도 수행되지 않습니다. 실행가능 명령문과 매개변수가 System 언어 환경으로 전달되면, 언어 환경은 named pipe를 작성하고 환경 변수 DTWPIPE를 파이프 이름으로 설정합니다.

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

예 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()
}%
```

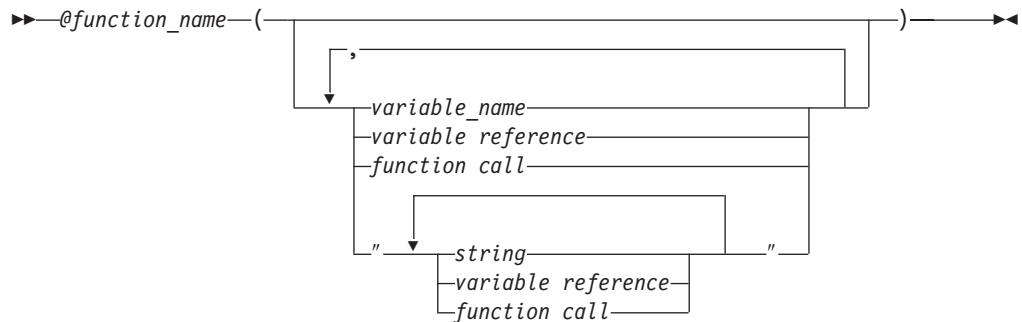
EXEC 블록이 사용되는 방법을 보려면 이 예를 예 4와 비교해 보십시오. 예 4에서, System 언어 환경은 Perl 프로그램을 해석하는 방법은 알지 못하지만, 외부 프로그램을 호출하는 방법은 알고 있습니다. EXEC 블록은 perl 프로그램을 외부 프로그램으로 호출하도록 지시합니다. 실제 Perl 언어 명령문은 외부 Perl 프로그램에 의해 해석됩니다. 예 5에는 Perl 언어 환경이 Perl 언어 명령문을 직접 해석할 수 있으므로 EXEC 블록이 없습니다.

함수 호출 (@)

목적

FUNCTION 블록, MACRO_FUNCTION 블록 또는 인수가 지정된 내장 함수를 호출합니다. 함수가 내장 함수가 아닌 경우, 함수 호출을 지정하기 전에 이를 Net.Data 매크로에 정의해야 합니다.

구문

가
ㅂㅅ

@function_name

임의의 기존의 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. 구문에 대해서는 5 페이지의 『변수명』을 참조하십시오.

string

개행 문자를 제외한 영문자, 숫자 및 구두점의 임의 조합.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다.

문맥

다음의 컨텍스트에서 함수 호출을 찾을 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 선언부내의 모든 블록의 외부

제한

- 함수 호출은 다음과 같은 요소를 포함할 수 있습니다.
 - 주식 블록
 - 문자열
 - 함수 호출
 - 변수 참조
- OUT 또는 INOUT 매개변수 값에는 변수 참조, 함수 호출 또는 리터럴 문자열이 포함될 수 없습니다.

예

예 1: SQL 함수 formQuery에 대한 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()
%}
```

예 2: 입력 및 출력 매개변수를 포함한 REXX 함수에 대한 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)
%}

```

예 3: 변수 참조 및 함수 호출을 사용하는 입력 매개변수가 있는 REXX 함수에 대한 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)
%}

```

예 4: INOUT 매개변수의 사용을 나타내는 매크로.

```

%DEFINE a = "initial value of a"
%FUNCTION(DTW_REXX) func1(INOUT x) {
    Say 'value at start of function:<br>'
    Say 'x =' x
    Say '
<p>
'
    x = "new value of a"
    %REPORT {

<p>
value at start of report block:<br>
    x = $(x)<br>
    @dtw_assign(x, "newest value of a")
    value at end of report block:<br>
    x = $(x)<br>
    %}
%}
%HTML(report) {
    initial values:<br>
    a = $(a)<br>
    @func1(a)
    value after function call:<br>
    a = $(a)<br>
%}

```

결과 출력:

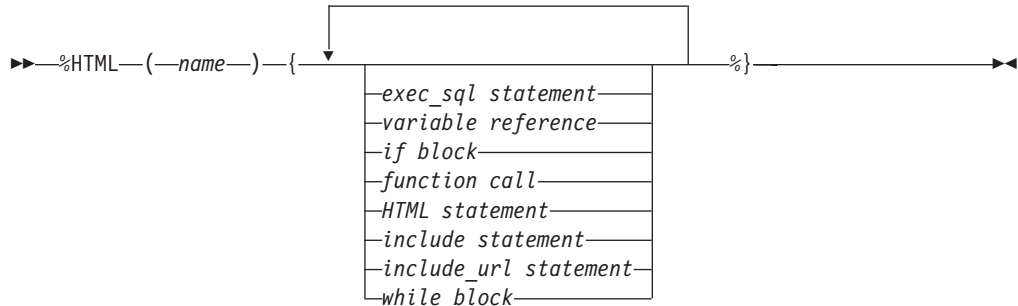
```
initial values:  
a = initial value of a  
value at start of function:  
x = initial value of a  
value at start of report block:  
x = new value of a  
value at end of report block:  
x = newest value of a  
value after function call:  
a = newest value of a
```

HTML 블록

목적

웹 페이지 표시 방법을 정의합니다. 실행할 HTML 블록의 이름은 Net.Data 호출 시에 URL에 지정됩니다. HTML 블록에는 대부분의 Net.Data 매크로 언어 명령문과 HTML 및 Javascript와 같은 유효한 표시 명령문이 포함될 수 있습니다.

구문



값

%HTML

클라이언트의 브라우저에 표시할 HTML 태그 및 텍스트가 포함된 블록을 지정하는 키워드.

name

영문자 또는 밑줄로 시작하며 마침표를 비롯하여 영문자, 숫자 또는 밑줄 문자의 조합이 들어 있는 영문자 또는 숫자 문자열.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 281 페이지의 『부록B. DB2 WWW 연결』 또는 DB2 World Wide Web 릴리스 1 책자를 참조하십시오.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수를 나타내는 문자열이고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다. 구문 및 예는 35 페이지의 『IF 블록』을 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

HTML statements

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

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 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

문맥

HTML 블록은 다음 컨텍스트에 들어 있습니다.

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

제한

HTML 블록에는 다음의 요소가 포함될 수 있습니다.

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

예

예 1: 머리말 및 꼬리말을 위한 include 파일이 있는 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"  
%}
```

예 2: 이름에 마침표가 포함된 HTML 블록

```
%HTML(my.report){  
%INCLUDE"header.html"  
<P>You can put <EM>any</EM> HTML in an HTML block.  
An SQL function call is made like this:  
@xmp1()  
%INCLUDE"footer.html"  
%}
```


IF 블록

목적

조건부 문자열 처리를 수행합니다. IF 블록은 하나 이상의 조건을 테스트하고 조건 테스트의 결과에 따라 명령문 블록을 수행하도록 합니다. 또다른 IF 블록내에 IF 블록을 중첩시킬 수 있을 뿐 아니라 Net.Data 매크로의 선언부에 있는 IF 블록, HTML 블록, MACRO_FUNCTION 블록, REPORT 블록, WHILE 블록 및 ROW 블록을 사용할 수 있습니다.

조건 목록의 문자열 값은 이것이 정수를 나타내는 문자열이고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다.

제한사항: Net.Data는 유동 소수점 수와 같이 정수가 아닌 숫자의 숫자 비교는 지원하지 않습니다.

중첩된 IF 블록: IF 블록 구문의 규칙은 매크로에서 블록의 위치에 의해 결정됩니다. IF 블록이 선언부에서 다른 블록의 외부에 있는 IF 블록내에 중첩되어 있는 경우, 외부 블록이 사용할 수 있는 모든 요소를 사용할 수 있습니다. IF 블록이 IF 블록에 있는 다른 블록내에 중첩되어 있는 경우, 내부에 있는 블록의 구문 규칙을 따릅니다.

다음 예에서 중첩된 IF 블록은 HTML 블록 내부에 있을 때 사용된 규칙을 따라야 합니다.

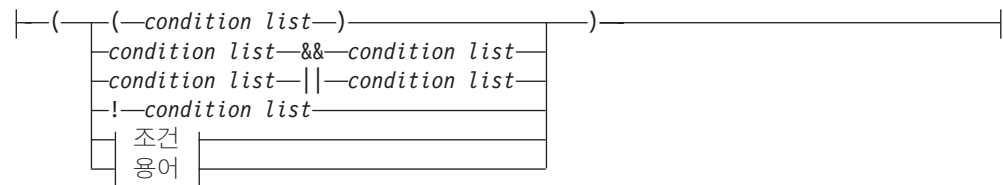
```
%IF block
...
%HTML block
...
%IF block
```

최대 1024개의 IF 블록을 중첩시킬 수 있습니다.

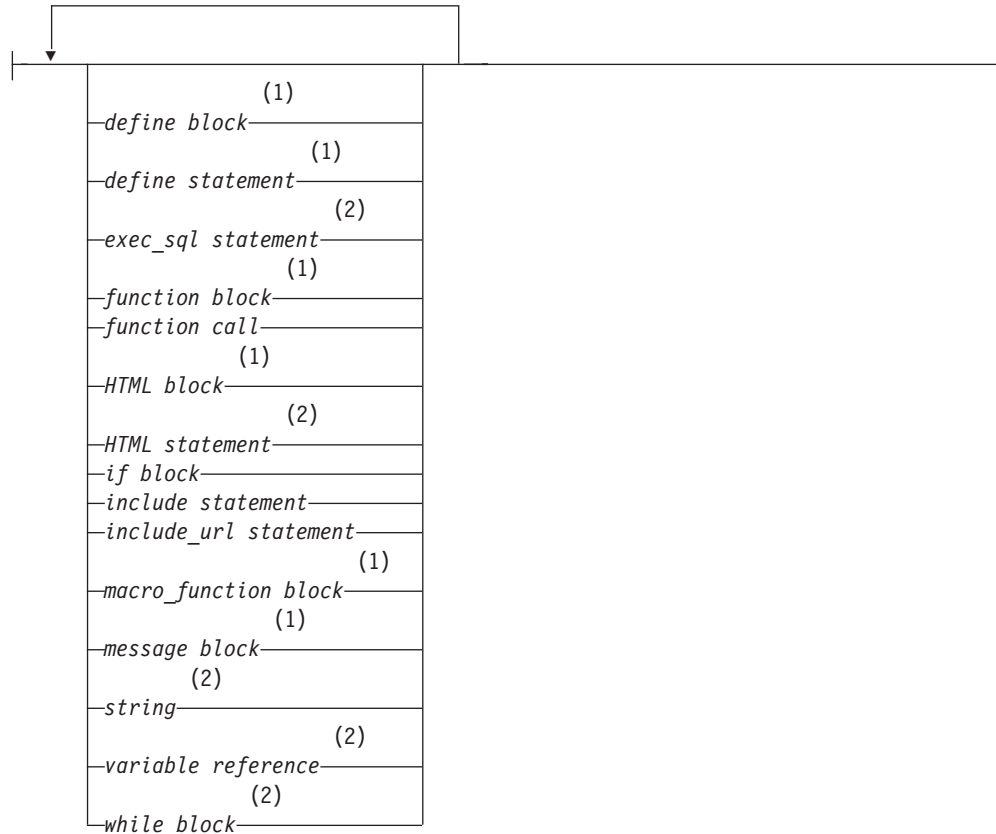
구문

```
►►%IF 조건 목록 statement_block else_if spec %ENDIF◄◄
```

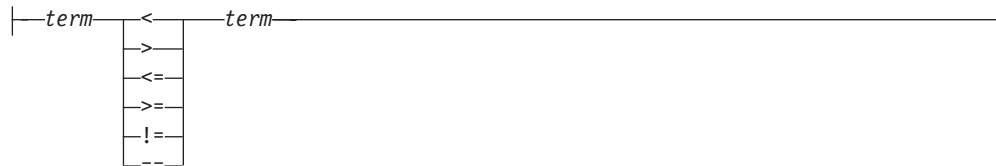
조건 목록:



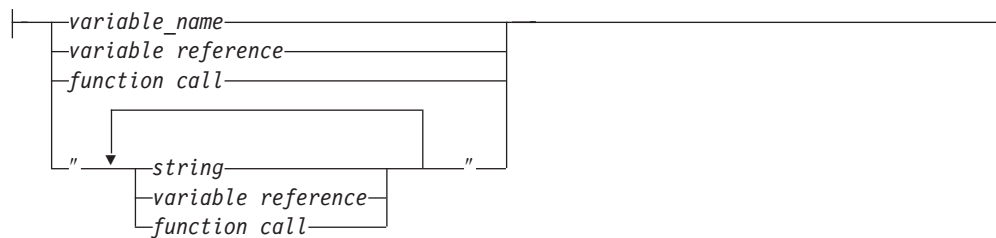
statement_block:



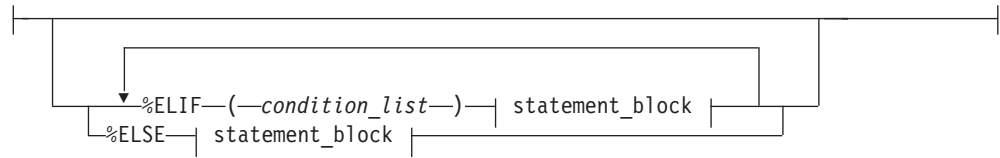
조건:



용어:



else_if spec:



주:

- 1 이 언어 구문은 IF 블록이 매크로의 선언부에 있는 다른 블록의 외부에 위치한 경우에 유효합니다.
- 2 이 언어 구문은 IF 블록이 HTML 블록, MACRO_FUNCTION 블록, REPORT 블록, ROW 블록 또는 WHILE 블록에 위치한 경우에 유효합니다.

값

%IF

조건부 문자열 처리를 지정하는 키워드.

condition list

조건의 값을 비교합니다. 조건 목록은 부울 연산자를 사용하여 연결할 수 있습니다. 조건 목록은 다른 조건 목록내에 중첩될 수 있습니다.

statement_block

다음과 같은 유효한 Net.Data 매크로 구문. 매크로 구문이 유효하게 되는 컨텍스트를 알려면 도표 및 제한사항을 참조하십시오.

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 예는 11 페이지의 『DEFINE 블록 또는 명령문』을 참조하십시오.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 281 페이지의 『부록B. DB2 WWW 연결』 또는 DB2 World Wide Web 릴리스 1 책자를 참조하십시오.

function block

Net.Data 매크로로부터 호출될 수 있는 서브루틴을 지정하는 키워드. FUNCTION 블록의 실행가능 명령문은 언어 환경에서 직접 해석되는 언어 명령문을 포함하거나 외부 프로그램에 대한 호출을 나타낼 수 있습니다. 구문 및 예는 20 페이지의 『FUNCTION 블록』을 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출(@)』을 참조하십시오.

HTML block

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

HTML statement

영문자 또는 숫자 문자와 클라이언트의 브라우저에 적합하게 형식화할 HTML 태그가 포함됩니다.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수를 나타내는 문자열이고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

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 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

macro_function block

Net.Data 매크로로부터 호출될 수 있는 서브루틴을 지정하는 키워드. MACRO_FUNCTION 블록의 실행가능 명령문은 Net.Data 매크로 언어 소스 명령문을 포함할 수 있습니다. 구문 및 예는 50 페이지의 『MACRO_FUNCTION 블록』을 참조하십시오.

message block

MESSAGE 블록. 리턴 코드 세트, 연관된 메시지, 함수 호출이 리턴될 때 Net.Data가 취하는 조치. 구문 및 예는 54 페이지의 『MESSAGE 블록』을 참조하십시오.

string

영문자, 숫자 및 구두점의 임의 조합. 문자열이 조건 목록의 용어에 들어 있는 경우, 개행 문자를 제외한 모든 문자를 포함할 수 있습니다. 문자열이 코드의 실행가능 블록에 있는 경우, 개행 문자를 포함한 모든 문자를 포함할 수 있습니다.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

condition

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

- 조건 연산자가 연산자 <,<=,>,>=,==,!= 중 하나입니다.
- 두 용어가 유효한 정수를 나타내는 문자열입니다. 여기서 유효한 정수란 플러스(+) 또는 마이너스(-) 기호가 선행할 수도 있는 숫자로 된 문자열이며 다른 공백이 없는 경우를 말합니다.

참이 아닌 조건이 있는 경우, 일반적인 문자열 비교가 수행됩니다.

term

변수명, 문자열, 변수 참조 또는 함수 호출.

%ELIF

대체 처리 경로를 시작하고 조건 목록과 대부분의 Net.Data 매크로 명령문을 포함할 수 있는 키워드.

%ENDIF

%IF 블록을 닫는 키워드.

%ELSE

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

문맥

IF 블록은 다음 컨텍스트에 들어 있습니다.

- Net.Data 매크로 선언부내의 다른 블록의 외부
- HTML 블록
- IF 블록
- MACRO_FUNCTION 블록
- REPORT 블록
- ROW 블록
- WHILE 블록

제한

IF 블록이 Net.Data 매크로의 선언 부분에 있는 다른 블록의 외부에 위치할 경우, 다음의 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 블록이 Net.Data 매크로의 HTML 블록, MACRO_FUNCTION 블록, REPORT 블록, ROW 블록 또는 WHILE 블록에 위치할 경우, 이 IF 블록에는 다음의 요소가 포함될 수 있습니다.

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

최대 1024개의 IF 블록을 중첩시킬 수 있습니다.

예

예 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){
...
%}

```

예 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
%}

```

예 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

```

숫자 비교는 내재된 테이블 변수, ROW_NUM가 항상 정수 값을 리턴하고 비교되는 값도 정수이기 때문에 수행됩니다.

예 4: 중첩된 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 INCLUDE 파일을 찾습니다.

대부분의 고급 언어에서와 동일한 방법으로 인클루드 파일을 사용할 수 있습니다. 인클루드 파일은 공통 머리말과 꼬리말을 삽입하고, 공통 변수 세트를 정의하거나 FUNCTION 블록 정의의 공통 서브루틴 라이브러리를 Net.Data 매크로에 통합시킬 수 있습니다.

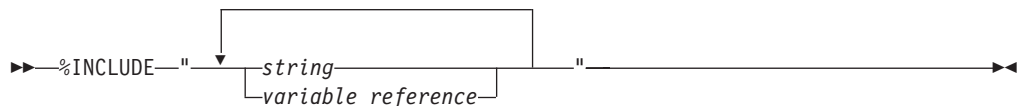
Net.Data는 매크로 처리시에만 INCLUDE 명령문을 실행하고 포함된 파일의 내용을 매크로의 INCLUDE 명령문 위치에 삽입합니다. 포함된 파일의 이름에 있는 변수 참조는 이 파일의 내용이 실행될 예정인 경우만 아니면 INCLUDE문이 처음 실행될 때 해석됩니다.

INCLUDE 명령문이 ROW 또는 WHILE 블록에 있는 경우, Net.Data는 INCLUDE 명령문을 반복적으로 실행하지 않습니다. Net.Data는 ROW 또는 WHILE 블록을 처음 실행할 때만 INCLUDE 명령문을 실행하고 포함된 파일의 내용을 블록에 통합한 다음 포함된 파일의 내용을 사용하여 ROW 또는 WHILE 블록을 반복적으로 실행합니다.

권한 부여 관련사항: Net.Data를 실행하는 사용자 ID는 INCLUDE 문에 참조된 모든 파일에 대해 액세스권을 가져야 합니다. 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장에서 Net.Data 파일에 대한 웹 서버 액세스권 지정과 관련한 부분을 참조하십시오.

참조: 국지 웹 서버의 HTML 파일을 포함시키고자 하는 경우, INCLUDE_URL에 대한 예 3에 보인 것처럼 INCLUDE_URL 구문을 사용하십시오. 예시된 구문을 사용하면 이미 웹 서버가 알고 있는 디렉토리를 지정하기 위해 Net.Data 초기화 파일의 INCLUDE_PATH를 갱신할 필요가 없습니다.

구문



값

%INCLUDE

파일이 읽혀 Net.Data 매크로에 통합되도록 지정하는 키워드.

name

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

string

개행 문자를 제외한 영문자, 숫자 및 구두점의 임의 조합.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

문맥

INCLUDE 명령문은 다음의 컨텍스트에서 찾을 수 있습니다.

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

제한

INCLUDE 명령문에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록
- 문자열
- 변수 참조

문자열에는 함수 호출을 사용할 수 없습니다.

최대 10개의 INCLUDE 명령문을 중첩시킬 수 있습니다.

예

예 1: HTML 블록 내의 INCLUDE 명령문

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

예 2: REPORT 블록 내의 INCLUDE 명령문

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

예 3: INCLUDE 명령문의 변수 참조

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

INCLUDE_URL 명령문

목적

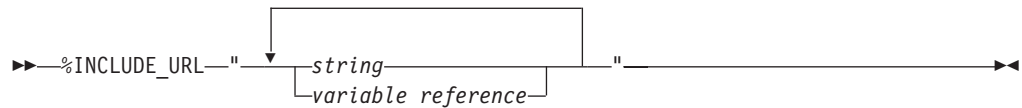
다른 파일을 읽어 명령문이 지정된 Net.Data 생성 출력에 통합합니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다.

INCLUDE_URL 명령문을 사용하면 응용프로그램 사용자가 제출 버튼을 선택하지 않아도 다른 매크로에서 매크로를 호출할 수 있습니다.

Net.Data는 매크로 처리시에만 INCLUDE_URL 명령문을 실행하고 포함된 파일의 내용을 매크로의 INCLUDE_URL 명령문 위치에 삽입합니다. 포함된 파일의 이름에 있는 변수 참조는 이 파일의 내용이 실행될 예정인 경우만 아니면 INCLUDE_URL 명령문이 처음 실행될 때 해석됩니다.

INCLUDE_URL 명령문이 ROW 또는 WHILE 블록에 있는 경우, Net.Data는 INCLUDE_URL 명령문을 반복적으로 실행하지 않습니다. Net.Data는 ROW 또는 WHILE 블록을 처음 실행할 때만 INCLUDE_URL 명령문을 실행하고 포함된 파일의 내용을 블록에 통합한 다음 포함된 파일의 내용을 사용하여 ROW 또는 WHILE 블록을 반복적으로 실행합니다.

구문



값

%INCLUDE_URL

국지 또는 원격 서버에서 파일이 읽혀 Net.Data 매크로에 통합되도록 지정하는 키워드.

string

개행 문자를 제외한 영문자, 숫자 및 구두점의 임의 조합.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

문맥

INCLUDE_URL 명령문은 다음의 컨텍스트에서 찾을 수 있습니다.

- DEFINE 블록

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

제한

INCLUDE_URL 명령문에는 다음의 요소가 포함될 수 있습니다.

- 주식 블록
- 문자열
- 변수 참조

INCLUDE_URL 파일에는 다음과 같은 파일 크기 제한이 적용됩니다.

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

최대 10개의 INCLUDE_URL 명령문을 중첩시킬 수 있습니다.

INCLUDE_URL은 OS/400 환경에서 지원되지 않습니다.

예

예 1: 다른 서버의 HTML 파일 포함

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

예 2: 서버명을 호출하여 원격 서버의 HTML 파일 포함

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

여기서 myserver는 서버명입니다.

예 3: 국지 웹 서버의 HTML 파일 포함

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

참고: 이 방법을 이용하면 이미 웹 서버가 알고 있는 디렉토리를 지정하기 위해 Net.Data 구성 파일의 INCLUDE_URL 경로를 갱신할 필요가 없습니다. *string*이 슬래쉬로 시작되지 않는 경우, Net.Data는 문자열이 서버명인 것으로 간주하고 해당 이름을 갖는 서버에서 파일을 검색합니다.

예 4: 원격 서버의 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)"
```

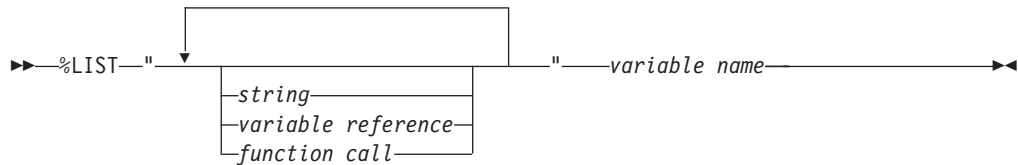
이 예에서는 매크로 hotpic.mac가 호출되고 custno가 변수로 전송됩니다. 문자열이 슬래쉬로 시작되면 Net.Data는 국지 웹 서버에서 INCLUDE 파일을 검색합니다.

LIST 명령문

목적

제한된 값 목록을 구축합니다. 일부 WHERE 또는 HAVING 절에 있는 것과 같은 여러 항목을 사용하여 SQL 조회를 구축할 때 LIST 명령문을 사용할 수 있습니다.

구문



값

%LIST

분리된 값 일람표를 구성하는 데 사용되는 변수를 지정하는 키워드.

string

개행 문자를 제외한 영문자, 숫자 및 구두점의 임의 조합.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, `VAR='abc'`일 경우, `$(VAR)`는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문에 대해서는 5 페이지의 『변수명』을 참조하십시오.

문맥

LIST 명령문은 다음의 컨텍스트에서 찾을 수 있습니다.

- DEFINE 명령문

제한

LIST 명령문에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록
- 변수 참조
- 함수 호출
- 문자열

예

예 1: 변수 목록

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

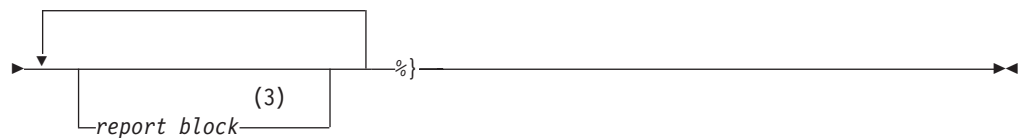
MACRO_FUNCTION 블록

목적

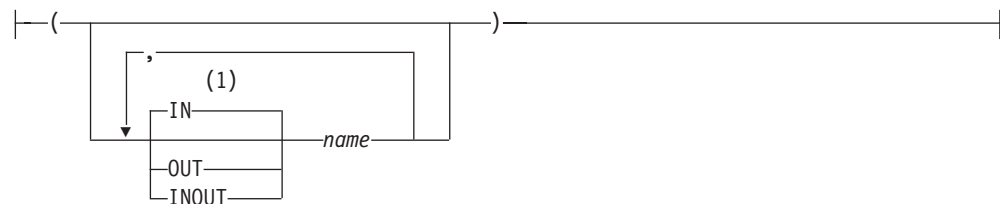
Net.Data 매크로로부터 호출할 수 있는 서브루틴을 정의합니다. MACRO_FUNCTION 블록의 실행가능 명령문은 Net.Data 매크로 언어 소스 명령문이어야 합니다.

구문

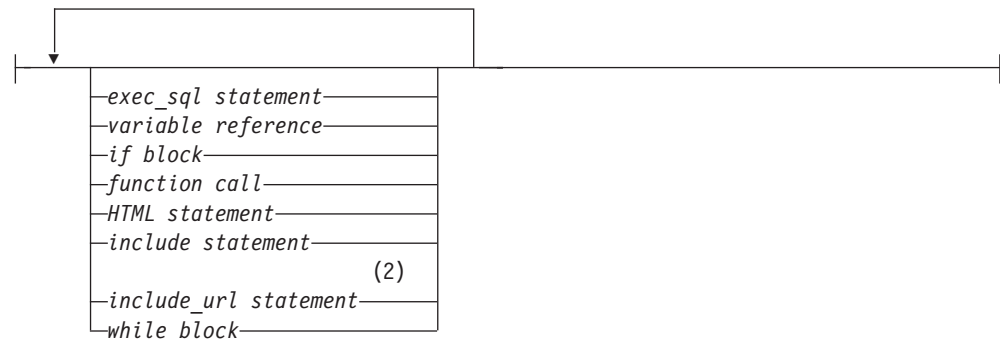
▶▶%MACRO_FUNCTION—*function_name*—| 매개변수 전달 스펙 |{—| 함수 내용 |—▶▶



매개변수 전달 스펙:



함수 내용:



주:

- 1 IN의 생략시 매개변수 유형은 매개변수 목록의 서두에 매개변수 유형이 지정되지 않은 경우 사용됩니다. 매개변수 유형이 없는 매개변수는 매개변수 목록에서 가장 최근에 지정된 유형을 사용하거나, 아무 유형도 지정되지 않은 경우 유형 IN을 사용합니다. 예를 들어, 매개변수 목록 -(-parm1 -, INOUT -parm2 -, -parm3 -, OUT -parm4 -, -parm5 -)에서, 매개변수 -parm1, -parm3 및 -parm5는 매개변수 유형이 없습니다. 매개변수 -parm1은 IN의 유형을 갖는 데, 처음부터 매개변수 유형이 지정되지 않았기 때문입니다. 매개변수 -parm3은 가장 최근에 지

정된 매개변수 유형인 INOUT 유형을 갖습니다. 마찬가지로, 매개변수 `-parm5`는 매개변수 목록에서 가장 최근에 지정된 유형인 OUT 유형을 갖습니다.

- 2 INCLUDE_URL 명령문은 OS/400에서 지원되지 않습니다.
- 3 MACRO_FUNCTION 블록 내의 여러 REPORT 블록은 OS/400, OS/2, Windows NT 및 UNIX가 지원합니다. 반복된 report 블록은 언어 환경을 호출하는 함수에만 사용할 수 있습니다.

값

%MACRO_FUNCTION

Net.Data 매크로로부터 호출될 수 있는 서브루틴을 지정하는 키워드. MACRO_FUNCTION 블록의 실행가능 명령문은 Net.Data가 직접 해석하는 언어 명령문을 포함하고 있어야 합니다.

function_name

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

매개변수 전달 스펙:

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

OUT

언어 환경이 출력 자료를 Net.Data에 리턴하도록 지정합니다.

INOUT

Net.Data가 입력 자료를 언어 환경에 전달하고 언어 환경이 Net.Data에 출력 자료를 리턴하도록 지정합니다.

name

영문자 또는 밑줄로 시작하는 영문자 또는 숫자 문자열로, 영문자, 숫자 또는 밑줄로 이루어진 모든 조합이 포함될 수 있습니다. name은 Net.Data 테이블이나 결과 세트를 나타낼 수 있습니다.

함수 내용:

exec_sql

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 281 페이지의 『부록B. DB2 WWW 연결』 또는 DB2 World Wide Web 릴리스 1 책자를 참조하십시오.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수를 나타내고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출(@)』을 참조하십시오.

HTML statement

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로에 통합합니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

report block

REPORT 블록. 함수 호출의 출력에 대한 포매팅 지침. 보고서에 머리말 및 꼬리말 정보를 사용할 수 있습니다. 구문 및 예는 59 페이지의 『REPORT 블록』을 참조하십시오.

문맥

MACRO_FUNCTION 블록은 다음의 컨텍스트에서 찾을 수 있습니다.

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

제한

MACRO_FUNCTION 블록은 다음과 같은 요소를 포함할 수 있습니다.

- 주식 블록
- EXEC_SQL 명령문
- HTML 명령문

- IF 블록
- INCLUDE 명령문
- INCLUDE_URL 명령문
OS/400에는 지원되지 않음
- REPORT 블록
- WHILE 블록
- 변수 참조
- 함수 호출

예

예 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
%}
```

예 2: 머리말 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

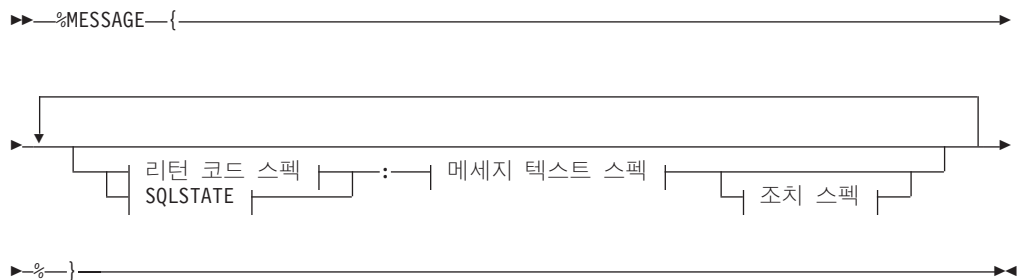
목적

함수로부터의 리턴 코드에 기초하여 표시할 메시지와 취할 조치를 지정합니다.

리턴 코드, 이의 관련 메시지 및 조치를 MESSAGE 블록에 정의하십시오. 함수 호출이 완료되면, Net.Data는 여기서 리턴된 리턴 코드를 MESSAGE 블록에 정의되어 있는 리턴 코드와 비교합니다. 함수의 리턴 코드가 MESSAGE 블록에 있는 리턴 코드와 일치하는 경우, Net.Data는 메시지를 표시하고 조치를 평가하여 처리를 계속할지 또는 Net.Data 매크로를 종료할지를 결정합니다.

MESSAGE 블록은 범위내에서 전역이거나 하나의 FUNCTION 블록에 대해 국지가 될 수 있습니다. MESSAGE 블록이 가장 바깥의 매크로 계층에 정의되어 있으면, 이는 범위내에서는 전역으로 간주됩니다. 여러 개의 전역 MESSAGE 블록이 정의되면, 마지막으로 처리된 블록만이 활동중인 것으로 간주됩니다. MESSAGE 블록이 FUNCTION 블록 내부에 정의되어 있으면, 블록이 정의된 FUNCTION 블록에 대한 범위내에서는 국지입니다. 규칙을 처리하는 리턴 코드에 대해서는 *Net.Data Administration and Programming Guide*의 MESSAGE 블록 부분을 참조하십시오.

구문



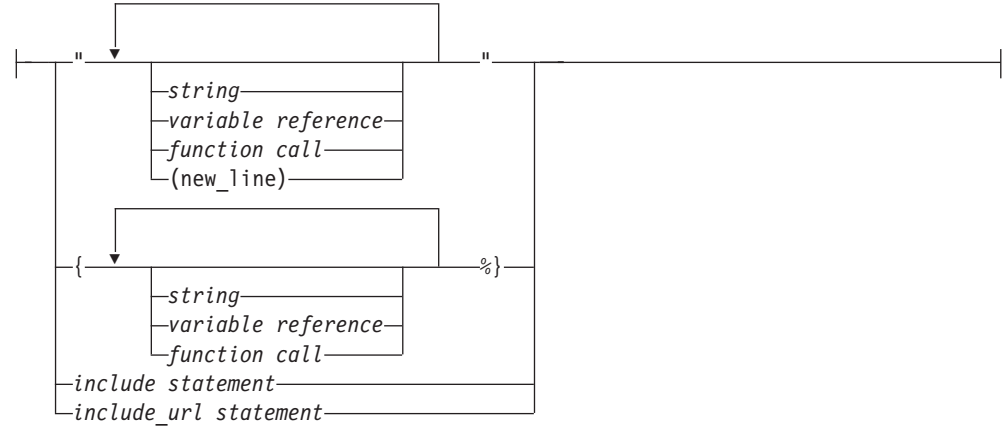
리턴 코드 스펙:



SQLSTATE:



메세지 텍스트 스펙:



조치 스펙:



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 변수의 값이 *return code spec* 값과 일치하면, 메세지 명령문의 나머지 정보는 함수 호출을 처리하는 데 사용됩니다. MESSAGE 블록에 입력되지 않은 리턴 코드에 대해서도 메세지를 지정할 수 있습니다.

+DEFAULT

양수의 생략시 메세지 코드를 지정하는 데 사용되는 키워드. RETURN_CODE가 0보다 크고 정확히 일치하는 값이 지정되어 있지 않으면, Net.Data는 이 메세지 명령문내의 정보를 함수 호출을 처리하는 데 사용합니다.

-DEFAULT

음수의 생략시 메세지 코드를 지정하는 키워드. RETURN_CODE가 0보다 작고 정확히 일치하는 값이 지정되어 있지 않으면, Net.Data는 이 메세지 명령문내의 정보를 함수 호출을 처리하는 데 사용합니다.

DEFAULT

생략시 메세지 코드를 지정하는 키워드. 다음 조건이 모두 충족되면, Net.Data는 이 메세지 명령문내의 정보를 함수 호출을 처리하는 데 사용합니다.

- RETURN_CODE가 0보다 크거나 작고, 0이 아닙니다.
- 리턴 코드에 정확히 일치하는 값이 지정되지 않았습니다.
- RETURN_CODE가 0보다 크거나 작을 때 +DEFAULT 또는 -DEFAULT 값이 지정되지 않았습니다.

msg_code

처리중에 발생할 수 있는 오류나 경고를 지정하는 메시지 코드. 0에서 9까지의 값을 갖는 숫자 문자열.

SQLSTATE

공통 오류 조건에 대한 공통 코드를 응용프로그램에 제공하는 키워드. SQLSTATE 값은 SEL 표준에 들어 있는 SQLSTATE 스펙에 기초하며 코드 체계는 IBM의 모든 SQL에서 동일합니다.

state_id

SQLSTATE. 형식이 ccsss인 5자(바이트)로 구성된 영숫자 문자열로 여기서 cc는 클래스를 나타내며 sss는 부속 클래스를 나타냅니다.

메시지 텍스트 스펙

RETURN_CODE가 현재의 메시지 명령문내의 *return_code* 값과 일치할 경우 웹 브라우저에 전송되는 문자열.

string

영문자, 숫자 및 구두점의 임의 조합. 문자열이 큰 따옴표안에 나타나면, 개행 문자를 사용할 수 없습니다.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출(@)』을 참조하십시오.

조치 스펙

RETURN_CODE가 현재의 메시지 명령문내의 *return_code* 값과 일치할 경우에 Net.Data가 취할 조치를 판별합니다.

EXIT

지정된 메시지 코드에 해당하는 오류나 경고가 발생했을 때 매크로가 즉시 종료되도록 지정하는 키워드. 이것이 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. MESSAGE 내에서 INCLUDE 명령문의 위치에는 제약이 없습니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로에 통합합니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

문맥

MESSAGE 블록은 다음의 컨텍스트에서 찾을 수 있습니다.

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

제한

MESSAGE 블록에는 다음의 요소가 포함될 수 있습니다.

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

OS/390, OS/2, Windows NT 및 UNIX 운영 체제의 경우: SQL 함수를 SQL 함수 내에서 호출할 수 없습니다.

예

예 1: 국지 MESSAGE 블록

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

예 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
links and forms, in this message. %}      : continue
  %}

%{ local message block inside a FUNCTION block %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE {
    -100      : "Return code -100 message"      : exit
    100      : "Return code 100 message"       : continue
    -default : {
This is a long message that spans more
than one line. You can use HTML tags, including
links and forms, in this message. %}      : exit
  %}
}
```

예 3: INCLUDE 명령문이 포함된 MESSAGE 블록

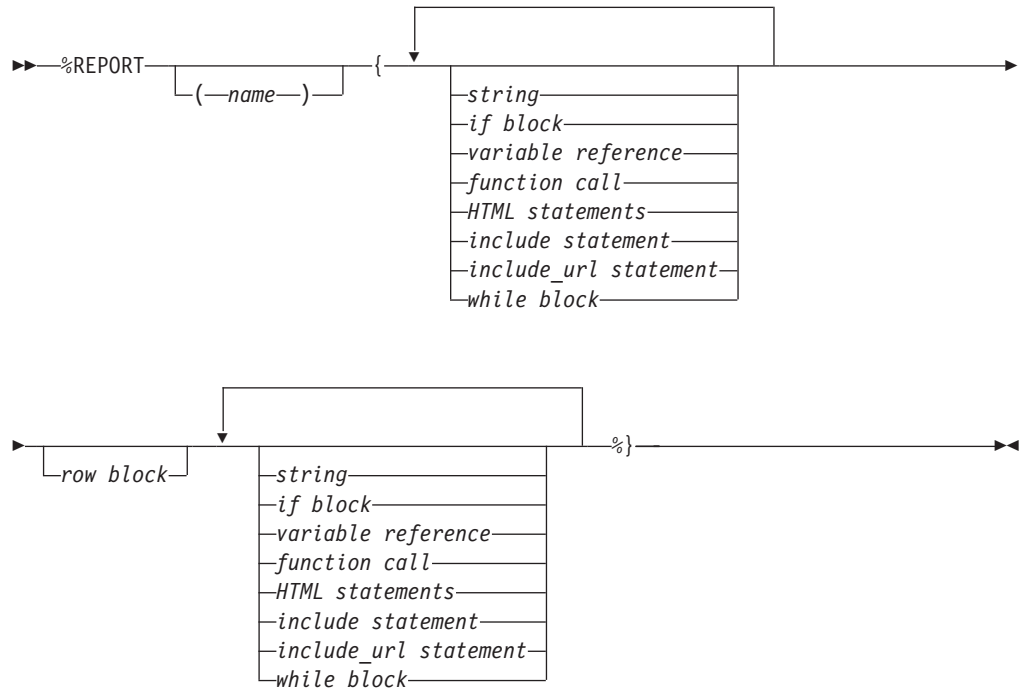
```
%message {
  %include "rc1000.msg"
  %include "rc2000.msg"
  %include "defaults.msg"
}
```


REPORT 블록

목적

함수 호출의 출력을 포맷합니다. 테이블명 매개변수를 지정하여 명명된 테이블의 자료가 보고서에 사용되도록 지정할 수 있습니다. 그렇지 않으면, 함수 호출 매개변수 목록의 첫번째 출력 테이블을 사용하여 보고서가 생성되거나, 목록에 테이블명이 없는 경우 생략시 테이블 자료가 보고서 생성에 사용됩니다.

구문



값

%REPORT

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

name

이 값은 Net.Data 테이블이나 결과 세트를 나타냅니다. 자세한 내용은 *Net.Data Administration & Programming Guide*의 보고서 블록 부분을 참조하십시오.

string

영문자, 숫자 및 구두점의 임의 조합.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수를 나타내고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여

기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다. 구문 및 예는 35 페이지의 『IF 블록』을 참조하십시오.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 Net.Data 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

HTML statements

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로에 통합합니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

row block

ROW 블록. 함수 호출로부터 리턴된 각 자료 행에 대해 HTML 형식 자료를 표시합니다. 구문 및 예는 62 페이지의 『ROW 블록』을 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

문맥

REPORT 블록은 다음의 컨텍스트에서 찾을 수 있습니다.

- FUNCTION 명령문 또는 블록
- MACRO_FUNCTION 블록

제한

REPORT 블록에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록
- IF 블록

- INCLUDE 명령문
- INCLUDE_URL 명령문
- ROW 블록
- WHILE 블록
- 함수 호출

OS/390, OS/2, Windows NT 및 UNIX 운영 체제의 경우: SQL 함수를 SQL 함수 내에서 호출할 수 없습니다.

- HTML 명령문
- 문자열
- 변수 참조

OS/390의 경우: REPORT 블록은 MACRO_FUNCTION 블록에서 허용되지 않습니다.

예

예 1: 이름 및 위치의 목록을 나타내는 두 컬럼으로 구성된 HTML 테이블

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

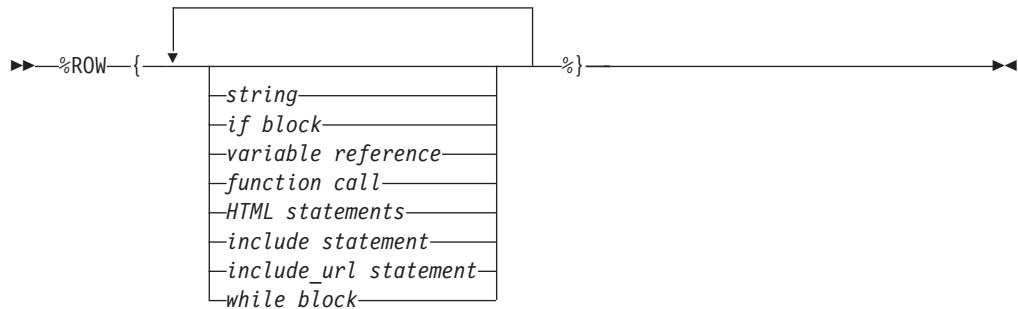
테이블에 있는 이름을 선택하면 *name.mac* Net.Data 매크로의 *details* HTML 블록이 호출되어 URL의 일부인 두 값에 전송됩니다. 이 예에서, 이름에 대한 좀 더 자세한 정보를 얻으려면 *name.mac*에 값을 사용할 수 있습니다.

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 형식화 자료를 지정하는 키워드.

string

영문자, 숫자 및 구두점의 임의 조합.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수를 나타내는 문자열이고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다. 구문 및 예는 35 페이지의 『IF 블록』을 참조하십시오.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

HTML statements

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

include_url statement

INCLUDE_URL 명령문. 다른 파일을 읽어 명령문이 지정된 Net.Data 매크로에 통합합니다. 지정된 파일은 국지 또는 원격 서버에 존재할 수 있습니다. 구문 및 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

문맥

ROW 블록은 다음의 컨텍스트에서 찾을 수 있습니다.

- REPORT 블록

제한

ROW 블록에는 다음의 요소가 포함될 수 있습니다.

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

OS/390, OS/2, Windows NT 및 UNIX 운영 체제의 경우: SQL 함수를 SQL 함수 내에서 호출할 수 없습니다.

- 변수 참조
- HTML 명령문
- 문자열

예

예 1: 이름 및 위치의 목록을 나타내는 두 컬럼으로 구성된 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>
%}
```

테이블에 있는 이름을 선택하면 *name.mac* Net.Data 매크로의 *details* HTML 블록이 호출되어 URL의 일부인 두 값에 전송됩니다. 이 예에서, 이름에 대한 좀 더 자세한 정보를 얻으려면 *name.mac*에 값을 사용할 수 있습니다.

TABLE 명령문

목적

관련 자료 집합인 변수를 정의합니다. 여기에는 동일한 레코드 배열, 행 및 각 행의 필드를 설명하는 컬럼 이름의 배열이 포함됩니다. 테이블 명령문은 DEFINE 명령문이나 블록에서만 정의할 수 있습니다.

구문

▶▶%TABLE 상한

상한:

(number)
ALL

값

%TABLE

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

상한

테이블에 포함시킬 수 있는 행 수. 상한 값이 지정되지 않으면, 테이블에는 수에 관계없이 행을 포함시킬 수 있습니다.

number

0에서 9까지의 값을 갖는 숫자 문자열. 값 0은 행을 제한없이 테이블에 포함시킬 수 있도록 합니다.

ALL

테이블내의 행의 수를 제한하지 않는 키워드.

문맥

TABLE 명령문은 다음의 컨텍스트에서 찾을 수 있습니다.

- DEFINE 명령문

제한

TABLE 명령문에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록
- 숫자

예

예 1: 상한이 30행인 Net.Data 테이블

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

예 2: 생략시 값(모든 행)을 사용하는 Net.Data 테이블

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

예 3: 모든 행을 지정하는 Net.Data 테이블

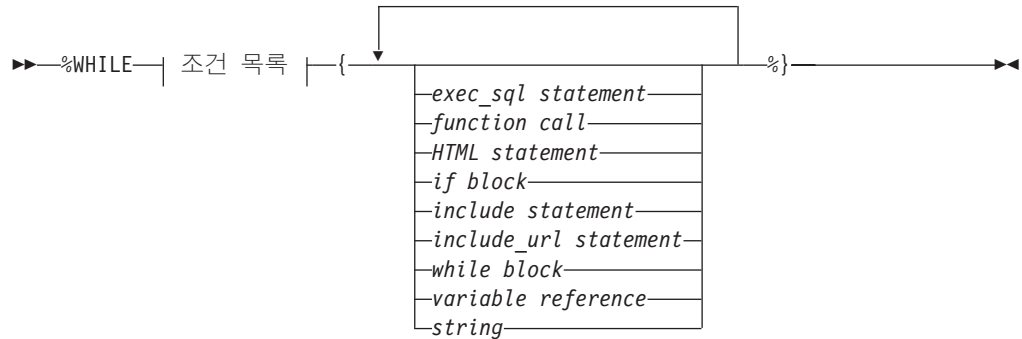
```
%DEFINE myTable3=%TABLE(ALL)
```


WHILE 블록

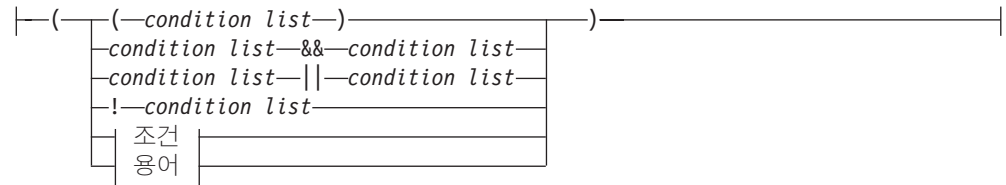
목적

조건부 문자열 처리에 기초하여 루핑 구문을 제공합니다. WHILE 블록은 HTML 블록, REPORT 블록, ROW 블록, IF 블록 및 MACRO_FUNCTION 블록에 사용할 수 있습니다. 조건 목록의 문자열 값은 이것이 정수를 나타내는 문자열이고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다.

구문



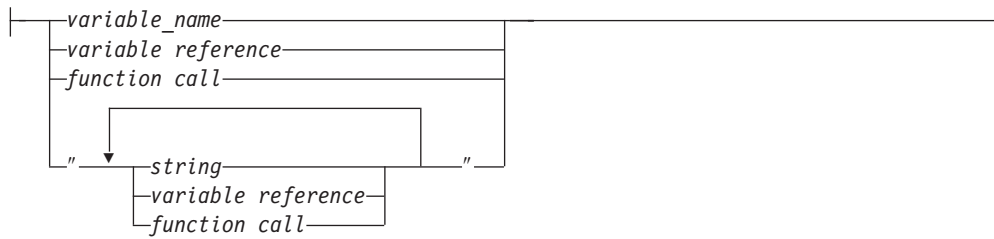
조건 목록:



조건:



용어:



값

%WHILE

루프 처리를 지정하는 키워드.

condition list

조건의 값을 비교합니다. 조건 목록은 부울 연산자를 사용하여 연결할 수 있습니다. 조건 목록은 다른 조건 목록내에 중첩될 수 있습니다.

condition

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

- 조건 연산자가 연산자 <, <=, >, >=, ==, != 중 하나입니다.
- 두 용어가 유효한 정수를 나타내는 문자열입니다. 여기서 유효한 정수란 플러스 (+) 또는 마이너스(-) 기호가 선행할 수도 있는 숫자로 된 문자열이며 다른 공백이 없는 경우를 말합니다.

참이 아닌 조건이 있는 경우, 일반적인 문자열 비교가 수행됩니다.

term

변수명, 문자열, 변수 참조 또는 함수 호출.

exec_sql statement

호환성을 위해 지원되는 DB2WWW 릴리스 1 언어 요소. 281 페이지의 『부록B. DB2 WWW 연결』 또는 DB2 World Wide Web 릴리스 1 책자를 참조하십시오.

function call

하나 이상의 FUNCTION 또는 MACRO_FUNCTION 블록이나 인수가 지정된 내장 함수를 호출합니다. 구문 및 예는 28 페이지의 『함수 호출 (@)』을 참조하십시오.

HTML statement

클라이언트의 브라우저에 적합하게 형식화할 HTML 태그뿐만 아니라 영문자 또는 숫자 문자가 포함됩니다.

if block

IF 블록. 조건부 문자열 처리를 수행합니다. 조건 목록의 문자열 값은 이것이 정수

를 나타내고 선행 또는 후미 공백이 없는 경우 비교를 위해 숫자로 취급됩니다. 여기에는 하나의 플러스(+) 또는 마이너스(-) 기호가 선행할 수 있습니다. 구문 및 예는 35 페이지의 『IF 블록』을 참조하십시오.

include statement

INCLUDE 명령문. 파일을 읽어 Net.Data 매크로에 통합시킵니다. 구문 및 예는 42 페이지의 『INCLUDE 명령문』을 참조하십시오.

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 예는 45 페이지의 『INCLUDE_URL 명령문』을 참조하십시오.

while block

WHILE 블록. 조건부 문자열 처리와 더불어 루핑을 처리합니다. 구문 및 예는 67 페이지의 『WHILE 블록』을 참조하십시오.

variable reference

변수의 값을 리턴하며, \$ 및 ()으로 지정됩니다. 예를 들어, VAR='abc'일 경우, \$(VAR)는 값 'abc'를 리턴합니다. 구문에 대해서는 5 페이지의 『변수 참조』를 참조하십시오.

string

영문자, 숫자 및 구두점의 임의 조합. 조건 목록의 용어에 들어 있는 문자열은 개행 문자를 제외한 모든 문자를 포함할 수 있습니다.

variable name

하나 이상의 이름, 추가되는 각 이름은 점(.)으로 결합됩니다. 구문에 대해서는 5 페이지의 『변수명』을 참조하십시오.

문맥

WHILE 블록은 다음의 컨텍스트에서 찾을 수 있습니다.

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

제한

WHILE 블록에는 다음의 요소가 포함될 수 있습니다.

- 주석 블록

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

예

예 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)
  </TR>

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

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

제2장 변수

Net.Data는 사용자 정의 변수와 Net.Data 변수를 제공합니다.

72 페이지의 『사용자 정의 변수』

사용자가 응용프로그램에 대해 정의하는 변수. 다음과 같은 작업을 수행하는 변수를 정의할 수 있습니다.

- 72 페이지의 『조건 변수』

다른 변수나 문자열의 값에 따라 변수 값을 설정합니다.

- 74 페이지의 『환경 변수』

ENVVAR 언어 구문을 사용하여 환경 변수를 참조합니다.

- 74 페이지의 『실행가능 변수』

EXEC 언어 구문을 사용하여 변수 참조로부터 다른 프로그램을 호출합니다.

- 75 페이지의 『숨겨진 변수』

HTML 소스에 변수 참조를 숨깁니다.

- 76 페이지의 『목록 변수』

LIST 언어 구문을 사용하여 분리된 값의 문자열을 구축합니다.

- 77 페이지의 『테이블 변수』

값 배열을 함수에 송수신합니다. 보고서 출력에는 사용할 수 없습니다.

Net.Data 변수

여러 가지 처리, 파일 조작, 테이블 처리, 보고서 포매팅 및 언어 환경을 위한 변수.

일부 변수의 값은 사용자가 정의 또는 수정할 수 있으며 또다른 변수는 Net.Data에 의해 정의됩니다. 변수에 대한 설명은 값을 정의할 수 있는지의 여부를 나타냅니다. 값을 정의하는 방법에 대해서는 변수의 설명을 참조하십시오.

Net.Data는 다음과 같은 유형의 변수를 제공합니다.

- 79 페이지의 『Net.Data 테이블 처리 변수』

Net.Data에 의해 정의되며 Net.Data 테이블을 처리할 수 있도록 합니다. SQL 조회와 함수 호출로 얻어진 자료에 액세스하려면 이 변수를 사용하십시오. 이 변수는 별다른 언급이 없는 한 REPORT 또는 ROW 블록 내에서만 인식됩니다.

- 89 페이지의 『Net.Data 보고서 변수』

함수를 통해 얻어진 보고서를 조정할 수 있도록 합니다. 임의의 Net.Data 매크로 블록에 보고서 변수를 정의하거나 참조할 수 있습니다.

- 98 페이지의 『Net.Data 언어 환경 변수』

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

- 118 페이지의 『Net.Data 기타 변수』

Net.Data에 의해 정의되어, Net.Data의 처리에 영향을 주고 함수 호출의 상태를 알아내며 데이터베이스 조회의 결과 세트에 대한 정보를 얻습니다. 일부 변수는 Net.Data에 의해 정의되며 변경할 수 없습니다.

대다수 Net.Data 변수의 출력은 수행되는 운영 체제에 따라 달라집니다.

Net.Data 매크로에서 상수는 최대 256KB가 될 수 있습니다. 따라서, 매크로에서는 길이가 256KB 이상인 변수를 초기화하거나 기본값을 설정할 수 없습니다.

이 장에는 각 변수에 지원되는 운영 체제가 명시되어 있습니다. 다음 목록에는 운영 체제의 약어가 정의되어 있습니다.

HP-UX	Hewlett Packard UNIX 운영 체제
SCO	Santa Cruz Operation OpenServer
SUN	Solaris 운영 체제
Win NT	Microsoft Windows NT 운영 체제

사용자 정의 변수

여기서는 사용자 정의 변수에 대해 설명합니다. 매크로 내에 이 변수를 정의하십시오.

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 삼원 조작이라고도 합니다.

조건 변수의 구문은 다음과 같습니다.

`test ? trueValue : falseValue`

여기서,

test 테스트할 조건.

trueValue

테스트가 참일 때 사용할 값.

falseValue

테스트가 거짓일 때 사용할 값.

예 1: 두 개의 가능한 값이 정의된 조건 변수

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

varB가 존재하면 varA=value_1, 존재하지 않으면 varA=value_2.

예 2: 변수 참조를 사용하여 정의된 조건 변수

```
varname = ? "$(value_1)"
```

이 경우 *value_1*이 널(NULL)이면, *varname*은 널(NULL)이고 그렇지 않으면 *varname*은 *value_1*로 설정됩니다.

예 3: LIST 명령문 및 WHERE 절이 사용된 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 proddtable $(where_clause)
%}
```

조건 변수와 LIST 변수가 함께 사용될 때 가장 효과적입니다. 위의 예는 DEFINE 블록에 WHERE 절을 설정하는 방법을 보여줍니다. 변수 *cust_inp*와 *prod_inp*는 웹 브라우저, 보통 HTML 양식으로부터 전달된 HTML 입력 변수입니다. 변수 *where_list*는 각각 웹 브라우저로부터 전달된 변수를 담고 있는 두 개의 조건 명령문으로 구성된 LIST 변수입니다.

웹 브라우저가 변수 *cust_inp* 및 *prod_inp*에 대한 값을 리턴하는데, IBM과 755C을 예로 들면, *where_clause*는 다음과 같습니다.

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

변수 *cust_inp* 또는 *prod_inp*가 널(NULL)이거나 정의되지 않은 경우, WHERE 절은 널(NULL)을 생략하도록 변경됩니다. 예를 들어, *prod_inp*가 널(NULL)이면 WHERE 절은 다음과 같이 변경됩니다.

```
WHERE custid = IBM
```

두 값이 널(NULL)이거나 정의되어 있지 않으면, 변수 *where_clause*는 널(NULL)이 되고 \$가 들어 있는 SQL 조회에는 WHERE 절이 나타나지 않습니다(*where_clause*).

환경 변수

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

환경 변수는 Net.Data ENVVAR 언어 구문을 사용하여 Net.Data가 수행되는 프로세스에 존재하는 환경 변수를 참조할 수 있도록 합니다.

예 1: 환경 변수의 값이 변수에 지정됩니다.

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

...

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

환경 변수 `SERVER_NAME`은 현재의 서버 이름을 값으로 갖는데, 이 예에서는 `www.software.ibm.com`입니다.

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

ENVVAR 명령문에 대한 자세한 내용은 16 페이지의 『ENVVAR 명령문』을 참조하십시오.

실행가능 변수

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

실행가능 변수는 실행가능 변수 기능을 이용하여 변수 참조에서 다른 프로그램을 호출할 수 있도록 합니다. 실행가능 변수는 EXEC 언어 요소를 사용하여 Net.Data 매크로에 정의됩니다. EXEC 언어 요소에 대한 자세한 내용은 17 페이지의 『EXEC 블록 또는 명령문』을 참조하십시오.

Net.Data는 매크로에서 실행가능 변수를 발견하면, 다음의 방법으로 참조된 실행가능 프로그램이 있는지 조회합니다.

1. Net.Data 초기화 파일에서 EXEC_PATH를 탐색합니다. EXEC_PATH에 대한 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장을 참조하십시오.
2. 프로그램을 찾지 못할 경우 Net.Data는 시스템에서 정의한 디렉토리를 탐색합니다. 실행가능 프로그램을 찾을 경우, Net.Data는 그 프로그램을 수행합니다.

예 1: 실행가능 변수 정의

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

변수 `runit`는 실행가능 프로그램 `testProg`를 실행하도록 정의되어 있습니다. `runit`이 실행가능 변수가 됩니다.

Net.Data는 Net.Data 매크로에서 실행가능 변수 참조를 발견하면 실행가능 프로그램을 수행합니다. 예를 들어, Net.Data 매크로에서 변수 *runit*에 대해 실행가능 변수 참조가 만들어지면 프로그램 *testProg*가 실행됩니다.

간단하게는 또다른 변수 정의에서 실행가능 변수를 참조하는 방법이 있습니다. 예 2는 이 방법을 보여줍니다. 변수 *date*가 실행가능 변수로 정의되고 *dateRpt*는 실행가능 변수를 포함하는 변수 참조로 정의됩니다.

예 2: 변수 참조로서의 실행가능 변수

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

Net.Data가 변수 참조 *\$(dateRpt)*를 해석하면, Net.Data는 실행가능 날짜를 찾아 프로그램을 실행하고 다음을 리턴합니다.

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

실행가능 변수는 절대 자신이 호출한 실행가능 프로그램의 출력 값으로 설정되지는 않습니다. 위의 예를 보면 날짜 값은 널(NULL)입니다. DTW_ASSIGN 함수 호출에서 이를 사용하여 이의 값을 다른 변수에 지정하는 경우, 지정 후 새로운 변수의 값도 역시 널(NULL)이 됩니다. 실행가능 변수의 목적은 자신이 정의한 프로그램을 호출하는 것입니다.

또한 변수 정의에 프로그램 이름과 매개변수를 함께 지정하여 실행될 프로그램에 매개변수를 전달할 수도 있습니다.

예 3: 매개변수가 있는 실행가능 변수

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

distance 및 *time*의 값이 프로그램 *calcMPH*에 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 이용하면 HTML 소스에 실제 변수 이름이 숨겨져 있더라도 변수를 참조할 수 있습니다. 숨겨진 변수를 사용하려면 다음과 같이 하십시오.

1. 숨기고자 하는 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.

예 1: HTML 양식의 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 양식이 웹 브라우저에 나타날 경우, $$(name)$ 및 $$(addr)$ 는 각각 $$(name)$ 와 $$(addr)$ 로 대체되어, 실제 테이블 및 컬럼명이 HTML 양식에 나타나지 않으므로 실제 변수명이 숨겨져 있는지는 아무도 모릅니다. 고객이 양식을 제출하면, HTML(REPORT) 블록이 호출됩니다. @mySelect()가 FUNCTION 블록을 호출하면 SQL문에서 $$(Field)$ 는 SQL 조회의 customer.name 또는 customer.addr로 대체 됩니다.

목록 변수

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

목록 변수를 사용하여 분리된 값의 문자열을 구축할 수 있습니다. 이들 중 일부는 일부 WHERE 또는 HAVING 절에서 발견된 것과 같은 복수의 항목을 사용하여 SQL 조회를 구축하는 데 유용합니다.

공백은 중요합니다. 값의 양쪽에 공백을 두어 값을 분리하는 것일 보통입니다. 대부분의 조회에서는 부울 또는 산술 연산자를 사용합니다(예: AND, OR 및 >). 구문 및 자세한 내용은 48 페이지의 『LIST 명령문』을 참조하십시오.

예 1: 조건적이고 숨겨져 있는 list 변수 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 양식에 체크 표시된 박스가 없으면, *conditions*은 널(NULL)이므로 조회시 *whereClause*도 널(NULL)이 됩니다. 그렇지 않으면, *whereClause*은 부울 연산자 OR 로 분리된 선택된 값을 갖습니다. 예를 들어, 세 도시가 모두 선택되었으면 SQL 조회는 다음과 같습니다.

```

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

```

예 2: 값 분리문자

```

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

```

이 예에서 테이블 처리 변수 VLIST는 두 개의 인용 부호와 OR 막대(|)를 값 분리 문자로 사용합니다. 문자열 값은 따옴표로 분리됩니다.

테이블 변수

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

테이블 변수에는 값의 배열 및 연관된 컬럼명이 들어 있습니다. 배열의 각 요소는 행입니다. 그룹 값을 함수에 전달하려면 테이블 변수를 사용하십시오. 함수의 REPORT 블록에서 테이블의 각 요소(행)를 참조할 수 있습니다. 테이블 변수는 종종 SQL 함수의 출력 및 보고서에 대한 입력에 사용되기도 하지만, 이들을 비SQL 함수에 IN, OUT 또

는 INOUT 매개변수로 전달할 수도 있습니다. 테이블은 SQL 함수에 OUT 매개변수로 전달될 수도 있습니다. 구문 및 자세한 내용은 65 페이지의 『TABLE 명령문』을 참조하십시오.

예 1: REXX 프로그램에 전달되는 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
}%

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

HTML REPORT 블록은 SQL 조회를 호출하고 결과를 테이블 변수에 저장한 다음 그 변수를 REXX 함수에 전달합니다.

Net.Data 테이블 처리 변수

Net.Data는 별다른 언급이 없는 한 이들 변수를 REPORT 및 ROW 블록에 사용하도록 정의합니다. 조회 결과 리턴된 값을 참조하려면 이 변수를 사용하십시오.

제한사항: DEFINE 섹션에는 이들 변수의 값을 정의하지 마십시오.

- 80 페이지의 『*Nn*』
- 81 페이지의 『*NLIST*』
- 82 페이지의 『*NUM_COLUMNS*』
- 83 페이지의 『*NUM_ROWS*』
- 84 페이지의 『*ROW_NUM*』
- 85 페이지의 『*TOTAL_ROWS*』
- 86 페이지의 『*V_columnName*』
- 88 페이지의 『*Vn*』
- 87 페이지의 『*VLIST*』

Nn

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

목적

컬럼 n 에 대한 조회나 함수 호출로부터 리턴된 컬럼명.

REPORT 및 ROW 블록에서 Nn 을 참조할 수 있습니다.

이들 변수는 사전 정의된 변수로 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하여 참조하려는 컬럼의 번호를 지정하십시오.

예

예 1: 컬럼명에 대한 변수 참조

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

예 2: DET_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 the first column is $(col1)
%}
```

이 예는 DTW_ASSIGN을 사용하여 REPORT 블록의 외부에 이 변수를 사용하는 방법을 보여줍니다. 자세한 내용은 176 페이지의 『DTW_ASSIGN』을 참조하십시오.

예 3: 컬럼명을 정의하기 위한 HTML 테이블내의 Nn

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

%}
```

NLIST

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

목적

함수 호출이나 조회의 결과로부터 얻어진 모든 컬럼명의 목록이 들어 있습니다. 생략시 분리문자는 공백입니다.

REPORT 및 ROW 블록에서 NLIST를 참조할 수 있습니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 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 ALIGN을 YES로 지정하면 컬럼명들 사이에 공백을 사용합니다.

예 2: 분리문자를 " | "로 변경하기 위한 %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

report 블록에서 Net.Data가 처리되고 있는 테이블 컬럼의 번호. 컬럼은 함수 호출이나 조회에 의해 리턴됩니다.

REPORT 및 ROW 블록에서 NUM_COLUMNS를 참조할 수 있습니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 1: NLIST에서 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

REPORT 블록에서 Net.Data가 처리되는 테이블의 행 수. 행 수는 자료가 들어 있는 Net.Data 테이블에 정의된 *upper limit* 매개변수 값의 영향을 받습니다. 예를 들어 *upper limit*가 30으로 설정되었는데 SELECT 명령문이 1000행을 리턴하는 경우, NUM_ROWS의 값은 30입니다. 또한, *upper limit*가 30으로 설정되었고 SELECT 명령문이 20행을 리턴하는 경우, NUM_ROWS는 20입니다. TABLE 명령문과 *upper limit* 매개변수에 대한 자세한 내용은 65 페이지의 『TABLE 명령문』을 참조하십시오.

NUM_ROWS는 START_ROW_NUM이 언어 환경에 전달되지 않는 한 START_ROW_NUM 값의 영향을 받지 않습니다. 예를 들어 START_ROW_NUM이 5(웹 페이지에 표시되는 테이블이 행 5에서 시작됨을 나타냄)로 설정되었고 SELECT 명령문이 25행을 리턴하는 경우, NUM_ROWS는 21이 아닌 25로 설정됩니다. 처음 4행은 테이블에서 없어지지만 NUM_ROWS의 값에 포함됩니다. 그러나 START_ROW_NUM이 환경 변수에 전달되면, NUM_ROWS에는 START_ROW_NUM이 지정한 행 이후의 행들만 포함됩니다. 위 예에서 NUM_ROWS는 21로 설정됩니다.

REPORT 및 ROW 블록에서 NUM_ROWS를 참조할 수 있습니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 1: REPORT 블록에서 처리되고 있는 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 값을 증가시키는 테이블 변수. 변수는 계수기로 작용하며 이의 값은 처리중인 현재 행의 번호입니다.

RPT_MAX_ROWS는 ROW_NUM의 값에 영향을 줄 수 있습니다. 예를 들어, 테이블에 100행이 있고 RPT_MAX_ROWS를 20으로 설정하면 행 20이 마지막으로 처리된 행이기 때문에 ROW_NUM의 최종 값은 20이 됩니다.

ROW 블록내에서만 ROW_NUM을 참조할 수 있습니다.

이 변수는 사전 정의된 변수이며, 이의 값을 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 1: 테이블의 각 행에 레이블을 표시하기 위해 ROW_NUM을 사용하여 HTML 출력에 컬럼을 이식합니다.

```
%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 아래 보인 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	X

목적

TABLE 언어 구문에 대한 *upper_limit* 값에 관계없이 조회를 통해 리턴되는 총 행 수. 예를 들어, RPT_MAX_ROWS가 최대 20행을 표시하도록 설정되어 있는데 조회를 통해 100행이 리턴되면, 이 변수는 ROW 처리후에 100으로 설정됩니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

운영 체제상의 차이:

- OS/400 운영 체제에서, 이 변수는 REPORT 또는 ROW 블록의 임의의 지점에서 참조할 수 있습니다.
- OS/390, OS/2, Windows NT 및 UNIX 운영 체제에서 이 변수는 REPORT 꼬리 말에서만 참조할 수 있습니다.

언어 환경 제한사항: 다음의 데이터베이스 언어 환경에서만 이 변수를 사용하십시오.

- SQL
- ODBC
- Oracle
- Sybase

필수: 이 변수를 사용하려면 DTW_SET_TOTAL_ROWS를 YES로 설정해야 합니다. 자세한 내용은 108 페이지의 『DTW_SET_TOTAL_ROWS』를 참조하십시오.

예

예 1: 발견된 총 행 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. 정의되지 않은 컬럼명에 대해서는 변수가 설정되지 않습니다. 동일한 이름을 가진 두개의 컬럼명이 들어 있는 조회에서는 예상치 않은 결과가 나타납니다. 중복된 컬럼명의 이름을 변경하려면 SQL에 AS 절을 사용하도록 하십시오.

ROW 블록내에서만 V_columnName을 참조할 수 있습니다.

이들 변수는 사전 정의된 변수로 값을 수정할 수 없습니다. 이 변수를 변수 참조로 사용하여 참조하려는 컬럼의 이름을 지정하십시오.

값

V_columnName

표 1. V_columnName 값

값	설명
columnName	데이터베이스 테이블의 현재 행의 컬럼명.

예

예 1: V_columnName을 변수 참조로 사용

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

%ROW{

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

VLIST

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

목적

ROW 블록내에서 처리중인 현재 행의 모든 필드 값 목록.

ROW 블록내에서만 VLIST를 참조할 수 있습니다. 생략시 분리문자는 공백입니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 1: list 태그를 사용하여 조회 결과 표시

```
%DEFINE ALIGN="YES"

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

예 2: list 변수를 사용하여 분리문자를 <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

현재 행에 대해 지정된 컬럼 번호 n 의 값.

ROW 블록내에서만 Vn 을 참조할 수 있습니다.

Net.Data는 각 필드에 대한 변수를 테이블에 지정하고 그 변수를 변수 참조에 사용하여, 참조하고자 하는 필드의 번호를 지정합니다. 블록 외부에서 이 변수를 사용하려면, Vn 의 값을 이전에 정의된 전역 변수나 OUT 또는 INOUT 함수 매개변수 값에 지정하십시오.

예

예 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.
%}
```

두번째 컬럼은 전자 메일 주소를 나타냅니다. 링크를 클릭하여 사람에게 메시지를 송신할 수 있습니다.

Net.Data 보고서 변수

이 변수를 사용하여 보고서를 조정할 수 있습니다. 각 변수에는 기본값이 있습니다. 변수에 새로운 값을 할당하여 기본값을 대체할 수 있습니다.

- 90 페이지의 『ALIGN』
- 91 페이지의 『DTW_DEFAULT_REPORT』
- 92 페이지의 『DTW_HTML_TABLE』
- 93 페이지의 『RPT_MAX_ROWS』
- 95 페이지의 『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 VLIST에 사용되는 선행 및 후미 공백을 제어합니다.

성능 향상: ALIGN을 사용하면, Net.Data는 채움 요건을 계산하기 위해 테이블의 모든 컬럼에 대한 최대 컬럼 길이를 파악해야 하므로 필요할 경우에만 ALIGN을 사용하십시오. 이 프로세스는 성능에 영향을 줄 수 있습니다.

YES로 설정되면, ALIGN은 표시장치에 맞춰 테이블 처리 변수를 조정하기 위해 채움 공간을 제공합니다. HTML 링크나 양식 조치에 조회 결과를 내포시키려면, 생략시 값인 NO를 사용하여 Net.Data가 보고서 변수 주변에 선행 및 후미 공백을 두지 않도록 하십시오.

DEFINE문을 사용하거나 @DTW_ASSIGN() 함수를 사용하여 이 변수의 값을 지정 하십시오.

값

ALIGN="YES"|"NO"

표 2. ALIGN 값

값	설명
YES	Net.Data는 표시장치에 맞게 정렬하기 위해 보고서 변수에 선행 및 후미 공백을 추가합니다.
NO	Net.Data는 선행 및 후미 공백을 추가하지 않습니다. NO가 생략시 값입니다.

예

예 1: ALIGN 변수를 사용하여 공백으로 각 컬럼을 분리

```
%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

Net.Data가 REPORT 블록이 없는 함수에 대해 생략시 보고서를 생성할지 여부를 결정합니다. 이 변수가 YES로 설정되면, Net.Data는 생략시 보고서를 생성합니다. NO로 설정되면, Net.Data는 생략시 보고서를 생성하지 않습니다. 테이블 변수로 함수 호출의 결과를 수신하고 그 결과를 다른 함수에 전달하여 처리하는 경우, 이와 같이 생략시 보고서를 생성하지 않는 것이 좋습니다.

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

값

DTW_DEFAULT_REPORT="YES" | "NO"

표 3. DTW_DEFAULT_REPORT 값

값	설명
YES	Net.Data는 REPORT 블록이 없는 함수에 대해 생략시 보고서를 생성하고 그 결과를 브라우저에 표시합니다. YES가 생략시 값입니다.
NO	Net.Data는 REPORT 블록이 없는 함수에 대해 생략시 보고서를 생성하지 않습니다.

예

예 1: Net.Data가 생성한 생략시 보고서 대체

```
%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 형식으로 테이블을 표시하는 대신 결과를 HTML 테이블로 표시합니다(즉, PRE 태그 대신 TABLE 태그를 사용).

생성된 TABLE 태그에는 경계와 셀 패딩 스펙이 포함되어 있습니다.

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

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

값

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

표 4. 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

예 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

테이블에서 함수 REPORT 블록이나 REPORT 블록이 지정되지 않은 경우에는 기본 보고서의 생성 중에 처리되는 테이블 안에서 행의 수를 지정합니다.

데이터베이스 언어 환경은 이 변수를 사용하여 리턴되는 행의 수를 제한하므로 결과 세트가 큰 경우에 성능을 상당히 향상시킵니다. 결과 세트가 큰 조회들을 각각 자체의 HTML 페이지에 위치하는 보다 작은 테이블들로 나누려면, 이 변수를 START_ROW_NUM과 함께 사용하십시오.

OS/400, Windows NT, OS/2 및 UNIX 사용자: 이 변수를 언어 환경으로 전달하려면, Net.Data 초기화 파일에 있는 데이터베이스 언어 환경의 ENVIRONMENT 명령문에 IN 매개변수로 포함시키십시오. 데이터베이스 언어 환경 명령문에 대해 자세히 알려면, 운영 체제에 대한 *Net.Data* 관리 및 프로그래밍 안내서의 구성 장을 참조하십시오.

OS/390 사용자: RPT_MAX_ROWS는 매크로에 정의될 때 데이터베이스 언어 환경에 내재적으로 전달됩니다.

DEFINE 명령문이나 @DTW_ASSIGN() 함수를 사용하여 이 변수의 값을 지정하십시오.

값

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

표 5. RPT_MAX_ROWS 값

값	설명
ALL	함수 호출에 의해 생성되는 테이블에 표시될 행의 수에 제한이 없음을 나타냅니다. 모든 행이 표시됩니다.
0	테이블의 모든 행이 표시되도록 지정합니다. 이 값은 ALL을 지정하는 것과 동일합니다.
number	함수 호출에 의해 생성되는 테이블에 표시될 최대 행 수를 나타내는 양의 정수. FUNCTION 블록에 REPORT 및 ROW 블록이 포함되어 있는 경우, 이 수는 ROW 블록이 실행되는 횟수를 지정합니다.

예

예 1: DEFINE 명령문에 RPT_MAX_ROWS를 정의합니다.

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

위의 방법은 함수가 리턴하는 행의 수를 20행으로 제한합니다.

예 2: HTML 입력을 사용하여 HTML 양식으로 변수를 정의합니다.

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

위의 예에 있는 행들은 FORM 태그에 배치되어 응용프로그램 사용자가 조회를 통해 리턴하고자 하는 행의 수를 설정할 수 있도록 합니다.

START_ROW_NUM

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

목적

테이블에서 함수 REPORT 블록이나 REPORT 블록이 지정되지 않은 경우에는 기본 보고서의 생성 중에 처리되는 시작 행 수를 지정합니다.

데이터베이스 언어 환경은 처리를 시작하기 위해 이 변수를 사용하여 결과 세트에서 시작 행을 판별합니다. 큰 결과 세트의 경우에 성능을 상당히 향상시키려면, 이 변수를 RPT_MAX_ROWS와 함께 사용하여 큰 결과 세트가 있는 조회들을 보다 작은 테이블로 나누십시오.

OS/400, Windows NT, OS/2 및 UNIX 사용자: 이 변수를 언어 환경으로 전달하려면, Net.Data 초기화 파일에 있는 데이터베이스 언어 환경의 ENVIRONMENT 명령문에 IN 매개변수로 포함시키십시오. 데이터베이스 언어 환경 명령문에 대해 자세히 알려면, 운영 체제에 대한 *Net.Data* 관리 및 프로그래밍 안내서의 구성 장을 참조하십시오.

OS/390 사용자: START_ROW_NUM은 매크로에 정의될 때 데이터베이스 언어 환경에 내재적으로 전달됩니다.

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

값

START_ROW_NUM="number"

표 6. START_ROW_NUM 값

값	설명
<i>number</i>	<p>보고서 표시를 시작할 행 번호를 나타내는 양의 정수. 기본값은 1입니다.</p> <p>START_ROW_NUM이 초기화 파일에서 데이터베이스 언어 환경의 환경 명령문에 지정되면, 이 수는 데이터베이스 언어 환경에 의해 처리되는 결과 세트의 행 번호를 지정합니다.</p> <p>START_ROW_NUM이 언어 환경에 전달되지 않으면, 이 수는 보고서를 표시하는 데 사용되는 Net.Data 테이블의 행 번호를 지정합니다.</p>

예

예 1: 다음 및 이전 버튼으로 HTML 양식 화면이동

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

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

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

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

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

```

        @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

%{ 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

언어 환경에서 FUNCTION 블록을 처리하는 방법을 조정할 수 있도록 하려면 이 변수를 함수에 사용하십시오. 각 변수에는 기본값이 있습니다. 변수에 새로운 값을 할당하여 기본값을 대체할 수 있습니다.

- 99 페이지의 『DATABASE』
- 101 페이지의 『DB_CASE』
- 102 페이지의 『DB2PLAN』
- 103 페이지의 『DB2SSID』
- 104 페이지의 『DTW_APPLET_ALTTEXT』
- 105 페이지의 『DTW_EDIT_CODES』
- 106 페이지의 『DTW_MBMODE』
- 107 페이지의 『DTW_SAVE_TABLE_IN』
- 108 페이지의 『DTW_SET_TOTAL_ROWS』
- 110 페이지의 『LOCATION』
- 111 페이지의 『LOGIN』
- 112 페이지의 『NULL_RPT_FIELD』
- 113 페이지의 『PASSWORD』
- 114 페이지의 『SHOWSQL』
- 115 페이지의 『SQL_STATE』
- 116 페이지의 『TRANSACTION_SCOPE』

DATABASE

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

목적

데이터베이스 함수를 호출할 때 액세스할 데이터베이스나 ODBC 자료 소스를 지정합니다. 여러 데이터베이스 또는 ODBC 자료 소스에 액세스하기 위해 한 매크로내에서 이 변수를 여러 차례 변경할 수 있습니다.

OS/400 운영 체제: 이 변수는 생략가능합니다. Net.Data는 기본적으로 DATABASE=『*LOCAL』을 지정합니다. DTW_SQL 언어 환경은 국지 관계형 데이터베이스 디렉토리 항목을 사용합니다.

Windows NT, OS/2 및 UNIX 운영 체제: DTW_ORA(Oracle) 언어 환경을 사용하는 경우를 제외하고, 데이터베이스 함수를 호출하기 전에 이 변수를 정의하십시오. 또한, 동일한 HTML 블록에서 동일한 언어 환경을 통해 여러 데이터베이스에 액세스하는 경우 라이브 연결을 사용해야 합니다.

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

값

DATABASE="dbname"

표 7. DATABASE 값

값	설명
dbname	Net.Data가 연결하는 데이터베이스의 이름.

예

예 1: 임의의 SQL 조작에 대해 CELDIAL 데이터베이스에 연결하도록 지정합니다.

```
%DEFINE DATABASE="CELDIAL"

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

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

함수 getRpt가 호출되면 데이터베이스 CELDIAL에 액세스됩니다.

예 2: 이전의 DATABASE 정의를 DTW_ASSIGN로 대체합니다.

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

HTML 블록은 DATABASE에 대한 이전 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 문자(대문자 또는 소문자)를 지정하고 모든 문자를 대문자 또는 소문자로 변환합니다. 이 변수가 정의되지 않은 경우, 생략시 조치는 SQL 명령 문자를 변환하지 않는 것입니다.

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

값

DB_CASE="UPPER"|"LOWER"

표 8. DB_CASE 값

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

예

예 1: 모든 SQL 명령에 대해 소문자를 지정합니다.

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

DB2PLAN

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

목적

국지 DB2 부속시스템에 연결하기 위한 플랜을 할당합니다. 변수는 Net.Data가 액세스 할 국지 DB2 부속시스템에서 Net.Data SQL 언어 환경에 대한 플랜의 이름을 지정합니다.

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

요구사항: 매크로에서 이 변수의 값이 적용되려면, SQL 언어 환경에 대한 ENVIRONMENT 명령문에 나열되어야 합니다.

값

DB2PLAN="*plan_name*"

표 9. DB2PLAN 값

값	설명
<i>plan_name</i>	DB2 플랜의 이름. 이름은 최대 8문자가 될 수 있습니다.

예

예 1: DEFINE 명령문에 플랜을 지정합니다.

```
%DEFINE DB2PLAN="DTWGAV22"
```

DB2SSID

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

목적

국지 DB2 부속시스템에 대한 연결을 설정합니다. 변수는 Net.Data가 액세스할 국지 DB2 서브시스템의 서브시스템 ID를 지정합니다. 각 매크로당 하나의 국지 데이터베이스 연결만이 허용됩니다.

요구사항: 매크로에서 이 변수의 값이 적용되려면, SQL 언어 환경에 대한 ENVIRONMENT 명령문에 나열되어야 합니다.

값

```
DB2PLAN="subsytem_id"
```

표 10. DB2SSID 값

값	설명
<i>subsystem_id</i>	DB2 부속시스템의 이름. 이름은 최대 8문자가 될 수 있습니다.

예

예 1: DEFINE 명령문에 부속시스템 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 않는 브라우저에 대한 텍스트와 HTML 태그를 표시하고 애플릿 언어 환경에 사용됩니다.

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

값

DTW_APPLET_ALTTEXT="HTML_text_and_tags"

표 11. DTW_APPLET_ALTTEXT 값

값	설명
HTML_text_and_tags	APPLET 태그를 인식하지 않는 브라우저에 대한 텍스트와 HTML 태그.

예

예 1: 웹 브라우저의 제한사항을 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 결과 리턴된 NUMERIC, DECIMAL, INTEGER 및 SMALLINT 자료 유형을 변환합니다. 변수 DTW_EDIT_CODES는 DTW_SQL LE가 구축하는 테이블의 결과 컬럼에 해당하는 문자열입니다. 예를 들어, DTW_EDIT_CODES의 다섯번째 문자는 결과 세트의 다섯번째 컬럼에 적용됩니다(단, 이 컬럼이 지원되는 유형 중 하나인 경우). 이 단일 문자는 *Data Description Specification Reference*에 정의된 지원 시스템 제공 편집 코드 중 하나일 수 있습니다.

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

참고: 숫자가 아닌 문자로 된(예: 콤마 또는 통화 기호) 문자열을 만들어 내는 컬럼에 사용자 제공 편집 코드를 적용하면 그 문자열이 Net.Data 매크로내의 후속 처리시 서버에 다시 전송되는 경우 구문 오류를 초래할 수 있습니다. 예를 들어, 숫자가 아닌 컬럼 값이 후속되는 DTW_SQL 함수 호출에서 숫자 비교에 사용될 수도 있는데, 이 경우 구문 오류가 초래됩니다.

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

값

DTW_EDIT_CODES="edit_code"

표 12. DTW_EDIT_CODES 값

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

예

예 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 환경에 사용되는 문자열과 단어 함수에 다중 바이트 문자 세트(MBCS) 지원을 제공합니다. Net.Data 초기화 파일에 이 변수를 설정할 수 있지만 매크로에서 이를 사용하여 현재의 설정을 설정하거나 겹쳐쓸 수 있습니다.

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

OS/400을 사용하는 경우: OS/400용 Net.Data는 MBCS 지원을 위한 함수를 자동으로 작동시키며 이 변수를 필요로 하지 않습니다. OS/400용 Net.Data는 OS/400 운영 체제로 이주된 매크로에서 이 변수를 무시합니다.

값

DTW_MBMODE="YES"|"NO"

표 13. DTW_MBMODE 값

값	설명
YES	문자열 및 단어 함수에 대해 MBCS 지원을 지정합니다.
NO	문자열 및 단어 함수에 MBCS 지원이 되지 않도록 지정합니다. NO가 생략시 값입니다.

예

예 1: INI 파일의 값을 대체합니다.

INI 파일:

DTW_MBMODE NO

매크로:

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


DTW_SAVE_TABLE_IN

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

목적

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

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

값

DTW_SAVE_TABLE_IN="table_name_var"

표 14. DTW_SAVE_TABLE_IN 값

값	설명
table_name_var	조회로부터 리턴된 테이블 자료를 저장하는 SQL 언어 환경의 테이블명.

예

예 1: REXX 호출에 사용된 사전 정의된 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 자료 분석시 테이블 변수 theTable을 사용하는 REXX 프로그램 anzTbl.cmd를 호출합니다. 변수 theTable은 이전의 SQL 함수 호출로부터 리턴되었습니다.

DTW_SET_TOTAL_ROWS

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

목적

조회에 대한 결과 세트의 행 수가 TOTAL_ROWS에 지정되어야 함을 데이터베이스 언어 환경에 지정합니다.

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

OS/400, OS/2, Windows NT 및 UNIX 사용자: 이 변수를 언어 환경으로 전달하려면, Net.Data 초기화 파일에 있는 데이터베이스 언어 환경의 ENVIRONMENT 명령문에 IN 변수로 포함시키십시오. 데이터베이스 언어 환경 명령문에 대한 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장을 참조하십시오.

OS/390 사용자: DTW_SET_TOTAL_ROWS는 매크로에 정의될 때 데이터베이스 언어 환경에 내재적으로 전달됩니다.

성능 향상: DTW_SET_TOTAL_ROWS를 YES로 설정하면, 총 행 수 및 데이터베이스 언어 환경을 판별하기 위해 모든 행을 검색해야 하기 때문에 성능에 영향을 줄 수 있습니다.

값

DTW_SET_TOTAL_ROWS="YES"|"NO"

표 15. DTW_SET_TOTAL_ROWS 값

값	설명
YES	총 행 수 값을 TOTAL_ROWS 변수에 지정합니다. 중요: 조회로부터 리턴된 행 수를 판별하기 위해 변수 TOTAL_ROWS를 참조하고자 하는 경우 이 값을 설정해야 합니다.
NO	Net.Data는 TOTAL_ROWS 변수를 설정하지 않으며 TOTAL_ROWS는 매크로에서 참조할 수 없습니다. NO가 생략시 값입니다.

예

예 1: TOTAL_ROWS를 사용하도록 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 서버에 대한 연결을 설정합니다. 이 변수는 국지 DB2 서브시스템이 원격 서버를 인식하는 데 사용하는 이름을 지정합니다. LOCATION의 값은 통신 데이터베이스(CDB)의 SYSIBM.SYSLOCATIONS 테이블내에 정의되어 있어야 합니다. 이 변수가 매크로내에 정의되지 않은 경우, 매크로에 의해 이루어지는 모든 SQL 요청은 국지 DB2 부속시스템에서 실행됩니다.

요구사항: 매크로에서 이 변수의 값이 적용되려면, SQL 언어 환경에 대한 ENVIRONMENT 명령문에 나열되어야 합니다.

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

값

LOCATION="remote_dbase_name"

표 16. LOCATION 값

값	설명
remote_dbase_name	CDB의 SYSIBM.SYSLOCATIONS 테이블에 정의된 유효한 원격 데이터베이스 서버의 이름. 이름은 최대 8문자가 될 수 있습니다.

예

예 1: DEFINE 명령문에 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

사용자 ID를 데이터베이스 언어 환경에 전달하여 보호되는 자료에 대한 액세스를 제공합니다. DB2의 보안 알고리즘을 합치려면 이 변수를 PASSWORD와 함께 사용하십시오.

OS/400을 사용하는 경우: OS/400은 DATABASE 변수가 정의되지 않았거나 이의 값이 `"*LOCAL"`로 설정된 경우 LOGIN 및 PASSWORD를 무시합니다. 데이터베이스 액세스는 Net.Data가 수행되는 사용자 프로파일을 통해 경로지정됩니다.

보안 관련사항: 이 값을 Net.Data 매크로에 코드화할 수도 있지만, 응용프로그램 사용자가 HTML 양식에 사용자 ID를 입력하는 것이 좋습니다. 또한, 생략시 값인 웹 서버 ID를 사용하면 사용자의 보안 요구에 부합되지 않는 액세스 레벨이 제공됩니다.

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

값

LOGIN=`database_user_id`

표 17. LOGIN 값

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

예

예 1: 사용자 ID, DB2USER로 액세스 제한

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

예 2: HTML 양식 입력 행 사용

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

이 예는 응용프로그램 사용자가 자신들의 사용자 ID를 입력하도록 HTML 양식의 일부로 포함시킬 수 있는 행을 보여줍니다.

NULL_RPT_FIELD

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

목적

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

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

값

NULL_RPT_FIELD="null_char"

표 18. NULL_RPT_FIELD 값

값	설명
null_char	SQL 결과 세트에 리턴된 NULL 값을 나타내기 위한 문자열을 지정합니다. 생략시 값은 공백 문자열입니다.

예

예 1: NULL 값을 나타내는 문자열을 SQL 언어 환경에 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

암호를 데이터베이스 언어 환경에 전달하여 보호되는 자료에 대한 액세스를 제공합니다. DB2의 보안 알고리즘을 합치려면 이 변수를 LOGIN과 함께 사용하십시오.

OS/400을 사용하는 경우: OS/400은 DATABASE 변수가 정의되지 않았거나 이의 값이 `"*LOCAL"`로 설정된 경우 LOGIN 및 PASSWORD를 무시합니다. 데이터베이스 액세스는 Net.Data가 수행되는 사용자 프로파일을 통해 경로지정됩니다.

보안 관련사항: 이 값을 Net.Data 매크로에 코드화할 수도 있지만, 응용프로그램 사용자가 HTML 양식에 암호를 입력하는 것이 좋습니다.

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

값

PASSWORD=`"password"`

표 19. PASSWORD 값

값	설명
<code>password</code>	데이터베이스 언어 환경에 대한 자동 액세스를 제공하기 위한 유효한 암호를 지정합니다.

예

예 1: 암호 NETDATA를 갖는 응용프로그램 사용자로 액세스 제한

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

예 2: HTML 양식 입력 행

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

이 예는 응용프로그램 사용자가 암호를 입력하도록 HTML 양식의 일부로 포함시킬 수 있는 행을 보여줍니다.

SHOWSQL

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

목적

웹 브라우저에서 사용된 조회의 SQL을 숨기거나 표시합니다. 테스트중에 SQL을 표시하면 Net.Data 매크로를 디버그할 때 특히 유용합니다. SHOWSQL은 Net.Data 구성 파일에서 DTW_SHOWSQL이 YES로 설정된 경우에만 사용할 수 있습니다. DTW_SHOWSQL 구성 변수에 대한 자세한 내용은 운영 체제에 대한 *Net.Data 관리 및 프로그래밍 안내서*에서 구성 장을 참조하십시오.

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

값

SHOWSQL="YES"|"NO"

표 20. SHOW_SQL 값

값	설명
YES	데이터베이스로 전송된 조회의 SQL을 표시합니다.
NO	데이터베이스로 전송된 조회의 SQL을 숨깁니다. NO가 생략시 값입니다.

예

예 1: 모든 SQL 조회를 표시합니다.

구성 파일에서:

DTW_SHOWSQL YES

매크로에서:

%DEFINE SHOWSQL="YES"

예 2: HTML 형식 입력을 사용한 SQL 표시 여부 지정

구성 파일에서:

DTW_SHOWSQL YES

매크로에서:

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

데이터베이스에서 리턴된 SQL 상태 값에 액세스하거나 이를 표시합니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 변수를 변수 참조로 사용하십시오.

예

예 1: REPORT 블록에 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 범위를 지정하여, 각각의 SQL 명령이 실행된 후 또는 HTML 블록의 모든 SQL 명령이 성공적으로 완료된 후 Net.Data가 COMMIT를 발행할지의 여부를 결정합니다. 확약 전에 모든 SQL 명령이 성공적으로 완료되어야 함을 지정하면, 성공적으로 실행되지 않은 SQL 명령이 있을 경우 그 블록의 동일한 데이터베이스에 대해 이전에 실행된 SQL이 구간 복원됩니다.

TRANSACTION_SCOPE 변수가 적용되려면, Net.Data 구성 파일의 ENVIRONMENT 명령문에 포함시켜야 합니다. 그런 다음, DEFINE 명령문이나 @DTW_ASSIGN() 함수를 사용하여 이 변수의 값을 지정할 수 있습니다.

일관성 관련사항: OS/400 및 OS/390 이외의 플랫폼에서 다음 모든 조건이 참이면, 실패 응답을 수신한 데이터베이스에 대한 갱신은 동일한 HTML 블록에서 액세스된 기타 데이터베이스에 대한 갱신이 확약되는 동안에 구간 복원될 수 있습니다.

- TRANSACTION_SCOPE = "MULTIPLE"가 지정되었습니다.
- 복수의 데이터베이스가 한 HTML 블록내에서 액세스됩니다 (이는 라이브 연결을 사용할 경우에 가능합니다).
- SQL 요청으로부터 실패한 응답이 수신되었습니다.

IBM의 DataJoiner 소프트웨어를 사용하거나 OS/400에 설치된 Net.Data로부터 여러 개의 데이터베이스에 액세스할 경우, Net.Data로부터 갱신할 때 여러 데이터베이스 갱신의 조정 및 일관성을 달성할 수 있습니다.

OS/400 및 OS/390에서, TRANSACTION_SCOPE = "MULTIPLE"는 하나의 HTML 블록에서 발행된 모든 IBM 데이터베이스 갱신이 함께 확약 또는 구간 복원되도록 합니다.

OS/400이외의 플랫폼에서, REXX, Perl 및 자바 언어 환경은 자체의 개별 운용 체제 프로세스에 따라 수행됩니다. 따라서, 이러한 언어 환경에서 발행한 데이터베이스 갱신은 Net.Data TRANSACTION_SCOPE 값과 상관없이 Net.Data 매크로에서 발행된 데이터베이스 갱신과는 별도로 확약되거나 구간 복원됩니다.

값

TRANSACTION_SCOPE="SINGLE" | "MULTIPLE"

표 21. TRANSACTION_SCOPE 값

값	설명
SINGLE	Net.Data는 HTML 블록의 각 SQL 명령이 성공적으로 완료된 후 COMMIT을 발행합니다.
MULTIPLE	Net.Data가 HTML 블록내의 모든 SQL 명령이 성공적으로 완료된 후에만 COMMIT을 발행하도록 지정합니다. MULTIPLE이 생략시 값입니다.

예

예 1: 각 트랜잭션 후 COMMIT을 발행하도록 지정합니다.

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

Net.Data 기타 변수

이들 변수는 Net.Data 정의 변수로, Net.Data 처리에 영향을 주고 함수 호출의 상태를 알아내며 데이터베이스 조회의 결과 세트에 대한 정보를 얻고 파일 위치와 날짜에 대한 정보를 판별하는 데 이를 사용할 수 있습니다. Net.Data 매크로를 테스트할 때 함수에 이 변수를 쓰거나 사용할 경우에 이 변수가 유용하다는 사실을 느낄 수 있습니다.

- 119 페이지의 『DTW_CURRENT_FILENAME』
- 120 페이지의 『DTW_CURRENT_LAST_MODIFIED』
- 121 페이지의 『DTW_DEFAULT_MESSAGE』
- 122 페이지의 『DTW_LOG_LEVEL』
- 123 페이지의 『DTW_MACRO_FILENAME』
- 124 페이지의 『DTW_MACRO_LAST_MODIFIED』
- 125 페이지의 『DTW_MP_PATH』
- 126 페이지의 『DTW_MP_VERSION』
- 127 페이지의 『DTW_PRINT_HEADER』
- 128 페이지의 『DTW_REMOVE_WS』
- 129 페이지의 『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

현재의 입력 파일에 대한 이름 및 확장자. 입력 파일은 INCLUDE문에 지정된 파일이거나 Net.Data 매크로입니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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 매크로이거나 INCLUDE 명령문에 지정된 파일입니다. 출력 형식은 Net.Data가 수행되는 시스템에 따라 다릅니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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	X	X		X	X	X	X

목적

오류 발생시, 호출에서 내장 함수나 언어 환경으로 리턴된 메세지 텍스트가 들어 있습니다.

Net.Data 매크로의 어느 부분에서나 DTW_DEFAULT_MESSAGE 변수를 사용할 수 있습니다.

이 변수는 사전정의된 변수로, 이의 값은 수정하지 마십시오. 변수를 변수 참조로 사용하십시오.

예

예 1: 함수가 성공적으로 완료되었는지의 여부를 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
```

예 2: 함수가 0이 아닌 리턴 코드를 리턴하는 경우의 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 리턴하는 경우 생략시 오류 메세지가 표시됩니다.

DTW_LOG_LEVEL

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

목적

Net.Data가 로그 파일에 기록하는 메세지 레벨.

DEFINE 명령문이나 @DTW_ASSIGN() 함수를 사용하여 이 변수의 값을 지정할 수 있습니다.

요구사항: 기록을 시작하도록 Net.Data 초기화 파일에 DTW_LOG_DIR을 정의하십시오. 그렇지 않으면, Net.Data는 매크로에 DTW_LOG_LEVEL 변수를 지정해도 메세지를 기록하지 않습니다.

값

DTW_LOG_LEVEL="OFF|ERROR|WARNING"

표 22. DTW_LOG_LEVEL 값

값	설명
OFF	Net.Data가 오류를 기록하지 않습니다. OFF가 생략시 값입니다.
ERROR	Net.Data가 오류 메세지를 기록합니다.
WARNING	Net.Data가 오류 메세지뿐 아니라 경고도 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

현재의 Net.Data 매크로 파일에 대한 이름 및 확장자.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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 날짜 및 시간. 출력 형식은 Net.Data가 수행되는 시스템에 따라 다릅니다.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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 실행가능 파일의 경로와 이름. 사용중인 시스템에 따라 다르지만, 출력은 다음과 같은 샘플 경로 및 이름과 유사합니다.

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

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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 수행중인 Net.Data의 버전 및 릴리스 번호.

이 변수는 사전 정의된 변수이며, 이의 값은 변경할 수 없습니다. 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

HTTP 머리말에 대한 텍스트를 지정합니다.

Net.Data는 텍스트를 출력하기 전에 이 변수를 한번 읽고 다시는 읽지 않기 때문에, Net.Data가 웹 브라우저로 전송된 텍스트를 처리하기 전에 이 변수를 설정해야 합니다. DTW_PRINT_HEADER 변수에 변경이 생겨도 Net.Data가 브라우저로 송신된 후에는 무시됩니다.

DTW_PRINT_HEADER를 사용하여 사용자 머리말을 생성하는 경우 (DTW_PRINT_HEADER = "NO"), DTW_REMOVE_WS를 "NO"로 설정해야 합니다.

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

값

DTW_PRINT_HEADER="YES"|"NO"

표 23. DTW_PRINT_HEADER 값

값	설명
YES	Net.Data는 HTTP 머리말로 텍스트 Content-type: text/html을 출력합니다. YES가 생략시 값입니다.
NO	Net.Data는 HTTP 머리말을 인쇄하지 않습니다. 사용자가 원하는 HTTP 머리말 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 설정되고, 처음 세 행은 Content-type 머리말, Set-Cookie 명령문 및 공백 행이 되어야 합니다.

예 1: Net.Data가 쿠키를 전송하도록 허용

```
%DEFINE DTW_PRINT_HEADER="NO"
```

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

```
<P>
Any text
%}
```

DTW_REMOVE_WS

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

목적

도표 작성기, 공백 및 개행 문자 등으로 인한 잉여 공간을 압축하여 동적으로 생성된 웹 페이지의 크기를 줄입니다.

DEFINE 블록에 이 변수의 값을 지정하십시오.

<PRE></PRE> 태그 사용: 이 변수를 YES로 정의하면 인쇄되는 공간의 양과 유형이 영향을 받습니다. 이 변수가 YES로 설정되면, <PRE></PRE> 태그를 사용하는 HTML 페이지의 부분은 표시되지 않습니다.

DTW_PRINT_HEADER를 사용하여 사용자 자신의 머리말을 생성하는 경우 (DTW_PRINT_HEADER = "NO"), DTW_REMOVE_WS를 "NO"로 설정해야 합니다.

OS/390을 사용하는 경우: 매크로 전체에 대해 값을 지정하려면 Net.Data 초기화 파일에 이 변수를 설정하십시오. 매크로에 이 값을 정의하여 값을 겹쳐쓸 수 있습니다. 매크로에 DTW_REMOVE_WS가 정의되어 있지 않으면, 초기화 파일의 값을 사용합니다.

값

DTW_REMOVE_WS="YES"|"NO"

표 24. DTW_REMOVE_WS 값

값	설명
YES	Net.Data는 연속된 두 개 이상의 공백을 하나의 개행 문자로 압축하여 좀 더 짧은 HTML 결과 페이지를 생성합니다.
NO	Net.Data가 공백을 압축하지 않습니다. NO가 생략시 값입니다.

예

예 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

함수에 의해 내장 함수나 언어 환경에 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.

RETURN_CODE 변수는 Net.Data 매크로의 어느 부분에서나 참조할 수 있습니다.

이 값은 사전정의된 것입니다. 값을 수정하지 마십시오. 이를 변수 참조로 사용하지 않습니다.

예

예 1: 함수가 성공적으로 완료되었는지의 여부를 알려주는 메시지

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

예 2: 리턴 코드가 0이 아닌 경우의 생략시 메시지

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

함수가 0이 아닌 리턴 코드를 리턴하는 경우 생략시 메시지가 표시됩니다.

제3장 Net.Data 내장 함수

Net.Data는 FUNCTION 블록을 작성하지 않고도 사용할 수 있는 다양한 함수를 제공합니다. Net.Data 내장 함수는 다음과 같은 범주로 나뉩니다.

- 일반 함수는 Net.Data를 이용하여 웹 페이지를 개발할 수 있도록 하고 다른 범주에는 속하지 않습니다. 133 페이지의 『일반 함수』 참조.
- 수학 함수는 수학적 연산을 수행합니다. 162 페이지의 『수학 함수』 참조.
- 문자열 조작 함수는 문자열과 문자를 수정합니다. 175 페이지의 『문자열 함수』 참조.
- 단어 조작 함수는 단어나 단어 세트를 수정합니다. 192 페이지의 『단어 함수』 참조.
- 테이블 조작 함수는 테이블 자료를 토대로 양식과 보고서를 생성할 수 있도록 돕습니다. 202 페이지의 『테이블 함수』 참조.
- 플랫 파일 인터페이스 함수는 파일 입력 및 출력을 수행합니다. 234 페이지의 『플랫 파일 인터페이스 함수』 참조.
- 웹 레지스트리 함수 웹 레지스트리에서의 조작을 수행합니다. 258 페이지의 『웹 레지스트리 함수』 참조.
- 영속 매크로 함수는 Net.Data에서의 트랜잭션 처리를 지원합니다. 270 페이지의 『영속적 매크로 함수』 참조.

모든 Net.Data 변수는 문자열 유형이지만, 정수 및 부동이라고 하는 용어는 각각 정수 값 또는 부동값을 나타내는 문자열을 표시하는 데 사용됩니다.

함수명

Net.Data 내장 함수는 예약된 접두부인 DTW로 시작됩니다. 사용자 정의 함수에는 이 접두부를 사용할 수 없습니다.

Net.Data 내장 함수가 아닌 함수에 대해 DTW 접두부를 사용하면 예상치 못한 결과가 발생할 수 있습니다.

내장 함수명은 대소문자가 구분되지 않습니다.

입력 및 출력 매개변수

함수는 Net.Data가 입력, 출력 또는 입출력 모두에 대해 매개변수를 사용할지의 여부를 결정하는 매개변수 전달 스펙을 가질 수 있습니다. 이러한 매개변수 전달 스펙은 다음 키워드에 의해 지정됩니다.

- IN** 매개변수가 Net.Data에서 언어 환경으로 입력 자료를 전달하도록 지정합니다.
- OUT** 매개변수가 언어 환경에서 Net.Data로 출력 자료를 리턴하도록 지정합니다.
- INOUT**
매개변수가 입력 자료를 언어 환경에 전달하고 언어 환경의 출력 자료를 Net.Data로 리턴하도록 지정합니다.

함수 결과 포매팅

대부분의 함수는 다음과 같은 형식을 취합니다.

- 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 매개변수상에서 함수를 수행합니다. 각 매개변수는 입력 매개변수와 출력 매개변수의 역할을 모두 수행합니다. 함수는 매개변수에 기초하여 수행되며 이의 결과가 매개변수에 리턴됩니다. 다음 예는 세 개의 입력 매개변수를 일관성 있게 보이도록 모두 대문자로 변환합니다.

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

- DTW_, DTWF_ 및 DTWR_로 시작하는 함수는 출력 매개변수에 결과를 리턴합니다. 출력 매개변수를 지정해야 합니다. 다음 예는 서버 시간을 나타냅니다.

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

함수 매개변수 규칙

함수 매개변수를 정확한 순서로 배치하십시오. 모든 입력 매개변수를 지정한 후에 마지막 입력 매개변수를 지정할 수 있으며, 생략시 값을 사용하려면 널(『』) 값을 지정하십시오. 예를 들어, 다음과 같이 DTW_TB_INPUT_TEXT를 호출할 수 있습니다.

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

위 예에서 네번째와 다섯번째 매개변수는 생략시 값을 사용합니다. 생성된 HTML에서 『32』가 MAXLENGTH의 값을 나타내려면 이들 매개변수를 널(NULL)이 되게 하십시오. 마지막 매개변수가 지정되지 않았으므로 생략시 값이 사용됩니다. MAXLENGTH와 앞의 두 매개변수에 대해 생략시 값을 사용하려면, 아래와 같이 이들 매개변수를 생략하십시오.

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

후속하는 널이 아닌 입력 매개변수가 존재하는 경우 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 이용하여 웹 페이지를 개발할 수 있도록 하며 다른 범주에는 속하지 않는 함수입니다. 다음과 같은 함수가 일반 함수입니다.

- 134 페이지의 『DTW_ADDQUOTE』
- 136 페이지의 『DTW_CACHE_PAGE』
- 140 페이지의 『DTW_DATE』
- 142 페이지의 『DTW_EXIT』
- 143 페이지의 『DTW_GETCOOKIE』
- 145 페이지의 『DTW_GETENV』
- 146 페이지의 『DTW_GETINIDATA』
- 147 페이지의 『DTW_HTMLENCODER』
- 149 페이지의 『DTW_QHTMLENCODER』
- 150 페이지의 『DTW_SENDMAIL』
- 154 페이지의 『DTW_SETCOOKIE』
- 157 페이지의 『DTW_SETENV』
- 158 페이지의 『DTW_TIME』
- 160 페이지의 『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 들어 있어야 SQL문이 정확히 처리되므로 교체가 필요합니다.

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

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

DTW_ADDQUOTE 함수를 사용하면 SQL문이 변경되어 오류 발생을 막아줍니다.

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('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: 각각의 작은 따옴표(')가 두개의 작은 따옴표로 교체되는 입력 문자열이 들어 있는 변수.

예

예 1: 추가의 작은 따옴표를 OUT 매개변수에 추가합니다.

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

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

예 2: 추가의 작은 따옴표를 함수 호출의 리턴 값에 추가합니다.

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

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

예 3: 추가의 작은 따옴표를 함수 호출의 각 INOUT 매개변수에 추가합니다.

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

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

예 4: 추가의 작은 따옴표를 DB2 테이블에 삽입되는 데이터에 추가합니다.

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

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

DTW_CACHE_PAGE

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

목적

매크로에서 함수의 위치 다음에 연속되는 모든 HTML 출력의 캐싱을 시작합니다. 이 함수가 호출되면, 캐쉬로부터 지정된 페이지를 검색하여 이것이 매크로에서 생성된 출력 페이지인 것처럼 웹 브라우저에 이를 전송합니다. 페이지를 찾았는데 이 페이지가 아직 만기되지 않은 경우, Net.Data는 매크로 처리를 중지하고 매크로에서 나온 다음 캐쉬된 페이지를 웹 브라우저에 전송합니다.

요청된 페이지가 캐쉬에 없거나 캐쉬된 기존 페이지가 *age* 값보다 오래된 경우, Net.Data는 새로운 출력 페이지를 생성합니다. 매크로가 성공적으로 완료되면, Net.Data는 새로운 페이지를 브라우저에 전송하고 그 페이지를 캐쉬합니다.

매크로에서 **DTW_CACHE_PAGE** 함수의 위치 판별:

- 대다수의 캐싱 응용프로그램에서, 모든 HTML 출력을 캐쉬하려면 **DTW_CACHE_PAGE**를 매크로의 상단에 지정하십시오. 이와 같이 하면 새로운 보고서 블록이 추가될 때 매크로를 유지보수하는 것이 훨씬 쉬워집니다. 예를 들어, 함수의 매크로의 중간에 있을 경우 HTML 보고서 섹션이 매크로에서 더 앞에 추가되는 경우 이것이 확인되지 않을 수도 있습니다. 이 경우 Net.Data는 새로운 보고서 출력을 캐쉬하지 않습니다. 또한, 이 방법을 이용할 경우 페이지가 캐쉬된다고 판단되면 Net.Data는 더 이상의 처리를 중단하므로 성능 향상을 꾀할 수 있습니다.
- 고급 캐싱 응용프로그램의 경우 매크로의 도입부에서가 아니라 처리중의 특정 시점에서 캐쉬 조작하도록 결정을 내려야 할 때 HTML 출력 섹션에 함수를 배치할 수 있습니다. 예를 들어, 조회나 함수 호출로부터 리턴되는 행 수에 기초하여 캐싱을 결정해야 할 수도 있습니다.

형식

@DTW_CACHE_PAGE(cacheid, url, age, status)

값

표 26. DTW_CACHE_PAGE 매개변수

매개변수	사용	설명
<i>cache_id</i>	IN	페이지가 놓일 캐쉬를 식별하는 문자열 변수.
<i>cached_page_ID</i>	IN	후속하는 DTW_CACHE_PAGE 캐쉬 요청에서 캐쉬된 페이지를 찾는 데 사용되는 식별자가 들어 있는 문자열 변수. 문자열이 URL일 수 있습니다.

표 26. DTW_CACHE_PAGE 매개변수 (계속)

매개변수	사용	설명
<i>age</i>	IN	<p>초 단위의 시간 길이를 담고 있는 문자열 변수. 이 매개변수는 페이지가 만기되었는지의 여부를 판별합니다. 이 페이지가 <i>age</i> 값보다 오래된 경우, 페이지가 브라우저에 전송되지 않습니다.</p> <p><i>age</i>가 -1로 지정되고 페이지가 캐쉬에 존재하는 경우, Net.Data는 페이지의 만기 여부에 관계없이 캐쉬에서 직접 페이지를 웹 브라우저에 전송합니다. Net.Data가 캐쉬의 페이지를 대체하지는 않습니다.</p>
<i>status</i>	OUT	<p>캐쉬된 페이지의 상태를 나타내는 문자열 변수. 가능한 값은 다음과 같은 소문자값입니다.</p> <ul style="list-style-type: none"> • ok: 매크로 실행이 종료되면 출력 페이지가 캐쉬됩니다. • new: 페이지가 캐쉬에 없습니다. • renew: 페이지가 캐쉬에 있지만 만기되었습니다. • no_cache: 지정된 캐쉬 식별자가 존재하지 않습니다. 캐쉬 식별자는 캐쉬 구성 파일에 정의되어 있어야 합니다. 페이지를 캐쉬하지 않고도 매크로의 실행은 계속됩니다. • inactive: 지정한 캐쉬가 비활성으로 표시되어 있습니다. 페이지를 캐쉬하지 않고도 매크로의 실행은 계속됩니다. • busy: 이 실행 이전에 사용자 매크로에서 DTW_CACHE_PAGE 내장 함수를 호출하였습니다. 매크로의 실행은 계속됩니다. • error: 캐쉬와 대화하는 중에 오류가 발생했습니다.

예

예 1: 모든 HTML 출력을 캡처하기 위해 DTW_CACHE_PAGE 함수를 매크로의 서두에 배치합니다.

```
%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>

%}
```

예 2: 캐쉬 여부는 HTML 출력의 예상 크기에 따라 결정되므로 함수를 HTML 블록에 배치합니다.

```
%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 예에서, 페이지는 HTML 출력의 예상 크기에 따라 캐쉬 또는 검색됩니다. 데이터베이스 테이블에 100개보다 많은 행이 들어 있는 경우 HTML 출력 페이지는 캐쉬 전용으로 간주됩니다. Net.Data는 반드시 매크로 실행 후 OUTPUT 블록의 텍스트, This is the customer list를 브라우저에 전송합니다. 즉, 텍스트는 캐쉬되지 않습니다. 함수 호출 @count_rows() 다음의 행은 IF 블록의 조건이 충족되면 캐쉬 또는 검색됩니다. 이 두 부분이 합해져서 완전한 Net.Data 출력 페이지를 형성합니다.

예 3: 캐쉬 ID와 캐쉬된 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!  
  </p>  
  </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 - 연중의 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

예 1: 일반 날짜 형식

```
@DTW_DATE(results)
```

- 결과: results = "25 Apr 1997"

예 2: 유럽 날짜 형식

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

- 결과: results="25/04/97"

예 3: 미국 날짜 형식

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

- 결과: 04/25/97

DTW_EXIT

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

목적

즉시 매크로에서 나가도록 지정합니다. 지금까지 매크로가 작성한 페이지는 Net.Data가 브라우저에 전송합니다.

성능 정보: DTW_EXIT를 사용하면 출력이 생성될 때 매크로의 처리가 중지되어 Net.Data의 전체 파일 처리 시간을 절약할 수 있습니다.

중요! DTW_EXIT 함수를 추가하기 전에 전체 매크로의 구문이 정확한지 확인하십시오. DTW_EXIT()를 사용하면 이 함수에 대한 호출이 있을 경우 Net.Data가 매크로의 처리를 중지하므로, DTW_EXIT() 함수가 처리된 이후 발생하는 오류를 파악할 수 없게 됩니다.

형식

@DTW_EXIT()

예

예 1: 매크로에서 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_GETCOOKIE

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

목적

읽혀질 쿠키의 이름을 지정하고 쿠키의 값을 리턴합니다.

참고: 하나의 쿠키를 두 개의 개별 HTTP 요청으로 정의하고 검색하십시오. 쿠키는 클라이언트에 전송된 후에만 가시화되므로 매크로가 동일한 HTTP 요청에 정의된 쿠키를 얻고자 하면 예상치 못한 결과가 초래될 수 있습니다.

OS/400 및 OS/390 사용자: 쿠키 값에 URL 형식 코드(예: "%20")가 포함된 경우, 이 쿠키 값은 리턴되기 전에 해독됩니다.

최신 정보는 『Persistent Client State HTTP Cookies』에서 Netscape의 스펙을 참조하십시오.

형식

@DTW_GETCOOKIE(IN cookie_name, OUT cookie_value)

@DTW_rGETCOOKIE(IN cookie_name)

값

표 28. DTW_GETCOOKIE 매개변수

자료 유형	매개변수	사용	설명
문자열	cookie_name	IN	쿠키의 이름을 지정하는 변수 또는 리터럴 문자열.
문자열	cookie_value	OUT	사용자 상태 정보와 같이 함수에 의해 검색된 쿠키의 값이 들어 있는 변수.

예

예 1: 사용자 ID와 암호 정보가 들어 있는 쿠키를 검색합니다.

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

예 2: 사용자 정보를 수집하기 전에 사용자에게 대한 쿠키가 존재하는지 판별합니다.

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

%HTML(welcome) {
    <html>
<body>
    <h1>Net.Data Club</h1>
    @DTW_GETCOOKIE("NDC_name", name)
```

```

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

```

HTML welcome 섹션에서는 쿠키 NDC_name이 존재하는지 점검합니다. 쿠키가 존재하면 브라우저는 개별화된 환영 화면을 표시합니다. 쿠키가 존재하지 않으면, 사용자 이름을 입력하라고 하는 프롬프트가 표시되고 HTML remember 섹션에 이를 기입하여, 아래와 같이 사용자 이름을 쿠키 NDC_name에 설정합니다.

```

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

```

DTW_GETENV

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

목적

지정된 환경 변수의 값을 리턴합니다. ENVVAR를 사용하여 환경 변수의 값을 참조할 수도 있습니다. 자세한 내용은 16 페이지의 『ENVVAR 명령문』을 참조하십시오.

형식

@DTW_GETENV(envVarName, envVarValue)

@DTW_rGETENV(envVarName)

값

표 29. 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

예 1: PATH문에 대한 값을 OUT 매개변수에 리턴합니다.

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

- 입력: myEnvVarName = "PATH"
- 결과: myEnvVarValue = "/usr/bin"

예 2: PATH문에 대한 값을 리턴합니다.

```
@DTW_rGETENV(myPath)
```

- 입력: myPath = "PATH"
- 결과: "/usr/bin"

예 3: 서버의 프로토콜에 대한 값을 리턴합니다.

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

- 결과: "HTTP/1.0"

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: OS/400 운영 체제가 아닌 경우, ENVIRONMENT 명령문 뿐만 아니라 구성 경로 변수(MACRO_PATH, EXEC_PATH 및 INCLUDE_PATH)는 이 호출로 검색할 수 없습니다. OS/400 운영 체제에서, 이 제한은 ENVIRONMENT 명령문에만 적용됩니다.

형식

@DTW_GETINIDATA(iniVarName, iniVarValue)

@DTW_rGETINIDATA(iniVarName)

값

표 30. 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

예 1: Net.Data 경로 변수 값을 리턴합니다.

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

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

DTW_HTMLENCODE

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

목적

HTML 문자 이탈 코드를 사용하여 선택된 문자를 코드화합니다. 이 함수를 사용하여, 웹 브라우저가 HTML로 해석하지 않기를 바라는 문자 자료를 코드화할 수 있습니다. 예를 들어, 적절한 이탈 코드를 사용하여 웹 페이지 내에 보다 작음(<) 및 보다 큼(>) 기호와 같은 문자를 표시할 수 있으며 이외의 경우에 이 문자는 브라우저에 의해 HTML 태그의 구성요소로 해석됩니다.

표31에는 DTW_HTMLENCODE 함수에 의해 코드화되는 문자가 나와 있습니다.

표 31. HTML의 문자 이탈 코드

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

형식

@DTW_HTMLENCODE(stringIn, stringOut)

@DTW_rHTMLENCODE(stringIn)

값

표 32. DTW_HTML_ENCODE 매개변수

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

예

예 1: 공백 문자를 코드화합니다.

```
@DTW_HTML_ENCODE(string1,string2)
```

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

예 2: 공백, 보다 작음 기호 및 같음 기호를 코드화합니다.

```
@DTW_rHTML_ENCODE("X <= 10")
```

- 결과: "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_HTMLENCODE와 동일한 기능을 수행하며, 작은 따옴표 문자(')를 '로 코드화합니다. DTW_QHTMLENCODE가 사용하는 HTML 문자 이탈 코드는 147 페이지의 표31에 나와 있습니다.

형식

@DTW_QHTMLENCODE(stringIn, stringOut)

@DTW_rQHTMLENCODE(stringIn)

값

표 33. DTW_QHTMLENCODE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
문자열	<i>stringOut</i>	OUT	특정 문자가 HTML 문자 이탈 코드로 대체된 수정 된 형식의 <i>stringIn</i> 이 들어 있는 변수.

예

예 1: 어포스트로피와 공백을 코드화합니다.

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

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

예 2: 어포스트로피, 공백 및 앰퍼샌드를 코드화합니다.

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

- 결과: "John's & Jane's"

DTW_SENDMAIL

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

목적

전자 우편(e-mail) 메시지를 동적으로 구축하여 전송합니다.

이 함수는 전자 우편 메시지 전송에 사용할 SMTP 서버를 지정하는 선택적 구성 변수, DTW_SMTP_SERVER와 함께 작동합니다. 이 매개변수의 값은 호스트명, IP 주소이거나 OS/390 서버의 지정 시에는 노드 및 이름이 될 수 있습니다. 이 변수가 정의되지 않은 경우, Net.Data는 국지 호스트를 SMTP 서버로 사용합니다. 이 변수에 대한 자세한 내용은 *Net.Data* 관리 및 프로그래밍 안내서의 구성 장을 참조하십시오.

OS/400, OS/2, Windows NT, UNIX에 대한 자국어 관련사항: SMTP(Standard Simple Mail Transfer Protocol) 서버는 미국 ASCII 문자와 같은 7비트 자료만을 인정합니다. 메시지에 8비트 문자가 들어 있는 경우, ESMTP(Extended Simple Mail Transfer Protocol) 서버를 지정하도록 하십시오. ESMTP 서버는 8비트 문자를 승인합니다. Net.Data는 8비트 자료를 7비트 자료로 코드화하지 않습니다. ESMTP 서버에 대한 액세스권이 없는 경우, 전자 우편 메시지에서 8비트 문자를 모두 제거하십시오.

형식

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

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

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

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

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

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

값

표 34. DTW_SENDMAIL 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>sender</i>	IN	<p>저자의 주소를 지정하는 변수 또는 리터럴 문자열. 이 매개변수는 필수입니다. 유효한 형식은 다음과 같습니다.</p> <ul style="list-style-type: none"> • Name <user@domain> • <user@domain> • user@domain
문자열	<i>recipient</i>	IN	<p>이 메시지가 전송되는 전자 우편 주소를 지정하는 변수 또는 리터럴 문자열. 이 값은 콤마(,)로 구분되는 여러 개의 수신인을 가질 수 있습니다. 이 매개변수는 필수입니다. 유효한 <i>recipient</i> 형식은 다음과 같습니다.</p> <ul style="list-style-type: none"> • Name <user@domain> • <user@domain> • user@domain
문자열	<i>message</i>	IN	전자 우편 메시지의 텍스트가 들어 있는 변수 또는 리터럴 문자열. 이 매개변수는 필수입니다.
문자열	<i>subject</i>	IN	주제 행의 텍스트가 들어 있는 변수 또는 리터럴 문자열. 이것은 선택적 매개변수입니다. 추가 매개변수를 지정하려면 널 문자열("")을 지정해야 합니다.
문자열	<i>CarbonCopy</i>	IN	전자 우편 주소나, 추가 수신인의 이름과 전자 우편 주소가 들어 있는 변수 또는 리터럴 문자열. 이 값은 콤마(,)로 구분된 여러 명의 수신인을 가질 수 있습니다. 유효한 수신인 형식을 알려면 <i>Recipient</i> 매개변수를 참조하십시오. 이것은 선택적 매개변수입니다. 추가 매개변수를 지정하려면 널 문자열("")을 지정해야 합니다.
문자열	<i>BlindCarbonCopy</i>	IN	전자 우편 주소나 추가 수신인의 이름과 전자 우편 주소가 들어 있는 변수 또는 리터럴 문자열로, 수신인이 전자 우편 표제에 나타나지는 않습니다. 이 값은 콤마(,)로 구분된 여러 명의 수신인을 가질 수 있습니다. 유효한 수신인 형식을 알려면 <i>Recipient</i> 매개변수를 참조하십시오. 이것은 선택적 매개변수입니다. 추가 매개변수를 지정하려면 널 문자열("")을 지정해야 합니다.
문자열	<i>ReplyTo</i>	IN	<p>이 메시지에 대한 응답을 전송해야 하는 전자 우편 주소가 들어 있는 변수 또는 리터럴 문자열. 이것은 선택적 매개변수입니다. 추가 매개변수를 지정하려면 널 문자열("")을 지정해야 합니다. 유효한 <i>ReplyTo</i> 형식은 다음과 같습니다.</p> <ul style="list-style-type: none"> • Name <user@domain> • <user@domain> • user@domain

표 34. DTW_SENDMAIL 매개변수 (계속)

자료 유형	매개변수	사용	설명
문자열	<i>Organization</i>	IN	송신자의 소속 단체명이 들어 있는 변수 또는 리터럴 문자열. 이것은 선택적 매개변수입니다.

예

예 1: 간단한 전자 우편 메시지를 작성하여 전송하는 함수 호출

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

DTW_SENDMAIL 함수는 다음 정보와 함께 전자 우편 메시지를 전송합니다.

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

There is a meeting at 9:30.

Date 정보는 시스템 날짜 및 시간 함수를 사용하여 구성되어 SMTP 날짜 형식으로 포맷됩니다.

예 2: 여러 명의 수신인, 카본 카피 및 블라인드 카본 카피 수신인, 회사 이름과 함께 전자 우편 메시지를 작성하여 전송하는 함수 호출

```
@DTW_SENDMAIL("IBM User 1 <ibmuser1@ibm.com>", "IBM User 2 <ibmuser2@ibm.com>,
IBM User 3 <ibmuser3@ibm.com>, IBM User 4 <ibmuser4@ibm.com>",
"There is a meeting at 9:30.", "Status meeting", "IBM User 5 <ibmuser5@ibm.com>",
"IBM User 6 <ibmuser6@ibm.com>", "meeting@ibm.com", "IBM")
```

DTW_SENDMAIL 함수는 다음 정보와 함께 전자 우편 메시지를 전송합니다.

```
Date: Mon, 3 Apr 1998 09:54:33 PST
To: IBM User 2 <ibmuser2@ibm.com>, IBM User 3 <ibmuser3@ibm.com>, IBM User 4 <ibmuser4@ibm.com>
CC: IBM User 5 <ibmuser5@ibm.com>
BCC: IBM User 6 <ibmuser6@ibm.com>
From: IBM User 1 <ibmuser1@ibm.com>
ReplyTo: meeting@ibm.com
Organization: IBM
Subject: Status meeting
```

There is a meeting at 9:30.

예 3: 웹 양식 인터페이스를 통해 전자 우편을 작성 및 전송하는 매크로

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

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

```

    <p>E-mail has been sent out.
  </body>
</html>
%}

```

이 매크로는 웹 양식 인터페이스를 통해 전자 우편을 전송합니다. HTML start 섹션에는 수신인의 전자 우편 주소, 주제 및 메시지를 입력할 수 있는 양식이 표시됩니다. 사용자가 전자 우편 전송 버튼을 누르면, HTML(sendemail) 섹션에 지정된 수신인에게 메시지가 전송됩니다. 이 섹션은 DTW_SENDMAIL를 호출하고 웹 양식에서 확보한 매개변수를 사용하여 수신인 및 수령인뿐 아니라 전자 우편 메시지의 내용을 판별합니다. 전자 우편 메시지가 전송되면 확인 메시지가 표시됩니다.

예 4: SQL 조화를 사용하여 수신인 목록을 결정하는 매크로

```

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

```

이 매크로는 고객 데이터베이스에서의 SQL 조화를 통해 결정된 지정된 고객 그룹에 자동화된 전자 우편 메시지를 전송합니다. 또한 SQL 조화는 고객의 전자 우편 주소를 검색하기도 합니다. 전자 우편의 내용은 *message* 값에 의해 결정되며, 정적이거나 동적일 수 있습니다(예를 들어, 다른 SQL 조화를 사용하여 제품의 버전 번호나 여러 가지 상품의 가격을 동적으로 지정할 수 있습니다).

DTW_SETCOOKIE

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

목적

쿠키 이름, 값 및 옵션(예: 만기 일자 및 보안 요건)을 정의합니다.

쿠키를 검색하려면 DTW_GETCOOKIE() 함수를 사용하십시오. 쿠키를 정의하는 방법에 대해서는 143 페이지의 『DTW_GETCOOKIE』를 참조하십시오.

참고:

- 하나의 쿠키를 두 개의 개별 HTTP 요청으로 정의 및 검색하십시오. 쿠키는 클라이언트에 전송된 후에만 가시화되므로 매크로가 동일한 HTTP 요청에 정의된 쿠키를 얻고자 하면 예상치 못한 결과가 초래될 수 있습니다.
- 쿠키에는 세미콜론, 콤마 및 공백을 사용하지 마십시오. 이들을 사용해야 하는 경우, DTW_SETCOOKIENet.Data 함수 DTW_rURLESCSEQ를 사용하여 특수 문자가 들어 있는 문자열을 DTW_SETCOOKIE에 전달하기 전에 이 문자열을 처리하십시오. 예를 들면 다음과 같습니다.

```
@DTW_SETCOOKIE("my_cookie_name", @DTW_rURLESCSEQ("my cookie value"))
```

제한사항:

- 클라이언트 웹 브라우저가 자바 스크립트를 지원하지 않는 경우, 브라우저는 쿠키를 설정하지 않습니다.
- DTW_SETCOOKIE는 자바 스크립트 코드를 생성하므로, <SCRIPT> 또는 <NOSCRIPT> HTML 요소 내부에서는 DTW_SETCOOKIE를 호출하지 마십시오.

형식

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

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

값

표 35. DTW_SETCOOKIE 매개변수

자료 유형	매개변수	사용	설명
문자열	cookie_name	IN	쿠키의 이름을 지정하는 변수 또는 리터럴 문자열
문자열	cookie_value	IN	쿠키의 값을 지정하는 변수 또는 리터럴 문자열

표 35. DTW_SETCOOKIE 매개변수 (계속)

자료 유형	매개변수	사용	설명
문자열	<i>advanced_options</i>	IN	<p>쿠키를 정의하는 데 사용되는 선택적 속성(콤마로 구분됨)이 들어 있는 문자열. 속성은 다음과 같습니다.</p> <p>expires = date 쿠키의 유효 기간을 정의하는 날짜 문자열을 지정합니다. 날짜가 만기되고 나면, 쿠키는 더 이상 저장 또는 검색되지 않습니다. 구문은 다음과 같습니다.</p> <p><i>weekday,%20DD-month-YYYY%20HH:MM:SS%20GMT</i></p> <p>여기서,</p> <p><i>weekday</i> 완전한 요일 이름을 지정합니다.</p> <p><i>DD</i> 월의 일자를 지정합니다.</p> <p><i>month</i> 월에 대한 세 문자 약어를 지정합니다.</p> <p><i>YYYY</i> 네 문자의 연도 수를 지정합니다.</p> <p><i>HH:MM:SS</i> 시간, 분 및 초로 시간소인을 지정합니다.</p> <p>domain = domain_name 일치하는 도메인 속성에 사용하도록 쿠키의 도메인 속성을 지정합니다.</p> <p>path = path 쿠키가 효력을 갖는 도메인에서 URL의 서브세트를 지정합니다.</p> <p>secure 보안 채널을 통해서만 쿠키가 HTTPS 서버로 전송되도록 지정합니다.</p> <p>secure 옵션이 지정되지 않은 경우, 비보안 채널을 통해 쿠키를 전송할 수 있습니다. 보안 옵션에서는 브라우저가 쿠키를 코드화할 필요가 없으며, DTW_SETCOOKIE 문이 들어 있는 페이지가 SSL을 통해 전송되지도 않습니다.</p> <p>확장 옵션에 대한 추가 정보는 http://home.netscape.com에서 Netscape 쿠키 스펙을 참조하십시오.</p>

예

예 1: 보안 확장 옵션과 더불어 사용자 ID 및 암호 정보가 들어 있는 쿠키를 정의합니다.

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

예 2: 만기일 확장 옵션을 가지는 쿠키를 정의합니다.

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1",
    "expires=Wednesday%2001-Dec-2010%2000:00:00")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT",
    "expires=Wednesday,%2001-Dec-2010%2000:00:00;secure")
```

함수 호출은 한 행에 작성해야 합니다. 이 예에서는 보기 쉽게 행을 분리했습니다.

예 3: 사용자 정보를 수집하기 전에 사용자에게 쿠키가 존재하는지 판별합니다.

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

HTML(welcome) 섹션에서는 쿠키 NDC_name이 존재하는지 점검합니다. 쿠키가 존재하면 브라우저는 개별화된 환영 화면을 표시합니다. 쿠키가 존재하지 않으면, 브라우저는 사용자 이름을 입력하라는 프롬프트를 표시하고 이를 HTML(remember) 섹션에 기재합니다. 이 섹션은 아래와 같이 사용자 이름을 쿠키 NDC_name에 기록합니다.

```
%HTML(remember) {
    <html>
    <body>
        <H1>Net.Data Club>
        @DTW_SETCOOKIE("NDC_name", name, "expires=Wednesday,%2001-Dec-2010%2000:00:00;path=/")
        <p>Thank you.
        <p><a href="welcome">Come back</a>
    </body>
</html>
%}
```

DTW_SETENV

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

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 36. DTW_SETENV 매개변수

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

예

예 1: 이전 경로의 값을 리턴합니다.

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

- 입력: envVarName = "PATH", envVarValue = "myPath"
- 결과: prevValue = "myPreviousPath"

예 2: 이전 경로의 값을 리턴하고 PATH 값에 대한 값을 지정합니다.

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

- 입력: envVarName = "PATH", envVarValue = "myPath"
- 결과: "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 37. 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 - 12 시간제(hh:mmAM/PM) L - 현지 시간(hh:mm:ss) N - 24시간제(hh:mm:ss); 생략시 값 X - 확장 시간 형식(24시간 시계를 사용하는 hh:mm:ss.ccc로 여기서 ccc는 밀리초로 나타 낸 수) H - 자정 이후 시간 M - 자정 이후 분 S - 자정 이후 초
문자열	<i>stringOut</i>	OUT	지정된 형식의 시간을 포함하는 변수.

예

예 1: 24시간제 형식

```
@DTW_TIME(results)
```

- 결과: results = "10:30:53"

예 2: 12시간제 형식

```
@DTW_TIME("C", results)
```

- 결과: results = "10:30AM"

예 3: 함수 호출과 함께 자정 이후의 분 수를 리턴합니다.

```
@DTW_rTIME("M")
```

- 결과: "630"

예 4: 함수 호출로 생략시 시간 및 날짜 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 선택된 문자를 이탈 코드로 대체합니다. 표38에 나열된 문자가 매크로가 다른 매크로나 HTML 블록을 참조하는 데 사용하는 URL내에 나타날 경우, 이 문자를 암호화하는 데 이 기능을 사용하십시오.

표 38. URL의 문자 이탈 코드

문자	이름	코드
SPACE	공간	%20
"	큰 따옴표	%22
#	숫자 기호	%23
%	퍼센트	%25
&	앰퍼샌드	%26
+	플러스	%2B
\	역슬래쉬	%2F
:	콜론	%3A
;	세미콜론	%3B
<	보다 작음	%3C
=	같음	%3D
>	보다 큼	%3E
?	물음표	%3F
@	@ 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 39. 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 대체된 문자가 있는 입력 문자열이 포함된 변수.

예

예 1: *string1*의 공백과 앰퍼샌드 문자를 이탈 코드로 대체하고 결과를 *string2*에 할당합니다.

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

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

예 2: 공백 및 앰퍼샌드 문자를 이탈 코드로 대체합니다.

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

- 결과: "Guys%20%26%20Dolls"

예 3: ROW 블록에 DTW_rURLESCSEQ를 사용하고 공백과 @ 기호를 이탈 코드로 대체합니다.

```
%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>
```

응용프로그램 사용자가 이름 "Patrick O'Brien"을 클릭하면, 이름과 전자 메일 주소로 지정된 값은 URL의 조회 문자열 내에서 흐르며 이에 따라 Net.Data는 fullrpt.mac 매크로의 입력 섹션을 실행하게 됩니다.

수학 함수

다음 함수를 사용하여 수학적 연산을 수행할 수 있습니다.

수학 함수에 대한 NLS 관련사항: Net.Data는 Net.Data가 수행되는 웹 서버에 지정된 국지 설정에 기초하여 숫자 값의 소수점을 표시합니다. 예를 들어, 웹 서버에 소수점이 콤마(,)로 지정된 경우 Net.Data는 콤마를 사용하여 십진 자료를 포맷합니다. 소수점을 지정하는 데 사용할 문자를 결정할 때 Net.Data는 다음 설정을 사용합니다.

OS/390, Windows NT, OS/2 및 UNIX 운영 체제의 경우:

웹 서버가 실행하는 LOCALE

OS/400 운영 체제의 경우:

- V4R2 또는 이후 릴리스: 프로세스가 수행되는 사용자 프로파일에 의해 지정.
- V4R1 또는 이전 릴리스: QDECFMT 시스템 값에서 검색.

수학 계산에는 다음 함수를 사용할 수 있습니다.

- 163 페이지의 『DTW_ADD』
- 164 페이지의 『DTW_DIVIDE』
- 165 페이지의 『DTW_DIVREM』
- 167 페이지의 『DTW_FORMAT』
- 170 페이지의 『DTW_INTDIV』
- 171 페이지의 『DTW_MULTIPLY』
- 172 페이지의 『DTW_POWER』
- 173 페이지의 『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_ADD(number1, number2, result)

@DTW_rADD(number1, number2, precision)

@DTW_rADD(number1, number2)

값

표 40. 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> 및 <i>number2</i> 의 합계가 들어 있는 변수.

예

예 1:

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

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

예 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_DIVIDE(number1, number2, result)

@DTW_rDIVIDE(number1, number2, precision)

@DTW_rDIVIDE(number1, number2)

값

표 41. 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	결과와 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

예 1:

@DTW_DIVIDE("8.0", NUM2, result)

- 입력: NUM2 = "2"
- 결과: result = "4"

예 2:

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

- 입력: "1", NUM2 = "3"
- 결과: "0.33333"

예 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_DIVREM(number1, number2, result)

@DTW_rDIVREM(number1, number2, precision)

@DTW_rDIVREM(number1, number2)

값

표 42. 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

예 1:

@DTW_DIVREM(NUM1, NUM2, result)

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

예 2:

@DTW_rDIVREM("10", NUM2)

- 입력: NUM2 = "0.3"
- 결과: "0.1"

예 3:

@DTW_rDIVREM("3.6", "1.3")

- 결과: "1.0"

예 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 지정하는 경우, 그 결과는 @DTW_rADD(number,『0』)가 수행된 것처럼 형식화됩니다. 다른 옵션이 지정되는 경우, 다음 규칙에 따라 숫자가 형식화됩니다.

- *before* 및 *after* 매개변수는 *result* 매개변수의 정수 부분과 소수 부분에 사용되는 숫자의 수를 나타냅니다. 이들 매개변수 중 하나 또는 두 매개변수를 모두 생략하는 경우, 해당 부분에는 필요한 수 만큼의 숫자가 사용됩니다.
- *before* 매개변수가 숫자의 정수 부분(음수의 경우 부호까지 포함하여)보다 작은 경우, 오류가 발생합니다. *before* 매개변수가 해당 부분에 필요한 것보다 큰 경우, *number* 매개변수 값의 왼쪽이 공백으로 채워집니다. *after* 매개변수의 크기가 *number* 매개변수의 소수 부분과 같지 않은 경우, 크기가 일치하도록 반올림됩니다(또는 0으로 채워집니다). 0을 지정하면 숫자는 정수로 반올림됩니다.
- 또한, *expp*와 *expt* 매개변수는 결과의 지수 부분을 제어합니다. *expp* 매개변수는 지수 부분의 자리수를 설정합니다. 생략시 값은 필요한 만큼 사용하는 것입니다(0이 될 수도 있습니다). *expt* 매개변수는 지수 표기에 사용할 트리거 포인트를 설정합니다. 생략시 값은 precision 매개변수의 생략시 값입니다.
- *expp*가 0이면, 지수가 제공되지 않고, 숫자는 필요한 만큼 0이 추가된 단순한 양식으로 표시됩니다. *expp*의 크기가 지수를 포함할 수 없을 정도이면 오류가 발생합니다.
- 정수나 소수 부분에 필요한 자리수가 각각 *expt* 또는 *expt*의 두 배를 초과할 경우, 지수 표기법을 사용하십시오. *expt*가 0인 경우, 지수가 0이 아니면 항상 지수 표기법이 사용됩니다(*expp*가 0이면 이것이 *expt*의 0 값을 대체합니다). *expp*가 0이 아닐 때 지수가 0이면, *expp*+2개의 공백이 결과의 지수 부분에 제공됩니다. 지수가 0이고 *expp*가 지정되지 않으면, 단순한 양식이 사용됩니다.

형식

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

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

@DTW_FORMAT(number, before, after, expp, result)

@DTW_FORMAT(number, before, after, result)

@DTW_FORMAT(number, before, result)

@DTW_FORMAT(number, result)

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

@DTW_rFORMAT(number, before, after, exp, expt)

@DTW_rFORMAT(number, before, after, exp)

@DTW_rFORMAT(number, before, after)

@DTW_rFORMAT(number, before)

@DTW_rFORMAT(number)

값

표 43. 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 지정하려면 널(NULL) 문자열("")을 입력해야 합니다.
정수	<i>after</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 이것은 선택적 매개변수입니다. 추가 매개변수를 지정하려면 널(NULL) 문자열("")을 입력해야 합니다.
정수	<i>exp</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 추가 매개변수를 지정하려면 널(NULL) 문자열("")을 지정해야 합니다.
정수	<i>expt</i>	IN	양수를 나타내는 리터럴 문자열 또는 변수. 추가 매개변수를 지정하려면 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

예 1:

@DTW_FORMAT(NUM, BEFORE, result)

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

예 2:

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

- 결과: result = " 2"

예 3:

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

- 결과: result = " 1.730"

예 4:

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

- 결과: result = "-12.7300"

예 5:

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

- 결과: result = "1.234573E+04"

예 6:

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

- 결과: result = "1.235"

예 7:

@DTW_rFORMAT(" - 12.73")

- 결과: " - 12.73"

예 8:

@DTW_rFORMAT("0.000")

- 결과: "0"

예 9:

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

- 결과: "12345.73"

예 10:

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

- 결과: "123456700000.000"

예 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_INTDIV(number1, number2, result)

@DTW_rINTDIV(number1, number2, precision)

@DTW_rINTDIV(number1, number2)

값

표 44. 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

예 1:

@DTW_INTDIV(NUM1, NUM2, result)

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

예 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 그 결과를 리턴합니다.

형식

@DTW_MULTIPLY(number1, number2, precision, result)

@DTW_MULTIPLY(number1, number2, result)

@DTW_rMULTIPLY(number1, number2, precision)

@DTW_rMULTIPLY(number1, number2)

값

표 45. 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

예 1:

@DTW_MULTIPLY(NUM1, NUM2, result)

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

예 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

첫번째 매개변수를 두번째 매개변수 만큼 제공하고 그 결과를 리턴합니다.

형식

@DTW_POWER(number1, number2, precision, result)

@DTW_POWER(number1, number2, result)

@DTW_rPOWER(number1, number2, precision)

@DTW_rPOWER(number1, number2)

값

표 46. 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>number1</i> 을 <i>number2</i> 제공한 결과가 들어 있는 변수.

예

예 1:

@DTW_POWER(NUM1, NUM2, result)

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

예 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 그 결과를 리턴합니다.

형식

@DTW_SUBTRACT(number1, number2, precision, result)

@DTW_SUBTRACT(number1, number2, result)

@DTW_rSUBTRACT(number1, number2, precision)

@DTW_rSUBTRACT(number1, number2)

값

표 47. 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

예 1:

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

- 입력: NUM2 = "2.07"
- 결과: "-0.77"

다음 예는 Net.Data의 문자열인 숫자 값을 비교하는 방법을 보여줍니다.

예 2:

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

- 입력: NUM1 = "1.3, NUM2 = "1.07"
- 결과: result = "0.23"

예 3:

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

- 입력: NUM2 = "2.07"
- 결과: "-0.77"

문자열 함수

다음 함수는 Net.Data가 지원하는 표준 문자열 함수 세트입니다.

- 176 페이지의 『DTW_ASSIGN』
- 177 페이지의 『DTW_CONCAT』
- 178 페이지의 『DTW_DELSTR』
- 179 페이지의 『DTW_INSERT』
- 181 페이지의 『DTW_LASTPOS』
- 182 페이지의 『DTW_LENGTH』
- 183 페이지의 『DTW_LOWERCASE』
- 184 페이지의 『DTW_POS』
- 185 페이지의 『DTW_REVERSE』
- 186 페이지의 『DTW_STRIP』
- 187 페이지의 『DTW_SUBSTR』
- 189 페이지의 『DTW_TRANSLATE』
- 191 페이지의 『DTW_UPPERCASE』

OS/390, OS/2, Windows NT 및 UNIX에 대한 MBCS 지원: DTW_MBMODE 구성 값을 갖는 단어 및 문자열 함수에 대해 다중 바이트 문자 세트(MBCS) 지원을 지정할 수 있습니다. Net.Data 초기화 파일에 이 값을 지정하십시오. 생략시 값은 지원되지 않습니다. Net.Data 매크로에서 DTW_MBMODE 변수를 설정하여 초기화 파일의 값을 겹쳐쓸 수 있습니다. 자세한 내용은 *Net.Data Administration and Programming Guide*와 106 페이지의 『DTW_MBMODE』의 구성 변수 절을 참조하십시오.

OS/400에 대한 MBCS 지원: DBCS 지원이 자동으로 제공되지 않으며 DBCS 지원에 이 변수가 필요하지 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. 이 함수를 사용하여 매크로에서 변수를 변경하십시오.

형식

@DTW_ASSIGN(stringOut, stringIn)

값

표 48. DTW_ASSIGN 매개변수

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

예

예 1:

@DTW_ASSIGN(RC, "0")

- RC를 "0"으로 설정합니다.

예 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 49. 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> '가 들어 있는 변수. <i>string1</i> 은 <i>string2</i> 에 연결됩니다.

예

예 1:

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

- 결과: result = "This is a test."

예 2:

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

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

예 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

n 번째 문자에서 시작하여 length 문자 수만큼 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 50. 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 시작되는 문자의 위치. n 이 <i>stringIn</i> 길이보다 큰 경우 <i>stringOut</i> 은 <i>stringIn</i> 값으로 설정됩니다.
정수	<i>length</i>	IN	삭제할 부속 문자열의 길이. 생략시 값은 <i>stringIn</i> 끝까지의 모든 문자를 삭제하는 것입니다.
문자열	<i>stringOut</i>	OUT	수정된 <i>stringIn</i> 유형이 들어 있는 변수.

예

예 1:

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

- 결과: result = "abe"

예 2:

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

- 결과: "abce"

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 문자열에 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 51. 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	<i>stringIn1</i> 이 삽입된 이후 <i>stringIn2</i> 내의 문자 위치. n 이 <i>stringIn2</i> 의 길이보다 큰 경우, 충분한 문자가 생길 때까지 채움 문자 <i>pad</i> 로 채워집니다. 생략시 값은 <i>stringIn2</i> 의 시작 부분에 삽입하는 것입니다.
정수	<i>length</i>	IN	삽입할 <i>stringIn1</i> 의 문자 수. 이 매개변수가 <i>stringIn1</i> 의 길이보다 큰 경우, 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 전부 또는 일부를 삽입하여 수정된 <i>stringIn2</i> 가 들어 있는 변수.

예

예 1:

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

- 결과: result = "123abc"

예 2:

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

- 결과: result = "abc 123"

예 3:

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

- 결과: result = "abc 123"

예 4:

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

- 결과: result = "abc//123///"

예 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 역방향으로(오른쪽에서 왼쪽으로) 가면서, 또다른 문자열에 있는 마지막 문자열의 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 52. DTW_LASTPOS 매개변수

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

예

예 1:

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

- 결과: result = "8"

예 2:

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

- 결과: result = "8"

예 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 53. 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 수가 들어 있는 기호.

예

예 1:

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

- 결과: result = "8"

예 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 54. 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 소문자인 <i>stringIn</i> 이 들어 있는 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

예 1:

@DTW_LOWERCASE("This", stringOut)

- 결과: stringOut = "this"

예 2:

@DTW_rLOWERCASE(string1)

- 입력: string1 = "Hello"
- 결과: "hello"

예 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 검색 패턴을 사용하여 또다른 문자열에 있는 첫번째 문자열의 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 55. 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> 내의 문자 위치. 생략시 값은 <i>stringIn2</i> 의 첫번째 문자에서 탐색을 시작하는 것입니다.
정수	<i>nOut</i>	OUT	<i>stringIn2</i> 에서 첫번째 <i>stringIn1</i> 의 위치가 들어 있는 변수. <i>stringIn1</i> 이 발견되지 않는 경우, 0이 리턴됩니다.

예

예 1:

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

- 결과: result = "6"

예 2:

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

- 결과: result = "7"

예 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 56. 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 형식이 들어 있는 변수.

예

예 1:

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

- 결과: result = ".ti si sihT"

예 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 뒤 공백, 또는 입력 문자열의 선행 공백 모두를 제거합니다.

형식

@DTW_STRIP(stringIn, option, stringOut)

@DTW_STRIP(stringIn, stringOut)

@DTW_rSTRIP(stringIn, option)

@DTW_rSTRIP(stringIn)

값

표 57. 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 - 선행 및 뒤 공백을 모두 제거 L 또는 l - 선행 공백만 제거 T 또는 t - 뒤 공백만 제거
문자열	<i>stringOut</i>	OUT	옵션으로 지정된 대로 공백이 제거된 <i>stringIn</i> 이 들어 있는 변수.

예

예 1:

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

• 결과: result = "day"

예 2:

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

• 결과: result = "day"

예 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 58. DTW_SUBSTR 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	탐색될 리터럴 문자열 또는 변수.
정수	<i>n</i>	IN	부속 문자열의 첫번째 문자 위치. 생략시 값은 <i>stringIn</i> 의 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 크거나, <i>length</i> 가 <i>stringIn</i> 보다 긴 경우 사용되는 채움 문자. 생략시 값은 공백입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 부속 문자열이 들어 있는 변수.

예

예 1:

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

- 결과: result = "bc"

예 2:

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

- 결과: result = "bc"

예 3:

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

- 결과: result = "bc.."

예 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 문자열의 문자를 변환합니다. *tableI*, *tableO* 및 *default* 문자가 매개변수 목록에 없는 경우, *stringIn* 매개변수가 대문자로 변환됩니다. *tableI* 및 *tableO*이 목록에 있는 경우, *tableI*에서 입력 문자열의 각 문자가 탐색되고 *tableO*의 해당 문자로 변환됩니다. *tableI*의 문자에 대해 *tableO*에 해당 문자가 없는 경우, *default* 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 59. 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 <i>default</i> 를 지정하려면 널 값("")을 사용하십시오. 그렇지 않은 경우 이 매개변수는 생략 가능합니다.
문자열	<i>tableI</i>	IN	<i>stringIn</i> 에서 탐색될 변수 또는 리터럴 문자열. <i>default</i> 를 지정하려면 널 값("")을 사용하십시오. 그렇지 않은 경우 이 매개변수는 생략 가능합니다.
문자열	<i>default</i>	IN	사용할 default 문자. 생략시 값은 공백입니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 변환된 결과가 들어 있는 변수.

예

예 1:

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

- 결과: result = "ABBC"

예 2:

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

- 결과: result = "aRR "

예 3:

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

- 결과: "12..ef"

예 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 60. 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 대문자인 <i>stringIn</i> 이 들어 있는 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

예 1:

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

- 결과: result = "TEST"

예 2:

```
@DTW_rUPPERCASE(string1)
```

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

예 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.Data는 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 갖는 단어 및 문자열 함수에 대해 다중 바이트 문자 세트(MBCS) 지원을 지정할 수 있습니다. Net.Data 초기화 파일에 이 값을 지정하십시오. 생략시 값은 지원되지 않습니다. Net.Data 매크로에서 DTW_MBMODE 변수를 설정하여 초기화 파일의 값을 겹쳐쓸 수 있습니다. 자세한 내용은 *Net.Data Administration and Programming Guide*와 106 페이지의 『DTW_MBMODE』의 구성 변수 절을 참조하십시오.

OS/400에 대한 MBCS 지원: DBCS 지원이 자동으로 제공되지 않으며 DBCS 지원에 이 변수가 필요하지 않습니다.

다음 함수는 Net.Data가 지원하는 단어 함수입니다.

- 193 페이지의 『DTW_DELWORD』
- 194 페이지의 『DTW_SUBWORD』
- 196 페이지의 『DTW_WORD』
- 197 페이지의 『DTW_WORDINDEX』
- 198 페이지의 『DTW_WORDLENGTH』
- 199 페이지의 『DTW_WORDPOS』
- 201 페이지의 『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 61. DTW_DELWORD 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	n	IN	삭제될 첫번째 단어의 단어 위치.
정수	<i>length</i>	IN	삭제할 단어 수. 생략시 값은 n 에서 <i>stringIn</i> 끝까지의 모든 단어를 삭제하는 것입니다. 생략 가능 매개변수.
문자열	<i>stringOut</i>	OUT	수정된 <i>stringIn</i> 유형이 들어 있는 변수.

예

예 1:

```
@DTW_DELWORD("Now is the time", "5", result)
```

- 결과: result = "Now is the time"

예 2:

```
@DTW_DELWORD("Now is the time", "2", result)
```

- 결과: result = "Now"

예 3:

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

- 결과: result = "Now time"

예 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 n 단어에서 시작하여 $length$ 에 의해 지정된 단어 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 62. DTW_SUBWORD 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	n	IN	부속 문자열 첫번째 단어의 단어 위치. 이 값이 <i>stringIn</i> 의 단어 수보다 큰 경우 널(NULL) 값이 리턴됩니다.
정수	<i>length</i>	IN	부속 문자열에 있는 단어 수. 이 값이 n 에서 <i>stringIn</i> 끝까지의 단어 수보다 큰 경우, <i>stringIn</i> 끝까지의 모든 단어가 리턴됩니다. 생략시 값은 n 에서 <i>stringIn</i> 끝까지의 모든 단어를 리턴하는 것입니다.
문자열	<i>stringOut</i>	OUT	n 과 <i>length</i> 에 의해 지정된 <i>stringIn</i> 의 부속 문자열이 들어 있는 변수.

예

예 1:

```
@DTW_SUBWORD("Now is the time", "5", result)
```

- 결과: result = ""

예 2:

```
@DTW_SUBWORD("Now is the time", "2", result)
```

- 결과: result = "is the time"

예 3:

```
@DTW_SUBWORD("Now is the time", "2", "2", result)
```

- 결과: result = "is the"

예 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 63. 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 큰 경우 널(NULL) 값이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	단어 위치 <i>n</i> 에 단어가 들어 있는 변수.

예

예 1:

```
@DTW_WORD("Now is the time", "3", result)
```

- 결과: result = "the"

예 2:

```
@DTW_WORD("Now is the time", "5", result)
```

- 결과: result = ""

예 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 문자열의 n 번째 단어 첫번째 문자의 위치를 리턴합니다.

형식

@DTW_WORDINDEX(stringIn, n, stringOut)

@DTW_rWORDINDEX(stringIn, n)

값

표 64. DTW_WORDINDEX 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>stringIn</i>	IN	변수 또는 리터럴 문자열.
정수	n	IN	색인할 단어의 단어 위치. 이 값이 입력 문자열의 단어 수보다 큰 경우 0이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 n 번째 단어의 문자 위치가 들어 있는 변수.

예

예 1:

```
@DTW_WORDINDEX("Now is the time", "3", result)
```

- 결과: result = "8"

예 2:

```
@DTW_WORDINDEX("Now is the time", "6", result)
```

- 결과: result = "0"

예 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 문자열의 n 번째 단어의 길이를 리턴합니다.

형식

@DTW_WORDLENGTH(stringIn, n, stringOut)

@DTW_rWORDLENGTH(stringIn, n)

값

표 65. 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 단어 위치. 이 값이 입력 문자열의 단어 수보다 큰 경우 0이 리턴됩니다.
문자열	<i>stringOut</i>	OUT	<i>stringIn</i> 의 n 번째 단어의 길이가 들어 있는 변수.

예

예 1:

```
@DTW_WORDLENGTH("Now is the time", "1", result)
```

- 결과: result = "3"

예 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 문자열에 있는 문자열의 첫번째 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 66. 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 클 경우 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

예 1:

```
@DTW_WORDPOS("the", "Now is the time", result)
```

- 결과: result = "3"

예 2:

```
@DTW_WORDPOS("The", "Now is the time", result)
```

- 결과: result = "0"

예 3:

```
@DTW_WORDPOS("The", "Now is the time", "5", result)
```

- 결과: result = "0"

예 4:

```
@DTW_WORDPOS("is the", "Now is the time", result)
```

- 결과: result = " 2"

예 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 67. 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 수가 들어 있는 변수.

예

예 1:

```
@DTW_WORDS("Now is the time", result)
```

- 결과:

```
result = "4"
```

예 2:

```
@DTW_rWORDS(" ")
```

- 결과: "0"

테이블 함수

이 함수는 Net.Data 테이블에 대한 작업을 단순화시키고, REXX, C 또는 Perl을 사용하여 사용자 함수를 작성하는 것보다 효율적입니다.

- 203 페이지의 『DTW_TB_APPENDROW』
- 204 페이지의 『DTW_TB_COLS』
- 205 페이지의 『DTW_TB_DELETEROW』
- 206 페이지의 『DTW_TB_DELETECOL』
- 207 페이지의 『DTW_TB_DLIST』
- 209 페이지의 『DTW_TB_DUMP』
- 210 페이지의 『DTW_TB_DUMPV』
- 211 페이지의 『DTW_TB_GETN』
- 212 페이지의 『DTW_TB_GETV』
- 213 페이지의 『DTW_TB_HTMLENCODE』
- 214 페이지의 『DTW_TB_INPUT_CHECKBOX』
- 215 페이지의 『DTW_TB_INPUT_RADIO』
- 216 페이지의 『DTW_TB_INPUT_TEXT』
- 218 페이지의 『DTW_TB_INSERTCOL』
- 220 페이지의 『DTW_TB_INSERTROW』
- 221 페이지의 『DTW_TB_LIST』
- 223 페이지의 『DTW_TB_MAXROWS』
- 224 페이지의 『DTW_TB_QUERYCOLNONJ』
- 225 페이지의 『DTW_TB_ROWS』
- 226 페이지의 『DTW_TB_SELECT』
- 228 페이지의 『DTW_TB_SETCOLS』
- 229 페이지의 『DTW_TB_SETN』
- 230 페이지의 『DTW_TB_SETV』
- 231 페이지의 『DTW_TB_TABLE』
- 233 페이지의 『DTW_TB_TEXTAREA』

DTW_TB_APPENDROW

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

목적

테이블의 끝에 하나 이상의 행을 추가합니다. 행이 테이블에 추가된 후 DTW_TB_SETV() 함수를 사용하여 새로운 행에 값을 할당하거나 테이블을 처리하도록 언어 환경으로 전달하십시오.

DTW_TB_APPENDROW()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

테이블에 허용된 총 행 수에 한계가 있고 추가할 행 수가 이 한계를 초과하는 경우, 호출측에 오류가 리턴됩니다.

형식

@DTW_TB_APPENDROW(table, rows)

값

표 68. DTW_TB_APPENDROW 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	행이 추가될 매크로 테이블 변수.
정수	<i>rows</i>	IN	<i>table</i> 에 추가할 행의 수.

예

예 1: 10개의 행을 테이블에 추가합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_APPENDROW(myTable, "10")
```

DTW_TB_COLS

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

목적

테이블의 현재 컬럼 수를 리턴합니다.

형식

@DTW_TB_COLS(table, cols)

@DTW_TB_rCOLS(table)

값

표 69. DTW_TB_COLS 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	컬럼 수가 리턴될 매크로 테이블 변수.
정수	<i>cols</i>	OUT	<i>table</i> 의 컬럼 수가 들어 있는 변수.

예

예 1: 컬럼 수를 검색하여 그 값을 *cols*에 할당합니다.

```
%DEFINE myTable = %TABLE
%DEFINE cols = ""
...
@FillTable(myTable)
...
@DTW_TB_COLS(myTable, cols)
```

예 2: 테이블의 현재 컬럼 수에 대한 값을 검색 및 표시합니다.

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>My table contains @DTW_TB_rCOLS(myTable) columns.
```

DTW_TB_DELETEROW

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

목적

start_row에 지정된 행에서 시작하여 하나 이상의 행을 삭제합니다.

DTW_TB_DELETEROW()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_DELETEROW(table, start_row, rows)

값

표 70. DTW_TB_DELETEROW 매개변수

자료 유형	매개변수	사용	설명
테이블	table	INOUT	행이 삭제될 매크로 테이블 변수.
정수	start_row	IN	삭제할 table의 첫번째 행의 행 번호.
정수	rows	IN	table에서 삭제할 행의 수.

예

예 1: 테이블의 행 10에서 시작하여 5개 행을 삭제합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETEROW(myTable, "10", "5")
```

예 2: 테이블의 모든 행을 삭제합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETEROW(myTable, "1", @DTW_TB_rROWS(myTable))
```

DTW_TB_DELETECOL

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

목적

*after_col*에 지정된 컬럼 다음에서 시작하여 테이블에서 하나 이상의 컬럼을 삭제합니다.

형식

@DTW_TB_DELETECOL(table, after_col, cols)

값

표 71. DTW_TB_DELETECOL 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	컬럼이 삭제될 매크로 테이블 변수.
정수	<i>after_col</i>	IN	후속 컬럼이 삭제될 컬럼의 컬럼 번호. 첫번째 컬럼을 삭제하려면, 0을 지정하십시오.
정수	<i>cols</i>	IN	<i>table</i> 에서 삭제할 컬럼 수.

예

예 1: 테이블에서 3번째 및 4번째 컬럼을 삭제합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETECOL(myTable, "3", "2")
```

예 2: 테이블에서 첫번째 컬럼을 삭제합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETECOL(myTable, "0", "1")
```

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, link, link_u, image, image_u)

@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u, image)

@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u)

@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link)

@DTW_TB_DLIST(table, term, def, termstyle, defstyle)

@DTW_TB_DLIST(table, term, def, termstyle)

@DTW_TB_DLIST(table, term, def)

@DTW_TB_DLIST(table, term)

@DTW_TB_DLIST(table)

값

표 72. DTW_TB_DLIST 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	HTML 정의 목록으로 표시할 매크로 테이블 변수를 지정하는 기호.
정수	<i>term</i>	IN	<i>term</i> 이름 값(<DT> 태그 뒤에 오는 텍스트)을 포함하는 <i>table</i> 내 컬럼의 컬럼 번호. 생략시 값은 첫번째 컬럼을 사용하는 것입니다.
정수	<i>def</i>	IN	term 정의 값(<DD> 태그 뒤에 오는 텍스트)을 포함하는 <i>table</i> 내 컬럼의 컬럼 번호. 생략시 값은 두번째 컬럼을 사용하는 것입니다.
문자열	<i>termstyle</i>	IN	<i>term</i> 이름 값에 대한 HTML 요소 목록이 들어 있는 변수 또는 리터럴 문자열. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
문자열	<i>defstyle</i>	IN	<i>term</i> 정의 값에 대한 HTML 요소 목록이 들어 있는 변수 또는 리터럴 문자열. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
문자열	<i>link</i>	IN	HTML 링크가 생성될 HTML 요소를 지정합니다. 유효값은 DT와 DD입니다. 생략시 값은 HTML 링크를 생성하지 않는 것입니다.

표 72. DTW_TB_DLIST 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>link_u</i>	IN	HTML 참조에 대한 URL이 들어 있는 <i>table</i> 내 컬럼 번호. 생략시 값은 HTML 링크를 생성하지 않는 것입니다.
문자열	<i>image</i>	IN	인라인 이미지조가 작성될 HTML 요소를 지정합니다. 유효값은 DT와 DD입니다. 생략시 값은 인라인 이미지(DT)를 생성하지 않는 것입니다.
정수	<i>image_u</i>	IN	인라인 이미지에 대한 URL이 들어 있는 <i>table</i> 내 컬럼 번호. 생략시 값은 인라인 이미지를 생성하지 않는 것입니다.

예

예 1: 아래 보인 것과 같이 테이블 자료에 따라 HTML을 산출하는 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/link1.html"><strong>link1text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b><i>image2text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link2.html"><strong>link2text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b><i>image3text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link3.html"><strong>link3text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT=""><b><i>image4text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link4.html"><strong>link4text</strong></A>
</DT>
</DL>
```

DTW_TB_DUMP

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

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 73. DTW_TB_DUMP 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	표시할 매크로 테이블 변수를 지정하는 기호.

예

예 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 74. DTW_TB_DUMPV 매개변수

자료 유형	매개변수	사용	설명
테이블	table	IN	표시할 매크로 테이블 변수를 지정하는 기호.

예

예 1:

@DTW_TB_DUMPV(Mytable)

이 예의 HTML은 다음과 같습니다.

```
<PRE>
http://www.mycompany.com/images/image1.gif
http://www.mycompany.com/link1.html
image1text
link1text
http://www.mycompany.com/images/image2.gif
http://www.mycompany.com/link2.html
image2text
link2text
http://www.mycompany.com/images/image3.gif
http://www.mycompany.com/link3.html
image3text
link3text
http://www.mycompany.com/images/image4.gif
http://www.mycompany.com/link4.html
image4text
link4text
</PRE>
```


DTW_TB_GETN

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

목적

*col*에 지정된 컬럼 번호가 있는지 컬럼 표제를 검색합니다.

DTW_TB_GETN()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_GETN(table, col, name)

@DTW_TB_rGETN(table, col)

값

표 75. DTW_TB_GETN 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	컬럼명이 리턴될 매크로 테이블 변수.
정수	<i>col</i>	IN	이름이 리턴될 컬럼의 컬럼 번호.
문자열	<i>name</i>	OUT	<i>col</i> 에 지정된 컬럼의 이름이 들어 있는 변수.

예

예 1: 컬럼 4의 컬럼 이름을 검색합니다.

```
%DEFINE myTable = %TABLE
%DEFINE name = ""
...
@FillTable(myTable)
...
@DTW_TB_GETN(myTable, "4", name)
```

예 2: 테이블의 마지막 컬럼의 컬럼 이름을 검색합니다.

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>The column name of the last column is @DTW_TB_rGETN(myTable, @DTW_TB_rCOLS(myTable))
```

DTW_TB_GETV

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

목적

*row*와 *col*에 지정된 행 및 컬럼 수에 대해 테이블 값을 검색합니다.

DTW_TB_GETV()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_GETV(table, row, col, value)

@DTW_TB_rGETV(table, row, col)

값

표 76. DTW_TB_GETV 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	테이블 값이 리턴될 매크로 테이블 변수.
정수	<i>row</i>	IN	리턴될 값의 행 번호.
정수	<i>col</i>	IN	리턴될 값의 컬럼 번호.
문자열	<i>value</i>	OUT	<i>row</i> 및 <i>col</i> 에 지정된 행과 컬럼의 값이 들어 있는 변수.

예

예 1: 행 6, 컬럼 3의 테이블 값을 검색합니다.

```
%DEFINE myTable = %TABLE
%DEFINE value = ""
...
@FillTable(myTable)
...
@DTW_TB_GETV(myTable, "6", "3", value)
```

예 2: 행 1, 컬럼 1의 테이블 값을 검색합니다.

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>The table value of row 1, column 1 is @DTW_TB_rGETV(myTable, "1", "1").
```

DTW_TB_HTMLencode

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

목적

다음 HTML 문자로 코드화된 입력 매크로 테이블을 리턴합니다.

이름	문자	코드
앰퍼샌드	&	&
큰 따옴표	"	"
보다 큼	>	>
보다 작음	<	<

형식

@DTW_TB_HTMLencode(table, collist)

값

표 77. DTW_TB_HTMLencode 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	수정할 매크로 테이블 변수.
문자열	<i>collist</i>	IN	코드화할 <i>table</i> 내의 컬럼 수. 생략시 값은 모든 컬럼을 코드화하는 것입니다.

예

예 1:

@DTW_TB_HTMLencode(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 체크박스 입력 태그를 리턴합니다.

형식

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol, valuecol, rows, checkedrows)

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol, valuecol, rows)

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol, valuecol)

@DTW_TB_INPUT_CHECKBOX(table, prompt, namecol)

값

표 78. DTW_TB_INPUT_CHECKBOX 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	체크박스 입력 태그로 표시할 매크로 테이블 변수.
문자열	<i>prompt</i>	IN	선택란 옆에 표시할 텍스트가 들어 있는 문자열 또는 <i>table</i> 내 컬럼 번호. 이 매개변수는 필수이지만, 널("") 값을 가질 수 있습니다. <i>prompt</i> 가 널 (NULL)인 경우, 사용되는 값은 <i>namecol</i> 에 대해 정의된 값입니다.
문자열	<i>namecol</i>	IN	입력 필드명이 포함된 문자열 또는 <i>table</i> 내 컬럼 번호.
정수	<i>valuecol</i>	IN	<i>table</i> 에서 입력 필드 값이 들어 있는 컬럼 번호. 생략시 값은 1입니다.
정수	<i>rows</i>	IN	입력 필드 생성에 사용되는 <i>table</i> 의 행 목록. 생략시 값은 모든 행을 사용하는 것입니다.
정수	<i>checkedrows</i>	IN	점검할 <i>table</i> 의 <i>rows</i> 을 지정하는 행 목록. 생략시 값은 필드를 점검하지 않는 것입니다.

예

예 1: 세 개의 선택란 입력 태그에 대한 HTML을 생성합니다.

```
@DTW_TB_INPUT_CHECKBOX(Mytable,"3","4","","2 3 4","1 3 4")
```

결과:

```
<INPUT TYPE="CHECKBOX" NAME="link2text" VALUE="1">image2text<BR>
<INPUT TYPE="CHECKBOX" NAME="link3text" VALUE="1" CHECKED>image3text<BR>
<INPUT TYPE="CHECKBOX" NAME="link4text" VALUE="1" CHECKED>image4text<BR>
```

DTW_TB_INPUT_RADIO

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

목적

매크로 테이블 변수로부터 하나 이상의 HTML 단일선택 버튼 입력 태그를 리턴합니다.

형식

@DTW_TB_INPUT_RADIO(table, prompt, namecol, valuecol, rows, checkedrows)

@DTW_TB_INPUT_RADIO(table, prompt, namecol, valuecol, rows)

@DTW_TB_INPUT_RADIO(table, prompt, namecol, valuecol)

값

표 79. DTW_TB_INPUT_RADIO 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	단일선택 버튼 입력 태그로 표시할 매크로 테이블 변수.
문자열	<i>prompt</i>	IN	단일선택 버튼 옆에 표시할 텍스트가 포함된 문자열 또는 <i>table</i> 내의 컬럼 번호. 이 매개변수는 필수이지만, 널("") 값을 가질 수 있습니다. <i>prompt</i> 가 널(NULL)인 경우, <i>valuecol</i> 의 값을 사용합니다.
문자열	<i>namecol</i>	IN	입력 필드명이 포함된 문자열 또는 <i>table</i> 내 컬럼 번호.
정수	<i>valuecol</i>	IN	<i>table</i> 에서 입력 필드 값이 들어 있는 컬럼 번호.
문자열	<i>rows</i>	IN	입력 필드 생성에 사용되는 <i>table</i> 의 행 목록. 생략시 값은 모든 행을 사용하는 것입니다.
정수	<i>checkedrows</i>	IN	해당 단일선택 버튼을 선택된 것으로 표시하기 위한 <i>table</i> 내 행 번호. 값은 하나만 허용됩니다.

예

예 1: 세 개의 단일선택 버튼 입력 태그에 대한 HTML을 생성합니다.

```
@DTW_TB_INPUT_RADIO(Mytable,"3","Radio4","4","2 3 4","4")
```

결과:

```
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link2text">image2text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link3text">image3text<BR>
<INPUT TYPE="RADIO" NAME="Radio4" VALUE="link4text" CHECKED>image4text<BR>
```

DTW_TB_INPUT_TEXT

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

목적

Net.Data 테이블의 지정된 행에 대해 HTML <INPUT> 태그를 리턴합니다. 예를 들어, Net.Data는 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)
```

```
@DTW_TB_INPUT_TEXT(table, prompt, namecol, valuecol, size, maxlen)
```

```
@DTW_TB_INPUT_TEXT(table, prompt, namecol, valuecol, size)
```

```
@DTW_TB_INPUT_TEXT(table, prompt, namecol, valuecol)
```

```
@DTW_TB_INPUT_TEXT(table, prompt, namecol)
```

값

표 80. DTW_TB_INPUT_TEXT 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	텍스트 입력 태그로 표시할 매크로 테이블 변수.
문자열	<i>prompt</i>	IN	입력 필드 옆에 표시될 텍스트가 포함된 문자열 또는 <i>table</i> 내 컬럼 번호. <i>prompt</i> 가 널(NULL)인 경우 텍스트가 표시되지 않습니다.
문자열	<i>namecol</i>	IN	<i>table</i> 에서 입력 필드명이 들어 있는 컬럼 번호.
정수	<i>valuecol</i>	IN	<i>table</i> 에서 기본 입력 필드 값이 들어 있는 컬럼 번호로, INPUT 태그의 VALUE 속성으로 지정됩니다. 생략시 값은 VALUE 속성 값을 생성하지 않는 것입니다.
정수	<i>size</i>	IN	입력 필드의 문자 수로서, INPUT 태그의 SIZE 속성에 대해 지정됩니다. 생략시 값은 가장 긴 생략시 입력 값의 길이이고, 생략시 입력 값이 없는 경우 10입니다.
정수	<i>maxlen</i>	IN	입력 문자열의 최대 길이로서, INPUT 태그의 MAXLENGTH 속성에 대해 지정됩니다. 생략시 값은 MAXLENGTH 속성 값을 생성하지 않는 것입니다.
정수	<i>rows</i>	IN	입력 필드 생성에 사용되는 <i>table</i> 의 행 목록. 생략시 값은 모든 행을 사용하는 것입니다.

예

예 1: 세개의 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="link1text" SIZE="35" MAXLENGTH="40">  
<P>image2text  
<INPUT TYPE="TEXT" NAME="image2text" VALUE="link2text" SIZE="35" MAXLENGTH="40">  
<P>image3text  
<INPUT TYPE="TEXT" NAME="image3text" VALUE="link3text" SIZE="35" MAXLENGTH="40">
```

DTW_TB_INSERTCOL

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

목적

*after_col*에 지정된 컬럼 다음에서 시작하여 하나 이상의 컬럼을 삽입합니다.

형식

@DTW_TB_INSERTCOL(table, after_col, cols)

값

표 81. DTW_TB_INSERTCOL 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	컬럼이 삽입될 매크로 테이블 변수.
정수	<i>after_col</i>	IN	다음에 새로운 컬럼이 삽입될 컬럼의 컬럼 번호. 테이블의 시작 부분에 컬럼을 삽입하려면 0을 지정하십시오.
정수	<i>cols</i>	IN	<i>table</i> 에 삽입할 컬럼 수.

예

예 1: 테이블의 끝에 5개의 컬럼을 삽입합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTCOL(myTable, @DTW_TB_rCOLS(myTable), "5")
```

예 2: 테이블의 시작 부분에 한 개의 컬럼을 삽입합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTCOL(myTable, "0", "1")
```

DTW_TB_INSERTROW

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

목적

*after_row*에 지정된 행 다음에서 시작하여 하나 이상의 행을 삽입합니다.

DTW_TB_INSERTROW()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_INSERTROW(table, after_row, rows)

값

표 82. DTW_TB_INSERTROW 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	행이 삽입될 매크로 테이블 변수.
정수	<i>after_row</i>	IN	뒤에 새로운 행이 삽입될 행의 번호. 테이블의 시작 부분에 행을 삽입하려면 0을 지정하십시오.
정수	<i>rows</i>	IN	<i>table</i> 에 삽입할 행 수.

예

예 1: 테이블의 다섯번째 행 다음에 하나의 행을 삽입합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTROW(myTable, "5", "1")
```

예 2: 테이블의 시작 부분에 세 개의 행을 삽입합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTROW(myTable, "0", "3")
```

DTW_TB_LIST

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

목적

HTML 목록을 리턴합니다.

형식

@DTW_TB_LIST(table, listtype, listitem, itemstyle, link_u, image_u)

@DTW_TB_LIST(table, listtype, listitem, itemstyle, link_u)

@DTW_TB_LIST(table, listtype, listitem, itemstyle)

@DTW_TB_LIST(table, listtype, listitem)

@DTW_TB_LIST(table, listtype)

값

표 83. DTW_TB_LIST 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	HTML 목록으로 표시할 매크로 테이블 변수를 지정하는 기호.
문자열	<i>listtype</i>	IN	생성할 목록 유형. 허용 가능한 값들은 다음과 같습니다. DIR MENU OL UL
정수	<i>listitem</i>	IN	목록 값(태그 뒤에 오는 텍스트)이 포함된 <i>table</i> 내 컬럼 번호. 생략시 값은 첫번째 컬럼을 사용하는 것입니다.
문자열	<i>itemstyle</i>	IN	용어 이름 값에 대한 HTML 요소 목록이 포함된 리터럴 문자열 또는 변수. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
정수	<i>link_u</i>	IN	HTML 링크에 대한 URL이 들어 있는 <i>table</i> 내 컬럼 번호. 이 값이 지정되지 않는 경우, HTML 링크가 생성되지 않습니다.
정수	<i>image_u</i>	IN	인라인 이미지에 대한 URL이 들어 있는 <i>table</i> 내 컬럼 번호. 이 값이 지정되지 않는 경우, 인라인 이미지가 생성되지 않습니다.

예

예 1: 순서화된 목록에 대한 HTML 태그를 생성합니다.

```
@DTW_TB_LIST(Mytable,"OL","4","TT U","2","1")
```

결과:

```
<TT><U>
<OL>
<LI><A HREF="http://www.mycompany.com/link1.html">
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT="">link1text</A>
<LI><A HREF="http://www.mycompany.com/link2.html">
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT="">link2text</A>
<LI><A HREF="http://www.mycompany.com/link3.html">
<
IMG SRC="http://www.mycompany.com/images/image3.gif" ALT="">link3text</A>
<LI><A HREF="http://www.mycompany.com/link4.html">
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT="">link4txt</A>
</OL>
</U></TT>
```

DTW_TB_MAXROWS

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

목적

테이블에 허용된 최대 행 수를 리턴합니다.

형식

@DTW_TB_MAXROWS(table, maxrows)

@DTW_TB_rMAXROWS(table)

값

표 84. DTW_TB_MAXROWS 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	최대 행 수가 리턴될 매크로 테이블 변수.
정수	<i>maxrows</i>	OUT	<i>table</i> 에 허용된 최대 행 수가 들어 있는 변수. 행 수에 대한 제한 없이 테이블이 정의된 경우, 0이 리턴됩니다.

예

예 1: 테이블에 허용된 최대 행 수를 검색하고 그 값을 리턴합니다.

```
%DEFINE myTable = %TABLE
%DEFINE maxrows = ""

@DTW_TB_MAXROWS(myTable, maxrows)
%IF (maxrows != "0")
    The maximum number of rows allowed is $(maxrows).
%ELSE
    There is no limit on the number of rows allowed.
%ENDIF
```

예 2: 최대 행 수에 대한 값을 검색 및 표시합니다.

```
%DEFINE myTable = %TABLE

%IF (@DTW_TB_rMAXROWS(myTable) != "0")
    The maximum number of rows allowed is @DTW_TB_rMAXROWS(myTable).
%ELSE
    There is no limit on the number of rows allowed.
%ENDIF
```

DTW_TB_QUERYCOLNONJ

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

목적

컬럼 표제와 연관된 컬럼 번호를 리턴합니다. 테이블에 컬럼 표제가 없는 경우, 0이 리턴됩니다.

DTW_TB_QUERYCOLNONJ()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_QUERYCOLNONJ(table, name, col)

@DTW_TB_rQUERYCOLNONJ(table, name)

값

표 85. DTW_TB_QUERYCOLNONJ 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	리턴될 컬럼 번호가 있는 매크로 테이블 변수.
문자열	<i>name</i>	IN	컬럼 번호가 리턴될 컬럼 표제의 이름. 테이블에 컬럼 표제가 없는 경우, 0이 리턴됩니다.
정수	<i>col</i>	OUT	<i>name</i> 에 이름이 지정된 컬럼의 컬럼 번호가 들어 있는 변수.

예

예 1: 이름이 SERIAL_NUMBER인 컬럼의 컬럼 번호를 검색합니다.

```
%DEFINE myTable = %TABLE
%DEFINE col = ""
```

```
@DTW_TB_QUERYCOLNONJ(myTable, "SERIAL_NUMBER", col)
```

예 2: 이름이 SERIAL_NUMBER인 컬럼의 컬럼 번호를 리턴합니다.

```
%DEFINE myTable = %TABLE
<P>The "SERIAL_NUMBER" column is column number @DTW_TB_rQUERYCOLNONJ(myTable, "SERIAL_NUMBER")
```

DTW_TB_ROWS

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

목적

테이블의 현재 행 수를 리턴합니다.

형식

@DTW_TB_ROWS(table, rows)

@DTW_TB_rROWS(table)

값

표 86. DTW_TB_ROWS 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	현재 행 수가 리턴될 매크로 테이블 변수.
정수	<i>rows</i>	OUT	<i>table</i> 의 현재 행 수가 들어 있는 변수.

예

예 1: 테이블의 현재 행 수를 검색하여 그 값을 *rows*에 할당합니다.

```
%DEFINE myTable = %TABLE
%DEFINE rows = ""
...
@FillTable(myTable)
...
@DTW_TB_ROWS(myTable, rows)
```

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, selectedrows)

@DTW_TB_SELECT(table, name, optioncol, size, multiple, rows)

@DTW_TB_SELECT(table, name, optioncol, size, multiple)

@DTW_TB_SELECT(table, name, optioncol, size)

@DTW_TB_SELECT(table, name, optioncol)

@DTW_TB_SELECT(table, name)

값

표 87. DTW_TB_SELECT 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	IN	SELECT 필드로서 표시할 매크로 테이블 변수.
문자열	<i>name</i>	IN	SELECT 필드의 NAME 속성 값.
정수	<i>optioncol</i>	IN	SELECT 필드의 OPTION 태그에 사용할 값이 있는 <i>table</i> 내 컬럼 번호. 생략시 값은 첫번째 컬럼을 사용하는 것입니다.
정수	<i>size</i>	IN	SELECT 필드의 OPTION 태그에 사용할 <i>table</i> 내 행 수. 생략시 값은 모든 행을 사용하는 것입니다.
문자열	<i>multiple</i>	IN	복수 선택이 허용되는지를 지정합니다. 생략시 값은 복수 선택을 허용하지 않는 N입니다.
문자열	<i>rows</i>	IN	SELECT 필드에 사용할 <i>table</i> 의 행 수. 생략시 값은 모든 행을 사용하는 것입니다.
문자열	<i>selectedrows</i>	IN	OPTION 태그가 체크표시되는 테이블의 행 목록. 하나 이상의 행을 지정하려면, 여러 매개변수가 Y로 설정되어 있어야 합니다. 생략시 값은 첫번째 항목을 선택하는 것입니다.

예

여러 개의 선택이 있는 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_SETCOLS

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

목적

테이블의 컬럼 수를 설정합니다. DTW_TB_SETN() 함수를 사용하여 컬럼 표제를 지정하십시오. 테이블에 대해 한 번만 DTW_TB_SETCOLS() 함수를 사용할 수 있습니다. 테이블의 컬럼 수를 변경하려면 DTW_TB_DELETECOL() 또는 DTW_TB_INSERTCOL() 함수를 사용하십시오.

형식

@DTW_TB_SETCOLS(table, cols)

값

표 88. DTW_TB_SETCOLS 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	컬럼 수가 설정될 매크로 테이블 변수.
정수	<i>cols</i>	IN	<i>table</i> 에 할당할 최초 컬럼 수.

예

예 1: 테이블에 대해 세 개의 컬럼을 할당하고 컬럼에 이름을 지정합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETCOLS(myTable, "3")
@DTW_TB_SETN(myTable, "Name", "1")
@DTW_TB_SETN(myTable, "Address", "2")
@DTW_TB_SETN(myTable, "Phone", "3")
```

DTW_TB_SETN

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

목적

컬럼 표제에 이름을 지정합니다. 컬럼 표제를 삭제하려면, 컬럼 표제 값을 널(NULL)로 지정하십시오.

DTW_TB_SETN()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

```
@DTW_TB_SETN(table, name, col)
```

값

표 89. DTW_TB_SETN 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	컬럼명이 설정될 매크로 테이블 변수.
문자열	<i>name</i>	IN	<i>col</i> 에 지정된 컬럼의 컬럼 표제에 지정된 문자열.
정수	<i>col</i>	IN	표제가 설정되고 있는 컬럼의 컬럼 번호.

예

예 1: 컬럼 표제 1-3에 대해 이름을 지정합니다.

```
%DEFINE myTable = %TABLE

@DTW_TB_SETCOLS(myTable, "3")
@DTW_TB_SETN(myTable, "Name", "1")
@DTW_TB_SETN(myTable, "Address", "2")
@DTW_TB_SETN(myTable, "Phone", "3")
```

예 2: 컬럼 2의 컬럼 표제를 삭제합니다. 이러한 삭제는 정의되지 않은 함수 호출에 대한 변수를 전달하면 수행됩니다. 기본적으로 이 변수는 널(NULL)의 값을 갖습니다.

```
%DEFINE myTable = %TABLE

@DTW_TB_SETN(myTable, nullVar, "2")
```

DTW_TB_SETV

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

목적

테이블에 값을 할당합니다. 테이블 값을 삭제하려면 값을 널(NULL)로 지정하십시오.

DTW_TB_SETV()를 호출하기 전에 테이블의 컬럼 수를 설정해야 합니다. DTW_TB_SETCOLS() 또는 DTW_TB_INSERTCOL() 함수를 사용하거나 설정할 언어 환경에 테이블을 전달하여 컬럼 수를 설정할 수 있습니다.

형식

@DTW_TB_SETV(table, value, row, col)

값

표 90. DTW_TB_SETV 매개변수

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	INOUT	테이블 값이 설정될 매크로 테이블 변수.
문자열	<i>value</i>	IN	<i>row</i> 및 <i>col</i> 에 지정된 행과 컬럼의 테이블 값에 지정된 문자열.
정수	<i>row</i>	IN	설정될 값의 행 번호.
정수	<i>col</i>	IN	설정될 값의 컬럼 번호.

예

예 1: 행 3 컬럼 3에 값을 지정합니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, "value3.3", "3", "3")
```

예 2: 행 4, 컬럼 2의 테이블 값을 삭제합니다. 이러한 삭제는 정의되지 않은 함수 호출에 대한 변수를 전달하면 수행됩니다. 기본적으로 이 변수는 널(NULL)의 값을 갖습니다.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, nullVar, "4", "2")
```

DTW_TB_TABLE

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

목적

매크로 테이블 변수로부터 HTML 테이블을 리턴합니다.

형식

@DTW_TB_TABLE(table, options, collist, cellstyle, link_u, image_u, url_text, url_style)

@DTW_TB_TABLE(table, options, collist, cellstyle, link_u, image_u, url_text)

@DTW_TB_TABLE(table, options, collist, cellstyle, link_u, image_u)

@DTW_TB_TABLE(table, options, collist, cellstyle, link_u)

@DTW_TB_TABLE(table, options, collist, cellstyle)

@DTW_TB_TABLE(table, options, collist)

@DTW_TB_TABLE(table, options)

@DTW_TB_TABLE(table)

값

표 91. 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 값은 속성을 사용하지 않는 것입니다. 유효한 값은 다음과 같습니다. <ul style="list-style-type: none"> • BORDER • CELLSPACING • WIDTH
문자열	<i>collist</i>	IN	HTML 테이블에서 사용할 <i>table</i> 의 컬럼 수. 생략시 값은 모든 컬럼을 사용하는 것입니다.
문자열	<i>cellstyle</i>	IN	각 TD 태그에 있는 텍스트 주위에 오게 되는 B와 I 같은 HTML 스타일 요소 목록. 생략시 값은 스타일 태그를 사용하지 않는 것입니다.
정수	<i>link_u</i>	IN	HTML 링크를 작성하는 데 사용되는 URL이 포함된 <i>table</i> 내 컬럼 번호. <i>collist</i> 에도 컬럼을 지정해야 합니다. 생략시 값은 HTML 링크를 생성하지 않는 것입니다.
정수	<i>image_u</i>	IN	인라인 이미지 작성에 사용되는 URL이 포함된 <i>table</i> 내 컬럼 번호. <i>collist</i> 에도 컬럼을 지정해야 합니다. 생략시 값은 이미지 태그를 생성하지 않는 것입니다.

표 91. DTW_TB_TABLE 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>url_text</i>	IN	HTML 링크나 인라인 이미지에 표시할 텍스트가 포함된 <i>table</i> 내 컬럼 번호. 생략시 값은 URL 자체를 사용하는 것입니다.
문자열	<i>url_style</i>	IN	<i>url_text</i> 에 지정된 텍스트에 대한 HTML 스타일 요소 목록. 생략시 값은 스타일 태그를 생성하지 않는 것입니다.

예

예 1: 경계선과, B(볼드) 및 I(이탤릭) 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>LINKURL
<TH>IMAGEURL
<TR>
<TD><i>link1text</i>
<TD><A HREF="http://www.mycompany.com/link1.html"><b>link1text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b>link1text</b>
<TR>
<TD><i>link2text</i>
<TD><A HREF="http://www.mycompany.com/link2.html"><b>link2text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b>link2text</b>
<TR>
<TD><i>link3text</i>
<TD><A HREF="http://www.mycompany.com/link3.html"><b>link3text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b>link3text</b>
</TABLE>
```

DTW_TB_TEXTAREA

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

목적

매크로 테이블 변수로부터 HTML TEXTAREA 태그를 리턴합니다.

형식

```
| @DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol, rows)
| @DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol)
| @DTW_TB_TEXTAREA(table, name, numrows, numcols)
| @DTW_TB_TEXTAREA(table, name, numrows)
| @DTW_TB_TEXTAREA(table, name)
```

값

표 92. 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 <i>table</i> 의 행 수입니다.
정수	<i>numcols</i>	IN	컬럼에 지정된 텍스트 영역의 폭. 생략시 값은 <i>table</i> 에서 가장 긴 행의 길이입니다.
정수	<i>valuecol</i>	IN	그 값이 텍스트 영역에 표시되는 <i>table</i> 내 컬럼 번호. 생략시 값은 첫번째 컬럼입니다.
문자열	<i>rows</i>	IN	TEXTAREA 태그 생성에 사용되는 <i>table</i> 내 행의 목록. 생략시 값은 모든 행을 사용하는 것입니다.

예

예 1: HTML TEXTAREA 태그를 생성하고 포함시킬 행을 지정합니다.

```
@DTW_TB_TEXTAREA(Mytable,"textarea5","3","70","4","1 3 4")
```

결과:

```
<TEXTAREA NAME="textarea5" ROWS="3" COLS="70">
link1text
link3text
link4text
<TEXTAREA>
```

플랫 파일 인터페이스 함수

플랫 파일 인터페이스(FFI)를 이용하면 플랫 파일에 자료를 저장할 수 있을 뿐만 아니라, 플랫 파일 소스(텍스트 파일)의 자료를 열고, 읽고, 조작할 수 있습니다. 다음과 같은 플랫 파일 인터페이스 내장 함수가 제공됩니다.

- 239 페이지의 『DTWF_APPEND』
- 241 페이지의 『DTWF_CLOSE』
- 242 페이지의 『DTWF_DELETE』
- 246 페이지의 『DTWF_OPEN』
- 248 페이지의 『DTWF_READ』
- 250 페이지의 『DTWF_REMOVE』
- 251 페이지의 『DTWF_SEARCH』
- 254 페이지의 『DTWF_UPDATE』
- 256 페이지의 『DTWF_WRITE』

다음 절에서는 FFI 내장 함수의 사용 방법 및 플랫 파일 소스에 대한 액세스 방법에 대해 설명합니다.

- 『플랫 파일 자료 소스에 액세스』
- 237 페이지의 『플랫 파일 인터페이스 분리문자』
- 238 페이지의 『파일 잠금』

플랫 파일 자료 소스에 액세스

FFI 함수 사용시 지정할 수 있는 디렉토리 및 서브디렉토리를 나열하고 path 문에 포함된 디렉토리에 없는 파일에 대한 보안을 제공하려면 *Net.Data* 초기화 파일에서 FFI_PATH 경로 구성 명령문을 사용하십시오. *Net.Data* 초기설정 파일에는 FFI_PATH 이 제공되어 있지 않습니다. 경로 구성 방법에 대해서는 *Net.Data Administration and Programming Guide*를 참조하십시오.

FFI_PATH는 다음 구문을 사용합니다.

FFI_PATH /path1;/path2;/path3...

매크로 기능으로 FFI 언어 환경을 호출하는 경우, FFI 함수의 *filename* 매개변수를 사용하여 FFI 함수가 작동되고 있는 플랫 파일에 대한 경로를 지정하십시오. 예를 들면 다음과 같습니다.

```
%DEFINE myfile = "/macros/myfile.txt" @DTWF_READ(myfile, ...)
```


다음 섹션에서 논의됩니다.

- 『Net.Data의 플랫폼 파일 위치 판별 방법』
- 236 페이지의 『플랫폼 파일 구성 규칙』
- 236 페이지의 『보안 관련 권장사항』
- 237 페이지의 『권한 부여 요건』

Net.Data의 플랫폼 파일 위치 판별 방법

Net.Data는 FFI 함수에 대한 *filename* 매개변수의 정보를 사용하여 Net.Data 초기설정 파일의 FFI_PATH 명령문을 탐색하고 지정된 디렉토리 또는 현재 디렉토리를 사용할지의 여부를 결정합니다.

FFI 함수에 파일명이 지정되면, Net.Data는 지정된 첫번째 경로에서 시작하여 FFI_PATH에 나열된 각 경로를 탐색하여 파일을 찾습니다. Net.Data는 이 때 발견된 첫번째 사본을 사용합니다. 파일이 없는 경우, Net.Data는 Net.Data가 수행되고 있는 프로세스나 스레드의 현재 작업 디렉토리에서 파일을 찾습니다.

예 : Net.Data는 FFI_PATH 구성 명령문을 사용하여 파일을 찾습니다.

FFI_PATH에는 다음 디렉토리가 들어 있습니다.

FFI_PATH /macros;/macros/org1;/macros/org2

그리고, 파일은 현재 디렉토리와 /macros/org1에 모두 위치합니다. 함수 호출이 다음과 같은 경우,

DTWF_READ("myfile.txt")

Net.Data는 /macros/org1/myfile.txt를 사용합니다.

DTWF_READ 함수가 기존 파일을 읽는 데 사용되고 myfile.txt의 파일명이 지정된 경우, Net.Data는 위에 지정된 경로 목록이 FFI_PATH에 들어 있다는 가정하에 /macros, /macros/org1 및 /macros/org2 디렉토리에서 파일을 탐색합니다.

현재 디렉토리 판별:

Net.Data의 현재 디렉토리는 사용중인 웹 서버의 구성에 따라 달라집니다.

- CGI를 사용하는 경우, 현재 디렉토리는 Net.Data가 수행되고 있는 디렉토리입니다.
- 웹 서버 API를 사용하고 있는 경우 현재 디렉토리는 다양해집니다. 서버의 생략시 요청 경로나 자원 맵핑이 변경되는 경우 현재 디렉토리도 변경될 수 있습니다.

플랫폼 파일 액세스 지정과 관련한 권장사항:

Net.Data가 플랫 파일 자료 소스에 액세스할 수 있는지 확인하려면 다음을 준수하십시오.

- DTWF_OPEN 함수를 사용하여 플랫 파일을 작성하는 경우, FFI_PATH에 있거나 현재 디렉토리인 디렉토리를 지정하십시오. 디렉토리를 지정하지 않으면, Net.Data가 현재 작업 디렉토리에 파일을 작성합니다.
- *filename* 매개변수에 디렉토리를 포함시킨 경우, Net.Data는 FFI_PATH에 지정된 디렉토리내의 서브디렉토리를 탐색하지 않으므로 FFI_PATH에 있는 경로와 일치하는 완전(full) 경로를 지정하십시오.
- 특히 웹 서버 API를 사용하고 있는 경우 *filename* 매개변수에 대한 절대 경로를 사용하십시오.

플랫 파일 구성 규칙

Net.Data 초기설정 파일에서 FFI_PATH를 추가 또는 갱신할 경우 다음 규칙을 따르십시오.

- FFI_PATH의 경로 명령문에는 유효한 인쇄가능 문자가 포함되어야 합니다. FFI에서 서는 의문 부호(?) 또는 큰 따옴표(“)가 들어 있는 경로를 사용할 수 없습니다.
- 매크로에서 *filename* 매개변수에 사용되는 모든 디렉토리와 서브디렉토리가 FFI_PATH에 지정되어야 합니다. FFI_PATH에 명시적으로 지정되지 않으면 *filename*에 나열된 경로의 서브디렉토리는 탐색되지 않습니다.
- FFI_PATH 명령문에 대해 절대 경로를 사용하십시오.

보안 관련 권장사항

FFI 함수가 Net.Data 초기설정 파일에서 FFI_PATH 명령문을 사용하여 액세스할 수 있는 파일을 지정할 수 있습니다. FFI는 명령문에 나열된 경로만 탐색하므로, 다른 디렉토리의 파일은 무단으로 액세스할 수 없도록 보호됩니다.

예를 들어, 아래와 같이 공용 또는 게스트 사용자 ID에 대한 디렉토리를 지정하여 FFI_PATH를 지정할 수 있습니다.

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

다음은 플랫 파일을 보호하기 위해 권장되는 사항들입니다.

- 플랫 파일 조작에 사용하기 적합한 디렉토리를 선택하십시오. 디렉토리에 대한 탐색을 제한하려면 이 디렉토리를 FFI_PATH에 추가해야 합니다.
- 매크로로 DTWF_REMOVE 또는 다른 반출(export) 조작을 수행할 때에는 현재 디렉토리에 있는 .dll 및 .cmd 확장자가 있는 파일을 제거하거나 교체하는 일이 없도록 주의하십시오.
- 시스템에 추가되는 매크로를 적절히 제어하여 시스템의 파일을 보호하기 위한 적절한 조치를 취하십시오.

- FFI_PATH에 경로를 지정하지 마십시오. 익명의 FTP 사용자가 경로에 기록할 수도 있습니다. 이 경우, 누군가가 이전에는 허용되지 않았던 조치를 허용하는 Net.Data 매크로를 시스템에 설치할 수 있습니다.
- Net.Data 초기설정 파일의 경로를 FFI_PATH에 추가하지 마십시오.

권한 부여 요건

Net.Data를 실행하는 사용자 ID는 FFI 내장 함수가 사용하는 파일에 대한 액세스권을 가져야 합니다. 자세한 내용은 *Net.Data Administration and Programming Guide*의 구성 장에서 Net.Data 파일에 대한 웹 서버 액세스권 지정과 관련된 부분을 참조하십시오.

플랫 파일 인터페이스 분리문자

성능을 향상시키려면, 일련의 SQL 요청을 통해 만들어진 Net.Data 표양식 출력을 플랫 파일로 유지하십시오. 차후의 요청에서는 SQL 요청을 다시 발행하는 대신 플랫 파일을 검색하면 됩니다.

Net.Data 플랫 파일은 Net.Data 테이블로부터 작성할 수 있고, Net.Data 테이블은 플랫 파일로부터 작성할 수 있습니다. 테이블과 플랫 파일간 변환을 하려면, 테이블내 컬럼과 플랫 파일내 레코드 사이의 대응을 정의해야 합니다. 분리문자는 요청된 변환 양식에 따라 파일을 여러 부분(행의 컬럼 등)으로 나눌 때 FFI가 사용하는 플래그 또는 분리자입니다. 분리문자는 플랫 파일의 레코드 부분을 구분하여 테이블의 컬럼으로 맵하는 방법과, 테이블의 컬럼을 플랫 파일의 레코드로 맵하는 방법을 정의할 수 있도록 하는 방안이 됩니다.

다음과 같은 두 가지 분리문자가 있습니다.

개행 문자(ASCII TEXT)

사용자 테이블이 한 컬럼으로 구성되어 있는 경우 이 변환을 사용하십시오. Net.Data는 해당 플랫 파일의 각 레코드를 테이블내 단일 행으로 대응시킵니다. 이런 경우, 플랫 파일내의 레코드들을 분리하는 정규 개행 문자가 유일하게 사용되는 분리문자입니다.

개행 문자 및 분리문자 문자열(DELIMITED)

사용자 테이블이 여러 컬럼으로 구성되어 있는 경우 이 변환을 사용하십시오. Net.Data가 테이블내 행으로부터 플랫 파일 레코드를 작성하는 경우, 항목간 분리자로서 분리문자 문자열을 씁니다. Net.Data가 플랫 파일로부터 테이블을 재작성하는 경우, 분리문자 문자열을 사용하여 테이블내 컬럼에 둘 각 행의 문자열을 결정합니다. 이런 경우, 정규 개행 문자는 테이블의 행에 해당하는 플랫 파일내의 레코드들을 분리하고, 분리문자 문자열은 단일 레코드내의 항목들을 분리합니다.

읽기 조작의 경우, 분리문자는 파일의 내용을 테이블의 행과 컬럼으로 분리합니다. 쓰기 조작의 경우, 분리문자는 테이블의 행과 컬럼에 값의 끝을 나타냅니다. Net.Data는 분리문자를 Net.Data 매크로 문자열로 FFI에 전달하며, DELIMITER 매개변수에 명시적으로 나열되지 않는 한 문자의 끝에 널(NULL) 문자를 포함시키지 않습니다.

분리문자에 널(NULL) 문자를 사용하려면, 두개의 큰 따옴표 안에 공백 문자열을 두는 (『"』) 대신 역슬래쉬와 0을 큰 따옴표로 묶어(『\0』) DELIMITER 매개변수를 지정하십시오. ASCIITEXT 변환을 지정하는 경우, Net.Data는 개행 문자를 분리문자로 사용하고 요청된 분리문자를 무시합니다.

쓰기 조작에 대해 읽기 조작과 다른 분리문자를 사용하는 경우 파일이 잘못 변경될 수 있습니다. Net.Data는 새로운 분리문자를 사용하여 파일을 작성합니다.

분리문자의 최대 길이는 256 문자입니다.

파일 잠금

DTWF_OPEN 및 DTWF_CLOSE 함수를 사용하여 플랫폼 파일을 잠글 수 있습니다. 이 기능을 사용하면, Net.Data는 플랫폼 파일을 예약하므로 다른 응용프로그램이 이 파일을 읽거나 갱신할 수 없게 됩니다.

파일을 잠그려면, DTWF_OPEN 함수를 사용하십시오. 이 함수는 파일을 다른 응용프로그램에 사용할 수 없도록 하고, 파일이 읽히고 갱신되는 중에 변경되는 것을 방지합니다.

파일의 잠금을 해제하려면, DTWF_CLOSE 함수를 사용하십시오. 이 기능은 파일을 해제하므로 다른 응용프로그램이 이 파일을 읽거나 갱신할 수 있습니다.

DTWF_APPEND

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

목적

테이블 변수 목차를 파일 끝에 기록합니다.

파일의 현재 내용은 DTWF_APPEND 사용시의 결과에 영향을 주며 특히, 마지막 행의 마지막 컬럼의 내용에 영향을 줍니다. 개행 문자가 파일의 마지막 행의 마지막 컬럼 값에 후속되는 경우, 추가된 자료는 새로운 행에 놓입니다. 그렇지 않은 경우 추가된 자료는 파일의 마지막 행의 일부가 됩니다.

형식

@DTWF_APPEND(filename, transform, delimiter, table, retry, rows)

@DTWF_APPEND(filename, transform, delimiter, table, retry)

@DTWF_APPEND(filename, transform, delimiter, table)

값

표 93. DTWF_APPEND 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	변수의 내용이 추가되는 파일의 이름. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	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. OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널 (NULL) 문자가 포함됩니다.

표 93. DTWF_APPEND 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>retry</i>	IN	파일을 즉시 첨부할 수 없는 경우, 재시도 횟수. 생략시 값은 재시도를 하지 않는 것입니다.
정수	<i>rows</i>	IN	첨부할 <i>table</i> 의 최대 행 수. 생략시 값은 모든 행을 첨부하는 것입니다. 0을 지정하면 모든 행을 첨부합니다.

예

예 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "DELIMITED", " ;", myTable)
```

예 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCIIITEXT", " ;", myTable)
```

예 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCIIITEXT", " ;", myTable, "0", "10")
```

DTWF_CLOSE

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

목적

DTWF_OPEN로 열린 파일을 닫습니다.

형식

@DTWF_CLOSE(filename, retry)

@DTWF_CLOSE(filename)

값

표 94. DTWF_CLOSE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	닫을 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
정수	<i>retry</i>	IN	파일을 즉시 닫을 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

예

예 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

목적

파일에서 레코드를 삭제합니다. (빈 파일은 삭제하지 않습니다.)

형식

@DTWF_DELETE(filename, transform, delimiter, retry, rows, startrow)

@DTWF_DELETE(filename, transform, delimiter, retry, rows)

@DTWF_DELETE(filename, transform, delimiter, retry)

@DTWF_DELETE(filename, transform, delimiter)

값

표 95. DTW_DELETE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	레코드를 삭제할 파일명. 호출이 성공적으로 완료 되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
정수	<i>retry</i>	IN	레코드를 즉시 삭제할 수 없는 경우 재시도 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 값은 1에서 시작하는 것입니다.

예

예 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "5000"  
    myRows = "2"  
%}  
@DTWF_DELETE(myFile, "Delimited", "|", myWait, myRows)
```

예 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myStart = "1"  
    myRows = "2"  
%}  
@DTWF_DELETE(myFile, "Asciitext", "|", "0", myRows, myStart)
```

DTWF_INSERT

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

목적

레코드를 파일에 삽입합니다.

형식

@DTWF_INSERT(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF_INSERT(filename, transform, delimiter, table, retry, rows)

@DTWF_INSERT(filename, transform, delimiter, table, retry)

@DTWF_INSERT(filename, transform, delimiter, table)

값

표 96. DTWF_INSERT 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	레코드를 삽입할 파일명. 호출이 성공적으로 완료 되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	파일에 삽입될 레코드의 테이블 변수. OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널 (NULL) 문자가 포함됩니다.
정수	<i>retry</i>	IN	파일에 즉시 기록할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.
정수	<i>rows</i>	IN	<i>table</i> 로부터 삽입할 최대 행 수. 생략시 값은 모든 행을 삽입하는 것입니다. 값 0은 모든 행을 삽입합니다.

표 96. DTWF_INSERT 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>startrow</i>	INOUT	삽입을 시작할 행 번호. 이 값이 파일의 전체 행 수보다 큰 경우, 마지막 레코드로 값이 변경되고 오류가 발생합니다. 0을 지정하면, 파일의 시작 부분에 삽입됩니다. 생략시 값은 1에서 시작하는 것입니다.

예

예 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "3000"
}%
@DTWF_INSERT(myFile, "Delimited", "|", myTable, myWait)
```

예 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

목적

파일을 엽니다. 파일이 존재하지 않으면, 파일명에 대한 절대 경로를 지정해야 하고 파일이 작성될 디렉토리는 FFI_PATH에 지정된 디렉토리와 일치해야 합니다. 절대 경로가 사용되지 않는 경우, 파일은 현재의 작업 디렉토리에서 열립니다. DTWF_OPEN은 파일을 열어줍니다. 그렇지 않은 경우 각 플랫폼 파일 조작 후 파일이 닫힙니다.

성능 정보: 파일이 열리는 횟수를 줄이려면 DTWF_OPEN을 사용하십시오.

파일은 DTWF_CLOSE를 사용하여 닫거나 매크로 처리가 종료될 때까지 열려 있습니다.

형식

@DTWF_OPEN(filename, mode, retry)

@DTWF_OPEN(filename, mode)

값

표 97. DTWF_OPEN 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	열 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>mode</i>	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 기존 파일을 덮어씁니다.) • a - 첨부할 파일을 엽니다. 해당 파일이 없는 경우 Net.Data는 파일을 작성합니다. • r+ - 읽기 및 쓰기를 위해 파일을 엽니다. • w+ - 읽기 및 쓰기를 위해 파일을 작성합니다. (같은 이름의 파일이 있는 경우 기존 파일을 덮어씁니다.) • a+ - 읽기 또는 첨부를 위해 첨부 모드로 파일을 엽니다. 해당 파일이 없는 경우 Net.Data는 파일을 작성합니다.
정수	<i>retry</i>	IN	파일을 즉시 열 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

예

예 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

목적

파일로부터 테이블 변수로 레코드를 읽습니다.

형식

@DTWF_READ(filename, transform, delimiter, table, retry, rows, startrow, columns)

@DTWF_READ(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF_READ(filename, transform, delimiter, table, retry, rows)

@DTWF_READ(filename, transform, delimiter, table, retry)

@DTWF_READ(filename, transform, delimiter, table)

값

표 98. DTWF_READ 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	테이블 변수로 레코드를 읽을 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블의 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	OUT	파일 레코드가 읽힐 테이블 변수. OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널 (NULL) 문자가 포함됩니다.
정수	<i>retry</i>	IN	파일을 즉시 읽을 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

표 98. DTWF_READ 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>rows</i>	INOUT	테이블로 읽어들이길 파일 레코드의 최대 수. 생략 시 값은 모든 레코드를 읽거나, 테이블이 꽉 찰 때까지 읽는 것입니다. 0의 값은 파일의 끝까지 읽는 것을 의미합니다. 결과 테이블의 행 수가 리턴됩니다.
정수	<i>startrow</i>	IN	읽기를 시작할 파일의 레코드. 생략시 값은 첫번째 레코드에서 읽기 시작하는 것입니다.
정수	<i>columns</i>	OUT	테이블의 컬럼 수를 리턴합니다.

예

예 1:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "1000"
}%
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait)
```

예 2:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "0"
  myRows = "0"
  myStartrow = "1"
  myColumns = ""
}%
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait, myRows,
            myStartrow, myColumns)
```

예 3:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
}%
@DTWF_READ(myFile, "ASCITEXT", ";", myTable)
@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

목적

전체 파일을 삭제합니다.

형식

@DTWF_REMOVE(filename, retry)

@DTWF_REMOVE(filename)

값

표 99. DTW_REMOVE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	삭제할 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
정수	<i>retry</i>	IN	파일을 즉시 삭제할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

예

예 1:

```
%DEFINE myFile = "c:/private/myfile"  
@DTWF_REMOVE(myFile)
```

예 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

목적

테이블 변수로 문자열 탐색 결과를 리턴합니다.

형식

@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows, startrow)

@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows)

@DTWF_SEARCH(filename, transform, delimiter, table, searchFor, retry)

@DTWF_SEARCH(filename, transform, delimiter, table, searchFor)

값

표 100. DTWF_SEARCH 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	탐색할 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블의 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.

표 100. DTWF_SEARCH 매개변수 (계속)

자료 유형	매개변수	사용	설명
테이블	<i>table</i>	OUT	<p>탐색 결과가 위치할 테이블 변수. <i>transform</i>이 DELIMITED인 경우 다음과 같은 3개 컬럼이 리턴됩니다.</p> <ul style="list-style-type: none"> • 일치가 발견된 행. • 일치가 발견된 컬럼. • 파일에서의 일치 컬럼. <p>OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널 (NULL) 문자가 포함됩니다.</p>
문자열	<i>searchFor</i>	IN	탐색할 문자들의 문자열.
정수	<i>retry</i>	IN	파일을 즉시 탐색할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.
정수	<i>rows</i>	INOUT	<i>table</i> 로 읽어들이는 최대 행 수. 생략시 값은 모든 행을 읽거나 <i>table</i> 이 가득찰 때까지 읽는 것입니다. 0을 지정하면 파일 끝까지 읽습니다. 이 매개변수에 의해 결과 테이블의 행 수가 리턴됩니다.
정수	<i>startrow</i>	IN	탐색을 시작할 파일의 레코드. 생략시 값은 1입니다. 즉, 첫번째 레코드에서 탐색을 시작합니다.

사용법

- DTWF_SEARCH 함수의 수행 결과 리턴되는 테이블에는 세 개의 컬럼이 있습니다. 처음 두 컬럼에는 일치하는 것이 발견된 행과 컬럼의 번호가 들어 있습니다. 마지막 컬럼에는 *SearchFor* 매개변수에 지정된 문자가 들어 있는 컬럼 값이 들어 있습니다. 예를 들어, 파일의 네번째 행에서 컬럼 3에 일치하는 문자가 들어 있는 경우, 리턴된 테이블에는 첫번째 컬럼에 숫자 4를 갖는 행이 있어서 출처가 되는 파일의 행을 나타냅니다. 리턴된 테이블의 두번째 컬럼에는 숫자 3이 있어서 일치하는 문자가 들어 있는 파일의 컬럼을 나타냅니다. 세번째 컬럼에는 완전한 컬럼 값이 있습니다.
- *SearchFor* 매개변수는 *delimiter* 매개변수의 내용을 포함할 수 없습니다.

예

예 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1000"  
    mySearch = "0123456789abcdef"  
@DTWF_SEARCH(myFile, "DELIMITED", ";",  
              myTable, mySearch, myWait)
```

예 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

목적

테이블 변수로부터 파일 레코드를 갱신합니다. 파일이 존재하지 않으면, 파일명에 대한 절대 경로를 지정해야 하고 파일이 작성될 디렉토리는 FFI_PATH에 지정된 디렉토리 와 일치해야 합니다. 절대 경로가 사용되지 않는 경우, 파일은 현재의 작업 디렉토리에 서 열립니다.

형식

@DTWF_UPDATE(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF_UPDATE(filename, transform, delimiter, table, retry, rows)

@DTWF_UPDATE(filename, transform, delimiter, table, retry)

@DTWF_UPDATE(filename, transform, delimiter, table)

값

표 101. DTWF_UPDATE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	테이블 변수로부터 레코드가 갱신되는 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블의 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	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. <p>OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널 (NULL) 문자가 포함됩니다.</p>
정수	<i>retry</i>	IN	파일에 즉시 기록할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.

표 101. DTWF_UPDATE 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>rows</i>	IN	<i>table</i> 에서 갱신될 최대 레코드 수. 생략시 값은 모든 레코드를 갱신하는 것입니다. 0의 값은 파일내 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

예 1:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "1500"
  myRows = "2"
}%
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

예 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

목적

테이블 변수 목차를 파일에 기록합니다. 파일이 존재하지 않으면, 파일명에 대한 절대 경로를 지정해야 하고 파일이 작성될 디렉토리는 FFI_PATH에 지정된 디렉토리와 일치해야 합니다. 절대 경로가 사용되지 않는 경우, 파일은 현재의 작업 디렉토리에서 열립니다.

형식

@DTWF_WRITE(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF_WRITE(filename, transform, delimiter, table, retry, rows)

@DTWF_WRITE(filename, transform, delimiter, table, retry)

@DTWF_WRITE(filename, transform, delimiter, table)

값

표 102. DTWF_WRITE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>filename</i>	INOUT	테이블 변수의 레코드를 기록할 파일명. 호출이 성공적으로 완료되면, 이 매개변수는 완전한 파일명을 리턴합니다.
문자열	<i>transform</i>	IN	파일의 형식은 다음과 같습니다. <ul style="list-style-type: none"> • ASCIITEXT - 컬럼 값들 사이에 개행 문자를 사용하여 테이블을 파일에 기록하고, <i>delimiter</i> 매개변수를 무시합니다. • DELIMITED - <i>delimiter</i> 매개변수에 지정된 분리문자를 사용하여 테이블을 파일에 기록합니다. 파일의 개행 문자는 ASCIITEXT 및 DELIMITED 변환에 대해 Net.Data 매크로 테이블의 행의 끝을 나타냅니다.
문자열	<i>delimiter</i>	IN	값의 끝을 표시하는 문자열. 이 매개변수는 대소문자를 구분합니다. <i>transform</i> 이 ASCIITEXT인 경우 무시됩니다.
테이블	<i>table</i>	IN	행을 파일로 반출(export)하는 데 사용되는 테이블 변수. OS/400을 사용하지 않는 경우: FFI 테이블의 최대 행 길이는 16383 문자입니다. 이 한도에는 Net.Data 매크로 테이블의 각 컬럼에 대한 널(NULL) 문자가 포함됩니다.

표 102. DTWF_WRITE 매개변수 (계속)

자료 유형	매개변수	사용	설명
정수	<i>retry</i>	IN	파일에 즉시 기록할 수 없는 경우 재시도되는 횟수. 생략시 값은 재시도를 하지 않는 것입니다.
정수	<i>rows</i>	IN	기록할 파일 레코드의 최대 수. 생략시 값은 전체 테이블을 쓰는 것입니다. 0의 값은 파일의 끝에 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

예 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "DELIMITED", ";", myTable)
```

예 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCIITEXT", ";", myTable, "5000")
```

예 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "ASCIITEXT", ";", myTable, "5000", "10", "50")
```

웹 레지스트리 함수

웹 레지스트리는 항목을 쉽게 추가, 검색, 삭제할 수 있도록 하기 위해 Net.Data에 의해 유지보수되는 키가 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 유지보수하기 위한 함수를 제공합니다.

- 259 페이지의 『DTWR_ADDENTRY』
- 260 페이지의 『DTWR_CLEARREG』
- 261 페이지의 『DTWR_CLOSEREG』
- 262 페이지의 『DTWR_CREATEREG』
- 263 페이지의 『DTWR_DELENTY』
- 264 페이지의 『DTWR_DELREG』
- 265 페이지의 『DTWR_LISTREG』
- 266 페이지의 『DTWR_LISTSUB』
- 267 페이지의 『DTWR_OPENREG』
- 268 페이지의 『DTWR_RTVENTRY』
- 269 페이지의 『DTWR_UPDATEENTRY』

제한사항: OS/2를 사용할 경우, *registry*, *registryVariable* 및 *registryData* 매개변수에 대해 별표(*)를 사용하지 마십시오.

DTWR_ADDENTRY

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

목적

항목을 웹 레지스트리에 추가합니다.

형식

@DTWR_ADDENTRY(registry, registryVariable, registryData, index)

@DTWR_ADDENTRY(registry, registryVariable, registryData)

값

표 103. DTWR_ADDENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	항목을 추가할 레지스트리명.
문자열	<i>registryVariable</i>	IN	추가할 레지스트리 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>registryData</i>	IN	추가할 레지스트리 항목의 <i>registryData</i> 문자열 부분의 값.
문자열	<i>index</i>	IN	추가할 색인 항목에 있는 <i>registryVariable</i> 문자열의 색인 부분 값. 이 매개변수는 선택적입니다. 색인 항목이 지정되면, 지정된 레지스트리에 추가됩니다.

예

예 1:

```
@DTWR_ADDENTRY("Myregistry", "Jones", "http://Advantis.com/~Jones/webproj")
```

예 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

목적

웹 레지스트리에서 항목을 제거합니다.

형식

@DTWR_CLEARREG(registry)

값

표 104. DTWR_CLEARREG 매개변수

자료 유형	매개변수	사용	설명
문자열	registry	IN	제거할 레지스트리명.

예

예 1:

```
@DTWR_CLEARREG("Myregistry")
```

DTWR_CLOSEREG

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

목적

웹 레지스트리를 닫습니다.

형식

@DTWR_CLOSEREG(registry)

값

표 105. DTWR_CLOSEREG 매개변수

자료 유형	매개변수	사용	설명
문자열	registry	IN	닫을 레지스트리명. 제한사항: 웹 레지스트리명에는 별표(*) 및 역슬래쉬(\)와 같은 특수 문자를 사용하지 마십시오.

예

예 1: 레지스트리를 닫습니다.

```
@DTWR_CLOSEREG("/qsys.lib/mylib.lib/myreg.file")
```

DTWR_CREATEREG

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

목적

새로 웹 레지스트리를 작성합니다.

Re

형식

@DTWR_CREATEREG(registry, security)

@DTWR_CREATEREG(registry)

값

표 106. DTWR_CREATEREG 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	작성할 레지스트리명. 제한사항: 웹 레지스트리명에는 별표(*) 및 역슬래쉬(\)와 같은 특수 문자를 사용하지 마십시오.
문자열	<i>security</i>	IN	레지스트리 작성시 보안 유형. UNIX 운영 체제에서, 생략시 보안은 레지스트리가 작성된 디렉토리과 동일합니다. 사용자, 그룹 및 공용의 3개 보안 그룹에 대해 보안을 지정하십시오. R은 읽기 권한을, W는 쓰기 권한을, X는 실행 권한을 부여합니다. 3개 그룹 모두에 전권을 부여하려면, 이 매개변수에 *RWX, *RWX, *RWX를 지정하십시오. 이 매개변수는 생략 가능합니다.

예

예 1:

@DTWR_CREATEREG("myRegistry")

예 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

목적

웹 레지스트리에서 항목을 삭제합니다.

형식

@DTWR_DELENTY(registry, registryVariable, index)

@DTWR_DELENTY(registry, registryVariable)

값

표 107. DTWR_DELENTY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	항목을 제거할 레지스트리명.
문자열	<i>registryVariable</i>	IN	제거할 레지스트리 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>index</i>	IN	색인 항목에 있는 <i>registryVariable</i> 문자열의 색인 부분 값. 이것은 선택적 매개변수입니다. 색인 항목이 지정된 경우, 레지스트리에서 제거됩니다.

예

예 1:

```
@DTWR_DELENTY("Myregistry", "Jones")
```

예 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

목적

웹 레지스트리 삭제

형식

@DTWR_DELREG(registry)

값

표 108. DTWR_DELREG 매개변수

자료 유형	매개변수	사용	설명
문자열	registry	IN	삭제할 레지스트리명.

예

예 1:

@DTWR_DELREG("Myregistry")

DTWR_LISTREG

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

목적

전체 웹 레지스트리를 나열합니다. DTWR_LISTREG는 사용자가 전달한 OUT 테이블 변수의 레지스트리 항목에 대한 정보를 리턴합니다. 테이블 변수는 매개변수로서 LISTREG 레지스트리 연산에 대한 FUNCTION 블록으로 전달되기 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 사용가능한 모든 레지스트리 항목이 나열됩니다. 한편 사용자가 테이블 행의 최대 수로 X 값을 지정했는데 지정된 레지스트리에 X 항목보다 많은 항목이 있는 경우, 첫번째 X 항목들만 표시되고, 오류 코드가 전송됩니다. 이 코드는 사용가능한 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 109. DTWR_LISTREG 매개변수

자료 유형	매개변수	사용	설명
문자열	registry	IN	표시할 레지스트리명.
문자열	registryTable	OUT	레지스트리 항목이 놓인 테이블 변수명.

예

예 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_LISTREG는 사용자가 전달한 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 사용가능한 모든 레지스트리 항목이 나열됩니다. 한편, 사용자가 테이블 행의 최대 수로 X 값을 지정했는데 지정된 레지스트리에 X 항목보다 많은 항목이 있는 경우, 첫번째 X 항목들만 표시되고, 오류 코드가 전송됩니다. 이 코드는 사용가능한 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 시스템상에서만 유효합니다.

형식

@DTWR_LISTSUB(registry, registryTable)

값

표 110. DTWR_LISTSUB 매개변수

자료 유형	매개변수	사용	설명
문자열	registry	IN	표시할 레지스트리명.
문자열	registryTable	OUT	레지스트리 항목이 놓인 테이블 변수명.

예

예 1:

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

@DTWR_LISTSUB("URLLIST", RegistryTable)
```


DTWR_OPENREG

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

목적

웹 레지스트리를 엽니다.

형식

@DTWR_OPENREG(registry, commit)

@DTWR_OPENREG(registry)

값

표 111. DTWR_OPENREG 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	열려는 레지스트리명.
문자열	<i>commit</i>	IN	<p>확약 제어하에서 레지스트리가 열릴지의 여부를 지정하는 리터럴 문자열 또는 단일 기호. 가능한 값은 다음과 같습니다.</p> <p>Y 확약 제어하에서 레지스트리를 엽니다.</p> <p>N 확약 제어하에서 레지스트리를 열지 않습니다.</p> <p>생략시 값은 N입니다.</p>

예

예 1: 확약 제어하에서 레지스트리를 엽니다.

```
@DTWR_OPENREG("/qsys.lib/mylib.lib/myreg.file", "Y")
```

DTWR_RTVENTRY

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

목적

웹 레지스트리 항목에서 registryData 문자열을 검색합니다.

형식

@DTWR_RTVENTRY(registry, registryVariable, registryData, index)

@DTWR_RTVENTRY(registry, registryVariable, registryData)

@DTWR_rRTVENTRY(registry, registryVariable, index)

@DTWR_rRTVENTRY(registry, registryVariable)

값

표 112. DTWR_RTVENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	검색할 항목이 들어 있는 레지스트리명.
문자열	<i>registryVariable</i>	IN	registryData 문자열이 검색되는 레지스트리 항목의 <i>registryVariable</i> 문자열 부분의 값.
문자열	<i>registryData</i>	OUT	<i>registryVariable</i> 에 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 부분 값. 이것은 선택적 매개변수입니다. 지정되는 경우, 색인 항목의 <i>registryData</i> 문자열이 리턴됩니다.

예

예 1:

```
%DEFINE RegistryData = ""
@DTWR_RTVENTRY("Myregistry", "Jones", RegistryData)
```

예 2:

```
@DTWR_RTVENTRY("URLLIST", "SMITH", RegistryData, "WORK_URL")
```

예 3:

```
@DTWR_rRTVENTRY("Myregistry", "Jones")
```

예 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

목적

지정된 레지스트리 항목에 대한 기존의 *registryData* 문자열 값을 호출자가 지정한 새로운 값으로 대체합니다. *registryVariable* 문자열은 변경할 수 없습니다.

형식

@DTWR_UPDATEENTRY(registry, registryVariable, newData, index)

@DTWR_UPDATEENTRY(registry, registryVariable, newData)

값

표 113. DTWR_UPDATEENTRY 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>registry</i>	IN	갱신할 항목이 들어 있는 레지스트리명.
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 신규 값.
문자열	<i>index</i>	IN	갱신할 색인 항목에 있는 <i>registryVariable</i> 문자열의 색인 부분 값. 이것은 선택적 매개변수입니다. 색인 항목이 지정된 경우, 갱신됩니다.

예

예 1:

```
@DTWR_UPDATEENTRY("Myregistry", "Jones", "http://advantis.com/~Jones/personal")
```

예 2:

```
@DTWR_UPDATEENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/personal", "WORK_URL")
```

영속적 매크로 함수

영속적 매크로 함수는 단일 트랜잭션내에서 영속성을 유지하는 매크로 블록을 정의하도록 함으로써 Net.Data에서의 트랜잭션 처리를 지원합니다. 이 함수를 사용하여 트랜잭션의 시작과 끝, 트랜잭션을 통해 영속성을 유지하는 HTML 블록, 트랜잭션내의 변경을 확약 또는 구간 복원할지의 여부를 정의할 수 있습니다.

- 271 페이지의 『DTW_ACCEPT』
- 273 페이지의 『DTW_COMMIT』
- 274 페이지의 『DTW_ROLLBACK』
- 275 페이지의 『DTW_RTVHANDLE』
- 276 페이지의 『DTW_STATIC』
- 277 페이지의 『DTW_TERMINATE』

DTW_ACCEPT

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

목적

영속적 매크로를 호출하는 데 사용되는 트랜잭션 핸들을 정의합니다. Net.Data에서는 웹 브라우저의 응답으로 매크로를 호출하는 URL에 트랜잭션 핸들이 포함되어야 합니다. 웹 서버로 요청이 전달되는 경우, 서버는 트랜잭션 핸들을 사용하여 트랜잭션을 처리하는 CGI 프로세스로 이 요청을 경로지정합니다.

DTW_TERMINATE()에 대한 호출이 들어 있는 마지막 논리 블록에 도달할 때까지 매크로에 있는 각 HTML 블록의 시작 부분에서 트랜잭션 핸들을 호출해야 합니다. 텍스트가 브라우저에 출력되기 전에 DTW_ACCEPT() 또는 DTW_TERMINATE()에 대한 호출이 없으면, Net.Data 오류가 발생합니다.

@DTW_STATIC() 함수에 지정된 시간종료 값을 대체하는 시간종료 값을 이 페이지에 대해 지정할 수 있습니다. 웹 서버는 사용자가 이 요청에 응답하기를 지정된 시간(초 단위) 동안 기다립니다.

매크로가 영속적 상태에 있지 않을 때 이 함수가 호출되면, Net.Data 오류가 발생합니다.

참고: 트랜잭션 핸들이 들어 있는 URL은 양식 누름 버튼의 조치나, 브라우저에 제공되는 페이지의 하이퍼텍스트 링크로 코드화할 수 있습니다.

형식

@DTW_ACCEPT(handle, timeout)

@DTW_ACCEPT(handle)

값

표 114. DTW_ACCEPT 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>handle</i>	IN	이 영속 트랜잭션에서 차후의 매크로 호출시 URL에 사용될 트랜잭션 핸들을 지정하는 변수 또는 리터럴 문자열.
정수	<i>timeout</i>	IN	이 포트에 대한 작업이 응답을 기다리는 초 단위 시간을 지정하는 변수 또는 리터럴 문자열. 이 값은 DTW_STATIC() 함수에 지정된 시간종료 값을 대체합니다.

예

예 1:

```
%DEFINE handle = ""
@DTW_RTVHANLDE(handle)

%HTML(REPORT){
@DTW_ACCEPT(handle)
...
%}
```

DTW_COMMIT

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

목적

최종 확약 범위 이후 확약 제어하에서 이루어진 자원에 대한 미결 변경사항을 영속적으로 만들고 새로운 확약 범위를 설정합니다.

형식

@DTW_COMMIT()

값

없음.

예

예 1: 확약을 지정합니다.

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

DTW_ROLLBACK

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

목적

최종 확약 범위를 현재의 확약 범위로 재설정합니다. 최종 확약 범위 이후 Net.Data가 수행되고 있는 프로세스에 대한 확약 제어하에서의 자원에 대한 모든 변경사항이 취소됩니다.

형식

@DTW_ROLLBACK()

값

없음.

예

예 1: 구간 복원을 지정합니다.

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


DTW_RTVHANDLE

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

목적

각 호출에서 이 매크로에 대해 고유하고, 스레드 정보, 시간소인 및 현재 사용자(있을 경우)의 조합에 기초하여 계산된 트랜잭션 핸들을 생성하여 리턴합니다. 트랜잭션 핸들은 연속적 트랜잭션의 일부로 지정된 URL이 HTTP 서버에 대해 고유하게 하고 유효한 요청으로 식별될 수 있도록 하는 데 사용할 수 있습니다.

형식

@DTW_RTVHANDLE(handle)

값

표 115. DTW_RTVHANDLE 매개변수

자료 유형	매개변수	사용	설명
문자열	<i>handle</i>	OUT	현재의 연속적 매크로에 대한 고유한 트랜잭션 핸들이 들어 있는 변수.

예

예 1: 트랜잭션 핸들을 검색하는 데 사용되는 handle 변수를 정의합니다.

```
%DEFINE handle = ""
@DTW_RTVHANDLE(handle)
```

DTW_STATIC

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

목적

전체 매크로가 영속적임을 나타냅니다. 이것이 매크로의 첫번째 명령문이 되어야 합니다. 별다른 언급이 없는 한, 여러 매크로 호출에 걸쳐 이 함수 호출이 영속적이게 된 다음에 매크로에 정의된 모든 변수가 호출되거나 프로세스가 종료됩니다.

함수 호출에 초단위 시간 종료 값을 지정하여 Net.Data가 수행되는 프로세스가 브라우저의 응답을 기다리는 시간의 양을 나타낼 수 있습니다. 시간종료 값이 만료되면, 프로세스가 종료되고 최종 확약 범위 이후 확약 제어하에서의 자원에 대한 모든 변경이 취소됩니다.

후속하는 @DTW_ACCEPT() 호출에 시간종료 값이 지정되면, Net.Data는 후속 호출의 값으로 이 값을 대체합니다. 이 호출이나 후속하는 @DTW_ACCEPT() 호출에 시간종료 값이 지정되지 않으면, 웹 서버의 생략시 시간종료 값이 사용됩니다.

형식

@DTW_STATIC(timeout)

@DTW_STATIC()

값

표 116. DTW_STATIC 매개변수

자료 유형	매개변수	사용	설명
정수	<i>timeout</i>	IN	이 트랜잭션을 처리하는 프로세스가 응답을 기다리는 초 단위 시간을 지정하는 변수 또는 리터럴 문자열.

예

예 1: 60초의 시간종료 값을 지정하는 DTW_STATIC()에 대한 호출

```
@DTW_STATIC("60")
```

DTW_TERMINATE

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

목적

영속 트랜잭션을 종료합니다. 최종 확약 범위 이후 확약 제어하에서 자원에 대해 행해진 모든 변경이 영속적이 됩니다.

DTW_TERMINATE 함수는 텍스트가 브라우저에 출력되기 이전에 영속 트랜잭션의 논리적 최종 HTML 블록이 시작될 때 호출됩니다. 블록내에서 함수 앞에 텍스트 출력이 나타나면 Net.Data 오류가 발생합니다. 응용프로그램 작성 방법에 따라 둘 이상의 논리적 최종 HTML 블록이 있을 수 있습니다. 매크로가 영속적 상태에 있지 않을 때 이 함수가 호출되면 Net.Data 오류가 발생합니다.

형식

@DTW_TERMINATE()

값

없음

예

예 1: 영속 트랜잭션을 종료합니다.

```
%HTML(QUIT){  
@DTW_TERMINATE()  
...  
%}
```

부록A. Net.Data Technical Library

Net.Data Technical Library는 다음의 Net.Data 웹 사이트에서 사용할 수 있습니다.

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

문서	설명
<i>OS/390용 Net.Data 관리 및 프로그래밍 안내서, OS/2, Windows NT 및 UNIX용 Net.Data 관리 및 프로그래밍 안내서, OS/400용 Net.Data 관리 및 프로그래밍 안내서</i>	Net.Data의 설치, 구성 및 호출에 대한 개념 및 TASK 정보에 들어 있습니다. 또한, Net.Data 매크로의 작성, Net.Data 성능 기술의 사용, Net.Data 언어 환경의 사용, 연결 관리 및 문제점 해결과 성능 조정을 위한 Net.Data 기록 및 추적의 사용 방법들을 설명합니다.
<i>Net.Data 참조서</i>	Net.Data 매크로 언어, 변수 및 내장 함수를 설명합니다.
<i>Net.Data 언어 환경 인터페이스 참조서</i>	Net.Data 언어 환경 인터페이스를 설명합니다.
<i>Net.Data 메시지 및 코드 참조서</i>	Net.Data 오류 메시지와 리턴 코드를 나열합니다.

부록B. DB2 WWW 연결

DB2 WWW 연결이 설치되어 있으면 Net.Data에서 기존의 응용프로그램을 수행할 수 있습니다. Net.Data 버전 2 기능의 장점을 이용하려면 응용프로그램을 갱신하는 것이 좋습니다.

DB2 WWW 언어 구문은 다음과 같습니다.

- 『EXEC_SQL』
- 『HTML_INPUT』
- 『HTML_REPORT』
- 『SQL』
- 282 페이지의 『SQL_MESSAGE』
- 283 페이지의 『SQL_REPORT』
- 283 페이지의 『SQL_CODE』

EXEC_SQL

이는 SQL 블록을 호출합니다. 함수라고 하기 보다는 SQL문이라고 부르십시오. 자세한 내용은 20 페이지의 『FUNCTION 블록』을 참조하십시오.

HTML_INPUT

이는 HTML 블록, INPUT과 동일합니다. 자세한 내용은 32 페이지의 『HTML 블록』을 참조하십시오.

HTML_REPORT

이는 HTML 블록, REPORT와 동일합니다. 자세한 내용은 32 페이지의 『HTML 블록』을 참조하십시오.

SQL

이는 Net.Data에서 FUNCTION(DTW_SQL) 함수로 호출되는 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.

예):

예 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>
%}
```

예 1: 동등한 Net.Data 매크로

```
%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

이는 Net.Data MESSAGE문과 동일합니다. 예를 보려면 54 페이지의 『MESSAGE 블록』을 참조하십시오.

SQL_REPORT

이는 Net.Data REPORT문과 동일합니다. 예를 보려면 59 페이지의 『REPORT 블록』을 참조하십시오.

SQL_CODE

이는 DB2 WWW 연결에 들어 있으며, 호환을 위해 Net.Data에서 지원됩니다. 이는 129 페이지의 『RETURN_CODE』와 동일합니다.

부록C. Net.Data 운영 체제

모든 Net.Data 기능이 각 운영 체제에서 지원되는 것은 아닙니다. 이 절에는 사용중인 운영 체제에 지원되는 기능이 나열되어 있습니다. **X**는 기능이 지원됨을 나타냅니다.

표 117. Net.Data 언어 환경

언어 환경	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
플랫 파일 인터페이스	X		X	X	X	X	X	X
IMS 웹	X		X	X			X	X
자바 애플릿	X	X	X	X		X	X	X
자바 응용프로그램	X		X				X	X
ODBC	X	X	X	X		X	X	X
Oracle	X							X
Perl	X	X	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

표 118. Net.Data 구성 변수

구성 변수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
CACHE_MACHINE	X							X
CACHE_PORT	X							X
DB2INSTANCE	X	X	X			X	X	X
DB2MSG5				X				
DB2PLAN				X				
DB2SSID				X				
DefaultDBCp				X				
DSNAOINI				X				
DTW_CACHE_MACRO				X				
DTW_CM_PORT	X	X	X					X
DTW_DIRECT_REQUEST	X	X	X	X		X	X	X
DTW_DO_NOT_CACHE_MACRO				X				
DTW_INST_DIR	X		X			X	X	X
DTW_LOG_DIR	X	X	X			X	X	X
DTW_LOG_LEVEL	X	X	X			X	X	X
DTW_MBMODE	X	X	X	X		X	X	X
DTW_REMOVE_WS				X				
DTW_SHOWSQL	X	X	X	X	X	X	X	X
DTW_SMTP_CHARSET					X			

표 118. Net.Data 구성 변수 (계속)

구성 변수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_SMTP_SERVER	X	X	X	X		X	X	X
DTW_SQL_ISOLATION					X			
DTW_SQL_NAMING_MODE					X			
DTW_VARIABLE_SCOPE	X	X	X			X	X	X
DTWR_CLOSE_REGISTRIES					X			
DTWR_CLOSE_REGISTRIES					X			
DTW_UNICODE	X	X	X			X	X	X
DTWR_CLOSE_REGISTRIES					X			

표 119. 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
DB_CASE	X	X	X	X	X	X	X	X
DB2PLAN				X				
DB2SSID				X				
DTW_APPLET_ALTTEXT	X	X	X	X		X	X	X
DTW_CURRENT_FILENAME	X	X	X	X	X	X	X	X
DTW_CURRENT_LAST_MODIFIED	X	X	X	X	X	X	X	X
DTW_DEFAULT_MESSAGE	X	X	X		X	X	X	X
DTW_DEFAULT_REPORT	X	X	X	X	X	X	X	X
DTW_EDIT_CODES					X			
DTW_HTML_TABLE	X	X	X	X	X	X	X	X
DTW_LOG_LEVEL	X	X	X			X	X	X
DTW_MACRO_FILENAME	X	X	X	X	X	X	X	X
DTW_MACRO_LAST_MODIFIED	X	X	X	X	X	X	X	X
DTW_MBMODE	X	X	X	X		X	X	X
DTW_MP_PATH	X	X	X	X	X	X	X	X
DTW_MP_VERSION	X	X	X	X	X	X	X	X
DTW_PRINT_HEADER	X	X	X	X	X	X	X	X
DTW_REMOVE_WS	X	X	X	X	X	X	X	X
DTW_SAVE_TABLE_IN	X	X	X	X	X	X	X	X
DTW_SET_TOTAL_ROWS	X	X	X	X	X	X	X	X
LOCATION				X				
LOGIN	X	X	X		X	X	X	X
Nn	X	X	X	X	X	X	X	X
NLIST	X	X	X	X	X	X	X	X
NULL_RPT_FIELD					X			
NUM_COLUMNS	X	X	X	X	X	X	X	X
NUM_ROWS					X			

표 119. Net.Data 변수 (계속)

변수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
PASSWORD	X	X	X		X	X	X	X
RETURN_CODE	X	X	X	X	X	X	X	X
ROW_NUM	X	X	X	X	X	X	X	X
RPT_MAX_ROWS	X	X	X	X	X	X	X	X
SHOWSQL	X	X	X	X	X	X	X	X
SQL_CODE	X	X	X	X	X	X	X	X
SQL_STATE	X	X	X	X	X	X	X	X
START_ROW_NUM	X	X	X	X	X	X	X	X
TOTAL_ROWS	X	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 120. Net.Data 함수

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_ACCEPT					X			
DTW_ADD	X	X	X	X	X	X	X	X
DTW_ADDQUOTE	X	X	X	X	X	X	X	X
DTW_ASSIGN	X	X	X	X	X	X	X	X
DTW_CACHE_PAGE	X							X
DTW_COMMIT					X			
DTW_CONCAT	X	X	X	X	X	X	X	X
DTW_DATE	X	X	X	X	X	X	X	X
DTW_DELSTR	X	X	X	X	X	X	X	X
DTW_DELWORD	X	X	X	X	X	X	X	X
DTW_DIVIDE	X	X	X	X	X	X	X	X
DTW_DIVREM	X	X	X	X	X	X	X	X
DTW_EXIT	X	X	X	X	X	X	X	X
DTW_FORMAT	X	X	X	X	X	X	X	X
DTW_GETCOOKIE	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
DTW_POS	X	X	X	X	X	X	X	X

표 120. Net.Data 함수 (계속)

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_POWER	X	X	X	X	X	X	X	X
DTW_QHTMLENCODE	X	X	X	X	X	X	X	X
DTW_REVERSE	X	X	X	X	X	X	X	X
DTW_ROLLBACK					X			
DTW_RVTHANDLE					X			
DTW_SENDMAIL	X	X	X	X	X	X	X	X
DTW_SETCOOKIE	X	X	X	X	X	X	X	X
DTW_SETENV	X	X	X	X	X	X	X	X
DTW_STATIC					X			
DTW_STRIP	X	X	X	X	X	X	X	X
DTW_SUBSTR	X	X	X	X	X	X	X	X
DTW_SUBTRACT	X	X	X	X	X	X	X	X
DTW_SUBWORD	X	X	X	X	X	X	X	X
DTW_TB_APPENDROW	X	X	X	X	X	X	X	X
DTW_TB_COLS	X	X	X	X	X	X	X	X
DTW_TB_DELETEROW	X	X	X	X	X	X	X	X
DTW_TB_DELETECOL	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_GETN	X	X	X	X	X	X	X	X
DTW_TB_GETV	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_INSERTCOL	X	X	X	X	X	X	X	X
DTW_TB_INSERTROW	X	X	X	X	X	X	X	X
DTW_TB_LIST	X	X	X	X	X	X	X	X
DTW_TB_MAXROWS					X			
DTW_TB_QUERYCOLNONJ	X	X	X	X	X	X	X	X
DTW_TB_ROWS	X	X	X	X	X	X	X	X
DTW_TB_SELECT	X	X	X	X	X	X	X	X
DTW_TB_SETCOLS	X	X	X	X	X	X	X	X
DTW_TB_SETN	X	X	X	X	X	X	X	X
DTW_TB_SETV	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_TERMINATE					X			
DTW_TIME	X	X	X	X	X	X	X	X

표 120. Net.Data 함수 (계속)

함수	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_TRANSLATE	X	X	X	X	X	X	X	X
DTW_UPPERCASE	X	X	X	X	X	X	X	X
DTW_UNLESCSEQ	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
DTWF_CLOSE	X		X	X	X		X	X
DTWF_DELETE	X		X	X	X		X	X
DTWF_INSERT	X		X	X	X		X	X
DTWF_OPEN	X		X	X	X		X	X
DTWF_READ	X		X	X	X		X	X
DTWF_REMOVE	X		X	X	X		X	X
DTWF_SEARCH	X		X	X	X		X	X
DTWF_UPDATE	X		X	X	X		X	X
DTWF_WRITE	X		X	X	X		X	X
DTWR_ADDENTRY	X		X		X			X
DTWR_CLEARREG	X		X		X			X
DTWR_CLOSEREG					X			
DTWR_CREATEREG	X		X		X			X
DTWR_DELENTY	X		X		X			X
DTWR_DELREG	X		X		X			X
DTWR_LISTREG	X		X		X			X
DTWR_LISTSUB	X		X					X
DTWR_OPENREG					X			
DTWR_RTVENTRY	X		X		X			X
DTWR_UPDATEENTRY	X		X		X			X

표 121. Net.Data 인터페이스

인터페이스 유형	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
FastCGI	X						X	
CGI	X	X	X	X	X	X	X	X
Java Beans								X
ICAPI(Internet Connection API)	X		X	X				X
ISAPI(Internet Server API)								X
Live Connection	X		X				X	X
Lotus Domino Go Web Server(GWAPI)	X		X	X				X
Netscape API(NSAPI)	X						X	X

표 121. Net.Data 인터페이스 (계속)

인터페이스 유형	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
Servlets	X			X				X

표 122. Net.Data 툴

툴	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
Administration Tool	X		X					X
NetObjects Fusion Plug-ins								X
Wizards	X		X			X	X	X

부록D. 주의사항

이 책은 국내에서 제공되는 제품과 서비스를 위해 개발되었습니다. 다른 나라에서는 이 책에 논의된 제품, 서비스 또는 기능이 제공되지 않을 수도 있습니다. 해당 지역에서 제공되는 제품 및 서비스에 대해서는 현지의 IBM 영업 대표에게 문의하십시오. 이 책에서 IBM의 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM의 제품, 프로그램 또는 서비스가 사용되어야 함을 의미하지는 않습니다. IBM의 지적 재산을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. 그러나, 비IBM 제품, 프로그램 또는 서비스의 작동을 평가 및 검증하는 것은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 출원중일 수 있습니다. 이 책을 제공한다고 해서 이 특허에 대한 사용권을 제공하는 것은 아닙니다. 특허 사용권에 대해서는 IBM 지적 재산권부(02-782-6028)로 문의하시기 바랍니다.

150-010

서울특별시 영등포구 여의도동 25-11, 한진해운빌딩
한국 아이.비.엠 주식회사
지적재산권부

다음 단락은 영국이나 이러한 조항이 현지법과 상충되는 국가에서는 적용되지 않습니다. IBM은 적법성, 판매가능성 또는 특정 목적에의 적합성 등 명시적이든 묵시적이든 어떤 종류의 보증도 제공함이 없이 『현상대로』 이 책을 제공합니다. 일부 국가에서는 거래시 명시적 또는 묵시적 보증을 포기할 수 없도록 하고 있으므로, 이 조항이 적용되지 않을 수도 있습니다.

이 책에는 기술상 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 책의 내용은 정기적으로 변경됩니다. 변경된 사항은 신판에 통합됩니다. IBM은 사전 통지없이 언제든지 이 책에 기술된 제품 및 프로그램을 개선 또는 변경할 수도 있습니다.

이 프로그램의 사용권 소유자가 (i) 독자적으로 작성된 프로그램과 다른 프로그램(이 프로그램을 포함하여) 간의 정보 교환이나 (ii) 교환된 정보의 상호이용 등의 목적으로 정보를 필요로 하는 경우에는 다음에 문의해야 합니다.

150-010

서울특별시 영등포구 여의도동 25-11, 한진해운빌딩
한국 아이.비.엠 주식회사
소프트웨어 사업본부

이러한 정보는 간혹 사용권료 지불을 포함하여 계약 조건에 따라 사용이 가능합니다.

이 책에 기술된 사용권 프로그램과 여기에 사용할 수 있는 기타 모든 사용권 제품은 IBM 고객 협약 또는 기타 양자간 동등한 협약의 조건하에서 IBM이 제공합니다.

비IBM 제품에 대한 정보는 이들 제품의 공급업자, 간행물 또는 기타 일반에 공개된 자료를 통해 얻으실 수 있습니다. IBM이 이들 제품을 테스트하지는 않았으므로 비IBM 제품과 관련한 성능의 정확성, 호환성 또는 기타 배상 청구에 대해서는 확인할 수 없습니다. 비IBM 제품의 성능에 대해서는 이들 제품의 공급업자에게 문의하십시오.

사용권:

이 책자에는 프로그래밍 기술과 다양한 작동 플랫폼을 보여주는 샘플 응용프로그램이 원어로 제공됩니다. 샘플 프로그램이 작성된 작동 플랫폼의 응용프로그래밍 인터페이스와 일치하는 응용프로그램의 개발, 사용, 판매 또는 배포 목적으로는 IBM에 비용을 지불하지 않고 어떤 형태로든 샘플 프로그램을 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건 하에서 충분한 테스트를 받지 않았습니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증할 수 없습니다. IBM의 응용프로그래밍 인터페이스와 일치하는 응용프로그램의 개발, 사용, 판매 또는 배포 목적으로는 IBM에 비용을 지불하지 않고 어떤 형태로든 샘플 프로그램을 복사, 수정 및 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 Microsystems사의 등록상표입니다.

Microsoft, Windows, Windows NT[®] 및 Windows 95 로고는 Microsoft Corporation의 등록상표입니다.

UNIX는 미국과 다른 나라에서 사용되는 X/Open Company Limited에 독점권이 있는 등록상표입니다.

그 밖의 회사, 제품 및 서비스명은 다른 회사의 등록상표이거나 서비스 상표입니다.

용어

가

경로명(path name). 오브젝트의 위치를 시스템에 알려줍니다. 경로명은 디렉토리명과 오브젝트명순으로 표시합니다. 각 디렉토리나 오브젝트명은 슬래쉬(/) 또는 역슬래쉬(\) 문자로 구분합니다.

경로(path). 파일을 찾는 데 사용되는 탐색 경로.

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.

라

라이브 연결(Live Connection). 연결 관리 프로그램과 여러 개의 client로 구성된 Net.Data 구성요소. 라이브 연결은 데이터베이스 및 자바 가상 기계 연결의 재사용을 관리합니다.

레지스트리(registry). 문자열을 저장 및 검색할 수 있는 저장소.

마

미들웨어(middleware). 응용프로그램과 네트워크 사이를 중재하는 소프트웨어. 이는 네트워크를 통해 클라이언트 응용프로그램과 서버 사이의 상호작용을 관리합니다.

바

방화벽(firewall). 외부의 무단 액세스로부터 내부의 네트워크를 보호하기 위한 소프트웨어가 설치된 컴퓨터.

사

상대 경로명(relative path name). 최상위 레벨 또는 "루트" 디렉토리에서 시작되는 않는 경로명. 시스템은 프로세스의 현재 작업 디렉토리에서 경로명이 시작하는 것으로 가정합니다.

아

애플릿(applet). HTML 페이지에 포함된 자바 프로그램. 애플릿은 Netscape Navigator와 같은 Java 가능 브라우저와 함께 작동하며, HTML 페이지가 처리될 때 로드됩니다.

언어 환경(language environment). Net.Data 매크로에서 DB2와 같은 외부 자료 소스 또는 Perl과 같은 프로그래밍 언어에 대한 액세스를 제공하는 모듈.

연결 관리 프로그램(Connection Manager). 라이브 연결 지원에 필요한 Net.Data내의 실행 파일인 dtwcm.

영속(persistence). 전체 트랜잭션에 대해 지정된 값을 계속 유지하는 상태. 여기서 트랜잭션은 여러 개의 Net.Data 호출에 관련됩니다. 변수만이 영속적일 수 있습니다. 또한, 확약 제어의 영향을 받는 자원에 대한 조작은 명시적 확약 또는 구간 복원이 수행될 때까지 또는 트랜잭션이 완료될 때 활성 상태입니다.

웹 서버(Web server). 인터넷 연결과 같은 HTTP 서버 소프트웨어를 수행하는 컴퓨터.

응용프로그램 프로그래밍 인터페이스(application programming interface (API)). 기능 인터페이스는 응용 체제 또는 사용권 프로그램의 특정 자료나 기능을 사용하기 위해 고급 언어로 쓰여진 응용프로그램을 허용하는 응용 체제 또는 별도로 주문가능한 사용권 프로그램에 의해 제공됩니다. Net.Data는 CGI 프로세스에서의 성능 향상을 위해 ICAP, GWAPI, ISAPI 및 NSAPI와 같은 독점적 웹 서버를 지원합니다.

인터넷(Internet). 국제 공용 TCP/IP 컴퓨터 네트워크.

인트라넷(Intranet). 회사 방어벽 내부의 TCP/IP 네트워크.

자

자료 유형(data type). 컬럼 및 리터럴의 속성.

자바(Java). 인터넷 응용프로그램에 특히 유용하며 운영 체제에 독립적이고 오브젝트 지향인 프로그래밍 언어.

작업 단위(unit of work). 하나의 조작으로 취급되는 회복가능한 일련의 조작. 작업 단위내의 모든 조작은 단일 조작인 것처럼 완료(확약) 또는 실행 취소(구간 복원)할 수 있습니다. 확약 제어의 영향을 받는 자원에 대한 조작만 확약 또는 구간 복원할 수 있습니다.

절대 경로(absolute path). 오브젝트의 전체 경로명. 절대 경로명은 최상위 레벨이나 "루트" 디렉토리(슬래쉬(/) 또는 역슬래쉬(\) 문자로 식별됨)로 시작합니다.

카

캐쉬 관리 프로그램(Cache Manager). 한 기계의 캐쉬를 관리하는 프로그램. 여러 개의 캐쉬를 관리할 수 있습니다.

캐쉬(cache). 최근에 액세스한 자료를 담고 있는 메모리나 디스크 공간의 일부로, 동일한 자료에 대한 차후 액세스의 속도를 높이기 위한 것입니다. 캐쉬는 종종 네트워크를 통해 액세스되는 자주 사용되는 자료의 국지 사본을 저장하는 데에도 사용됩니다.

캐싱(caching). 정보 갱신 시기가 될 때까지, 자주 사용되는 요청 결과를 신속한 검색을 위해 웹 서버에 국지로 저장하는 프로세스.

쿠키(cookie). HTTP 서버가 웹 브라우저에 전송하며, 브라우저가 HTTP 서버에 액세스할 때마다 브라우저가 반송하는 정보 패킷. 쿠키에는 서버가 선택하는 임의의 정보가 들어갈 수 있으며 별다른 언급이 없는 HTTP 트랜잭션 사이의 상태를 유지관리하는 데 사용됩니다. *Free Online Dictionary of Computing*

타

트랜잭션(transaction). 한 번의 Net.Data 호출. 영구 Net.Data가 사용되는 경우, 한 트랜잭션이 여러 개의 Net.Data 호출에 관련될 수 있습니다.

파

포트(port). TCP/IP와 고급 프로토콜 또는 응용프로그램간의 통신에 사용되는 16 비트 수.

플랫 파일 인터페이스(flat file interface). 일반 텍스트 파일의 자료를 읽고 쓸 수 있도록 하는 Net.Data 내장 함수 세트.

하

하이퍼텍스트 마크업 언어(hypertext markup language). 웹 문서를 작성하는 데 사용되는 태그 언어.

하이퍼텍스트 전송 프로토콜(hypertext transfer protocol). 웹 서버와 브라우저간에 사용되는 통신 프로토콜.

현재의 작업 디렉토리(current working directory). 모든 상대 경로명을 파악할 수 있는 프로세스의 생략시 디렉토리.

확약 제어(commitment control). 자원에 대한 조작이 작업 단위의 일부인 경우 Net.Data가 수행되는 프로세스내에서의 경계 설정.

A

API. 응용프로그램 인터페이스. Net.Data는 CGI 프로세스에서의 성능 향상을 위해 3가지 웹 서버 API를 지원합니다.

C

CGI. 공통 게이트웨이 인터페이스.

CGI(Common Gateway Interface). Web 서버가 응용프로그램에 제어를 전달하고 다시 자료를 수신할 때 사용하는 동기화 방법.

cliette. 웹 서버로부터의 요청을 처리하는 Net.Data 라이브 연결의 장시간 수행 프로세스. 연결 관리 프로그램은 이러한 요청을 처리하도록 cliette 프로세스의 일정을 계획합니다.

D

DBMS. 데이터베이스 관리 시스템.

Domino Go 웹 서버. Lotus Corp. 및 IBM에서 제공하는 웹 서버로서 일반 및 보안 연결을 모두 제공합니다. ICAPI와 GWAPI는 이 서버와 함께 제공되는 인터페이스입니다.

G

GWAPI. Go 웹 서버 API.

H

HTML. 하이퍼텍스트 마크업 언어(Hypertext markup language).

HTTP. 하이퍼텍스트 전송 프로토콜(Hypertext transfer protocol).

I

ICAPI. 인터넷 접속 API(Internet Connection API). *Domino Go* 웹 서버 참조.

ISAPI. Microsoft의 인터넷 서버 API.

L

LOB. 대형 오브젝트.

N

NSAPI. Netscape API.

P

Perl. 해석된 프로그래밍 언어의 하나.

T

TCP/IP. Transmission Control Protocol / Internet Protocol.

TCP/IP(Transmission Control Protocol / Internet Protocol). 근거리 통신망 및 광역 네트워크에 대해 피어 투 피어(Peer-to-peer) 연결성 기능을 제공하는 통신 프로토콜 세트.

U

URL. Uniform resource locator.

URL(uniform resource locator). HTTP 서버 및 선택적으로 디렉터리와 파일명을 지정하는 주소(예: <http://www.software.ibm.com/data/net.data/index.html>).

색인

[가]

값 그룹 전달 77
국지 DB2 부속시스템, ID 103
권한 부여 요건, FFI_PATH 237
기타 변수
 설명 118
 DTW_CURRENT_FILENAME 119
 DTW_CURRENT_LAST_MODIFIED 120
 DTW_DEFAULT_MESSAGE 121
 DTW_MACRO_LAST_MODIFIED 124
 DTW_MP_PATH 125
 DTW_MP_VERSION 126
 DTW_PRINT_HEADER 127
 DTW_REMOVE_WS 128
 RETURN_CODE 129
꼬리말 42

[나]

날짜 변수 118
날짜 형식, UTF-8 140
내장 함수 131

[다]

다음 버튼, RPT_MAX_ROWS 96
단어 함수
 DTW_DELWORD 193
 DTW_SUBWORD 194
 DTW_WORD 196
 DTW_WORDINDEX 197
 DTW_WORDLENGTH 198
 DTW_WORDPOS 199
 DTW_WORDS 201
 MBCS 지원 192
대문자, 지정 101
대소문자, SQL 명령에 지정 101
대체 텍스트, 웹 브라우저 104
데이터베이스 액세스 제한 111, 113
데이터베이스 일관성, 트랜잭션 범위 116
데이터베이스에 연결, DATABASE 변수 99

[라]

루핑 67

[마]

매개변수 전달, System 언어 환경 26
매개변수, 전달 26
매크로
 공통 구문 요소 5
 샘플 3
 선언 부분 2
 언어 구문 1
 전역 구문 1
 처리 중단 142
 포맷 3
 표시 부분 2
 행 길이 한계 4
매크로에서 전자 메일 송신 150
머리말 42
메세지, 생략시 텍스트 121
문자열
 값, 분리된 76
 설명 6
 숫자 비교 35, 67
 조건부 처리 35, 67

문자열 함수
 DTW_ASSIGN 176
 DTW_CONCAT 177
 DTW_DELSTR 178
 DTW_INSERT 179
 DTW_LASTPOS 181
 DTW_LENGTH 182
 DTW_LOWERCASE 183
 DTW_POS 184
 DTW_REVERSE 185
 DTW_STRIP 186
 DTW_SUBSTR 187
 DTW_TRANSLATE 189
 DTW_UPPERCASE 191
 MBCS 지원 175
문자열의 숫자 비교 35, 67

[바]

변수
 기타 118
 목록 76

변수 (계속)

- 보고서 89
- 숨겨진 75
- 실행가능 74
- 언어 환경 98
- 조건 72
- 테이블 77, 79
- 환경 74
- Net.Data, 개요 71

변수 참조 5

변수명 5

변수명 숨기기 75

보고서

- 포매팅 59
- Net.Data의 생략시 값 대체 91

보고서 변수

- 설명 89
- ALIGN 90
- DTW_DEFAULT_REPORT 91
- DTW_HTML_TABLE 92
- RPT_MAX_ROWS 93
- START_ROW_NUM 95

보안

- 로그인 ID 111
- 암호 113

보안 관련 권장사항, FFI_PATH 236

부속시스템 ID, DB2 부속시스템에 연결 103

분리된 값의 문자열 76

분리된 문자열 나열 76

분리문자, FFI 언어 환경

- ASCHTEXT 237
- DELIMITED 237

[사]

상한 65

선언 부분, 매크로 2

성능, DTW_EXIT 142

소문자, 지정 101

수학 함수

- DTW_ADD 163
- DTW_DIVIDE 164
- DTW_DIVREM 165
- DTW_FORMAT 167
- DTW_INTDIV 170

수학 함수 (계속)

- DTW_MULTIPLY 171
- DTW_POWER 172
- DTW_SUBTRACT 173

숨겨진 변수

- 단계 75
- 설명 75
- 예, HTML 양식 75

시간 형식, UTF-8 158

실행가능 변수

- 매개변수 사용 75
- 변수 참조로 75
- 설명 74
- 예 74

[아]

언어 구문

공통 구문 요소 5

매크로

- 구문 1
- 설명 7

문자열 6

변수 참조 5

변수명 5

함수 호출 28

COMMENT 블록 9

DB2 WWW 연결 281

DEFINE 블록 또는 명령문 11

ENVVAR 명령문 16

EXEC 블록 또는 명령문 17

FUNCTION 블록 20

HTML 블록 32

IF 블록 35

INCLUDE 명령문 42

INCLUDE_URL 명령문 45

LIST 명령문 48

MACRO_FUNCTION 블록 50

MESSAGE 블록 54

REPORT 블록 59

ROW 블록 62

TABLE 명령문 65

WHILE 블록 67

언어 환경 변수

설명 98

언어 환경 변수 (계속)

DATABASE 99
DB2PLAN 102
DB2SSID 103
DB_CASE 101
DTW_APPLET_ALTTEXT 104
DTW_EDIT_CODES 105
DTW_MBMODE 106
DTW_SAVE_TABLE_IN 107
DTW_SET_TOTAL_ROWS 108
LOCATION 110
LOGIN 111
NULL_RPT_FIELD 112
PASSWORD 113
SHOWSQL 114
SQL_STATE 115
TRANSACTION_SCOPE 116

영속적 매크로 함수

DTW_ACCEPT 271
DTW_COMMIT 273
DTW_ROLLBACK 274
DTW_RTVHANDLE 275
DTW_STATIC 276
DTW_TERMINATE 277

오류 처리 54

용어집 292

운영 체제 283

원격 DB2 부속시스템, 위치 110

웹 레지스트리 함수

DTWR_ADDENTRY 259
DTWR_CLEARREG 260
DTWR_CLOSEREG 261
DTWR_CREATEREG 262
DTWR_DELENTY 263
DTWR_DELREG 264
DTWR_LISTREG 265
DTWR_LISTSUB 266
DTWR_OPENREG 267
DTWR_RTVENTRY 268
DTWR_UPDATEENTRY 269

위치, DB2 부속시스템에 연결 110

위치, 플랫폼 파일 235

이전 버튼, RPT_MAX_ROWS 96

일반 함수 133

DTW_ADDQUOTE 134

일반 함수 133 (계속)

DTW_CACHE_PAGE 136
DTW_DATE 140
DTW_EXIT 142
DTW_GETCOOKIE 143
DTW_GETENV 145
DTW_GETINIDATA 146
DTW_HTMLENCODER 147
DTW_QHTMLENCODER 149
DTW_SENDMAIL 150
DTW_SETCOOKIE 154
DTW_SETENV 157
DTW_TIME 158
DTW_URLESCSEQ 160

[자]

절대 경로, 플랫폼 파일의 235

조건 변수

변수 참조 73

설명 72

예 76

LIST 명령문이 있는 73

조건부 문자열 처리 35, 67

주의사항 291

지원되는 가능 테이블 283

[카]

쿠키

전송 127

DTW_GETCOOKIE 143

DTW_PRINT_HEADER 127

DTW_SETCOOKIE 154

[타]

테이블

HTML로 된 결과 92

Net.Data, 행 수 지정 93

테이블 명령문 77

테이블 변수

설명 77

예 78

테이블 처리 변수

설명 79

테이블 처리 변수 (계속)

NLIST 81
NUM_COLUMNS 82
NUM_ROWS 83
Nn 80
ROW_NUM 84
SQL 언어 환경에 대해 지정 107
TOTAL_ROWS 85
VLIST 87
Vn 88
V_columnName 86

테이블 함수

DTW_TB_APPENDROW 203
DTW_TB_COLS 204
DTW_TB_DELETECOL 206
DTW_TB_DELETETROW 205
DTW_TB_DLIST 207
DTW_TB_DUMP 209
DTW_TB_DUMPV 210
DTW_TB_GETN 211
DTW_TB_GETV 212
DTW_TB_HTMLENCODE 213
DTW_TB_INPUT_CHECKBOX 214
DTW_TB_INPUT_RADIO 215
DTW_TB_INPUT_TEXT 216
DTW_TB_INSERTCOL 218
DTW_TB_INSERTTROW 220
DTW_TB_LIST 221
DTW_TB_MAXROWS 223
DTW_TB_QUERYCOLNONJ 224
DTW_TB_ROWS 225
DTW_TB_SELECT 226
DTW_TB_SETCOLS 228
DTW_TB_SETN 229
DTW_TB_SETV 230
DTW_TB_TABLE 231
DTW_TB_TEXTAREA 233

[파]

파일 위치 변수 118

파일 잠금해제, FFI 함수 238

파일 잠금, FFI 함수 238

표시 부분, 매크로 2

플랜, DB2 부속시스템에 연결 102

플랫 파일

구성 규칙 236

권한 부여 요건 237

보안 관련 권장사항 236

분리문자 237

액세스 234

액세스 권장사항 235

위치

현재 디렉토리 235

FFI_PATH 235

자료 소스 234

절대 경로 235

정의 234

파일 잠금 238

현재 디렉토리에 작성 235

FFI_PATH에 일치 235

플랫 파일에 액세스 234

플랫폼 지원 283

[하]

함수

값 그룹 전달 77

단어 192

명명 규칙 131

문자열 175

범용 133

설명 131

수학 162

영속적 270

웹 레지스트리 258

테이블 202

플랫 파일 인터페이스(FFI) 234

함수 호출

구문 28

설명 28

출력 포매팅 59

테이블 행 처리 62

INOUT 변수의 사용 30

함수에 대한 MBCS 지원

단어 함수 192

문자열 함수 175

행 길이 한계, 매크로 4

현재 디렉토리, 플랫 파일 판별 235

호출

외부 프로그램 17

호출 (계속)
 함수 28
 화면 이동, 다음 및 이전 버튼으로 96
 환경 변수
 설명 74
 예 74
 ENVVAR 명령문 16

A

ALIGN 90
 APPLET 태그, 대체 텍스트 104

C

COMMENT 블록
 구문 9
 설명 9

D

DATABASE 99
 DB2 WWW 연결, 언어 구문 281
 DB2 부속시스템에 연결
 부속시스템 ID 103
 위치 110
 DB2 플랜 102
 DB2PLAN 102
 DB2SSID 103
 DB_CASE 101
 DEFINE 명령문
 구문 11
 설명 11
 DEFINE 블록
 구문 11
 설명 11
 DTWF_APPEND 238
 DTWF_CLOSE 238, 241
 DTWF_DELETE 242
 DTWF_INSERT 244
 DTWF_OPEN 238, 246
 DTWF_READ 248
 DTWF_REMOVE 250
 DTWF_SEARCH 251
 DTWF_UPDATE 254
 DTWF_WRITE 256

DTWR_ADDENTRY 258
 DTWR_CLEARREG 260
 DTWR_CLOSEREG 261
 DTWR_CREATEREG 262
 DTWR_DELENTY 263
 DTWR_DELREG 264
 DTWR_LISTREG 265
 DTWR_LISTSUB 266
 DTWR_OPENREG 267
 DTWR_RTVENTRY 268
 DTWR_UPDATEENTRY 269
 DTW_ACCEPT 271
 DTW_ADD 162
 DTW_ADDQUOTE 134
 DTW_APPLET_ALTTEXT 104
 DTW_ASSIGN 80, 175, 176
 DTW_CACHE_PAGE 136
 DTW_COMMIT 273
 DTW_CONCAT 177
 DTW_CURRENT_FILENAME 119
 DTW_CURRENT_LAST_MODIFIED 120
 DTW_DATE 140
 DTW_DEFAULT_MESSAGE 121
 DTW_DEFAULT_REPORT 91
 DTW_DELSTR 178
 DTW_DELWORD 192
 DTW_DIVIDE 164
 DTW_DIVREM 165
 DTW_EDIT_CODES 105
 DTW_FORMAT 167
 DTW_GETCOOKIE 143
 DTW_GETENV 145
 DTW_GETINIDATA 146
 DTW_HTML_ENCODE 147
 DTW_HTML_TABLE 92
 DTW_INSERT 179
 DTW_INTDIV 170
 DTW_LASTPOS 181
 DTW_LENGTH 182
 DTW_LOG_LEVEL 122
 DTW_LOWERCASE 183
 DTW_MACRO_FILENAME 123
 DTW_MACRO_LAST_MODIFIED 124
 DTW_MBMODE 106
 DTW_MP_PATH 125

DTW_MP_VERSION 126
 DTW_MULTIPLY 171
 DTW_POS 184
 DTW_POWER 172
 DTW_PRINT_HEADER 127
 DTW_QHTMLENCODE 149
 DTW_REMOVE_WS 128
 DTW_REVERSE 185
 DTW_ROLLBACK 274
 DTW_RTVHANDLE 275
 DTW_SAVE_TABLE_IN 107
 DTW_SENMAIL 150
 DTW_SETCOOKIE 154
 DTW_SETENV 157
 DTW_SET_TOTAL_ROWS 108
 DTW_STATIC 276
 DTW_STRIP 186
 DTW_SUBSTR 187
 DTW_SUBTRACT 173
 DTW_SUBWORD 194
 DTW_TB_APPENDROW 203
 DTW_TB_COLS 204
 DTW_TB_deleteCOL 206
 DTW_TB_DELETEROW 205
 DTW_TB_DLIST 207
 DTW_TB_DUMPH 209
 DTW_TB_DUMPV 210
 DTW_TB_GETN 211
 DTW_TB_GETV 212
 DTW_TB_HTMLLENODE 213
 DTW_TB_INPUT_CHECKBOX 214
 DTW_TB_INPUT_RADIO 215
 DTW_TB_INPUT_TEXT 216
 DTW_TB_INSERTCOL 218
 DTW_TB_INSERTROW 220
 DTW_TB_LIST 218
 DTW_TB_MAXROWS 223
 DTW_TB_QUERYCOLNONJ 224
 DTW_TB_ROWS 225
 DTW_TB_SELECT 226
 DTW_TB_SETCOLS 228
 DTW_TB_SETN 229
 DTW_TB_SETV 230
 DTW_TB_TABLE 231
 DTW_TB_TEXTAREA 233

DTW_TERMINATE 277
 DTW_TIME 158
 DTW_TRANSLATE 189
 DTW_UPPERCASE 191
 DTW_URLESCSEQ 160
 DTW_WORD 196
 DTW_WORDINDEX 197
 DTW_WORDLENGTH 198
 DTW_WORDPOS 199
 DTW_WORDS 201

E

ENVVAR 명령문 74
 구문 16
 설명 16
 EXEC 명령문 74
 구문 17
 설명 17
 EXEC 블록
 구문 17
 설명 17
 EXEC_PATH 17
 EXEC_SQL 281

F

FFI 언어 환경
 구성 규칙 236
 권한 부여 요건 237
 보안 관련 권장사항 236
 분리문자 237
 파일 위치 235
 파일에 액세스 234
 현재 디렉토리 235
 FFI 언어 환경 구성 236
 FFI 언어 환경 호출 234
 FFI 함수
 파일 잠금 238
 파일 잠금해제 238
 DTWF_APPEND 239
 DTWF_CLOSE 241
 DTWF_DELETE 242
 DTWF_INSERT 244
 DTWF_OPEN 246
 DTWF_READ 248

FFI 함수 (계속)

DTWF_REMOVE 250

DTWF_SEARCH 251

DTWF_UPDATE 254

DTWF_WRITE 256

FFI_PATH

구문 234

구성 규칙 236

보안 관련 권장사항 236

플랫 파일 위치 235

플랫 파일에 액세스 234

filename 매개변수와 일치하는 경로 235

FUNCTION 블록

구문 21

설명 20

H

HTML

변수명 숨기기 75

양식, 사용자 ID 입력 111

양식, 암호 입력 113

테이블 결과 표시 92

HTML 블록

구문 32

설명 32

HTML_INPUT 블록 281

HTML_REPORT 블록 281

I

IF 블록

구문 35

설명 35

IN 키워드 23, 51, 131

INCLUDE 명령문

구문 42

설명 42

include 파일 42

INCLUDE_PATH 42

INCLUDE_URL 명령문

구문 45

설명 45

INOUT 변수

예 30

INOUT 키워드 23, 51, 131

L

LIST 명령문

구문 48

설명 48

list 변수

값 분리문자 77

설명 76

예 76

LOCATION 110

LOGIN 111

M

MACRO_FUNCTION 블록

구문 50

설명 50

MESSAGE 블록

구문 54

설명 54

N

Net.Data 테이블

상한 65

정의 65

NLIST 81

NULL_RPT_FIELD 112

NUM_COLUMNS 82

NUM_ROWS 83

Nn 80

O

OUT 키워드 23, 51, 131

P

PASSWORD 113

R

REPORT 블록

구문 59

설명 59

테이블 변수 77

ALIGN 90

REPORT 블록 (계속)

DTW_DEFAULT_REPORT 91

DTW_HTML_TABLE 92

NLIST 81

NUM_COLUMNS 82

NUM_ROWS 83

Nn 80

RPT_MAX_ROWS 93

START_ROW_NUM 95

TOTAL_ROWS 85

RETURNS 키워드 23

RETURN_CODE 129

ROW 블록

구문 62

설명 62

NLIST 81

NUM_COLUMNS 82

NUM_ROWS 83

Nn 80

ROW_NUM 84

TOTAL_ROWS 85

Vn 87, 88

V_columnName 86

ROW_NUM 84

RPT_MAX_ROWS 93

S

SHOWSQL 114

SQL

명령, 대소문자 지정 101

숨기기 또는 표시하기 114

SQL 블록 281

SQL 상태, 표시 115

SQL_CODE 283

SQL_MESSAGE 블록 282

SQL_REPORT 블록 283

SQL_STATE 115

START_ROW_NUM 95

System 언어 환경, 매개변수 전달 26

T

TABLE 명령문

구문 65

설명 65

TOTAL_ROWS 85

TRANSACTION_SCOPE 116

U

UTF-8 형식

날짜 140

시간 158

V

VLIST 87

Vn 88

V_columnName 86

W

WHILE 블록

구문 67

설명 67

IBM 한글 지원에 관한 설문



FAX : (02) 781-7778

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: Net.Data
참조서
버전 2 릴리스 2

성명			직위/담당업무	
회사명			부서명	
주소				
전화번호			팩스번호	
전자우편 주소				
사용중인 시스템	<input type="checkbox"/> 중대형 서버 <input type="checkbox"/> UNIX 서버 <input type="checkbox"/> PC 및 PC 서버			

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
☐ 한글 책자 ☐ 영문 책자
 (이유: _____)
2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가
3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가
4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가
5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
☐ 그다지 필요성을 느끼지 않음
6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☞ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.

귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

IBM