4D バージョン 2003
ランゲージ追加/修正情報

目次

目次

この追加修正情報では、新しいバージョンに加えられた主な変更点について、抜粋して掲載しています。これらの変更点は『ランゲージリファレンス』マニュアルにはまだ反映されていません。

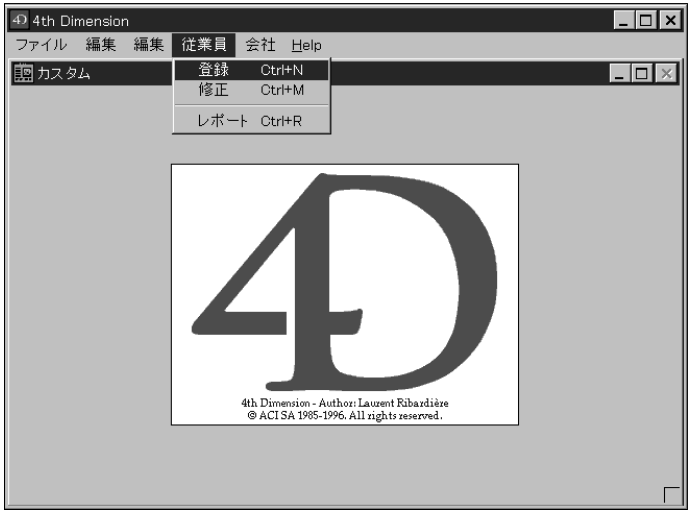
カスタムメニューの例

カスタムメニューは、「メニュー」エディタを使用して、各メニューコマンド（メニュー項目とも呼ばれる）にメソッドや自動アクションを関連付けるだけで簡単に作成することができます。

「ユーザの認識」の節では、ユーザがメニューを選択したときに起こる事象について説明します。次の節「メニュー表示の裏側」では、この作業を実現するための設計について説明します。事例はとても簡単ですが、カスタムメニューを用いると、データベースの使用や習得がいかに楽になるかがわかります。ユーザは、「ユーザ」モードの“一般的な”ツールやメニューではなく、ユーザの必要に応じた作業だけに目を向ければいいのです。

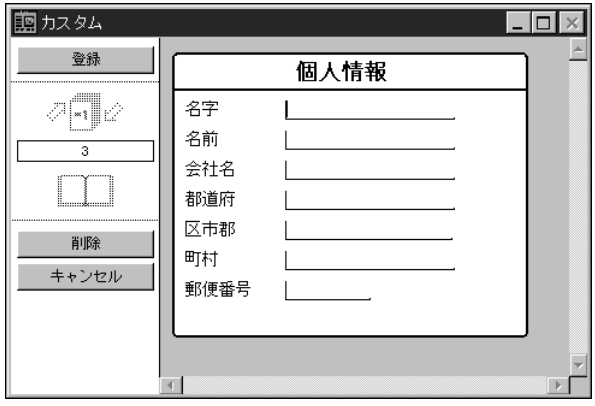
ユーザの認識

ユーザは新しい従業員を追加するために、“登録”というカスタムのメニューを選択します。

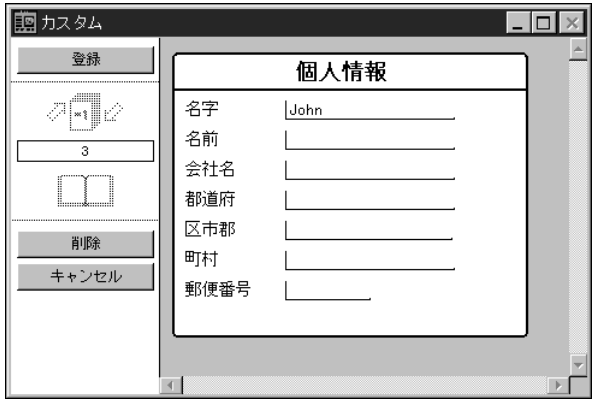


注：ここで使用している図の中には、バージョン3の4th Dimensionをもとに作成されたものが一部含まれています。

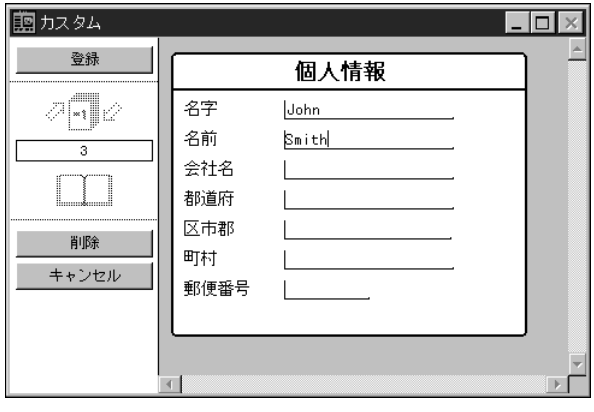
従業員テーブルの入力フォームが表示されます。



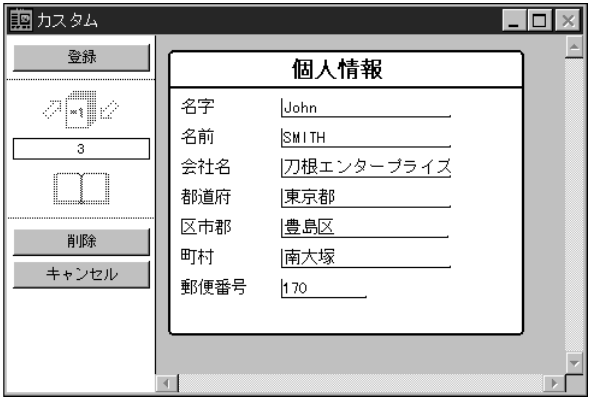
ユーザは、従業員の名字を入力します。



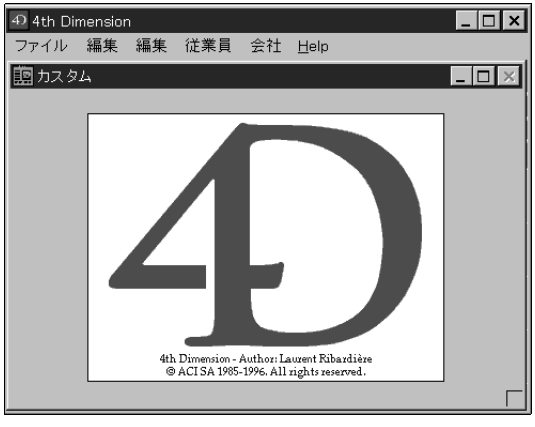
ユーザはタブで次のフィールドへ移動します。名前は大文字に変換されます。



ユーザは、レコードの入力を終了すると、「登録」ボタンをクリックします（通常は、ボタンバー上の最後のボタン）。



空の入力フォームが表示されるため、ユーザは「キャンセル」ボタンをクリックして「データ入力ループ」を終了します。再びメニューバーが表示されます。



アプリケーション開発用ツール

4Dアプリケーションの開発を進める上で、はじめは気付かなかったさまざまな機能を発見することでしょう。4D開発環境にその他ツールやプラグインを追加し、標準の4Dの機能を拡張することができます。

開発用ツール

4D社からアプリケーション開発に使用できる各種ツールが提供されています。これらのツールを使用して、別のデータベースからファイルやフィールド等のオブジェクトを移動したり、データベースのシンタックスチェックを行うことができます。次のようなツールがあります。

■「4D Insider」：このツールは、別のデータベースからテーブル、フィールド、メソッド、メニューバー、リスト、外部モジュール等を移動することができます。また、4th Dimensionデータベースのクロスリファレンスを作成することもできます。このツールはプロシージャや変数、コマンド、プラグイン、ストラクチャ、リスト、フォーム等を表示したり、印刷する場合に使用します。

■「4D Backup」：このツールは、入力されたデータを自動的に保存し、動作上の障害が発生した場合には情報を復元します。

4D プラグイン

4Dアプリケーションの機能は、4D開発環境に専門のプラグインを追加することにより拡張することができます。

4Dは、以下のプロダクティビティプラグインを提供しています。

■ 4D Draw：グラフィカル描画プログラム

■ 4D Write：ワードプロセッサ

■ 4D View：スプレッドシートとリストエディタ

4Dは、また以下のコネクティビティプラグインも提供しています。

■ 4D ODBC：ODBC経由の接続

■ 4D for OCI：ORACLE Call Interfaceとの接続用ツール

■ 4D Open for Java：Javaアプリケーションとの接続用ツール

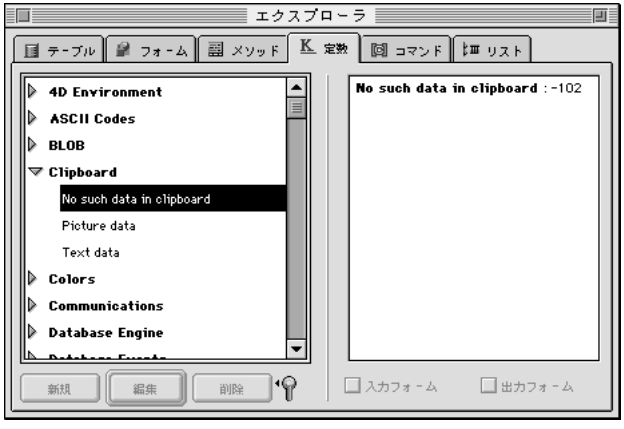
■ 4D Open：分散4D情報システムを構築するための4D間接続

定数

定数は、固定値を持つ式です。定数には、名前で選択できる「定義済定数」と実際の値を入力する「リテラル定数」の2種類があります。

定義済定数

4th Dimensionのバージョン6からあらかじめ定義されている「定義済定数」が導入されました。これらの定数は、「エクスプローラ」ウインドウに次のように一覧表示されています。



定義済定数は、テーマ別に一覧されています。「メソッドエディタ」ウインドウで定義済定数を使用するには、次の手順で行います。

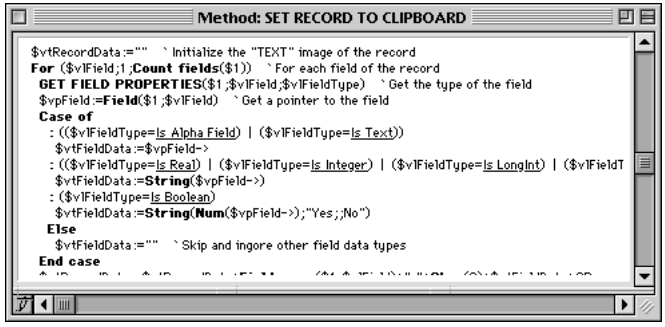
■「エクスプローラ」ウインドウから「メソッドエディタ」ウインドウに定数をドラッグ&ドロップします。

■「メソッドエディタ」ウインドウに定数の名前を直接入力します。

定義済定数名は、最大31文字まで指定できます。

Tips：定義済定数の名前を直接入力する場合に@記号（アットマーク）を使用すると、定数名すべてを入力しなくても済みます。例えば、「No such da@」と入力してからreturnキーまたはenterキーを押すと、4th Dimensionはコードの行の正当性を検査するときにその行に「No such data in clipboard」という定数を入れます。

「メソッドエディタ」ウインドウや「デバッグ」ウインドウにおいて、定義済定数はデフォルトとして下線付きで表示されます。



例えば、上記のウインドウでは「Is Alpha Field」が定義済定数です。

システム変数

OK

システム変数OKは、システム変数の中で最も頻繁に使用されます。一般的には、処理が成功すると1が代入され、成功しなかったときに0が代入されます。以下のコマンドは、システム変数OKに値を代入します。

ACCEPT	**ADD RECORD**	**ADD SUBRECORD**
Append document	**APPEND TO CLIPBOARD**	**APPLY TO SELECTION**
ARRAY TO LIST	**ARRAY TO SELECTION**	**ARRAY TO STRING**
LIST	**BLOB TO DOCUMENT**	**CALL WEB SERVICE**
CANCEL	**CHANGE ACCESS**	**CLOSE XML**
COMPRESS BLOB	**CONFIRM**	**Count XML attributes**
Count XML elements	**Create document**	**Create resource file**
DELETE DOCUMENT	**DELETE RESOURCE**	**DIALOG**
DISTINCT VALUES	**DOCUMENT TO BLOB**	**EXPAND BLOB**
EXPORT DIF	**EXPORT SYLK**	**EXPORT TEXT**
Get current printer	**Get First XML element**	**GET ICON RESOURCE**
Get indexed string	**Get Next XML element**	
GET PICTURE FROM LIBRARY	**GET PICTURE RESOURCE**	
GET PRINT OPTION	**GET RESOURCE**	**Get string resource**
Get text from clipboard	**Get text resource**	
GET XML ATTRIBUTE BY INDEX	**GET XML ATTRIBUTE BY NAME**	
Get XML element	**GET XML ELEMENT VALUE**	**GET XML ERROR**
IMPORT DIF	**IMPORT SYLK**	**IMPORT TEXT**

LOAD SET **LOAD VARIABLES** **MODIFY RECORD**
MODIFY SUBRECORD **Open document** **Open resource file**
ORDER BY **ORDER BY FORMULA** **Parse XML source**
Parse XML variable **PLAY** **PRINTERS LIST**
PRINT LABEL **PRINT SELECTION** **PRINT SETTINGS**
QUERY **QUERY BY EXAMPLE** **QUERY BY FORMULA**
QUERY SELECTION **QUERY SELECTION BY FORMULA**
RECEIVE PACKET **RECEIVE RECORD** **RECEIVE VARIABLE**
RELATE MANY SELECTION **RELATE ONE SELECTION** **REPORT**
Request **SAVE SET** **SAVE VARIABLES**
SELECT LOG FILE **SEND PACKET** **SEND RECORD**
SEND VARIABLE **SET CHANNEL** **SET CURRENT PRINTER**
SET PICTURE RESOURCE **SET PICTURE TO CLIPBOARD**
SET PRINT OPTION **SET RESOURCE** **SET RESOURCE NAME**
SET RESOURCE PROPERTIES **SET STRING RESOURCE**
SET TEXT RESOURCE **SET TEXT TO CLIPBOARD** **SET TIMEOUT**
START WEB SERVER **STRING LIST TO ARRAY** **USE ASCII MAP**
VALIDATE TRANSACTION

制御フロー

ユーザは、メソッドの単純性や複雑性に関係なく、プログラミング構造のシーケンス、分岐、ループの3種類のうち1つまたは複数をいつでも使用し、メソッド内でステートメントの実行や順序を制御します。

これらの構造のそれぞれを制御するステートメントがあります。この節では、これらのステートメントについての紹介にとどめます。

シーケンス構造

シーケンス構造は、単純な線形構造です。シーケンスは、4th Dimensionが最初から最後まで次々に実行する一連のステートメントです。例えば、以下のようなものです。

```
OUTPUT FORM (([人事]) ; "リスト")
ALL RECORDS ([人事])
DISPLAY SELECTION ([人事])
```

1行から成るルーチンも、オブジェクトメソッドに対しては頻繁に使用されます。最も簡単なシーケンス構造の例を次に示します。

[人事]名字:= **Uppercase** ([人事]名字)

分岐構造

分岐構造は、条件を判定した結果に応じて、メソッドに、別のパスを与えることができます。この条件は“True（真）”または“False（偽）”に評価される式、つまりブール式です。分岐構造には、**If...Else...End if**構造があります。これは、2つのパスのうちのいずれかにプログラムの流れを導きます。他の分岐構造には、**Case of...Else...End case**構造があり、これは多くのパスの中の1つだけにプログラムの流れを導きます。

注：4th Dimensionでは、512レベルの“深さ”まで入れ子構造のプログラム（If/While/For/Case of/Repeat）を作成することができます。

メソッド

コマンドや演算子がプログラミング言語として機能するには、それらをメソッド内に配置する必要があります。この章では、すべての種類のメソッドの共通機能について説明します。メソッドには、トリガ（テーブルメソッド）、フォームメソッド、プロジェクトメソッド、データベースメソッドがあります。オブジェクトメソッドも特別なメソッドの1つです。

メソッドは、ステートメントで構成されます。ステートメントとは、メソッドの1行のことで1つの命令を実行します。ステートメントは単純な場合もあれば、複雑な場合もあります。ステートメントは常に1行ですが最大32,000バイトまで使用することができます。これは、ほとんどの処理で十分な長さです。例えば、以下の行は[従業員]テーブルに新しいレコードを追加するステートメントです。

ADD RECORD ([従業員])

メソッドは、テストとループも含んでいます。これらは、制御フローをコントロールしています。制御フローに関する詳細は、後述の「制御フロー」の節を参照してください。

注：メソッドの最大サイズは、2GBのテキストまたはコマンド行32,000行までに制限されています。この制限を超えると、超過した行が表示されないことを知らせる警告メッセージが表示されます。

メソッドの例と用語

この節では、メソッドの用語、概念に共通する部分について説明するためにメソッドを詳しく調べてみることにします。この節に記述した内容の詳細は、このマニュアルの他の部分で説明されています。

メソッドは先頭の行から始まり、最後の行に到達するまで、各ステートメント（命令文）を実行します。基本的には、すべてのメソッドは同じです。次にプロジェクトメソッドの例を示します。

```
QUERY ([従業員])  `「クエリ」エディタを表示する
If (Records in selection ([従業員]) =0)  `何も見つからない場合、
    ADD RECORD ([従業員])  `レコードを追加する
End if `終了
```

まず、プログラミング言語の用語と機能を説明することにします。上記の各行を“ステートメント”または“コード行”と呼びます。プログラミング言語を使用して作成したものを、単にコードとも呼びます。4th Dimensionは、コードで指定した処理を実行します。

それでは、最初の行を詳しく見てみましょう。

```
QUERY ([従業員])  `「クエリ」エディタを表示する
```

この行の最初の要素である**QUERY**は、コマンドです。コマンドは、4th Dimensionのプログラミング言語の一部で、処理を実行します。**QUERY**コマンドは「クエリ」エディタを表示します。「ユーザ」モードの「クエリ」メニューから「検索」を選択することと同じ機能です。

この行の2番目の要素である括弧は、**QUERY**コマンドに対する引数（パラメータ）を指定します。引数は、コマンドが処理を実行するために必要なものです。この例では、[従業員]はテーブル名です。テーブル名は常に角カッコ（[...]）の中で指定します。つまり、“従業員テーブルが**QUERY**コマンドの引数である”ということを意味します。コマンドの中には複数の引数を持つものもあります。

3番目の要素は、行の終りに指定されたコメントです。コメントは逆アポストロフィ（`）によって示します。ユーザ（コードを解析する人）にこのコードが何を行っているのかを説明します。またコメント記号に続く内容は、コードを実行する時点で無視されます。コメントはそれ自体で1つの行になりますが、通常は上記の例で示すようにコードの右側に記述します。コメントを使用することにより、メソッドの内容を読みやすく、理解しやすいものにします。

注：コメントは、32,000桁まで入力することができます。

以下の行は、レコードが見つかったかどうかを調べます。

```
If (Records in selection ([従業員])=0)  `何も見つからない場合...
```

Ifはフロー制御ステートメント（“フロー制御文”または単に“文”）です。これはメソッドの実行の流れを制御します。**If**文は、カッコ内の式を判定し、結果が“True（真）”の場合、実行が以下の行に継続されます。本マニュアルでは、2つのブール値“True（真）”と“False（偽）”を使用します。

Records in selectionは関数です。これは値を返すコマンドです。ここでは、**Records in selection**関数は引数として渡されたテーブルのカレントセレクションのレコード数を返します。

注：関数名の頭文字だけが大文字になっていることに注意してください。これは、4th Dimensionの関数に対する命名規則によるものです。

既に、カレントセレクションとは何かについて説明しました。これは、その時点で作業対象となっているレコードの集まりのことです。レコードの数が0件の場合（レコードが全く見つからない場合）に以下の行を実行します。

```
ADD RECORD ([従業員])  ` レコードを追加する。
```

ADD RECORDコマンドは入力フォームを表示し、新しいレコードを追加します。この行はインデントされています。4th Dimensionは、自動的にコードをフォーマットします。この行は、先ほどのフロー制御ステートメント（**If**）に従属することを示すためにインデントされています。

```
End if ` 終了
```

End if文は**If**文の制御セクションを終了します。フロー制御ステートメントを使用した場合は、常に制御が終了する場所を示す対応ステートメントを指定する必要があります。

この節の概念をしっかりと把握し、理解するまで見直してください。

- オブジェクトメソッドとフォームメソッドに関する詳細は、第4章を参照してください。
- トリガに関する詳細は、第57章の「トリガコマンド」を参照してください。
- プロジェクトメソッドに関する詳細は、第6章を参照してください。
- データベースメソッドに関する詳細は、第6章を参照してください。

Structure file

Structure file→文字列

引数	タイプ		説明
このコマンドには、引数はありません。			
戻り値	文字列	←	データベースストラクチャファイルのロング名

説明

Structure file関数は、現在使用しているデータベースのストラクチャファイルのロング名を返します。

Windows上

例えば、ボリュームG上の「DOCS¥ MyCD」の中に配置されたデータベースを使って作業している場合、この関数は、「G:¥ DOCS¥ MyCD¥ myCD.4DB」を返します。

Macintosh上

例えば、ハードディスク「Macintosh HD」上の「MyCD f」フォルダの中に配置されたデータベースを使って作業している場合、この関数は、「Macintosh HD:MyCD f :My CD」を返します。

注：MacOS上で、4D Engineをマージしたアプリケーションからこの関数を呼び出した場合、関数はソフトウェアパッケージ内にあるストラクチャファイルのアクセスパスを返します。ソフトウェアパッケージ自体のアクセスパスを取得したい場合には、Application file関数を利用することをお勧めします。方法としては、Application type関数を使用してアプリケーションタイプを調べた後、その結果に応じてStructure file関数またはApplication type関数を実行します。

警告：4D Clientを実行している最中にこの関数を呼び出すと、ロング名ではなくストラクチャファイルの名前だけが返されます。

▼以下の例は、現在使用中のストラクチャファイルの名前と配置場所を表示します。

```
If (Application type#4D Client)
    $vsStructureFilename:= Long name to file name (Structure file)
    $vsStructurePathname:= Long name to path name (Structure file)
    ALERT("現在下記のデータベースを使用しています"+Char(34)
        +$vsStructureFilename+Char(34)+" located at "+Char(34)
        +$vsStructurePathname+Char(34)+".")
```

```
Else
    ALERT("次のデータベースに接続されます："+Char(34)
                                        +Structure file+Char(34))
End if
```

参照

Application file、Data file、DATA SEGMENT LIST

コンパイラコマンド

統合された4th Dimensionのコンパイラは、データベースアプリケーションをアセンブラ命令に翻訳します。コンパイラの利点は次の通りです。

■「スピード」：データベースの実行速度を3倍から1000倍速くします。

■「コードチェック」：データベースアプリケーションのコードを系統的にチェックし、論理的矛盾や構文的矛盾を検出します。

■「データベースの保護」：データベースをコンパイルすると、インタプリタコードが削除されます。コンパイルされたデータベースは、ストラクチャやメソッドを表示、または修正することができないこと以外は、オリジナルのデータベースと機能上は同一のものです。

■「スタンドアロンかつダブルクリックで起動するアプリケーション」：コンパイル後のデータベースは、独自のアイコンを持つスタンドアロンアプリケーション（.EXEファイル）に作り変えることもできます。

4D Compilerはカスタムアイコンを持ったスタンドアロンアプリケーション（.EXEファイル）を作成することができます。

この節のコマンドは、コンパイラの使用に関連があります。これらのコマンドは、データベース中のデータタイプを定義します。IDLEコマンドは、コンパイルされたデータベースで特別に使用されるコマンドです。

C_BLOB
C_BOOLEAN
C_DATE
C_GRAPH
C_INTEGER
C_LONGINT
C_PICTURE
C_POINTER
C_REAL
C_STRING
C_TEXT
C_TIME
IDLE

IDLEコマンド以外のこれらのコマンドは、変数を宣言し、それらを指定したデータタイプとしてキャストします。変数を宣言することによって、変数のデータタイプに関連する曖昧さが解決されます。変数がこれらのコマンドのいずれかで宣言されていない場合には、コンパイラが変数のデータタイプを判断しようとします。フォームで使用される変数のデータタイプは、多くの場合、コンパイラで判断するのは困難です。このため、開発者がこれらのコマンドによってフォームで使用される変数を宣言することが特に重要です。

注：時間を節約するために、コンパイラウインドウにあるオプションを使用し、変数定義メソッドの生成や更新を行うことができます。このオプションは、データベース内で使用されているすべての変数を調査してタイプ指定を行う変数定義メソッドを自動作成します。

コンパイルするコードの記述に関する一般規則

■ バージョン1の4th Dimensionで使用されていたような変数の間接参照はできません。パーセント記号（%）を使って間接的に変数を参照する文字型間接参照も、中カッコ（{...}）を用いる数値型間接参照も実行することはできません。中カッコ（{...}）は定義された配列の要素を指定する場合にのみ使用されます。しかし、引数による間接参照は使用できます。

■ あらゆる変数や配列のデータタイプは、変更することはできません。

■ 1次元配列を2次元配列に、また2次元配列を1次元配列に変更することはできません。

■ 文字(列)変数の長さや、文字列配列の要素の長さは変えられません。

■ コンパイラにより変数のタイプ定義は行われますが、フォーム上の変数のように、データタイプが明確でない場合は、コンパイラ命令を使用して変数のデータタイプを指定するべきです。

■ 変数を明示的にタイプ定義するもう1つの理由は、コードの最適化です。このことは、特にカウンタとして使用される変数に対して当てはまります。最大限の能力を得るために、可能な限り倍長整数型の変数を使用してください。

■ 変数をクリアする（ヌルに初期化する）には、変数の名前を用いて**CLEAR VARIABLE**コマンドを使用します。**CLEAR VARIABLE**コマンド内で変数の名前を表わす文字列を使用してはいけません。

■ **Undefined**関数は、常にFalseを返します。変数は常に定義されています。

■ 通常、倍長整数変数および整数変数に対する数値演算は、デフォルトの数値タイプ（実数）に対する演算よりもはるかに高速に実行されます。

例題

(1)以下は、コンパイラ用の基本的な変数宣言の例です。

```
C_BLOB (vxMyBlob) `プロセス変数vxMyBlobはBLOBタイプの変数として宣言される
C_BOOLEAN (<>On Windows) `インタープロセス変数<>OnWindowsはBoolean
                                                タイプの変数として宣言される
C_DATE ($vdCurDate) `ローカル変数$vdCurDateは日付タイプの変数として
                                                宣言される
C_GRAPH (vg1 ; vg2 ; vg3) `3つのプロセス変数vg1、vg2、vg3はグラフタイプの
                                                変数として宣言される
```

(2)以下の例では、プロジェクトメソッド「OneMethodAmongOthers」は3つの引数を宣言します。

```
` 「OneMethodAmongOthers」プロジェクトメソッド
` OneMethodAmongOthers (実数 ; 整数 { ; 倍長整数})
` OneMethodAmongOthers (合計 ; 割合 { ; 比率})
C_REAL ($1) ` 実数タイプの第1引数
C_INTEGER ($2) ` 整数タイプの第2引数
C_LONGINT ($3) ` 倍長整数タイプの第3引数
```

(3)以下の例では、プロジェクトメソッド「Capitalize」は文字列引数を受け付け、文字列の結果を返します。

```
` 「Capitalize」プロジェクトメソッド
` Capitalize (文字列) → 文字列
` Capitalize (元文字列) → 大文字に変換された文字列
C_STRING (255 ; $0 ; $1)
$0:=Uppercase (Substring ($1 ; 1 ; 1))+Lowercase (Substring ($1 ; 2))
```

(4)以下の例では、プロジェクトメソッド「SEND PACKETS」はテキスト引数の変数番号が後ろにある時間引数を受け付けます。

```
` 「SEND PACKETS」プロジェクトメソッド
` SEND PACKETS (時間 ; テキスト { ; テキスト2... ; テキストN})
` SEND PACKETS (ドキュメント参照番号 ; データ { ; データ2... ; データN})
C_TIME ($1)
C_TEXT (${2})
C_LONGINT ($vlPacket)
For ($vlPacket ; 2 ; Count parameters)
    SEND PACKET ($1 ; ${$vlPacket})
End for
```

(5)以下の例では、プロジェクトメソッド「COMPILER_Param_Predeclare28」はコンパイラ用の別のプロジェクトメソッドのシンタックスを事前に宣言します。

```
` 「COMPILER_Param_Predeclare28」プロジェクトメソッド
C_REAL (OneMethodAmongOthers ; $1) `OneMethodAmongOthers(実数;整数
                                                {;倍長整数})
C_INTEGER (OneMethodAmongOthers ; $2) ` ...
C_LONGINT (OneMethodAmongOthers ; $3) ` ...
C_STRING (Capitalize ; 255 ; $0 ; $1) ` Capitalize (文字列) → 文字列
C_TIME (SEND PACKETS ; $1) `SEND PACKETS (時間;テキスト{;テキスト2 ;
                                                テキストN})
C_TEXT (SEND PACKETS ; ${2}) ` ...
```

C_BLOB

C_BLOB ({メソッド ;}変数 {; 変数２ ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_BLOB コマンドは、指定されたそれぞれの変数をBLOBタイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_BLOB (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_BLOB (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_BOOLEAN

C_BOOLEAN ({メソッド ;}変数 {; 変数 2 ; …; 変数 N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_BOOLEAN コマンドは、指定されたそれぞれの変数をブール変数としてキャストします。

コマンドの第 1 の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第 2 の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2 等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2 番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_BOOLEAN (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_BOOLEAN (${5})」宣言は、4D とコンパイラに対して、5 番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters 関数を参照してください。

参照

前節での例を参照してください。

C_DATE

C_DATE ({メソッド ;}変数 {; 変数２ ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_DATEコマンドは、指定されたそれぞれの変数を日付変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_DATE (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_DATE (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_GRAPH

C_GRAPH ({メソッド ;}変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_GRAPHコマンドは、指定されたそれぞれの変数をグラフ変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_GRAPH (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_GRAPH (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_INTEGER

C_INTEGER ({メソッド ;}変数 {; 変数２ ; …; 変数 N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

注：このコマンドは、以前のデータベースとの互換性を保つために 4th Dimension に残されています。実際には、4D とコンパイラは整数を倍調整数へと内部的にタイプ変換します。

例えば

```
C_INTEGER($MyVar)
$TheType:=Type($MyVar) `$TheType = 9 (Is Longint)
```

説明

C_INTEGER コマンドは、指定されたそれぞれの変数を整数タイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_INTEGER (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_INTEGER (${5})」宣言は、4D とコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters 関数を参照してください。

参照

前節での例を参照してください。

C_LONGINT

C_LONGINT ({メソッド ;}変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_LONGINTコマンドは、指定されたそれぞれの変数を倍長整数タイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_LONGINT (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_LONGINT (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_PICTURE

C_PICTURE ({メソッド ;}変数 {; 変数２ ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_PICTURE コマンドは、指定されたそれぞれの変数をピクチャタイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_PICTURE (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_PICTURE (${5})」宣言は、4D とコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters 関数を参照してください。

参照

前節での例を参照してください。

C_POINTER

C_POINTER ({メソッド ;}変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_POINTERコマンドは、指定されたそれぞれの変数をポインタタイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_POINTER (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_POINTER (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_REAL

C_REAL ({メソッド ;}変数 {; 変数２ ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_REALコマンドは、指定されたそれぞれの変数を実数タイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_REAL (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_REAL (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_STRING

C_STRING ({メソッド ;}サイズ; 変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
サイズ	数値	→	文字列の長さ
変数	変数または${...}	→	定義する変数の名前

説明

C_STRING コマンドは、指定されたそれぞれの変数を文字列変数としてキャストします。

引数＜サイズ＞は、定義した変数に納められる文字（列）の最大の長さを指定します。文字は最大255バイトまでです。処理速度が問題となる場合、テキスト変数ではなくできるだけ文字（列）変数を使用してください。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_STRING (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_STRING (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_TEXT

C_TEXT ({メソッド ;}変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_TEXTコマンドは、指定されたそれぞれの変数をテキストタイプの変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_TEXT (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_TEXT (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

C_TIME

C_TIME ({メソッド ;}変数 {; 変数2 ; …; 変数N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

説明

C_TIMEコマンドは、指定されたそれぞれの変数を時間変数としてキャストします。

コマンドの第1の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第2の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2等）またはその両方をコンパイラ用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_TIME (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_TIME (${5})」宣言は、4Dとコンパイラに対して、5番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。詳細については、Count parameters関数を参照してください。

参照

前節での例を参照してください。

IDLE

IDLE

説明

IDLEコマンドは、コンパイラと一緒に使用する目的だけに作成されたコマンドです。このコマンドは、コンパイルしたデータベースにおいてのみ使用され、制御を4th Dimensionエンジンに戻す命令が、そのデータベース中のメソッドにない場合に使用します。例えば、ループ内に全くコマンドを含まない**For**ループを持ったメソッドを実行している場合に、**ON EVENT CALL**コマンドで割り込みメソッドをインストールしていてもループを抜け出すことはできません。また、他のアプリケーションに切り替えることができません。このような場合、**IDLE**コマンドを挿入して、4th Dimensionによりイベントがトラップされるようにします。割り込みを起こしたくない場合は、**IDLE**コマンドを削除します。

例題

以下の例は、**IDLE**コマンドを呼び出さないかぎり、コンパイルしたデータベースでループから抜け出すことができません。

```
` Do Something プロジェクトメソッド
ON EVENT CALL ("EVENT METHOD")
<>vbWeStop:=False
MESSAGE ("実行中..."+Char (13)+"どれかキーを押すと実行を停止します。")
Repeat
    IDLE ` 4D コマンドを含まない処理を実行する
Until (<>vbWeStop)
ON EVENT CALL ("")

` EVENT METHOD プロジェクトメソッド
If (Undefined (KeyCode))
    KeyCode:=0
End if
If (KeyCode#0)
    CONFIRM ("処理を停止しても良いですか？")
    If (OK=1)
        <>vbWeStop:=True
    End if
End if
```

参照

なし

データベースメソッド

「On Startup」データベースメソッド

「On Startup」データベースメソッドは、データベースを開くと呼び出されます。

この状態は、以下のような4D環境で発生します。

■ 4th Dimension

■ 4D Client（クライアント側で、接続が4D Serverにより受け付けられた後）

■ 4D Runtime

■ 4D Engineを組み込んだ4Dアプリケーション

注：「On Startup」データベースメソッドは、4D Serverによって起動されることはありません。

「On Startup」データベースメソッドは、4Dによって自動的に起動されます。プロジェクトメソッドとは異なり、このデータベースメソッドをユーザが呼び出すことはできません。「On Startup」データベースメソッド内から作業を呼び出して実行するには、プロジェクトメソッドの場合と同様に、サブルーチンを使用します。

「On Startup」データベースメソッドは、以下のような処理に最適です。

■ 作業セッション全体で使用するインタープロセス変数を初期化する。

■ データベースを開いた時にプロセスを自動的に開始する。

■ 以前の作業セッション中にこの目的で保存された初期設定やシステム定義をロードする。

■ 条件が一致しない（システムリソースがない等の）場合には、明示的に**QUIT 4D**コマンドを呼び出して、データベースを開かないようにする。

■ データベースを開く度に自動的に実行したい他の動作を実行する。

4th Dimension バージョン 3 との互換性

データベースメソッドは、バージョン6で導入された新しいメソッドです。バージョン3の4th Dimensionでは、データベースを開いた時に4Dが自動的に実行するメソッドは「Startup」プロシージャだけでした。バージョン5から変換されたデータベースを使用し、新しい「On Startup」データベースメソッドの機能を利用したい場合には、データベースの「環境設定」ダイアログボックスにある「V3.x.xのStartupプロシージャ方式を使う」チェックボックスが選択されていないことを確認してください。このプロパティは「Startup」メソッドと「On Startup」データベースメソッドの切り替えにだけ影響を与えます。このプロパティを非選択にせずに、例えば「On Exit」データベースメソッドを追加すると、この後者のデータベースメソッドが4Dによって起動されます。

「On Exit」データベースメソッド

「On Exit」データベースメソッドは、データベースを終了すると呼び出されます。

この状態は、以下のような4D環境で発生します。

■ 4th Dimension

■ 4D Client（クライアント側で、接続が4D Serverにより受け付けられた後）

■ 4D Runtime

■ 4D Engineを組み込んだ4Dアプリケーション

注：「On Exit」データベースメソッドは、4D Serverによっては起動されません。

「On Exit」データベースメソッドは、4Dによって自動的に起動されます。プロジェクトメソッドとは異なり、このデータベースメソッドをユーザが呼び出すことはできません。「On Exit」データベースメソッド内から作業を呼び出して実行するには、プロジェクトメソッドの場合と同様に、サブルーチンを使用します。

データベースは、以下のうちいずれかの状態になると終了します。

■ ユーザが「ユーザ」モードまたは「デザイン」モードの「ファイル」メニューから「終了」メニューを選択した場合

■ **QUIT 4D**コマンドへの呼び出しが実行された場合

■ 4Dのプラグインソフトから**QUIT 4D**エントリポイントへの呼び出しが実行された場合

データベースの終了がどのような方法で実行されたかに関わらず、4Dは以下のような処理を実行します。

■「On Exit」データベースメソッドがない場合には、4Dは実行プロセスそれぞれを区別せずに1つずつアボートします。ユーザがデータ入力を実行している場合には、レコードはキャンセルされ、保存されません。

■「On Exit」データベースメソッドがある場合には、4Dは新しく作成されたローカルプロセスの中でこのメソッドの実行を開始します。したがって、このデータベースメソッドを使用し、インタープロセス間通信を通じて、（データ入力を）終了、または処理の実行を中止しなければならないことを、他のプロセスに通知することができます。4Dは、いずれ終了するということに注意してください。「On Exit」データベースメソッドは、必要なクリーンアップや終了の処理を実行することができますが、中断処理を拒否することができず、ある時点で終了することになります。

「On Exit」データベースメソッドは、以下のような処理を実行するには最適です。

■ データベースを開いた時に自動的に開始されたプロセスを停止する。

■「On Startup」データベースメソッドの以下のセッションの開始時に再使用される初期設定やシステム定義を（ローカルに、ディスク上に）保存する。

■ データベースを終了する度に実行したい処理が他にあれば、それを実行する。

注：「On Exit」データベース・メソッドがローカル／クライアントプロセスであるのを忘てはいけません。それはデータ・ファイルにアクセスすることができません。

このように、「On Exit」データベースメソッドが検索またはソートを実行すると、終了しようとしている4D Clientは「フリーズ」して、実際には終了しません。

クライアントがアプリケーションを終了するとき、ユーザがデータにアクセスする必要があるならば、データ・ファイルにアクセスすることができる「On Exit」データベースメソッド内から、新しいグローバルなプロセスを作成します。

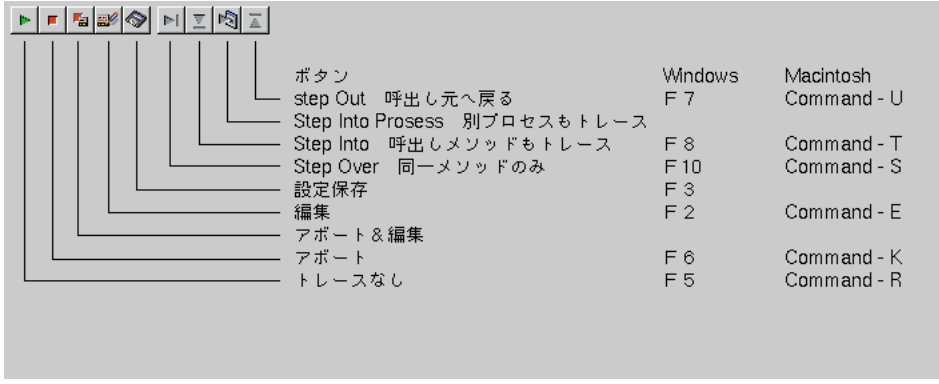
すると、新しいプロセスが「On Exit」データベースメソッド実行の終わりの前に、正常終了します（例によって、インタープロセス変数を使うことによって）。

デバッガのショートカット

この節では、「デバッグ」ウインドウで提供されているすべてのショートカットをリストしています。

「実行コントロール」ツールバーでのボタンショートカット

次の図では、「デバッグ」ウインドウの左上隅にある8個のボタンのショートカットを示しています。



「トレースなし」　F5 or command+r

「アボート」　F6 or command+k

「アボート＆編集」

「編集」　F2 or command+e

「設定保存」　F3

「Step Over（同一メソッドのみ）」　F10 or command+s

「Step Into（呼び出しメソッドもトレース）」　F8 or command+t

「Step Into Prosess（別プロセスもトレース）」

「Step Out（呼び出し元へ戻る）」　F7 or command+u

■ shift 押しながら F5、および shift を押しながらトレースなしボタンをクリック、実行を再開始します。さらに、現在のプロセスの次の **TRACE** コマンド呼び出しをすべて使用不可能にします。

「デフォルト表現式／値」エリアでのキーボードショートカット

■「デフォルト表現式／値」エリアで「マウスの右ボタンをクリック（Windows）」するか、または「control+ クリック（Macintosh）」を実行すると、「デフォルト表現式／値」エリアの「スピード」メニューがプルダウンされます。

■「デフォルト表現式／値」エリア内の項目をダブルクリックすると、その項目が「カスタム表現式／値」エリアにコピーされます。

「メソッド連鎖」エリアでのキーボードショートカット

■「メソッド連鎖」エリアでメソッドの名前をダブルクリックすると、メソッドが、「ソースコード」エリアの呼び出しチェーンにある呼び出しに対応する行に表示されます。

「カスタム表現式／値」エリアでのキーボードショートカット

■「カスタム表現式／値」エリアで「マウスの右ボタンをクリック（Windows）」するか、または「control+クリック（Macintosh）」を実行すると、「カスタム表現式／値」エリアの「スピード」メニューがプルダウンされます。

■「カスタム表現式／値」エリア内でダブルクリックすると、新しい表現式が作成されます。

「ソースコード」エリアでのキーボードショートカット

■左マージンをクリックすると、ブレークポイントが設定される（永続的ブレークポイントの場合）、またはブレークポイントが削除されます。

■「Alt+Shift+クリック（Windows）」または「option+shift+クリック（Macintosh）」により、一時的ブレークポイントが設定されます。

■「Alt+クリック（Windows）」または「option+クリック（Macintosh）」により、新しいブレークポイントや既存のブレークポイントの「ブレークポイントプロパティ」ウインドウが表示されます。

■単純にドラッグ&ドロップを行なうことによって、選択されたテキストをクリックし、それをカスタム表現式&ドロップします。

■Ctrlキー（Windows）、またはcommandキー（Macintosh）を押しながら、選択されたテキストをクリックすることによって。

■Ctrlキー（Windows）、またはcommandキー（Macintosh）を押しながらDのキーのコンビネーションを使うことによって。

全エリア共通のキーボードショートカット

■「Ctrl+*（Windows）」または「Command+*（MacOS）」により、「デフォルト表現式／値」エリアが強制的に更新されます。

■どのペインでも項目が選択されていない場合にenterキーを押すと、1行ずつ進みます。

■項目の値が選択されている場合には、矢印キーを使用してリスト内を移動します。

■項目が編集されている場合には、矢印キーを使用してカーソルを移動します。「Ctrl+a／x／c／v（Windows）」、または「command+a／x／c／v（Macintosh）」を「編集」メニューの「すべてを選択／切り取り／コピー／貼り付け」メニューへのショートカットとして使用します。

入力制御コマンド

CANCEL

CANCEL

説明

CANCELコマンドは、フォームやオブジェクトのメソッド（またはサブルーチン）で以下の目的のために使用されます。

■ **ADD RECORD**、**MODIFY RECORD**、**ADD SUBRECORD**、**MODIFY SUBRECORD**を使って開始されたデータ入力による新規または修正レコードやサブレコードをキャンセルするため

■ **DIALOG**コマンドで表示されたフォームをキャンセルするため

■ **DISPLAY SELECTION**や**MODIFY SELECTION**コマンドを使って、レコードセレクションを表示しているフォームを終了するため

■ **Print form**関数を使用して実行しようとするフォーム印刷をキャンセルするため（後述）。

データ入力中に、**CANCEL**コマンドは、ユーザがキャンセルキーコンビネーション（Windowsでは「Ctrl+.（ピリオド）」キー、Macintoshでは「command+.（ピリオド）」キー）を押した場合と同じ動作を実行します。

一般に**CANCEL**コマンドは、メニューコマンドの結果として実行されます。また、「動作なし」属性ボタンのオブジェクトメソッド内でもよく使用されます。

また、**Open window**関数におけるオプションの「クローズボックス」メソッド内でもよく使用されます。あるウインドウにコントロールメニューボックス（Macintosh版では、クローズボックス）がある場合、**ACCEPT**や**CANCEL**コマンドはコントロールメニューボックスがクリックされたり、または「閉じる」メニューが選択された際に実行されるメソッドの中で呼び出されます。

CANCELコマンドは、待ち行列を作成することができません。あるイベントに対してメソッド内で2つの**CANCEL**コマンドを続けて実行しても、1つの**CANCEL**コマンド実行をした場合と同じ効果しか得られません。

最後に、**Print form**関数を使用する際に、このコマンドを「On Printing Detail」フォームイベントで使用することができます。この状況において、**CANCEL**コマンドは実行直前の印刷を一時中断し、次のページに印刷を再開します。この仕組みを利用して、スペースが不足している場合やページブレークが必要な場合のフォーム印刷を管理することができます。

注：この操作は、PAGE BREAK(*) コマンドとは異なります。PAGE BREAK(*) コマンドは、印刷待ちであるすべてのフォームをキャンセルします。

例題

SET PRINT MARKERコマンドの例題を参照してください。

参照

ACCEPT、PAGE BREAK、Print form

システム変数とセット

CANCELコマンドを実行すると（フォームまたは印刷がキャンセルされ)、システム変数OKには0が代入されます。

データ読み込みとデータ書き出しコマンド

IMPORT DATA

IMPORT DATA (ファイル名 {; プロジェクト {; *}})

引数	タイプ		説明
ファイル名	文字列	→	読み込みファイルへのパス
プロジェクト	BLOB	→	読み込みプロジェクトの内容
		←	読み込みプロジェクトの新しい内容 (*引数が渡された場合)
*	*	→	データ読み込みダイアログボックスを表示し、プロジェクトを更新します。

説明

このコマンドは、データをファイル名ファイルから読み込めるようにします。4Dは以下のフォーマットのデータを読み込むことができます：テキスト、固定長のテキスト、XML、SYLK、DIF、DBF（dBase）および4th Dimension。

空白をファイル名に渡すと、**IMPORT DATA**は標準ファイルセーブダイアログボックスを表示して、ユーザが読み込むファイルの名前、タイプおよび位置を定義することができるようにします。ダイアログボックスが受け入れられると、Documentシステム変数にファイルパスがセットされます。ユーザがキャンセルをクリックすると、コマンドの実行は停止されて、システム変数OKは0になります。

■ オプション引数プロジェクトを省略した場合、データ読み込みダイアログボックスが表示され、インポートパラメータを定義するかまたは既存の定義ファイルからインポートプロジェクトを読み込むことができます。

注：インポートプロジェクトには、読み込むテーブルやフィールド、区切り符号（デリミタ）のようなインポートに関するすべてのパラメータが含まれています。これらのパラメータはデータ読み込みダイアログボックス内で定義します。プロジェクトはディスクにセーブされ、読み込んで使用する事ができるようになります。

■ 有効なインポートプロジェクトを持つBLOBをプロジェクト引数に渡した場合、ユーザの操作無しに直接データ読み込みが実行されます。プロジェクトはデータ読み込みダイアログボックス内で既に前もって定義し保存しておかなければいけません。これを実行するには、2つの方法があります。

■ プロジェクトをディスクに保存後、**DOCUMENT TO BLOB**コマンドを使用してフィールドまたはBLOB変数にセットします。

■ 空白のプロジェクト引数およびオプション引数 * を指定した**IMPORT DATA**コマンドを実行し、プロジェクト引数のBLOBに保存します（下記参照）。この方法は、ディスク上からプロジェクトを読み込む必要はなく、データと共にプロジェクトを保存することができます。

オプションの引数 * が指定されていれば、プロジェクト内に定義されたパラメータと共にデータ読み込みダイアログボックスを表示します。これは、パラメータの1つまたはそれ以上を変更できる可能性を持ちながら、前もって定義されたプロジェクトを使用できるようにするものです。さらに、データ読み込みダイアログボックスを閉じた後に、プロジェクト引数は、「新しい」プロジェクトのパラメータを持つことができ、新しいプロジェクトをBLOBフィールドやディスク上等に保存することができます。

データ読み込みが正常に終了すると、システム変数OKは1になります。

参照

EXPORT DATA

システム変数とセット

標準の保存ファイルのダイアログボックスかインポートのダイアログボックスのキャンセルボタンをクリックすると、システム変数OKには0がセットされます。読み込み処理が正常に終了すると1がセットされます。

EXPORT DATA

EXPORT DATA (ファイル名 {; プロジェクト {; *}})

引数	タイプ		説明
ファイル名	文字列	→	書き出しファイルへのパス
プロジェクト	BLOB	→	書き出しプロジェクトの内容
		←	書き出しプロジェクトの新しい内容 (*引数が渡された場合)
*	*	→	データ書き出しダイアログボックスを表示し、プロジェクトを更新します。

説明

このコマンドは、データをファイル名ファイルから書き出せるようにします。4Dは以下のフォーマットのデータを書き出すことができます：テキスト、固定長のテキスト、XML、SYLK、DIF、DBF（dBase）および4th Dimension。

空白をファイル名に渡すと、**EXPORT DATA**は標準ファイルセーブダイアログボックスを表示して、ユーザが書き出すファイルの名前、タイプおよび位置を定義することができるようにします。ダイアログボックスが受け入れられると、Documentシステム変数にファイルパスがセットされます。ユーザがキャンセルをクリックすると、コマンドの実行は停止されて、システム変数OKは0になります。

■ オプション引数プロジェクトを省略した場合、データ書き出しダイアログボックスが表示され、エクスポートパラメータを定義するかまたは既存の定義ファイルからエクスポートプロジェクトをロードことができます。

注：エクスポートパラメータには、書き出すテーブルやフィールド、区切り符号（デリミタ）のようなエクスポートに関するすべてのパラメータが含まれています。これらのパラメータはデータ書き出しダイアログボックス内で定義します。プロジェクトはディスクに保存され、読み込みして使用する事ができるようになります。

■ 有効なエクスポートプロジェクトを持つBLOBをプロジェクト引数に渡した場合、ユーザの操作無しに直接データ書き出しが実行されます。プロジェクトはデータ書き出しダイアログボックス内で既に前もって定義し保存しておかなければいけません。これを実行するには、2つの方法があります。

■ プロジェクトをディスクに保存後、**DOCUMENT TO BLOB**コマンドを使用してフィールドまたはBLOB変数にセットします。

■ 空白のプロジェクト引数およびオプション引数＊を指定した**EXPORT DATA**コマンドを実行し、プロジェクト引数のBLOBに保存します（下記参照）。この方法は、ディスク上からプロジェクトを読み込む必要はなく、データと共にプロジェクトを保存することができます。

オプションの引数＊が指定されていれば、プロジェクト内に定義されたパラメータと共にデータ書き出しダイアログボックスを表示します。これは、パラメータの1つまたはそれ以上を変更できる可能性を持ちながら、前もって定義されたプロジェクトを使用できるようにするものです。さらに、データ書き出しダイアログボックスを閉じた後に、プロジェクト引数は、「新しい」プロジェクトのパラメータを持つことができ、新しいプロジェクトをBLOBフィールドやディスク上等に保存することができます。

データ書き出しが正常に終了すると、システム変数OKは1になります。

参照

IMPORT DATA

システム変数とセット

標準のオープンファイルのダイアログボックスかエクスポートのダイアログボックスのキャンセルボタンをクリックすると、システム変数OKには0がセットされます。書き出し処理が正常に終了すると1がセットされます。

割り込みコマンド

FILTER EVENT

FILTER EVENT

引数	タイプ	説明

このコマンドには、引数はありません。

説明

FILTER EVENTコマンドは、**ON EVENT CALL**コマンドでインストールされたイベント管理用プロジェクトメソッドから呼び出します。

イベント管理メソッドでこのコマンドを呼び出すと、カレントイベントが4Dに渡されなくなります。

このコマンドを使用すると、イベントキューからカレントイベント（クリック、キー入力）を取り除くことができます。したがって、4Dはイベント管理用プロジェクトメソッド内で発生したイベントに対してそれ以上の処理は行いません。

警告：FILTER EVENTコマンドを呼び出すだけのイベント管理メソッドを作成しないようにしてください。すべてのイベントが4Dから無視されるためです。FILTER EVENTコマンドだけのイベント管理メソッドがある場合には、「Ctrl+Shift+Alt+Backspace」（Macintosh版では、「command+option+shift+control+delete」）キーを押します。これにより、ON EVENT CALLプロセスがイベントをまったく受け取らない通常のプロセスに切り替えます。

特別なケース：DISPLAY SELECTIONや MODIFY SELECTIONコマンドを用いてフォームを表示している場合に、FILTER EVENTコマンドを標準の出力フォームメソッド内で使用することもできます。この特別なケースでは、FILTER EVENTコマンドを使用してレコード上でのダブルクリックをフィルタリングすることができます（また、この方法でページモードでのレコードオープン以外の動作を実行します）。これを行うには、出力フォームメソッドに次の行を追加します。

```
If(Form event=On Double Clicked)
    FILTER EVENT
    ... `ダブルクリックの処理
End if
```

例題

前述の**ON EVENT CALL**コマンドの例を参照してください。

参照

ON EVENT CALL

メニュー

MENU BAR

MENU BAR (メニューバー番号 {; プロセス} {; *})

引数	タイプ		説明
メニューバー番号	数値	→	メニューバーの番号または名前
プロセス	数値	→	プロセス参照番号
*		←	メニューバーの状態を保存

説明

MENU BARコマンドは、カレントプロセスに対してのみ、カレントメニューバーを＜メニューバー番号＞で指定したメニューバーに変更します。引数＜メニューバー番号＞には、新しいメニューバーの番号または名前を渡します。

注：メニューバー名には、31桁までのユニークな名前を指定することができます。

オプション引数＜プロセス＞は、指定したプロセスのメニューバーを＜メニューバー＞に変更します。オプション引数＜*＞により、メニューバーの現在の状態を保存することができます。この引数が省略された場合、このコマンドが実行されると、**MENU BAR**コマンドはメニューバーを元の状態に戻します。

例えば、**MENU BAR** (1)を実行したとします。次に、**DISABLE MENU ITEM**コマンドを使い、複数のメニューを使用不可にします。

MENU BAR (1)を2度目に実行すると、その実行が同じプロセスからでも別のプロセスからでも、メニューはすべて、元の使用可の状態に戻ります。

MENU BAR (1 ; *)を実行すると、メニューバーは前と同じ状態を保っており、使用不可にしたメニューは使用不可のままです。

注：オプション引数＜プロセス＞を指定しない場合、＜*＞は2番目の引数になります。つまりMENU BAR (1 ; 2 ; *)とMENU BAR (1 ; *)はともに有効な命令文です。

ユーザが「カスタム」モードに移ると、最初の間メニューバー（メニューバー#1）が表示されます。データベースを開く際に「On Startup」データベースメソッド、またはStartupメソッドで目的のメニューバーを指定して、個々のユーザ用にメニューバーを変更することができます。

例題

1. 以下の例は、カレントメニューバーをメニューバー#3に変更し、メニューの状態を元に戻します。

```
MENU BAR (3)
```

2. 以下の例題は、カレントメニューバーを“FormMenuBar1”という名前のメニューバーに変更し、メニューコマンドの状態を保存します。

```
MENU BAR("FormMenuBar1";*)
```

3. 以下の例は、レコードの変更中にフォームのメニューバーをメニューバー#3に変更します。レコードの変更が済むと、メニューの状態を保存してメニューバーをメニューバー#2に戻します。

```
MENU BAR (3)        ` 以下のフォームにメニューバー#3を設定する
ALL RECORDS (顧客)
MODIFY SELECTION (顧客)        ` フォームを開く
MENU BAR (2 *)      ` 変更後メニューバーを戻す
```

参照

メニューの管理

SET ABOUT

SET ABOUT (アイテム ; メソッド)

引数	タイプ		説明
アイテム	文字列	→	アバウトメニュー項目の新しいテキスト
メソッド	文字列	→	メニュー項目が選択されたときに呼び出すメソッド

説明

SET ABOUTコマンドは、「ヘルプ」（Macintosh版では、「アップル」）メニューの「4th Dimension (R) について...」を＜アイテム＞に変更します。

SET ABOUTコマンドを実行した後、ユーザがこのメニューを選択すると、＜メソッド＞が実行されます。一般的に、このメソッドはデータベースに関するバージョン情報を示すダイアログボックスを表示します。

4th Dimensionアイコン、バージョン番号、ならびに著作権に関する注意がダイアログの上部に追加されます。

例題

1. 以下の例は、「4th Dimension (R) について」メニューを「スケジュールについて...」に置き換えます。メソッド“ABOUT”は、カスタムのアバウトボックスを表示します。

```
SET ABOUT ("スケジュールについて..." ; "ABOUT")
```

2. 以下の例は、「4th Dimension (R) について」メニューを元のアバウトボックスに戻します。

```
SET ABOUT ("4th Dimension (R) について..." ; "")
```

参照

なし

オブジェクトプロパティコマンド

MOVE OBJECT

MOVE OBJECT ({*;} オブジェクト ; 水平移動 ; 垂直移動 ;{; 水平リサイズ {; 垂直リサイズ}}{; *})

引数	タイプ		説明
*	*	→	指定された場合=オブジェクトは オブジェクトの名前（文字列） 省略された場合=オブジェクトは変数
オブジェクト	オブジェクト	→	オブジェクト名（*が指定されている場合）
			またはフィールドまたは変数 (*が省略された場合)
水平移動	倍長整数	←	オブジェクトの水平移動距離 (>0=右へ、<0=左へ)
垂直移動	倍長整数	←	オブジェクトの垂直移動距離 (>0=下へ、<0=上へ)
水平リサイズ	倍長整数	←	オブジェクトの水平方向のサイズ変更値
垂直リサイズ	倍長整数	←	オブジェクトの垂直方向のサイズ変更値
*	*		指定されている場合=絶対座標 省略された場合=相対座標

説明

このコマンドは、*とオブジェクトの引数で定義された、現在のフォーム内のオブジェクトを、水平方向にピクセル、垂直方向にピクセル移動させます。また、(オプションで)オブジェクトを水平方向に水平リサイズピクセル、垂直方向にピクセル、サイズの変更をすることもできます。

移動とサイズ変更の方向は、水平移動および垂直移動引数に渡された値に依ります。

- 値が正であれば、オブジェクトは右および下へそれぞれ移動され、サイズ変更されます。
- 値が負であれば、オブジェクトは左および上へそれぞれ移動され、サイズ変更されます。

最初のオプション引数*を指定すると、オブジェクト引数がオブジェクト名（文字列）であることを示し、最初のオプション引数 * を省略すると、オブジェクト引数がフィールドまたは変数であることを示します。この場合、文字列ではなくフィールドまたは変数の参照（オブジェクトタイプがフィールドまたは変数のみ）を指定します。

オブジェクトにオブジェクト名としてワイルドカード（@）を使用し、複数のオブジェクトを指定すると、関連する全オブジェクトが移動またはサイズが変更されます。

注：バージョン6.5からは、文字列に含まれるワイルドカード文字（@）の取り扱い方を設定することができます。このオプションは、「オブジェクトプロパティ」コマンドに影響を与えます。

デフォルトでは、水平移動、垂直移動、水平リサイズ、垂直リサイズの値は、オブジェクトの以前の位置からの相対的な値です。引数が絶対位置を表わすようにしたい場合は、最後のオプションの引数 * を渡します。

このコマンドは次のコンテクストで働きます。

■ **DIALOGUE**コマンドを使用して表示されたフォーム

■ インプットフォームのデータ入力

■ **MODIFY SELECTION**もしくは**DISPLAY SELECTION**コマンドで表示される出力フォームのヘッダーとフッター

■ フォーム出力イベント

例題

▼ 下記のコードは、"button_1"を右に10ピクセル、上に10ピクセル移動させ、幅を30ピクセル、高さを40ピクセルにサイズ変更します。

```
MOVE OBJECT (*;"button_1";10;-20;30;40)
```

▼ 下記のコード=は、"button_1"をを以下の座標に移動します：(10;20)(30;40)

```
MOVE OBJECT (*;"button_1";10;20;30;40;*)
```

参照

GET OBJECT RECT

クイックレポート

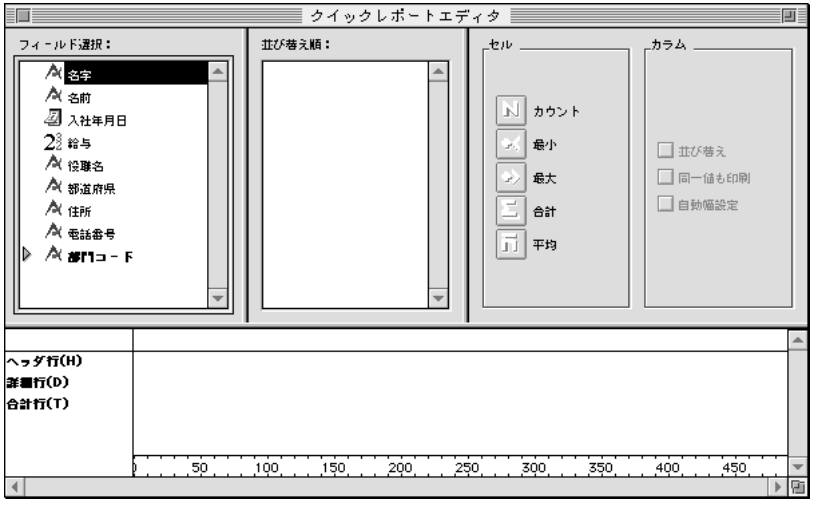
QR REPORT

QR REPORT ({テーブル ;} ドキュメント {; *})

引数	タイプ		説明
テーブル	テーブル	→	印刷するテーブル 省略した場合、デフォルトテーブル
ドキュメント	文字列	→	クィックレポートのドキュメント
*	*	→	プリンタダイアログボックスの表示取り消し

説明

QR REPORT コマンドは、下図に示した「クィックレポート」エディタで作成した＜テーブル＞に対するレポートを印刷します。



ユーザは、「クィックレポート」エディタでカスタムレポートを作成することができます。「クィックレポート」エディタを使用したレポートの作成に関する詳細は、『4th Dimension ユーザリファレンス』を参照してください。

＜ドキュメント＞は、「クィックレポート」エディタで作成され、ディスクに保存されたレポートドキュメントファイルです。「クィックレポート」エディタ内の「ファイル」メニューから「保存」または「新規保存」を選択してレポートドキュメントを保存します。これはレポートの形式を格納するだけで、印刷されるレコードを格納するわけではありません。

＜ドキュメント＞に対して空の文字列（""）を指定した場合に、**REPORT**コマンドは「ファイルを開く」ダイアログボックスを表示し、ユーザは、印刷するレポートを選択することができます。レポートが選択されると「用紙設定」ダイアログボックスが表示されます。オプション引数にアスタリスク（*）を指定すると、プリンタダイアログボックスは表示されません。レポートは、その後で印刷されます。

＜ドキュメント＞に存在しないドキュメント名を指定した場合は、「クィックレポート」エディタが表示されます。

この場合、任意の引数である＜ウィザード＞と＜クエリ＞を使用して、「クイックレポート」エディタウインドウ上に「ウィザードを開く」ボタンや「新規クエリ」ボタンを表示するかどうかを指定することができます。ボタンを表示するには“True”を渡し、ボタンを隠す場合には“False” を渡します。

デフォルトでは（これらの引数を省略した場合）、対応するボタンは表示されません。

「クイックレポート」エディタを使用しない場合、レポートが印刷されるとシステム変数OKに1がセットされます。印刷されない場合（ユーザが「用紙設定」ダイアログボックスでキャンセルをクリックした場合等）には0がセットされます。

▼ 例題

1. 以下の例は、ユーザが[従業員]テーブルを検索し、レポート“明細一覧”を自動的にプリントします。

```
QUERY ([従業員])
If (OK=1)
        REPORT ([従業員];"明細一覧";*)
End if
```

2. 以下の例は、ユーザは[従業員]テーブルを検索した後、プリントするレポートを選択します。

```
QUERY ([従業員])
If (OK=1)
        REPORT ([従業員];"")
End if
```

3. ユーザはウィザードを使用して（または使用せずに）任意のレポートの設計、保存、ロード、プリントを行えます。

```
QUERY ([従業員])
If (OK=1)
        REPORT([People];Char(1); True)
End if
```

参照

なし

印刷コマンド

PRINT FORM

PRINT FORM ({テーブル ;} フォーム)

引数	タイプ		説明
テーブル	テーブル	→	印刷するテーブル 省略した場合、デフォルトテーブル
フォーム	文字列	→	印刷するフォーム

説明

PRINT FORMコマンドは、フィールドや変数の現在の値を＜フォーム＞に印刷します。このコマンドは、フォームのディテイルエリア（ヘッダ行とディテイル行の間のエリア）だけを印刷します。通常は、印刷のプロセスをメソッドで完全に制御する必要のある非常に複雑なレポートを印刷するために使用します。**PRINT FORM**コマンドはレコード処理、ブレーク処理、ページブレーク（改ページ）処理、ヘッダ処理、フッタ処理を全く行いません。これらの処理は、すべてデザイナが行います。**PRINT FORM**コマンドは固定された大きさの枠のなかにフィールドや変数を印刷します。

PRINT FORMコマンドは、フォームの印刷の後にページブレーク（改ページ）を行わないため、同じページに異なるフォームを容易に配置することができます。したがって、**PRINT FORM**コマンドは、異なるテーブルや異なるフォームを含む複雑な印刷処理には最適です。ページブレーク（改ページ）を強制的に行うには**PAGE BREAK**コマンドを使用してください。印刷可能領域を超える高さのフォームの印刷を次のページに持ち越すには、**PAGE BREAK**コマンドを使用する前に**CANCEL**コマンドを呼び出してください。

PRINT FORMコマンドを使用する場合、プリンタダイアログボックスは表示されません。レポートでは「デザイン」モードでフォームに割り当てられた用紙設定が使用されません。**PRINT FORM**コマンドを実行する前に用紙設定を指定する方法は2通りあります。

■ **PRINT SETTINGS**コマンドを使用する。この場合、ユーザが設定を行う。

■ **PAGE SETUP**コマンドを使用する。この場合、用紙設定はプログラムで指定する。

PRINT FORMコマンドは、メモリ中にそれぞれ印刷するページを作成します。各ページはメモリのページがいっぱいになるか、**PAGE BREAK**コマンドを実行すると印刷されます。**PRINT FORM**コマンドの使用後の最後のページの印刷を確実に行うためには、**PAGE BREAK**コマンドで終了しなければなりません。そうでないと、最後のページはメモリ中に残り印刷されません。

警告：サブフォームや外部オブジェクトは、PRINT FORMコマンドでは印刷できません。このようなオブジェクトを持つフォームを1つだけ印刷するには、代りにPRINT RECORDコマンドを使用します。

PRINT FORMコマンドにより、フォームメソッドの**On Printing Detail**イベントが発生します。

▼以下の例は、単純な**PRINT SELECTION**コマンドをシミュレートします。ブレーク処理はありません。このレポートは、小切手レジスタに対するものです。

レコードが小切手用か預金用であるかによって2種類のフォーム内の1つを使用します。

```
QUERY ([レジスタ])          ` レコードの選択
If (OK=1)
    ORDER BY ([レジスタ])          ` レコードのソート
    If (OK=1)
        PRINT SETTINGS          ` 用紙の設定
        If (OK=1)
            For ($i ; 1 ; Records in selection ([レジスタ]))
                                        ` 選択した全レコードのループ
                If ([レジスタ]タイプ="小切手") ` “タイプ”フィールド
                                                が"小切手"の場合
                    PRINT FORM ([レジスタ] ; "小切手印刷")
                                                ` 小切手の印刷
                Else ` “タイプ”フィールドが"小切手"でない場合
                    PRINT FORM ([レジスタ] ; "預金印刷")
                                                ` 預金の印刷
                End if
                NEXT RECORD ([レジスタ]) ` 以下のレコードに移動
            End for
            PAGE BREAK ` 最後のページを印刷
        End if
    End if
End if
```

参照

なし

PAGE BREAK

PAGE BREAK {(* | >)}

引数	タイプ		説明
*¦>		→	*：で開始した印刷ジョブをキャンセル >：1つのプリントジョブを強制する

説明

PAGE BREAKコマンドは**Print form**関数とともに使用すると（「On Printing Detai」フォームイベント中に）、強制的にページブレークを行い、メモリ上に作成された最終ページを印刷します。**PAGE BREAK**コマンドは、**PRINT SELECTION**コマンドとともに使用しないでください。この代りに、**Subtotal**関数または**BREAK LEVEL**コマンドにオプション引数を使用してページブレークを行ってください。

＜*＞と＜>＞引数は両方とも省略できます。

＜*＞引数により、**PRINT FORM**コマンドによって開始したプリントジョブをキャンセルすることができます。このコマンドを実行すると、進行中のプリントジョブが直ちに中止されます。

＜>＞引数は、**PAGE BREAK**コマンドの振る舞いを変更します。この形式は2種類の効果を持ちます。

■ **PAGE BREAK**コマンドが引数なしで再度実行されるまで、プリントジョブの開始を止めます。

■ プリントジョブに優先権を与えます。プリントジョブが終了するまで、他のプリントは行われません。

2番目のオプションは、スプールされるプリントジョブとともに使用すると、特に有効です。＜>＞引数により、プリントジョブは1つのファイルにスプールされます。これは、プリント時間を減少させます。

例題

1. **PRINT FORM**コマンドの例を参照してください。

2. **SET PRINT MARKER**コマンドの例題を参照してください。

参照

CANSEL、PRINT FORM

PRINT RECORD

PRINT RECORD ({テーブル} {;*})

引数	タイプ		説明
テーブル	テーブル	→	カレントレコードを印刷するテーブル、または省略した場合デフォルトテーブル
*¦>	*¦>		*印刷ダイアログボックスを省略、または印刷設定の再初期化を行わない

説明

このコマンドは＜テーブル＞のカレントレコードを、カレントセレクションを変更せずに印刷します。カレント出力フォームが印刷に用いられます。＜テーブル＞にカレントレコードが存在しない場合、**PRINT RECORD**コマンドは何も行いません。

PRINT RECORDコマンドを使ってサブフォームや外部オブジェクトを印刷することができます。この操作は、**Print form**関数では実行できません。

注：レコードに対して行われた修正が保存されていない場合、ディスク上の修正前のフィールド値ではなく、修正後の値が印刷されます。

PRINT RECORDコマンドは印刷の前にはデフォルトで印刷ダイアログを表示します。

ダイアログをユーザーがキャンセルした場合は、**PRINT RECORD**コマンドの実行は

中止され、印刷は行われません。このダイアログの表示を省略するために、オプション引数「*」「>」を使います。

「*」引数を指定した時は、デフォルトの印刷設定、もしくはコマンド**PAGE SETUP**で定義した設定で印刷されます。

「>」引数を指定した時は、印刷設定の再初期化を行うことなく、カレント印刷設定のまま、印刷を行います。この引数の利用は、あらかじめカスタム化しておいた印刷設定を維持しながら繰り返し印刷する際に便利です。**PRINT RECORD**コマンドの使用例題を参照してください。

▼以下の例では、カレントレコードを印刷します。このコードは入力フォームのボタン内に記述されています。ユーザがそのボタンをクリックすると、レコードは指定した出力フォームで印刷されます。

```
OUTPUT FORM ([テーブル1] ; "レコード印刷")
PRINT RECORD ([テーブル1] ; *)
OUTPUT FORM ([テーブル1] ; "出力")
```

例題

```
PRINT SETTINGS  `印刷設定を定義
If (OK=1)
        OUTPUT FORM([従業員];"ディテイル")  `最初の印刷帳票を使う
⇒       PRINT RECORD([従業員];>)  `ユーザーが定義した印刷設定で印刷
        OUTPUT FORM([従業員];"Simple")  `二つ目の印刷帳票を使う
⇒       PRINT RECORD([従業員];>)  `ユーザーが定義した印刷設定で印刷
        OUTPUT FORM([従業員];"Output")  `デフォルト出力フォームへ戻す
End if
```

参照

PRINT FORM

SET PRINT MARKER

SET PRINT MARKER (マーカー番号; 位置{; * })

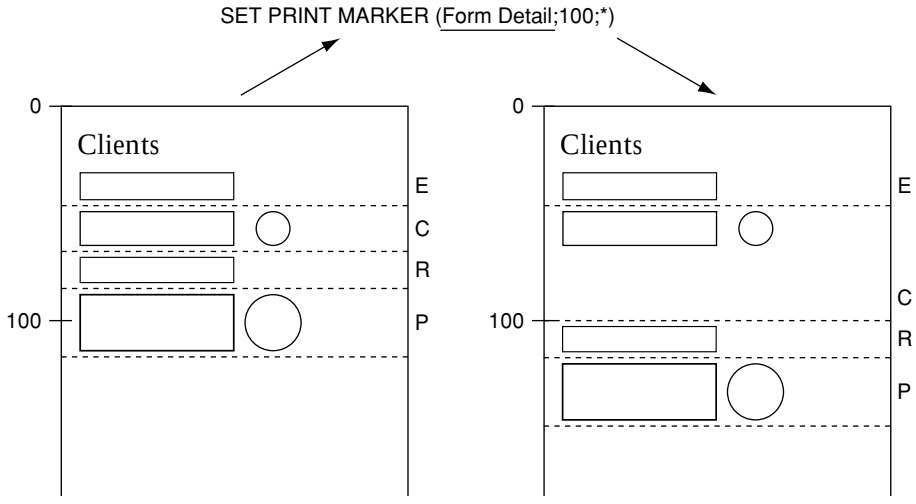
引数	タイプ		説明
マーカー番号	数値	→	マーカーの番号
位置	数値	→	マーカーの新しい位置
*	*	→	渡した場合、次のマーカーを移動する 省略した場合、次のマーカーを移動しない

■ **SET PRINT MARKER** コマンドは、「On Printing Detail」フォームイベント中に **Print form** 関数を使用した場合に呼び出せるようになりました。この操作により、カスタマイズしたレポートの印刷がスムーズに行えます（例題を参照）。

■ さらに、**SET PRINT MARKER** コマンドは、3番目の引数として * 記号を受け入れます。

この引数を渡すと、このコマンドの実行時に、＜マーカー番号＞で指定したマーカーより下側に位置するすべてのマーカーが、指定したマーカーと同じピクセル数だけ、同じ方向へ移動します。このマーカーより下側にあるエリア内のオブジェクトもすべて移動します。

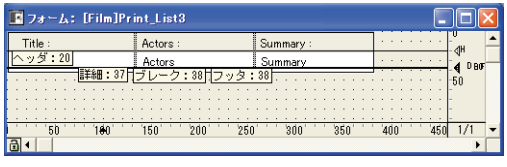
引数＜*＞を使用すると、後続の各マーカーの最初の位置より下側に＜マーカー番号＞で指定したマーカーを位置付けることができます。これら後続のマーカーも同時に移動します。



注：引数＜*＞を使用しない場合、移動するマーカを後続のマーカーよりも下側に位置付けることはできません。

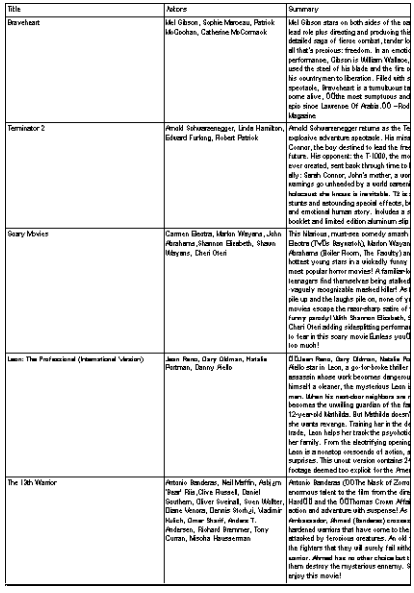
▼ この例題を使用して、3つのカラムがある印刷を実行することができます。各行の高さは、フィールド内容に応じて実行中に計算されます。

印刷に使用する出力フォームは次の通りです。



注：このフォームに対して、「On Printing Detail」フォームイベントが選択されています（印刷するエリアに関係なく、Print form関数はこのタイプのフォームイベントだけを生成する点に留意してください）。

レコードごとに、“Actors”または“Summary”カラムの内容に応じて行の高さを調整しなくてはなりません。目的とする結果を次に示します。



Title	Actors	Summary
Braveheart	Mel Gibson, Sophie Marceau, Patrick McGoohan, Catherine McCormack	Mel Gibson stars on both sides of the se lead role plus directing and producing this detailed saga of fierce combat, tender lo all that's precious: freedom. In an emoti performance, Gibson is William Wallace, used the steel of his blade and the fire o his countrymen to liberation. Filled with s spectacle, Braveheart is a tumultuous te some alive, 00the most sumptuous and epic since Lawrence Of Arabia.00 --Rol Magazine
Terminator 2	Arnold Schwarzenegger, Linda Hamilton, Edward Furlong, Robert Patrick	Arnold Schwarzenegger returns as the Te explosive adventure spectacle. His mine Connor, the boy destined to lead the free future. His opponent: the T-1000, the mo ever created, sent back through time to ally: Sarah Connor, John's mother, a wor warnings go unheeded by a world oblivio holocaust she knows is inevitable. T3 is stunts and astounding special effects, b and emotional human story. Includes a booklet and limited edition aluminum sli
Scary Movies	Carmen Electra, Marlon Wayans, John Abrahams, Shannon Elizabeth, Shawn Wayans, Cheri Oteri	The hilarious, must-see comedy smash Electra (TVÕs Baywatch), Marlon Wayan Abrahams (Boiler Room, The Faculty) ar hottest young stars in a violently funny most popular horror movies! A familiar-l teenagers find themselves being stalked -equally recognizable masked killer! As pile up and the laugh pile on, none of y movies escape the razor-sharp satire of funny parody! With Shannon Elizabeth, C Cheri Oteri adding side-splitting performa to fear in this scary movie.Eyeless yeaÕ too much!
Leon: The Professional (International Version)	Jean Reno, Gary Oldman, Natalie Portman, Danny Aiello	Ó Eileen Reno, Gary Oldman, Natalie Po Aiello star in Leon, a go-for-broke thriller assassin whose work becomes dangerou himself a cleaner, the mysterious Leon i man. When his next-door neighbors are becomes the unwilling guardian of the fa 12-year-old Mathilde. But Mathilde doesn she wants revenge. Training her in the de trade, Leon helps her track the psychoti her family. From the electrifying opening Leon in a nonstop crescendo of action, surprises. This uncut version contains 2 footage deemed too explicit for the Amer
The 13th Warrior	Antonio Banderas, Neil Marlin, Ashley 'Bear' Ris ,Clive Russel, Daniel Southern, Oliver Sveinall, Sven Walter, Diane Venora, Dennis Storhøi, Vladimir Kulich, Omer Sharif, Andrew T. Anderson, Richard Brammer, Tony Curran, Nikosha Haussman	Antonio Banderas (Ó0The Mask of Zorro Ó0Desperado) brings enormous talent to the film from the dire Hand Ó0 and the Ó0Thomas Crown Affai action and adventure with suspense! An Ambassador, Ahmad (Banderas) crosses hardened warriors that have come to the attacked by ferocious creatures. As old the fighters that they will surely fail with warrior. Ahmad has no other choice but t them destroy the mysterious enemy. S enjoy this movie!

印刷用のプロジェクトメソッドは次の通りです。

```
C_LONGINT(vLprint_height;$vLheight;vLprinted_height)
C_STRING(31;vSprint_area)
PAGE SETUP([Film];"Print_List3")
GET PRINTABLE AREA(vLprint_height)
vLprinted_height:=0
SELECT ALL([Film])

vSprint_area:="Header" `ヘッダエリアの印刷
$vLheight:=Print form([Film];"Print_List3";Form Header)
$vLheight:=21`Fixed height
vLprinted_height:=vLprinted_height+$vLheight

While(Not(End selection([Film])))
   vSprint_area:="Detail" `詳細エリアの印刷
   $vLheight:=Print form([Film];"Print_List3";Form Detail)
   `フォームメソッドにおいて詳細行の計算が実行される
   vLprinted_height:=vLprinted_height+$vLheight
   If(OK=0) `フォームメソッドでCANCELが実行された
      PAGE BREAK
      vLprinted_height:=0
      vSprint_area:="Header" `ヘッダエリアの再印刷
      $vLheight:=Print form([Film];"Print_List3";Form Header)
```

```
                $vLheight:=21
                vLprinted_height:=vLprinted_height+$vLheight
                vSprint_area:="Detail"
                $vLheight:=Print form([Film];"Print_List3";Form Detail)
                vLprinted_height:=vLprinted_height+$vLheight
        End if
        NEXT RECORD([Film])
End while
PAGE BREAK `最後のページが印刷されたことを確認
```

フォームメソッドPrint_List3は次の通りです。

```
C_LONGINT($l;$t;$r;$b;$fixed_wdth;$exact_hght;$l1;$t1;$r1;$b1)
C_LONGINT($final_pos;$i)
C_LONGINT($detail_pos;$header_pos;$hght_to_print;$hght_remaining)
Case of
        \ (vSprint_area="Detail") `詳細行の印刷が進行中
                GET OBJECT RECT([Film]Actors;$l;$t;$r;$b)
                $fixed_wdth:=$r-$l`Actorsテキストフィールドサイズの計算
                $exact_hght:=$b-$t
                BEST OBJECT SIZE([Film]Actors;$wdth;$hght;$fixed_wdth)
                        `内容に応じたフィールドの最適サイズ
                $movement:=$hght-$exact_hght
                GET OBJECT RECT([Film]Summary;$l1;$t1;$r1;$b1)
                $fixed_wdth1:=$r1-$l1`Summaryテキストフィールドサイズの計算
                $exact_hght1:=$b1-$t1
                BEST OBJECT SIZE([Film]Summary;$wdth1;$hght1;$fixed_wdth1)
                        `内容に応じたフィールドの最適サイズ
                $movement1:=$hght1-$exact_hght1
                If($movement1>$movement)
                        `最も高さがあるフィールドを決定する
                        $movement:=$movement1
                End if

                If($movement>0)
                        $position:=Get print marker(Form Detail)
                        $final_pos:=$position+$movement
                                `Detailマーカーと、これに続くマーカーを移動
                        SET PRINT MARKER(Form Detail;$final_pos;*)
                                `テキストエリアのサイズ変更
                        MOVE OBJECT([Film]Actors;$l;$t;$r;$hght+$t;*)
                        MOVE OBJECT([Film]Summary;$l1;$t1;$r1;$hght1+$t1;*)
                                `分割ラインのサイズ変更
```

```
            GET OBJECT RECT(*;"H1Line";$l;$t;$r;$b)
            MOVE OBJECT(*;"H1Line";$l;$final_pos-1;$r;$final_pos;*)
            Loop($i;1;4;1)
                  GET OBJECT RECT(*;"VLine"+文字列($i);$l;$t;$r;$b)
                  MOVE OBJECT(*;"VLine"+文字列($i);$l;$t;$r;
                                                $final_pos;*)
            End of loop
      End if

            `利用できるスペースの計算
      $detail_pos:=Get print marker(Form Detail)
      $header_pos:=Get print marker(Form Header)
      $hght_to_print:=$detail_pos-$header_pos
      $hght_remaining:=printing_height-vLprinted_height
      If($hght_remaining<$hght_to_print) `不十分な高さ
            CANCEL `フォームを次ページに移動
      End if
End case
```

ストラクチャアクセスコマンド

Get database parameter

Get database parameter ({テーブル;} セレクタ) →倍長整数

引数	タイプ		説明
テーブル	テーブル	→	属性の値を得るテーブル。この引数が省略されている場合はデフォルトテーブル
セレクタ	倍長整数	→	データベース属性コード
戻り値	倍長整数	←	属性の値

説明

このコマンドは、カレントプロセス用の4Dデータベース属性の値を読み込むことができます。

セレクタ引数は、読み込む属性を指定します。4th Dimensionは「データベース属性」のカテゴリー内に、前もって定義されている下記のような定数があります。

定数	タイプ	値
Seq Order Ratio	倍長整数	1
Seq Access Optimization	倍長整数	2
Seq Distinct Values Ratio	倍長整数	3
Index Compacting	倍長整数	4
Seq Query Select Ratio	倍長整数	5
Minimum Web Process	倍長整数	6
Maximum Web Process	倍長整数	7
Web Conversion Mode	倍長整数	8
Database Cache Size	倍長整数	9
4th Dimension Scheduler	倍長整数	10
4D Server Scheduler	倍長整数	11
4D Client Scheduler	倍長整数	12
4D Server Timeout	倍長整数	13
4D Client Timeout	倍長整数	14
Port ID	倍長整数	15
IP Addres to listen	倍長整数	16
Character set	倍長整数	17
Max Concurrent Web Processes	倍長整数	18
Client Minimum process Web	倍長整数	19
Client Maximum process Web	倍長整数	20

Client Maximum Web requests size	倍長整数	21
Client Port ID	倍長整数	22
Client IP Address to listen	倍長整数	23
Client Character set	倍長整数	24
Client Max Concurrent Web Proc	倍長整数	25
Cache Writing Mode	倍長整数	26
Maximum Web requests size	倍長整数	27

この関数によって返される値については、**SET DATABASE PARAMETER**コマンドの内容を参照してください。

データベースキャッシュサイズ（9）のセレクタはカレントのデータベースのメモリキャッシュのサイズを得ることができます。戻り値はバイトで示されます。最大キャッシュサイズはWindows、Macintoshの両方のプラットフォームでセットでき、最小キャッシュサイズはMacintoshのプラットフォームでのみセットできます。セットするには、データベースプロパティのダイアログボックスでします。実際のサイズはデータベースキャッシュはセットの仕方とカレントのシステムリソースの両方によって割り当てられます。**Get database parameter**関数は4Dによってデータベースキャッシュを割り当てたメモリの際のサイズを得ることができます。

注：ランゲージを使ってはデータベースキャッシュメモリのサイズをセットできません。つまり、データベースキャッシュサイズセレクタはSET DATABASE PARAMETERコマンドを使ってもセットされません。

例題

(1)以下のメソッドにより、4Dスケジューラの現在の値を得ることができます。

```
C_LONGINT($ticksbtwcalls;$maxticks;$minticks;$lparams)
If (Application type=4th Dimension) ` 4D シングルユーザが使われているとき
$lparams:=Get database parameter(4th Dimension scheduler)
$ticksbtwcalls:=$lparams & 0x00ff
$maxticks:=($lparams>>8) & 0x00ff
$minticks:=($lparams>>16) & 0x00ff
End if
```

(2)セレクタ16（IP Address to listen）により、HTTPのリクエストを受けた4D Webサーバの IPアドレスを取得することができます。以下の例は16進化10進数への分割を行います。

```
C_LONGINT($a;$b;$c;$d)
C_LONGINT($addr)
$addr:=Get database parameter(IP Address to listen)
$a:=($addr>>24)&0x000000ff
$b:=($addr>>16)&0x000000ff
```

```
$c:=($addr>>8)&0x000000ff
$d:=$addr&0x000000ff
```

参照

DISTINCT VALUES、QUERY SELECTION、SET DATABASE PARAMETER

SET DATABASE PARAMETER

SET DATABASE PARAMETER ({テーブル;} セレクタ;値)

引数	タイプ		説明
テーブル	テーブル	→	属性を設定するテーブル 引数が省略されている場合は デフォルトテーブル
セレクタ	倍長整数	→	変更するデータベース属性コード
値	倍長整数	→	属性の値

説明

このコマンドは、カレントプロセス用に4Dデータベース内部の様々な属性を変更することができます。

セレクタは、変更するデータベースの属性コードを指定します。4th Dimensionは「データベース属性」のカテゴリー内に、前もって定義されている下記のような定数があります。

定数	タイプ	値
Seq Order Ratio	倍長整数	1
Seq Access Optimization	倍長整数	2
Seq Distinct Values Ratio	倍長整数	3
Index Compacting	倍長整数	4
Seq Query Select Ratio	倍長整数	5
Minimum Web Process	倍長整数	6
Maximum Web Process	倍長整数	7
Web Conversion	倍長整数	8
Database Cache Size	倍長整数	9
4th Dimension Scheduler	倍長整数	10
4D Server Scheduler	倍長整数	11
4D Client Scheduler	倍長整数	12
4D Server Timeout	倍長整数	13
4D Client Timeout	倍長整数	14
Port ID	倍長整数	15

IP Addres to listen	倍長整数	16
Character set	倍長整数	17
Max Concurrent Web Processes	倍長整数	18
Client Minimum process Web	倍長整数	19
Client Maximum process Web	倍長整数	20
Client Maximum Web requests size	倍長整数	21
Client Port ID	倍長整数	22
Client IP Address to listen	倍長整数	23
Client Character set	倍長整数	24
Client Max Concurrent Web Proc	倍長整数	25
Cache Writing Mode	倍長整数	26
Maximum Web requests size	倍長整数	27

値は、属性の値を指定します。値の内容は変更しようとする属性によって違います。セレクタで指定する可能性のある値を示します。

セレクタ＝1 (Seq Order Ratio)(シーケンシャルソートの実行)

値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下ではソートがシーケンシャルモードで実行されます。この率は、100,000分の1単位で表わされます。デフォルト値は9,000（=9%）です。

セレクタ＝2 (Seq Access Optimization)(シーケンシャルソートの最適化)

値：0または1（0：最適化されず、1：最適化する）

内容：シーケンシャルアクセス（配列のソート、検索、選択）用の最適化モード。最適化モードでは、4Dはディスクからの多くのレコードを一度に読もうとしますが、これらをキャッシュ内には置きません。このモードは、キャッシュのサイズが低いためです。デフォルトでは、値は1になります（最適化モード）。

セレクタ＝3 (Seq Distinct Values Ratio)(シーケンシャルDistinct Valuesの実行)

値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下では**DISTINCT VALUES**コマンドがシーケンシャルモードで実行されます。この率は、100,000分の1単位で表わされます。デフォルト値は0です。

セレクタ＝4 (Index Compacting)(インデックスの圧縮)

値：0または1（0：no、1：yes）

内容：インデックスページ圧縮の可能または不可能。デフォルトでは値は1（インデックスは必要であればコンパクト化される）です。インデックスページは多くのインデックスやレコードを含むデータベースでは、4Dのメモリキャッシュを多量に使います。キャッシュがいっぱいで4Dが空きの追加を必要としていると、キャッシュ内のデータは正しくアンロードしません。データをアンロードする前に空きを増やさないと、プログラムはインデックスページの圧縮に空きスペースができるかチェックします。別の方法は後でデータを再ロードを避けることが可能です。

セレクタ＝5 (Seq Query Select Ratio)(シーケンシャル検索の実行)

値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下では**QUERY SELECTION**コマンドがシーケンシャルモードで実行されます。この率は、100,000分の1単位で表わされます。デフォルト値は0です。

セレクタ=6 (Minimum Web Process)(Webプロセスの最小数)

値：0→32,767

内容：4th Dimensionならびに4D Serverを使用した場合に、非コンテキストモードで保持するWebプロセスの最少数。デフォルトでは、値は0になります（下記参照）。

セレクタ＝7 (Maximum Web Process)(Webプロセスの最大数)

値：0→32,767

内容：4th Dimensionならびに4D Serverを使用した場合に、非コンテキストモードで保持するWebプロセスの最少数。デフォルトでは、値は10になります。

Webサーバが、非コンテキストモードでプロセスの再利用をするために、4DはWebプロセスを5秒間延滞し、次に起こりうるHTTPリクエストの実行のために待機させます。能力の面で言えば、各問い合わせに新しいプロセスを作成するよりも、この原理はずっと利点の多いものです。Webプロセスが再利用されると、もう一度5秒間延滞させられます。5秒以内に何のリクエストも発生しない場合、Webプロセス数が指定した最小数でなければプロセスはアボートされ、最小数に達した場合は再度延滞させられます。

これらの引数は、リクエストの数やメモリ等に応じて、Webサーバの機能を調整できるようにするものです。

セレクタ＝8 (Web conversion mode)(Web変換モード)

値：0、1，2または3

0＝（デフォルト）ブラウザが対応している場合はHTML 4.0フォーマットに変換し、対応していない場合はHTML3.2と配列を使います。

1＝6.0.x変換モード

2＝6.5変換モード

3＝HTML4.0フォーマット+CSS-Pに変換（バージョン6.5.2から）

説明：4th Dimensionならびに4D Serverで使用するWeb用の4Dフォームの変換モード。デフォルトでは、4D Web ServerはCSS1（cacading style sheets）を使用し、4th Dimensionで表示される4Dフォームと同様のHTMLページを生成します。この機能を使用すると、バージョン 6.7より前の4Dで作成されたデータベースに関しては、フォームが正しく変換されない可能性があります。このため、フォーム変換モードを設定する必要があるかもしれません。

このモードは、**SET DATABASE PARAMETER**が呼び出されたプロセス（Webコンテキスト）に対してのみ設定することができます。このコマンドは、「On Web Connection」データベースメソッド内で呼び出してデータベースのすべてのフォームを確実に統一したり、あるいは特定のフォームを表示する前にのみ呼び出すこともできます。このコマンドは、コンテストモード、またはWebプロセス以外の場所から呼び出すと、何も行いません。

注：セレクタの追加はGet database parameter関数のデーターベースキャッシュサイズ（9）でできます。このセレクタはSET DATABASE PARAMETER関数では出来ません。さらに詳しい説明はGet database parameterを参照してください。

セレクタ ＝ 10 (4th Dimension Scheduler)

セレクタ ＝ 11 (4D Server Scheduler)

セレクタ ＝ 12 (4D Client Scheduler)

値：これら3つのセレクタに対し、引数＜値＞は16進数、0x00aabbcc の形式で表わされます。詳細は次の通りです。

aa = システムへのコール毎の最小tick数（0 ～ 100）

bb = システムへのコール毎の最大tick数（0 ～ 100）

cc = システムへのコール間のtick数（0 ～ 20）

これらの値のうち1つが範囲外であれば、4Dによって最大数に設定されます。引数＜値＞には、次の定義済標準値のうちいずれかを渡すことができます。

値 = -1：4Dに割り当てられた最高優先度

値 = -2：4Dに割り当てられた平均優先度

値 = -3：4Dに割り当てられた最低優先度

説明：この引数を使用して、4Dシステム内部コールをダイナミックに設定することができます。セレクタの値に応じて、スケジューラの値は次のアプリケーションのために設定されます。

■ このコマンドが4th Dimensionから呼び出された場合、4th Dimension（シングルユーザ）および4D Tools（セレクタ = 10）。

■ このコマンドが4D Serverから呼び出された場合、4D Server（セレクタ = 11）。

■ このコマンドが4D Clientから呼び出された場合、4D Client（セレクタ = 12）。

（例題1を参照）

セレクタ = 13（4D Server Timeout）

説明：この引数を使用して、4D Serverのタイムアウトの値を変更することができます。4D Serverのタイムアウトのデフォルト値は、サーバ側の「データベースプロパティ」ダイアログボックスの「接続設定」ページで定義します。

セレクタ「4D Server Timeout」により、対応する引数＜値＞に新しいタイムアウト（分単位で指定）を設定できます。この機能は、クライアント側でＣＰＵを占有する時間がかかる処理を実行する前に、タイムアウト設定を長くしたい場合は特に便利です。例えば、膨大なページの印刷などは、予期しないタイムアウトになる可能性があります。

また、2種類のオプションがあります。

■ 引数＜値＞に正の値を渡すと、グローバルかつ永続的なタイムアウトが設定されます。この新しい値はすべてのプロセスに対して適用され、4Dアプリケーションの初期設定に保存されます（「データベースプロパティ」ダイアログボックスで変更した場合と同じ）。

■ 引数＜値＞に負の値を渡すと、ローカルで一時的なタイムアウトが設定されます。この新しい値は呼び出し元のプロセスに対してのみ適用され（他のプロセスではデフォルトの値を維持）、例えば処理の終了時のように、クライアントが動作していることを示す信号をサーバが受信すると即座に、デフォルト値へリセットされます。このオプションは、4Dプラグインにより開始された時間のかかる処理を管理する際に便利です。

“タイムアウトしない”オプションを設定するには、＜値＞に 0 を渡します。（例題2を参照）

セレクタ = 14（4D Client Timeout）

説明：この引数を使用して、4D Clientのタイムアウトの値を変更することができます。4D Clientのタイムアウトのデフォルト値は、クライアント側の「データベースプロパティ」ダイアログボックスの「接続設定」ページで定義します。

このセレクタに関する詳細は、「4D Server Timeout」の説明（13）を参照してください。

4D Clientのタイムアウトは、非常に特殊な状況において変更されます。

セレクタ = 15（Port ID）

説明：この引数を使用して、4th Dimensionおよび4D Serverによる4D Webサーバが使用するTCPポートのIDをオンザフライで変更することができます。デフォルト値は80で、この値は「データベースプロパティ」ダイアログボックスの「Web サーバI 」ページで設定することができます。

セレクタ「Port ID」は、コンパイルしてエンジンを組み込んだ4D Web サーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。TCPポートIDに関する詳細は、「Webサービス：システム設定」の節を参照してください。

セレクタ = 16（IP Address to listen）

説明：この引数を使用して、4th Dimensionおよび4D Serverによる4D WebサーバがHTTPリクエストを受信するIPアドレスをユーザがオンザフライで変更することができます。デフォルトでは、特定のアドレスは定義されていません（＜値＞= 0）。この引数は「データベースプロパティ」ダイアログボックスの「Web サーバI 」ページで設定することができます。

セレクタ「IP Address to listen」は、コンパイルしてエンジンを組み込んだ4D Webサーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。

引数＜値＞には、16 進数のIPアドレスを渡します。つまり、“a.b.c.d”のようなIPアドレスを指定するには、以下のようなコードを作成します。

```
C_LONGINT($addr)
$addr:=($a<<24)|($b<<16)|($c<<8)|$d
SET DATABASE PARAMETER(IP Address to listen;$addr)
```

例題3も参照してください。IPアドレスの設定方法に関する詳細は、「Webサービス：Webサーバセッティング」の節を参照してください。

セレクタ = 17（Character set）

値：

0：Western European（西ヨーロッパ）

1：Japanese（日本語）

2：Chinese（中国語）

3：Korean（韓国語）

4：User-defined（ユーザ定義）

5：Reserved（予備）

6：Central European（中央ヨーロッパ）

7：Cyrillic（キリル文字）

8：Arabic（アラビア語）

9：Greek（ギリシャ語）

10：Hebrew（ヘブライ語）

11：Turkish（トルコ語）

12：Baltic（バルト語）

説明：この引数を使用して、ユーザはデータベースに接続しているブラウザとの通信に、4D Webサーバ（4th Dimensionならびに4D Serverを使用）が使用する文字セットをオンザフライで変更することができます。実際のところ、デフォルト値はOSの言語に依存します。

この引数は「データベースプロパティ」ダイアログボックスの「Web サーバII 」ページで設定することができます。セレクタ「Character set」は、コンパイルしてエンジンを組み込んだ4D Webサーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。

セレクタ = 18（Max Concurrent Web Processes）

値：デフォルト値は32,000ですが、10から32,000までの任意の値を渡すことができます。

説明：この引数を使用して、4th Dimensionならびに4D Serverを用いた4D Webサーバでサポートされる任意のタイプの同時Webプロセス上限数（コンテキスト、非コンテキスト、または“プロセス再利用”に属するプロセス－セレクタ7、「Webプロセスの最大数」を参照）を正確に設定することができます。この上限数（マイナス1）に達した場合、4Dはそれ以上プロセスを作成しなくなり、HTTPステータス503（「Service Unavailable to all new requests」すべての新しいリクエストへのサービス不可）を返します。

この引数により、同時に行われる非常に膨大な数のリクエストやコンテキスト作成に関する過大な要求の結果として、サーバが飽和状態になることを防げます。また、この引数は「データベースプロパティ」ダイアログボックスでも設定することができます（「Webサービス：Webサーバセッティング」の節を参照）。

理論上、Webプロセスの最大数は次の計算式の結果になります：使用可能メモリ/Webプロセスのスタックサイズ。別の解決策は、ランタイムエクスプローラに表示されるWebプロセス情報を示す方法です。つまり現在のWebプロセス数およびWebサーバの開始以降に達した最大数が示されている情報です。

注：“プロセスの再利用”の上限数より小さい値を渡した場合、この上限数はセレクタ18の値に合わせるために減らされます。必要であれば、再利用の下限数（セレクタ6、Webプロセスの最小数）も変更できます。

セレクタ = 19(Client Minimum process Web)

セレクタ = 20(Client Maximum process Web)

セレクタ = 21(Client Max Web requests size)

セレクタ = 22(Client Port ID)

セレクタ = 23(Client IP Address to listen)

セレクタ = 24(Client Character set)

セレクタ = 25（Client Max Concurrent Web Proc）

値：4th Dimension や 4D Server の対応するセレクタと同じ（セレクタ 6 から 8、15 から 18、および 27 を参照）

説明：これらのセレクタを使用して、Web サーバとして使用する 4D Client マシンの操作用パラメータを指定することができます。

これらのセレクタを用いて指定された値は、Web サーバとして使用するすべての 4D Client マシンに対して適用されます。特定の 4D Client マシンに対してのみ値を指定したい場合には、4D Client の「環境設定」ダイアログボックスを使用してください。

セレクタ = 26（Cache writing mode）

値：0 または 1（0= 無効、1= 有効）

説明：最適化されたキャッシュ書き込みモードを有効、または無効に設定します。デフォルトでは、この値は 1 になります（有効モード）。

バージョン 2003 の 4th Dimension より、最適化されたキャッシュ書き込みモードはデフォルトとして有効に設定され、特に MacOS において 4D アプリケーションの処理速度が著しく向上します。

セレクタ = 27（Maximum Web requests size）

値：500 000 から 2 147 483 648

説明：Web サーバが処理を許可された受信 HTTP リクエスト（POST）の最大数（バイト単位）。デフォルトでは、この値は 2 000 000 になります。最大値（2 147 483 648）を渡すと、実際上は制限がなくなります。

この制限を使用し、受信するリクエストが多すぎるために Web サービスが限界に達してしまう危険性を回避します。リクエストがこの制限に達すると、4D Web サービスはリクエストを拒否します。

セレクタの有効範囲

以下の表に各セレクタの有効範囲を示します。

セレクタ	値	有効範囲
Seq Order Ratio	1	カレントテーブルとプロセス
Seq Access Optimization	2	カレントテーブルとプロセス
Seq Distinct Values Ratio	3	カレントテーブルとプロセス
Index Compacting	4	4D アプリケーション(*)
Seq Query Select Ratio	5	カレントテーブルとプロセス
Minimum Web Process	6	4th Dimension、4D Server(*)
Maximum Web Process	7	4th Dimension、4D Server(*)
Web conversion mode	8	カレントプロセス
Database cache size	9	4D アプリケーション(*) (**)
4th Dimension Scheduler	10	4D アプリケーション(*)
4D Server Scheduler	11	4D アプリケーション(*)
4D Client Scheduler	12	4D アプリケーション(*)
4D Server Timeout	13	4D アプリケーション（正の数の場合）(***)
4D Client Timeout	14	4D アプリケーション（正の数の場合）(***)
Port ID	15	4th Dimension、4D Server(*)
IP Address to listen	16	4th Dimension、4D Server(*)
Character set	17	4th Dimension、4D Server(*)
Max Concurrent Web Processes	18	4th Dimension、4D Server(*)
Client Minimum process Web	19	すべての 4D Client マシン(*)
Client Maximum process Web	20	すべての 4D Client マシン(*)
Client Max Web requests size	21	すべての 4D Client マシン(*)
Client Port ID	22	すべての 4D Client マシン(*)
Client IP Address to listen	23	すべての 4D Client マシン(*)
Client Character set	24	すべての 4D Client マシン(*)
Client Max Concurrent Web Proc	25	すべての 4D Client マシン(*)
Cache writing mode	26	4D アプリケーション(*)
Maximum Web requests size	27	4th Dimension、4D Server(*)

(*) この場合、引数＜テーブル＞は無視されます。
(**) このセレクタは読み込みのみ可能です（Get database parameter コマンドを参照）。
(***)引数＜値＞が負の値である場合、この設定はカレントプロセスにのみ影響し、次回のリクエストの際にはリセットされます。

例題

(1) シングルユーザ版の 4D を実行している場合、次のメソッドを使用して、スケジューラの値を定義することができます。

C_LONGINT($ticksbtwcalls;$maxticks;$minticks;$lparams)

```
If(Application type=4th Dimension)`シングルユーザの4Dを使用
    $ticksbtwcalls:=12
    $maxticks:=20
    $minticks:=7
    $lparams:=($minticks<<16)¦($maxticks<<8)¦$ticksbtwcalls
    SET DATABASE PARAMETER (4th Dimension scheduler;$lparams)
End if
```

(2)以下のコードでは、予期しないタイムアウトを回避しています。

```
`カレントプロセスに対してタイムアウトを3時間まで延長する
SET DATABASE PARAMETER(4D Server Timeout;-60*3)
`4Dの制御を受けずに、時間のかかる処理を実行する
...
WR PRINT MERGE (Area;3;0)
...
```

(3)IPアドレス192.193.194.195は、次のコードを使用して設定します。

```
SET DATABASE PARAMETER(IP Address to listen;0xC0C1C2C3)
```

参照

DISTINCT VALUES、GET database parameter、QUERY SELECTION

システムドキュメントコマンド

Create document

Create document (ドキュメント {; タイプ}) → ドキュメントファイル参照番号

引数	タイプ		説明
ドキュメント	文字列	→	ドキュメントファイル名、または ドキュメントへの完全なパス名、または 空の文字列の場合、標準のファイル ダイアログボックス表示
タイプ	文字列	→	Macintoshファイルタイプ（4桁の文字）、 または Windowsファイル拡張子（1～3桁の文字）、 または省略した場合、 テキストドキュメント（.TXT）
戻り値	DocRef	←	ドキュメントファイル参照番号

説明

Create document関数は、＜ドキュメント＞で指定した名前の新しいドキュメントファイルを作成し、そのドキュメントファイルのドキュメントファイル参照番号を返します。

＜ドキュメント＞には新しいドキュメントファイルの名前、または完全なパス名を渡します。＜ドキュメント＞が既にディスク上に存在する場合には、そのドキュメントファイルを上書きします。しかし、＜ドキュメント＞がロックされていたり、既に開かれている場合には、エラーが発生します。

＜ドキュメント＞が空の文字列（ヌル""）の場合には、「ファイル作成」ダイアログボックスを開きます。ユーザは、ここで新しいドキュメントファイルの名前を入力することができます。このダイアログをキャンセルすると、ドキュメントファイルは作成されず、**Create document**関数はドキュメントファイル参照番号にヌル値を返し、システム変数OKに0を代入します。

ドキュメントが正常に作成され、開かれると、**Create document**関数はドキュメントファイル参照番号を返し、システム変数OKに1を代入します。また、システム変数Documentが更新され、作成されたドキュメントのアクセスパスを返します。

「ファイル作成」ダイアログボックスを使用するかしないかに関わらず、**Create document**関数はデフォルトとしてTEXT（Windows）またはTEXT（Macintosh）タイプのドキュメントファイを作成します。別のタイプのドキュメントファイルを作成するには、引数＜ファイルタイプ＞を指定します。

Macintoshでは、ファイルタイプを渡します。Windowsでは、1から3文字のWindowsのファイル拡張子、または**MAP FILE TYPES**コマンドを使ってマップされるMacintoshのファイルタイプを指定します。拡張子なしのドキュメント、いくつかの拡張子を含むドキュメント、または3文字を超える拡張子を含むドキュメントを作成したい場合は、引数<タイプ>を使わずにフルパスネームを引数<ドキュメント>へ渡して下さい。

ドキュメントファイルを作成し、開いた後は、**RECEIVE PACKET**や**SEND PACKET**コマンドを使用してドキュメントへの読み込みや書き込みを行えます。また、これらのコマンドと**Get file position**や**SET FILE POSITION**コマンドを組み合わせて、ドキュメントの一部に直接アクセスすることもできます。

最後に、開かれたドキュメントファイルに対して**CLOSE DOCUMENT**を呼び出すことを忘れないようにしてください。

例題

1. 以下の例は、新しいドキュメントファイル“ノート”を作成し、それに“こんにちは”という文字列を書き込み、ドキュメントファイルを閉じます。

```
C_TIME (vDoc)
vドキュメント:=Create document ("ノート")` 新しいドキュメントファイル
                                                    "ノート"を作成
If (OK=1)
      SEND PACKET (vドキュメント ; "こんにちは") ` ドキュメントファイルに
                                                    書き込む
      CLOSE DOCUMENT (vドキュメント)     ` ドキュメントファイルを閉じる
End if
```

ドキュメントファイル“ノート”をワープロソフトウェアで開いてみると、“こんにちは”という文字列が含まれています。

2. 以下の例はWindows上で標準的ではない拡張子をともなうドキュメントを作成しています。

```
$vtMyDoc:=Create document("Doc.ext1.ext2") `Several extensions
$vtMyDoc:=Create document("Doc.shtml") `Long extension
$vtMyDoc:=Create document("Doc.") `No extension (the period "." is mandatory)
```

システム変数とセット

ドキュメントが正常に作成されると、システム変数OKに1が代入され、システム変数Documentにはドキュメントのアクセスパスが代入されます。

参照

Append document、Open document

PLATFORM PROPERTIES

PLATFORM PROPERTIES (プラットフォーム;システム;マシン)

引数	タイプ		説明
プラットフォーム	数値	→	1=68K ベースの Macintosh
		→	2=Power Macintosh
		→	3=Windows
システム	数値	→	動作している種類に依存
マシン	数値	→	動作している種類に依存

説明

PLATFORM PROPERTIES コマンドは、実行しているプラットフォーム、オペレーションシステム（OS）のバージョン、使用しているマシンに搭載されたプロセッサの種類に関する情報を返します。

PLATFORM PROPERTIES コマンドは、引数＜プラットフォーム＞、＜システム＞、＜マシン＞に4D環境の情報を返します。

＜プラットフォーム＞は、ユーザが実行している4th Dimensionのバージョンが68Kの Macintoshなのか、それともPower Macintosh、またはWindowsなのかを示します。この引数は、次のようにあらかじめ定義されている定数の1つを返します。

定数	タイプ	値
Macintosh 68K	倍長整数	1
Power Macintosh	倍長整数	2
Windows	倍長整数	3

＜システム＞と＜マシン＞の中に返される情報は、動作している4th Dimensionの種類によって異なります。

Macintosh版

Macintosh版の4th Dimensionを実行している場合、引数＜システム＞には32ビット（倍長整数）値が返され、上位16ビットは未使用で、下位16ビットは以下のような構造になっています。

■ 上位バイトには、主要なバージョン番号が入っています。

■ 下位バイトは、2つの部分（各4ビット）に分かれています。上位部分はアップデートの主要なバージョン番号で、下位部分はアップデート番号の枝番です。例えばSystem9.0.4は$0904のようにコード化されるため、10進数では2308になります。

注：4Dで、これらの値は%（モジューロ）と//（整数値を返す除算）を使った数値計算で求められます。また、「ビットワイズ」演算子を使っても求められます。

MacOSのメインバージョンを調べるには、次の式を使用します。

```
PLATFORM PROPERTIES($vlPlatform;$vlSystem)
$vlResult:=$vlSystem//256
    `If$vlResult= 8--> MacOS 8.xで動作しています
    `If$vlResult= 9--> MacOS 9.xで動作しています
    `If$vlResult= 16--> MacOS 10.xで動作しています
```

Windows版

Windows版の4th Dimensionを実行している場合、引数＜システム＞からは32ビット（倍長整数）値が返され、そのビットとバイトは以下のような構造になっています。

上位レベルのビットが0になっている場合、Windows NT 4またはWindows 2000の特色を持つシステムで実行していることを示します。ビットが1になっている場合は、Windows 95またはWindows 98のもとで実行しているということを意味します。

注：上位レベルのビットは倍長整数の符号に使われています。したがって、4Dでは、値が正か負かを調べるだけです。正ならWindows NT、Windows 2000、またはWindows XPが動作しています。

下位バイトには、Windowsの主要なバージョン番号が入っています。4が返された場合、Windows 95、98もしくはWindows NTが動作中であることを示します。5が返された場合、Windows 2000またはWindows XPが動作中であることを示します。

以下の下位バイトにはWindowsのバージョン番号の枝番が入っています。Windows 95が動作中である場合、この値は0になります。

注：4Dで、これらの値は%（モジューロ）と//（整数値を返す除算）を利用した数値計算で求められます。また、バージョン6から追加された「ビットワイズ」演算子を使っても求められます。

■ 引数＜マシン＞は、次の定義済定義のいずれかと比較できる値を返します。

定数	タイプ	数値
IINTEL 386	倍長整数	386
INTEL 486	倍長整数	486

Pentium	倍長整数	586
PowerPc 601l	倍長整数	601
PowerPc 603	倍長整数	603
PowerPc 604	倍長整数	604
PowerPc G3	倍長整数	510
その他G3以上	倍長整数	406

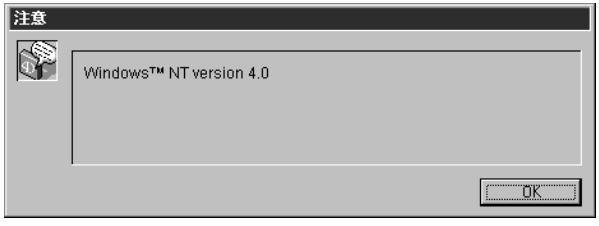
注：Macintoshの数値に関するアップデート一覧は、Apple Computer, Inc.よりDeveloperおよびTechnicalドキュメント内で公開されます。Apple社やその他メーカーが新しいMacintoshモデルを発表した場合には、新しい値が追加される可能性があります。

▼以下のプロジェクトメソッドは、使用しているOSを示した「警告」ボックスを表示します。

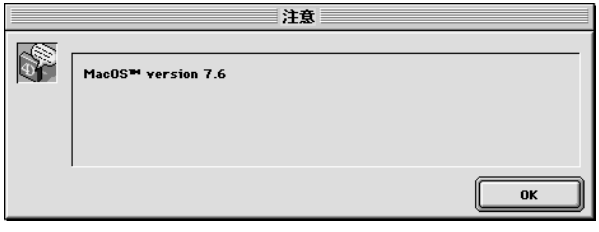
```
`「SHOW OS VERSION」プロジェクトメソッド
PLATFORM PROPERTIES ($vlPlatform ; $vlSystem ; $vlMachine)
If (($vlPlatform<1) | (3<$vlPlatform))
	$vsPlatformOS:=""
Else
	If ($vlPlatform=3)
		$vsPlatformOS:=""
		If ($vlSystem<0)
			$winMajVers:=((2^31)+$vlSystem)%256
			$winMinVers:=(((2^31)+$vlSystem)\256)%256
			Case of
				¥ ($winMajVers=4)
					$vsPlatformOS:="Windows™ NT"
				¥ ($winMajVers=5)
					If($winMinVers=0)
						$vsPlatformOS:="Windows™ 2000"
					Else
						$vsPlatformOS:="Windows™ XP"
					End if
			End case
		End if
		$vsPlatformOS:=$vsPlatformOS+" バージョン"
				+String ($winMajVers)+"."+String ($winMinVers)
	Else
		$vsPlatformOS:="MacOS® version"
		If(($vlSystem//256) = 16)
			$vsPlatformOS:=$vsPlatformOS+"10"
		Else
```

```
                $vsPlatformOS:=$vsPlatformOS+String($vlSystem\256)
            End if
            $vsPlatformOS:=$vsPlatformOS+"."+String(($vlSystem\16)%16)+
                (("."+String($vlSystem%16))*Num(($vlSystem%16) # 0))
        End if
    End if
    ALERT($vsPlatformOS)
```

Windows 上では、次の図のような「警告」ボックスを表示します。



Macintosh 上では、次の図のような「警告」ボックスを表示します。



参照

なし

Gestalt

Gestalt (セレクタ ; 値)→ 文字列

引数	タイプ		説明
セレクタ	文字列	→	4 文字の Gestalt セレクタ
値	数値	←	gestalt 値
戻り値	数値	←	エラーコード

説明

Gestalt 関数は、ユーザが＜セレクタ＞に渡すセレクタに基づいて、システムハードウェアおよびソフトウェアの特性を示す数値を＜値＞に返します。

必要な情報が得られると、**Gestalt** 関数の結果として 0 を、取得できなかった場合には、エラーコード -5550 を返します。セレクタがわからなければ、**Gestalt** 関数はエラーコード
-5551 を返します。

重要：Gestalt マネージャは Macintosh の一部です。セレクタのいくつかは Windows 上でも実現されていますが、このコマンドの有効性は Windows 上では限られています。

Gestalt 関数に渡すことができるセレクタについての詳細は、Gestalt マネージャに関する Apple 社の開発者向け（Developer）ドキュメントを参照してください。

オンラインでのアドレスは、以下の通りです。

http://developer.apple.com/techpubs/macosx/Carbon/oss/GestaltManager/gestaltmanager.html

例題

Machintosh 上で、MacOS のバージョン 7.6 を使用している場合、以下のコードは、「システムバージョン 0x0760 を実行しています。」という警告を表示します。

```
$vlErrCode:=Gestalt ("sysv" ; $vlInfo)
If ($vlErrCode=0)
        ALERT ("システムバージョン："+String ($vlInfo ; "&x")
                                                + "を実行しています。")
End if
```

参照

なし

Webサーバコマンド

概要

4th Dimension、4D Server、ならびに4D Clientには、4Dデータベースやあらゆるタイプの HTMLページをWeb上に公開できるWebサーバエンジンが組み込まれています。4D Web サーバエンジンの主な機能は次の通りです。

■ やさしい公開方法

Web上へのデータベース公開は、いつでも開始したり中止することができます。これを行うには、メニューコマンドを選択するか、またはランゲージコマンドを実行するだけです。

■ コンテキストモードと非コンテキストモード

4D Webサーバは、コンテキストモードと非コンテキストモードという2つのモードで動作することができます。4D Webサーバは、これらのうち一方のモードで使用するか、あるいは必要に応じてあるモードからもう一方のモードへオンザフライで切り替えることもできます。

■ コンテキストモード（4th Dimensionならびに4D ServerのWebサーバでのみ利用可能）では、ユニークで独創的な機能が提供されます。このモードにおいて、4DはWebブラウザを標準的なデータベースクライアントとして管理します。作成したデータベースは直接Web上に公開されます。データベースやWebサイト、ならびにこの2つの間の橋渡しをするCGIの開発は必要ありません。データベースがWebサイトになります。

データベースのストラクチャやデータに変更が行なわれると、そのデータベースに接続しているすべてのブラウザへ即座に反映されます。4Dはデータベースのメニューバーやフォーム、メソッドをHTMLに変換します。したがって、HTMLを知らなくてもWeb上に4Dデータベースを公開することができます。4Dは各Webブラウザのコンテキスト（セレクション、変数等）を用いてデータを自動的に管理します。

この代わりに、コンテキストモードにおけるWebナビゲーションには特有の制約があります。詳細については、「コンテキストモードの使用」の節を参照してください。

■ 非コンテキストモード（標準モード）で使用する場合、4D Webサーバは完全に標準的なHTTPサーバになります。つまり、Webページはコンテキストを保持する必要なく送信されます。ユーザは、Webブラウザにページを送信する前に4Dデータベースのデータにアクセスし、スタティックデータとデータベースのデータを共に含む“セミダイナミック”HTMLページを作成することができます。Webサーバによる処理を全く必要としないスタティックなWebページを送信することも可能です。

■ 専用のデータベースメソッド

「**On Web Authentication**」と「**On Web Connection Database**」データベースメソッドは、Webサーバにおけるリクエストの入り口点となります。これらのメソッドを使用して、あらゆるタイプのリクエストの評価やルート付けを行います。

■ 特別なタグとURLの使用

4D Webサーバでは、ユーザとのやりとりを可能にするさまざまなメカニズムが提供されています。特に次のような事柄です。

■ ブラウザ側に送信されるとWebサーバの処理が開始する特別なタグをWebページに組み込むことができます。

■ 任意のアクションを実行するために4Dを呼び出す特別なURLを使用することができます。

■ これらのURLをフォームアクションとして使用し、ユーザによるHTMLフォームの送信時に処理を開始するすることもできます。

■ アクセスセキュリティ

自動設定オプションを使用し、Webブラウザに対する特定のアクセス権を認可したり、4th Dimensionに統合されたパスワードシステムを使用することができます。また、データベース内でのアクセス管理を簡略化するために、“一般Webユーザ”を定義することができます。

「**On Web Authentication**」データベースメソッドを使用すると、Webサーバがリクエストを処理する前に、任意のリクエストの評価を行うことができます。さらに、デフォルトのHTMLルートフォルダを指定することにより、ディスク上のファイルへのアクセスを制限することができます。

最後に、Web経由で実行されるプロジェクトメソッドを個別に指定しなくてはなりません。

■ SSL接続

4D Web ServerはセキュアドモードでSSL(Secured Socket Layer)プロトコルを通じてブラウザに情報を送ることができます。このプロトコルは多くのブラウザと互換性があり、送信、受信について信頼性と変換された情報の完全性を実現しています。

■ インターネットフォーマットの拡張サポート

4D Web サーバは、XML 文書と WML（Wireless Markup Language）テクノロジーをサポートします。

■ CGI サポート

4D Web Server は他の CGI を通じた HTTP サーバにより呼び出されるのと同様に、CGI を非常に容易に呼び出すことができます。

■ データベースの同時処理

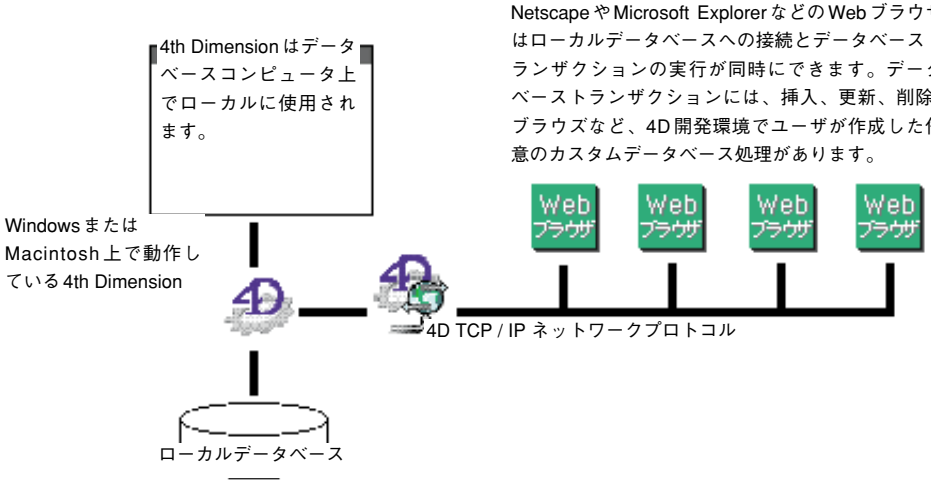
4th Dimension と Web

4th Dimension を使用して Web 上に 4D データベースを公開する場合、同時に以下のことができます。

■ 4D でローカルにデータベースを使用する

■ Web ブラウザを使用してデータベースに接続する

これを以下の図にまとめます。



4D Server と Web

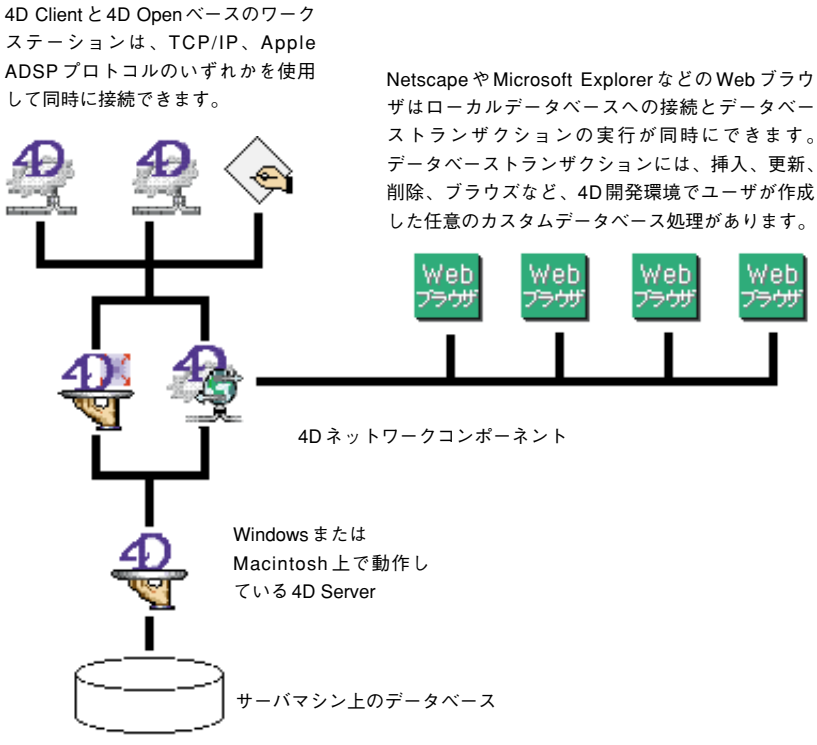
4D Server を使用して Web 上に 4D データベースを公開する場合には、以下を使用して 4D データベースへの接続とその処理を同時に行えます。

■ 4D Client ワークステーション

■ 4D Open をベースにしたアプリケーション

■ Web ブラウザ

これを以下の図にまとめます。



4D Client と Web

4D Client を使用して 4D データベースが Web 上に公開されると、次の方法で 4D データベースへ接続し、同時に使用することができます。

■ 4D Client マシン経由

■ 4D Open を使用したアプリケーション経由

■ Web ブラウザ経由。このケースでは、データが 4D Server からも公開されている場合に、Web ブラウザは 4D Client または 4D Server 経由で、公開されたデータベースに接続することができます。さらに、これにより異なるデータアクセスモードに対処できるようになります（パブリック、管理、等）。

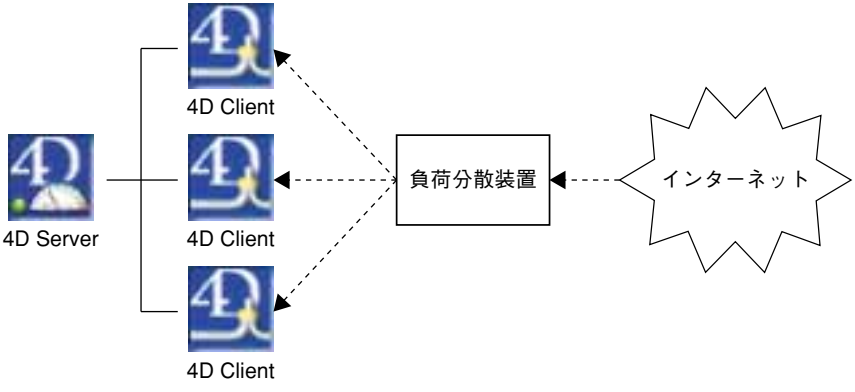
コンテキストモードの場合を除き、4D Clientでは4D Webサーバの基本的な仕組みが同じ方法で使用されます。ただし、4D ClientのWebサーバではコンテキストモードが使用できません（このモードに関する詳細は、「コンテキストモードの使用」の節を参照してください）。

同様に、ランゲージコマンドの動作は、4th Dimension、4D Server、4D Clientのいずれでコマンドが実行されても通常は同じです。重要なのは、コマンドはそれが実行されるマシンのWebサイトに対して適用されるという点です。したがって、**Execute on server**関数や**EXECUTE ON CLIENT**コマンドを使用して、これを管理しなくてはなりません。

■ 4D Clientを使用したロードバランシング（負荷分散）システム

任意の4D ClientマシンをWebサーバとして使用できるため、負荷分散装置（ロードバランサー）を備えたダイナミックなWebサーバシステムを構築することができます。このシステムにより大規模な開発が可能となり、次の事柄が実現します。

■ 4D Webサーバのパフォーマンスを最適化するためのロードバランシングシステムの構築：各4D Client Webサーバにインストールされる Webサイトミラーを使用し、現時点のそれぞれの負荷に基づいて負荷分散装置（ハードウェアまたはソフトウェア）が各クライアントマシンにリクエストを送信します。



■ 障害対策済のWebサービスの構築：4D Webサイトは、2台以上の4D Clientマシン上にミラー処理されます。ある4D Client Webサーバがダウンすると、もう一方が処理を引き継ぎます。

■ 例えば、リクエストの発信元に応じて、同一データからさまざまなビューを作成できます。社内ネットワークにおいて、プロテクトされた4D Client Webサーバがイントラネット上のリクエストを処理し、ファイヤウォール上にある別の4D Client Webサーバがインターネット上のリクエストを処理することができます。

■ 異なる4D Client Webサーバ間でのタスク分配：ある4D Client WebサーバがSOAPリクエストを処理し、別のサーバが標準的なリクエストを処理する、など。

参照

SEND HTML FILE、SET HTML ROOT、SET HTML ROOT、SET WEB DISPLAY LIMIT、SET WEB TIMEOUT、STOP WEB SERVER、Webサービス、接続セキュリティ、非コンテクストモード、Webサーバセッティング、CGIの使用、SSLプロトコルの使用

Webサーバー設定と接続管理

4th Dimension、4D Server、ならびに4D ClientにはWebサーバ機能が組み込まれ、透過的かつダイナミックにデータベースのデータをWeb上に公開することができます。

この節では、4Dデータベースの公開とブラウザへの接続、および接続管理プロセスに必要となる手順について説明します。

Web上への4Dデータベース公開条件

4th Dimension、4D Server、または4D Clientを使用してWeb上に4Dデータベースを公開するには、次に述べる項目が必要になります。

■ 必要とする4D Web接続ライセンス、4D Server Web接続ライセンス、または4D Client Web接続ライセンスがアプリケーションにインストールされていなくてはなりません。詳細については『4D Product Line インストールガイド』を参照してください。

■ Web接続は、TCP／IPプロトコルを使用してネットワーク上に構築されます。

したがって、

■ マシンにTCP／IPをインストールし、正しく設定しておく必要があります。詳細についてはコンピュータまたはオペレーティングシステムのマニュアルを参照してください。

■ SSLをネットワーク接続に使う場合には、要求されたコンポーネントが確実にインストールされているか確認してください（後述のWeb Service、SSLプロトコルの使用を参照）。

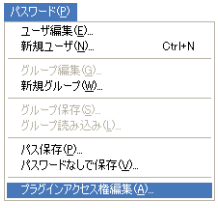
■ 上記の条件をすべてチェックし、対処した後は、4DでWebサービスを開始する必要があります。この点に関しては、この節でさらに詳しく説明します。

公開の認可（4D Client）

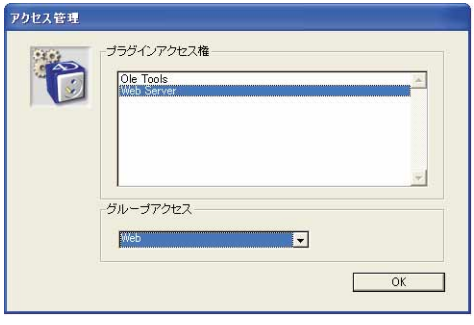
デフォルトとして、すべての4D Clientマシンは接続しているデータベースをWeb上に公開することができます。ただし、4Dのパスワードシステムを使用すると、各4D ClientのWeb公開を制御することができます。

実際のところ、4D Client の Web ライセンスは、4D Server からはプラグインランセンスとみなされます。したがって、プラグインと同じ方法で、Web サーバライセンスの使用権を特定グループのユーザに限定しなくてはなりません。

これを行うには、4D Client を使用してパスワードウインドウを表示します（これらのパラメータを変更するための適切なアクセス権が必要です）。パスワードウインドウが前面に表示されたら、「パスワード」メニューの「プラグインアクセス...」コマンドを選択します。



アクセスを管理するダイアログボックスにおいて、「Web サーバ」の選択後、「グループアクセス」ポップアップメニューを用いてユーザグループを関連付けます。



上図：“Web”グループに属するユーザだけが、それぞれの 4D Client マシンを Web サーバとして公開する権限を与えられます。

MacOS X における Web サーバの設定

MacOS X において、Web パブリッシング用に予約されている TCP／IP を使用するには、特定のアクセス権が必要となります。つまり、そのマシンの“ルート”ユーザだけが、これらのポートを使用してアプリケーションを起動することができます。

これらのポート番号は 0 から 1023 までです。デフォルトとして 4D データベースの公開には、標準モードでは TCP ポート 80、SSL モードではポート 443 が使用されます。

“ルート”ユーザとして接続せずに、デフォルトの TCP ポートを使用して 4D データベースを公開すると、警告ダイアログボックスが表示されます。

標準の HTTP 発行用のデフォルトポート番号は変更することができます。しかし、SSL での公開を行うには、ポート 443 を使用しなければなりません。

データベースの公開には 3 種類のオプションがあります。

■ 4D Web サーバで使用する TCP ポート番号を変更する。

番号の変更を行うには、「環境設定」ダイアログボックス（「Web サーバ設定」の節を参照）、または **SET DATABASE PARAMETER** コマンドを使用します。この場合、それぞれのデータベース接続用 URL に続けてポート番号を指定する必要があります（例えば、http://www.mydatabase.com/pages/mypage.html:8080）。

しかし、ポート番号を変更できるのは、標準の HTTP プロトコルで公開された 4D Web サーバ、言い換えれば、SSL プロトコルを使用しないサーバだけである点に留意してください。SSL を利用して 4D Server を公開するには、ポート 443 を使用する必要があります。また、暗号化モードで 4D Web サーバを公開するためには、“ルート”ユーザとして接続しなければなりません。

■“ルート”ユーザとして接続する。

デフォルトとして、MacOS X が動作するマシンでは“ルート”ユーザが有効ではありません。まず“ルート”ユーザを有効にしたあと、そのユーザ名を使用してログインしなければなりません。“ルート”ユーザを有効にするには、Apple 社より提供され、「Applications:Utilities」フォルダにインストールされている NetInfo Manager ユーティリティを使用します。

ユーティリティの起動後、「ドメイン」メニューから「セキュリティ」コマンドを選択し、さらに「ルートユーザを有効」オプションを選択します。まず最初に同じメニューにある「認証...」コマンドを使い、マシン管理者を指定しなければなりません（短い名前と管理者のパスワードを入力する）。

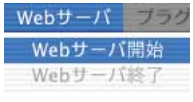
この操作に関する詳細は、MacOS Xのドキュメントを参照してください。

“ルート”ユーザを作成したら、このセッションをクローズし（Appleメニュー）、“ルート”ユーザ名を使用してログインします。これで、ポート番号80でWebサーバを起動したり、あるいは暗号化接続を使用して4D Webサーバを起動することができます。

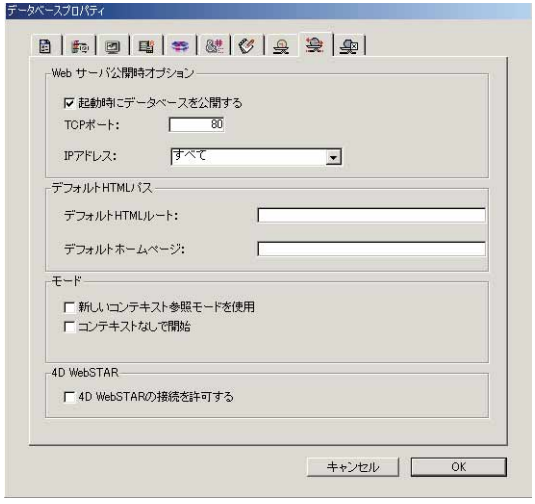
4D Webサーバの開始

4D Webサーバは、次に示す3つの方法で開始することができます。

■ 4D Serverのメインメニューバー、または4th Dimensionや4D Clientの「ユーザ」モードから「Webサーバ」メニューを使用します。「Webサーバ」メニューにより、いつでもWebサービスの開始や終了を行えます。



■ データベースが開かれるたびにデータベースを自動的に公開します。データベースをWeb上へ自動的に公開するには、4D Serverのメインメニューバー、または4th Dimensionや4D Clientの「デザイン」モードで「編集」メニューから「環境設定...」を選択します。すると、「環境設定」ウインドウが表示されます。ここで、「Web」テーマの「公開」ページをクリックしてください。



「Webサーバ公開時オプション」エリアで、「起動時にデータベースを公開する」チェックボックスを選択してから「OK」ボタンをクリックします。これを一度行えば、ユーザが4th Dimensionや4D Server、または4D Clientでデータベースを開くたびに、Web上へ自動的に公開されます。

■ プログラムで**START WEB SERVER**コマンドを呼び出す方法。

Tips：Web上へのデータベース公開を開始または終了する際に、4Dを停止させて、データベースを再オープンする必要はありません。Webサーバの終了や再開は、「Webサーバ」メニューを使用するか、またはSTART WEB SERVERコマンドとSTOP WEB SERVERコマンドを呼び出すことにより、必要なだけ何回でも行えます。

Web上に公開された4Dデータベースへの接続

Webへの4Dデータベースの公開を開始したら、ユーザはWebブラウザを使用して、そのデータベースに接続することができます。これを実行するには以下のように行います。

■ Webサイトに登録名（例："www.flowersforever.com"）がある場合には、その名前をブラウザのOpen、アドレス、場所エリアのいずれかに指定します。その後、enterキーを押して接続します。

■ Webサイトに登録名がない場合には、マシンのIPアドレス（例：192.168.0.99）をブラウザのOpen、アドレス、場所エリアのいずれかに指定します。その後、enterキーを押して接続します。

現時点で、お使いのブラウザにはWebサイトのホームページが表示されているはずです。標準設定のままでデータベースを公開した場合には、4th DimensionのWebサーバのデフォルトホームページが現われます。このページを使用して、接続やサーバ操作のテストを行うことができます。

また、次に示すいずれかの状況に直面する可能性があります。

1. 接続に失敗し、"...the server may not be accepting connections or may be busy..."（サーバは接続を受け付けないか、またはビジー状態です）といったメッセージを受け取った場合。

この場合、以下を確認します。

■ 入力した名前またはIPアドレスが正確なことを確認します。

■ 4th Dimensionや4D Server、または4D Clientが稼働しており、そのWebサーバが開始していることを確認します。

■ デフォルトのWeb TCPポート以外のTCPポートでサービスを受けるようにデータベースが設定されているかどうかを確認します（下記の4を参照）。

■ サーバマシンとブラウザマシンの両方でTCP/IPが正しく設定されているかどうかを確認します。両方のマシンは同一のネットとサブネット上にあるか、あるいはルータが正しく設定されている必要があります。

■ ハードウェアが正しく接続されていることを確認します。

■ サイトのテストをローカルでしているのではなく、インターネットやイントラネット上で他者がサービスを提供しているWebデータベースに接続しようとしているのであれば、表示されたメッセージが正しいこともあります。つまり、サーバがオフになっているかビジー状態であるということです。この場合には、ログオンできるまで少し待って再試行するか、Webデータベースの公開先に連絡してください。

2. 接続に成功したが、HTTPエラーの404“File not found（ファイルが見つかりません）”というメッセージが表示される場合。これは、そのサイトのホームページを提供できないことを表わしています。この場合、データベースの「環境設定」（「Webサーバ設定」の節を参照）やSET HOME PAGEコマンドを用いて指定した場所に、ホームページが実際に存在しているかどうかを確認してください。

3. 接続に成功したが、“メニューバー/このデータベースはWeb上に公開する準備が整っていません。まず始めにメニューバーを作成してください。”というメッセージを表示するページが現われた場合。

この場合は、コンテキストモードで公開されたデータベースに正しく接続したものの、ホームページとメニューバーが定義されていないことを表わします（コンテキストモードでは、HTMLページが指定されていない場合に4Dはメニューバー番号1をデフォルトのホームページとして公開します)。詳細については、「Webサーバ入門」の節を参照してください。

4. 接続はしたが、予想していたようなWebページを取得できない場合。

これは、1つのマシン上で同時に複数のWebサーバが稼働している場合に発生する可能性があります。以下の例を参照してください。

■ 独自のWebサーバが既に稼働しているWindowsシステム上で、4DのWebデータベースが1つだけ動作している場合。

■ 1つのマシン上で複数の4D Webデータベースを稼働している場合。

このような状況の元では、4D Webデータベースが発行されるTCPポート番号を変更する必要があります。これを実行するには、「Webサーバ設定」の節を参照してください。

注：データベースがパスワードシステムで守られていれば、正当なユーザ名とパスワードを入力する必要があります（詳しい説明は「接続セキュリティ」を参照してください)。

Webプロセスの管理

データベースのWeb公開やブラウザへの接続は、各種4Dプロセスがサポートします。この節では、これらのプロセスならびにその特性について説明します。

「Webサーバ」プロセス

Webサーバプロセスは、データベースがWebサイトとして公開されている時に稼働し、実行します。次に示すランタイムエクスプローラの「プロセスページ」では、「Webサーバ」プロセスは稼働中かつ実行中である3番目のプロセスです。

これは、4Dカーネルプロセスです。したがって、アボートボタンを使用してこのプロセスをアボートすることはできません。また、**CALL PROCESS**等のコマンドを使用してプロセス間通信を実行することもできません。「Webサーバ」プロセスはユーザインタフェースコンポーネント（ウインドウ、メニュー等）を持たないことに注意してください。

「Webサーバ」プロセスは、以下の方法で起動できます。

- 4D Serverや4th Dimensionまたは4D Clientの「Webサーバ」メニューより、「Webサーバ開始」を選択する（「ユーザ」モード）。
- 4Dコマンドの**START WEB SERVER**コマンドを呼び出す。
- 「環境設定」の「起動時にデータベースを公開する」オプションが選択されているデータベースをオープンする。

「Webサーバ」プロセスは、以下の方法で停止できます。

- 4D Serverや4th Dimensionまたは4D Clientの「Webサーバ」メニューより、「Webサーバ終了」を選択する（「ユーザ」モード）。
- 4Dコマンドの**STOP WEB SERVER**コマンドを呼び出す。
- 現在公開されているデータベースを停止する。

「Webサーバ」プロセスの目的は、Web接続を処理するだけです。「Webサーバ」プロセスを起動することによって、実際にWeb接続がオープンされるのではなく、Webユーザに対してWeb接続の開始を許可するだけです。また、「Webサーバ」プロセスの停止によって、現在稼働中の「Web接続」プロセスがあればそれをクローズするというのではなく、Webユーザに対して新しいWeb接続の開始を許可しなくなるということです。

「Webサーバ」プロセスを停止する際に、実行中の「Web接続」プロセスが存在する場合、これらのプロセスは通常通り実行を継続します。

結果として、Webサーバプロセスの終了は時間の延長を必要とします。

「Web 接続」プロセス

Webブラウザがデータベースへ接続しようとするたびに、接続リクエストは「Webサーバ」プロセスによって処理されます。「Webサーバ」プロセスは、以下の手順を実行します。

■ まず初めに、「Webプロセス」と呼ばれる一時的なローカル4Dプロセスをいくつか作成し、Webブラウザとの接続を評価し、管理します。

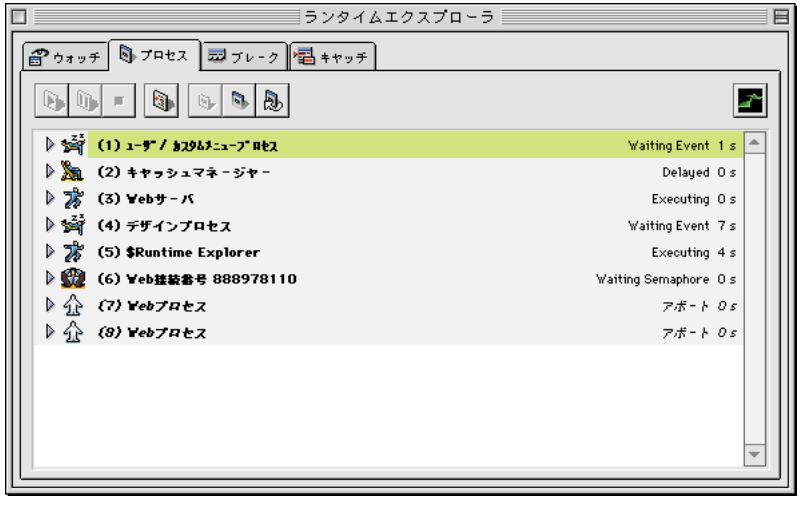
注：これらの一時的なプロセスは各HTTPリクエストを管理します。即座に実行した後アボートされるか遅延されます。非コンテキストモードに関連するWebサーバでは、4Dは5秒間Webプロセスのプールを凍結し、新たなHTTPクエリを実行する時に再利用します。SET DATABASE PARAMETERコマンドを使ってこの動作をカスタマイズできます。

■ リクエストにはコンテキストの作成が不要である場合、Webプロセスがリクエストの処理を行い、応答（必要な場合）をブラウザへ送信します。この後で、一時的なプロセスはアボートされるか、または遅延されます（上記を参照）。

■ リクエストにはコンテキストの作成が必要である場合、新規接続で利用できるリソースか存在するかどうかを確認します。そうでない場合には、Webブラウザに以下のメッセージを送信します。“This database has not been setup for the Web yet”（このデータベースはWeb用に設定されていません）。

Web接続が正常に開始された場合には、「Web接続」プロセスが起動します。このプロセスが、この接続の全Webセッションを扱います。

「プロセスリスト」ウインドウは、Webブラウザ接続が開始された後に起動された「Web接続」プロセス“Web接続番号152142900”を表示しています。



開始後にアボートされた6番目のプロセスがWeb接続の初期化を行っている点に注意してください。

注：コンテキストの管理についての詳しい説明は、次節の「コンテキストモード」を参照してください。

■ セッション中に、コンテキストモードから非コンテキストモードに接続をスイッチすると、（そのIDを持つ）Web接続プロセスはアボートされます。

逆に言えば、セッション中に、接続を非コンテキストモードからコンテキストモードへスイッチすると、番号付きのWeb接続プロセスが作成されます。

接続セキュリティ

4D Webサーバの接続セキュリティは、下記の要素に基づいています。

■ Webパスワード管理システムと「**On Web Authentication**」データベースメソッドとの組み合わせ。

■ "一般Webユーザ"の定義

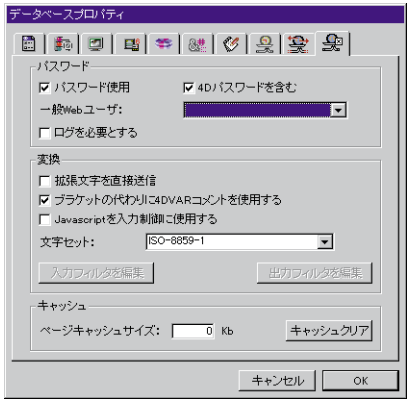
■ デフォルトのHTML、ルートフォルダの設定

■ データベースの各プロジェクトメソッドに対する「4DACTIONで利用可能」プロパティの設定。

注意：接続の安全性それ自体はSSLプロトコルを通して管理することができます。詳細は「SSLプロトコルの利用」を参照してください。

Webアクセス用のパスワードマネージメントシステム

「環境設定」ダイアログボックスにおいて、Webサーバに適用するアクセス管理システムを指定することができます。これを行うには、「環境設定」ダイアログボックスで、「Web」テーマの「公開」ページを選択します。



パスワードエリアでは､"パスワード使用"と"4Dパスワードを含む"の2つのオプションが選択できます。"4Dパスワードを含む"のチェックボックスは"パスワード使用"が選択された場合にのみ指定できます。

■ **パスワード使用：**Webサーバのパスワードシステムを起動させます。接続時にブラウザ上にダイアログボックスが表示され、名前とパスワードが入力できます。名前とパスワードならびに接続パラメータ（IPアドレスおよびポート、URL、...）は、**On Web Authentication**データベースメソッドに渡され、独自のパスワードシステムの構築等必要な処理ができるようになります。

注：この場合、On Web Authenticationデータベースメソッドが存在しなければ接続は拒絶されます。

■ **4Dパスワードを含む：**独自のパスワードシステムの代り、またはそれに付加するものとして、4Dで定義されているデータベースパスワードシステムを使用することができます。

注：

・4D Client Webサーバの場合、4D Clientマシンにより公開されたすべてのサイトは、同じユーザテーブルを共有するということを覚えておいてください。ユーザやパスワードの検証は、4D Serverアプリケーションにより実行されます。

・HTTPリクエストでは､ユーザが入力したパスワードは暗号化されません(基本モード)。

4DWebサーバのアクセスシステムのオーバービュー

4D Webサーバへの接続を判断するシステムは、2つのパラメータの組合せによります。

■「環境設定」ダイアログボックスの「Webパスワード」オプション

■ **On Web Authentication**データベースメソッドの存在

以下の場合があります。

オプションが何も指定されてない場合

注：デフォルトとして、新規データベースにはこれらのパラメータが設定されます。

■ **On Web Authentication**データベースメソッドが存在している場合、$1と$2の他は、$3と$4にブラウザとサーバのIPアドレスが渡され、$5と$6のユーザ名とパスワードは空白のままデータベースメソッドが実行されます。この場合、ブラウザのIPアドレスまたはサーバのIPアドレスを使用して接続を判断することができます。

■ **On Web Authentication**データベースメソッドが存在しない場合、接続は自動的に受け入れられます。

"パスワード使用"オプションが指定されて、"4Dパスワードを含む"オプションは指定されていない場合

■ **On Web Authentication**データベースメソッドが存在している場合、すべての引数が渡されます。したがって、ユーザ名、パスワード、ブラウザまたはサーバのIPアドレスに応じて、接続をより細かく判断することができます。

■ **On Web Authentication**データベースメソッドが存在しない場合、接続は自動的に拒否されて、認承メソッドが存在しないことを示すメッセージがブラウザに送られます。

注：ブラウザによって送られたユーザ名が空白で、On Web Authenticationデータベースメソッドが存在しない場合、「パスワード」ダイアログがブラウザに送られます。

"パスワード使用"および"4Dパスワードを含む"オプションが指定されている場合

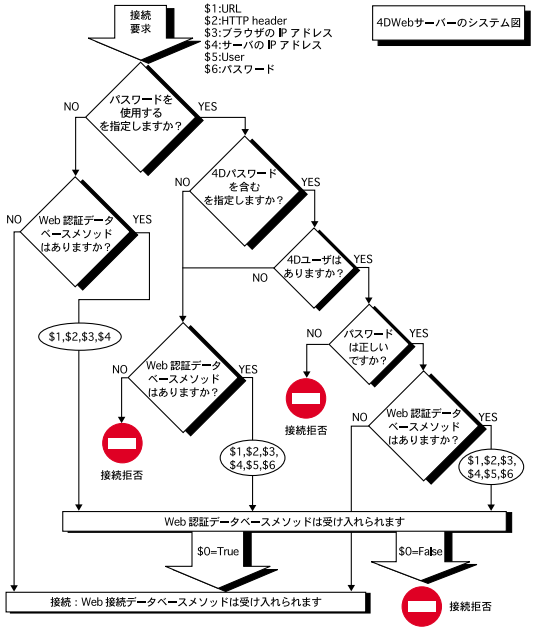
■ ブラウザによって送られたユーザ名が4Dのユーザとして登録されており、パスワードが正しければ、接続は受け入れられます。パスワードが誤っていれば接続は拒否されます。

■ ブラウザによって送られたユーザ名が4Dのユーザとして登録されてない場合、2つの可能性があります。

On Web Authenticationデータベースメソッドが存在している場合、引数$1、$2、$3、$4、$5、$6が渡されるので、ユーザ名、パスワード、ブラウザまたはWebサーバのIPアドレスを使用して接続を判断することができます。

On Web Authenticationデータベースメソッドが存在しない場合、接続は拒否されます。

4D Webサーバのアクセスシステムは、以下の図のようにまとめられます。



ロボットに関するセキュリティ注意事項

特定のロボット（query engines, spiders...）は、Webサーバやスタティックホームページを閲覧していきます。ロボットに、すべてのサイトをアクセスできるようにさせたい場合、どのURLへのアクセスを許さないのかを定義できます。

これを実行するには、ROBOTS.TXTファイルをサーバのルートに置きます。このファイルは下記の書式で構成されていなければなりません。

▼例：

```
User-Agent:*
Disallow:/4D
Disallow:/%23%23
Disallow:/GIFS/
```

"User-Agent:*"は、すべてのロボットに影響することを意味します。

"Disallow:/4D"は、/4Dで始まるURLにロボットがアクセスできないことを意味します。

"Disallow:/%23%23"は、/%23%23で始まるURLにロボットがアクセスできないことを意味します。

"Disallow:/GIFS/"は、/GIFS/フォルダまたはそのサブフォルダにロボットがアクセスできないことを意味します。

▼ 他の例として、

```
User-Agent:*
Disallow:/
```

この場合、ロボットはサイト全体へのアクセスが許されません。

一般Webユーザ

4Dパスワードテーブルに定義されたユーザを「一般Webユーザ」として指定することができます。この場合、データベースに接続するWebユーザは、この一般ユーザに設定されたアクセス権と制限が適応されます。したがって、データベースの異なる部分への閲覧を簡単に管理できます。

注：Webユーザがデータベースの異なる部分（テーブルやメニュー等）へのアクセス制限を受けるこのオプションを、パスワードシステムで管理されるWebサーバの接続管理システムと混同しないようにしてください。

▼ 一般Webユーザを定義するには、

1.「デザイン」モードにおいて、パスワードエディタで少なくとも1つのユーザを作成する。

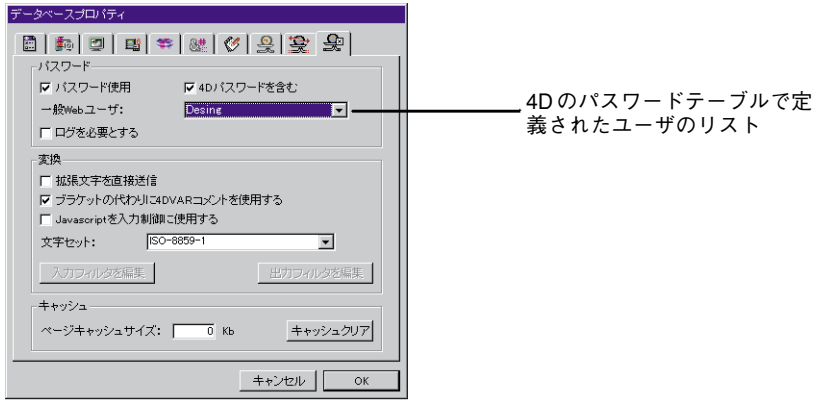
必要であれば、ユーザにパスワードを設定できます。

2. 各エディタで、このユーザのアクセス（可能または禁止）を設定する。

3.「環境設定」ダイアログボックスで、「Web」テーマの「公開」ページを選択する。

「Webパスワード」エリアには、「一般Webユーザ」のドロップダウンリストがあります。デフォルトでは、一般ユーザはデザイナであり、Webユーザはデータベース全体へのアクセス権を持ちます。

4. ドロップダウンリストからユーザを選択し、ダイアログボックスを有効にする。



データベースへの接続を許されたすべてのWebユーザは、この一般Webユーザに関連したアクセス権およびアクセスの制限が適応されます（"4Dパスワードを含む"オプションが選択され、接続するユーザが4Dパスワードテーブル内に存在しない場合を除きます。下記を参照してください）。

Webパスワードシステムとの相互作用

パスワード使用オプションは、一般のWebユーザがどのように操作するかには影響を与えません。このオプションの状態がどうであれ「一般Webユーザ」に関連したアクセス権と制限は、データベースに接続することを許されたすべてのWebユーザに適用されます。

しかし、4Dパスワードを含むオプションが選択されると、下記の2つの可能性があります。

- ユーザの名前とパスワードが4Dのユーザとして登録されていない場合、**On Web Authentication**データベースメソッドによって接続が受け入れられると、一般Webユーザのアクセス権が適用されます。
- ユーザの名前とパスワードが4Dのユーザとして登録されている場合には、「一般Webユーザ」のパラメータは無視され、ユーザ自身のアクセス権で接続します。

デフォルトHTMLルートフォルダを定義する

「環境設定」のこのオプションを使用し、4Dがブラウザに送信するスタティックHTMLページやセミダイナミックHTMLページ、ピクチャ等が含まれるフォルダを指定することができます。

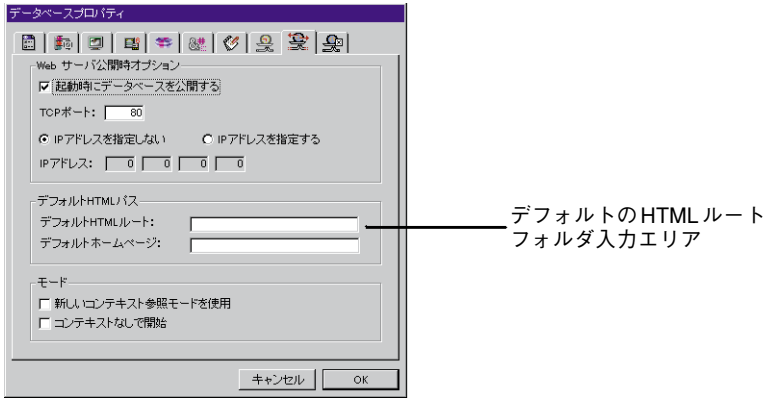
さらに、Webサーバドライブ上のHTMLルートフォルダより上位の階層へのアクセスができないように定義したことになります。

このアクセス制限は、Webブラウザに送られるURLならびに"**SEND HTML FILE**"のような4DのWebサーバコマンドに適用されます。ブラウザによって送られたURLまたは4Dコマンドが、HTMLルートフォルダの上位にあるファイルにアクセスしようとした場合に、ファイルが発見できなかったことを示すエラーが返されます。

デフォルトとして、4Dは「WebFolder」という名前のHTMLルートフォルダを定義します。このフォルダが存在しない場合は、Webサーバを初めて開始した際、ディスク上にHTMLルートフォルダが物理的に作成されます。デフォルトの場所をそのまま使用した場合、ルートフォルダは次の場所に作成されます。

- 4th Dimensionおよび4D Serverでは、データベースのストラクチャファイルと同じ階層内。
- 4D Clientでは、「4D Client.exe」ファイル（Windows）またはソフトウェアパッケージ（MacOS）と同じ階層内。

デフォルトのHTMLルートフォルダの名前や場所は、「環境設定」ダイアログボックスで変更することができます（「Web」テーマの「公開」ページ）。



"デフォルトHTMLルート"入力エリアに、定義したいフォルダの新しいアクセスパスを入力します。

このダイアログボックスに入力するアクセスパスは相対的です。つまり、データベース（4th Dimensionまたは4D Server）のストラクチャ、または4D Clientアプリケーションやソフトウェアパッケージ（4D Client）が含まれるフォルダを起点に指定します。

4D Webサーバはデータベースのマルチプラットフォームの互換性を保つため、アクセスパスを記述するための特別な書き込み規約を使用します。構文のルールは下記の通りです。

- フォルダはスラッシュ（"/"）で分離する
- アクセスパスはスラッシュ（"/"）で終了してはいけない

■ フォルダ階層の中で1レベル「上がる」には、".."（ピリオドを2つ）を、フォルダ名の前に入力する

■ アクセスパスはスラッシュ（"/"）で始めてはならない（HTMLルートフォルダをデータベースや4D Clientのフォルダに設定したい場合を除きます。下記を参照してください）。

HTMLルートフォルダをデータベースや4D Clientのフォルダに設定したいが、上位のフォルダへのアクセスを禁止したい場合は、"4Ddatabase/Web"を入力します。HTMLルートフォルダをデータベースフォルダにしたいが、上のフォルダへのアクセスを禁止したい場合は、"/"をエリア内に入力します。全ボリュームへアクセスを可能にするには、デフォルトHTMLルート"エリアを空にします。

警告：「環境設定」ダイアログボックスにおいて、デフォルトHTMLルートフォルダを指定しない場合、データベースのストラクチャファイルまたは4D Clientアプリケーションが含まれるフォルダが使用されます。

注：「環境設定」ダイアログボックスにおいて、HTMLルートフォルダが変更された場合、アクセスが制限されているファイルが保存されないようにキャッシュがクリアされます。

データベースの環境設定とSET HTML ROOT（コンテキストモード）

いったんHTMLルートフォルダが「データベースプロパティ」ダイアログで指定されると、**SET HTML ROOT**コマンド（コンテキストモードのみ）を使って変更することができますが、変更は現在のWebプロセスのみに適用され、HTMLページのキャッシュはクリアされます。

しかし、**SET HTML ROOT**コマンドは、「環境設定」で定義されたデフォルトHTMLルートフォルダを考慮します。「環境設定」ダイアログボックスで定義されたフォルダが“WebPages/”であり、SET HTML ROOT（"Folder"）をという命令を渡した場合、デフォルトのHTMLルートフォルダは“WebPages/Folder/”になります。また、この場合でも、接続制限は“WebPages”フォルダの上位階層のフォルダに対して適用されます。

注：Webサーバが非コンテキストモードの時は、SET HTML ROOTコマンドは何も行ないません。

4DACTIONで利用可能

4DACTION（非コンテキストモード）ならびに4DMETHOD（コンテキストモード）という特別なURLを使用し、Web上に公開された4Dデータベースのプロジェクトメソッドの実行を開始することができます。例えば、http://www.server.com/4DACTION/Erase_Allというリクエストの場合、Erase_Allプロジェクトメソッドが存在すれば、それが実行されます。

この結果、この仕組みにはデータベースに関するセキュリティ上のリスクが存在します。特に、インターネットユーザが故意に（または無意識に）Webを介して意図せずにメソッドの実行を起動してしまう危険性があります。

次の3つの方法により、この危険を回避することができます。

■ 4Dパスワードシステムを用いて、プロジェクトメソッドへのアクセスを制限する。

欠点：このシステムでは4Dのパスワードを使用する必要があり、あらゆるタイプのメソッドの実行を禁止します（HTMLタグの使用も含む）。

■「On Web Authentication」データベースメソッドを用いて、URL経由でのメソッド呼び出しをフィルタリングする。

欠点：データベースに多数のメソッドが存在する場合、このシステムの運用が難しい可能性があります。

■「メソッドプロパティ」ダイアログの「4DACTIONで利用可能」オプションを使用する。

このオプションを使用して、4DACTIONならびに4DMETHOD（コンテキストモード）という特別なURLを用いて呼び出されるプロジェクトメソッドを個別に指定します。このオプションが選択されていない場合、特別なURLを含むHTTPリクエストを使用して、そのプロジェクトメソッドを実行することができません。その代わりに、他のタイプの呼び出し（4Dタグ、他のメソッド等）を用いて実行されます。

4th Dimension 2003で作成されたデータベースに対し、デフォルトではこのオプションが選択されていません。4DACTIONや 4DMETHOD Web URLを使用して実行されるメソッドには、このオプションを必ず指定しなくてはなりません。

これとは逆に、互換性上の理由から、既存のデータベース（2003より以前のバージョンの4Dで作成）に対しては、このオプションが選択されています。デフォルトでは、特別な4DのURLを用いて、すべてのプロジェクトメソッドにアクセスすることができます。

4DACTIONを用いて利用できるプロジェクトメソッドには、特定のアイコンが使用されます。

参照

On Web Authentication データベースメソッド、On Web Connection データベースメソッド、SSLプロトコルの使用

On Web Authentication データベースメソッド

「**On Web Authentication**」データベースメソッドは、Webサーバエンジンへのアクセスを管理します。Webブラウザのリクエストにおいてサーバ上での4Dメソッドの実行が必要となる場合に（4DACTION、4DCGI URL、または4DSCRIPTタグ等を用いて呼び出されたメソッド）、4th Dimension、4D Server、または4D Clientによりこのデータベースメソッドが呼び出されます。

このメソッドには6つの引数$1、$2、$3、$4、$5、$6が渡されます。またひとつのブール値が返されます。これらの引数の説明は次の通りです。

パラメータ	タイプ	内容
$1	テキスト	URL
$2	テキスト	HTTPヘッダ + HTTPボディ(上限32kb)
$3	テキスト	Webクライアント（ブラウザ）のIPアドレス
$4	テキスト	サーバのIPアドレス
$5	テキスト	ユーザ名
$6	テキスト	パスワード
$0	ブール	True= リクエストが受け入れられた False= リクエストが拒否された

これらの引数は以下のように定義しなければなりません。

```
` On Web Authentication データベースメソッド
C_TEXT ($1;$2;$3;$4;$5;$6)
C_BOOLEAN($0)
` メソッドのコード
```

注：On Web Authenticationデータベースメソッドの全引数が有効とは限りません。データベースメソッドが受け取る情報は、「環境設定」ダイアログボックスで事前に選択したオプションよって異なります（「接続セキュリティ」の節を参照してください）。

URL

最初の引数（$1）は、ブラウザのロケーションエリアにユーザが入力したURLであり、ホストアドレスが削除されたものです。

イントラネット接続を例に説明しましょう。4D WebサーバマシンのIPアドレスが「192.168.0.99」であると仮定します。以下の表は、Webブラウザに入力されたURLに従った$1の値を示しています。

Webブラウザロケーションエリアに入力されたURL	引数$1の値
192.168.0.99	/
http://192.168.0.99	/
192.168.0.99/Customers	/Customers
http://192.168.0.99/Customers	/Customers
http://192.168.0.99/Customers/Add	/Customers/Add
192.168.0.99/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

特別なケース：非コンテキストモードにおいて、要求されたURLが既存のページか4Dの特別なURLのいずれでもない場合、引数 $1の値は“/”で始まりません。例えば、URLが“http://192.168.0.99/Clients/Add”であり、“Add”ページが「Clients」フォルダに存在しない場合、 “Clients/Add”を$1の値として「On Web Authentication」データベースメソッドが呼び出され、次に「On Web Connection」データベースメソッドが呼び出されます。

HTTP リクエストのヘッダとボディ

2番目の引数（$2）は、Webブラウザから送信されたHTTPリクエストのヘッダとボディです。このヘッダが**On Web Authentication**データベースメソッドにそのまま渡されていることに注意してください。そのコンテンツは接続を試行するWebブラウザの特徴によって変わります。

アプリケーションの中でこのヘッダとボディ情報を使用する場合、それを解析するのはプログラマの仕事です。

注：パラメータの詳しい説明はOn Web Connectionを参照してください。

Webクライアントの IP アドレス

$3には、ブラウザが動作しているマシンのIPアドレスが渡されます。この情報は、イントラネット接続とインターネット接続を区別できるようにするものです。

Webサーバの IP アドレス

$4には、4D Webサーバの呼び出しに利用するIPアドレスが渡されます。バージョン6.5は、複数のIPアドレスを持つマシンでマルチホーミングを可能にしています。より詳しい情報は、「Webサービス：Webサーバセッティング」を参照してください。

ユーザ名とパスワード

$5および$6には、ユーザによって入力されたユーザ名とパスワードが渡されます。「パスワード入力」ダイアログは、「環境設定」ダイアログボックスで「パスワード使用」オプションが選択された場合、各接続時に表示されます（より詳しい情報は、「Webサービス：接続セキュリティ」を参照してください）。

注：ユーザ名が4Dに存在するブラウザから送られた場合、$6パラメータ（ユーザのパスワード）はセキュリティの理由で返されません。

■ 引数 $0

On Web Authentication データベースメソッドはブールタイプの戻り値を返します。

■ 接続を受け入れる場合は、$0にTrueを設定します。

■ 接続を拒否する場合は、$0にFalseを設定します。

On Web Connection データベースメソッドは、**On Web Authentication** によって接続が受け入れられた場合にのみ実行されます。

警告：$0に値が入っていない、または$0がOn Web Authenticationデータベースメソッドで定義されていない場合、接続は接続は受け付けられたとみなされ、On Web Connectionデータベースメソッドが実行されます。

注：

・On Web Authenticationデータベースメソッド内では、いかなるインタフェースエレメント（ALERTやDIALOG等）も呼び出してはなりません。呼び出した場合、操作は中断され、接続は拒絶されます。データベースメソッドが実行されている間にエラーが発生した場合も同様です。

・「メソッドプロパティ」ダイアログボックスの「4DACTIONで利用可能」オプションを使用すると、各プロジェクトメソッドに対して4DACTIONや 4DMETHODによる実行を禁止することができます。この件に関する詳細は、「接続セキュリティ」の節を参照してください。

「On Web Authentication」データベースメソッドの呼び出し

リクエストや処理において4Dメソッドの実行が必要になると、モードに関わらず「On Web Authentication」データベースメソッドが自動的に呼び出されます。また、Webサーバが無効なスタティックURL（例えば、リクエストされたスタティックページが存在しない場合）を受信した場合にも呼びだされます。

したがって「On Web Authentication」データベースメソッドは次の状況で呼び出されます。

■ 4Dが“4DACTION/”で始まるURLを受信した場合

■ 4Dが“4DMETHOD/”で始まるURLを受信した場合

■ 4Dが“4DCGI/”で始まるURLを受信した場合

■ 4Dが存在しないスタティックページを要求するURLを受信した場合

■ 4Dがセミダイナミックページの4DSCRIPTタグを処理する場合

■ 4Dがセミダイナミックページのメソッドに基づいた4DLOOPタグを処理する場合

注：有効なスタティックページを要求するURLをサーバが受信した場合、「On Web Authentication」データベースメソッドは呼び出されません。

例題

▼ **On Web Authentication**データベースメソッドの代表的な例を示します。

```
a`Web認証データベースメソッド
C_TEXT($5;$6;$3;$4)
C_TEXT($user;$password;$BrowserIP;$ServerIP)
C_BOOLEAN($4Duser)
ARRAY TEXT($users;0)
ARRAY TEXT($nums;0)
C_LONGINT($upos)
C_BOOLEAN($0)
        $0:=False
        $user:=$5
        $password:=$6
        $BrowserIP:=$3
        $ServerIP:=$4
`セキュリティ上で,@（ワイルドカード）が含まれる名前を拒否します
If (WithWildcard($user) | WithWildcard($password))
        $0:=False
                `ワイルドカードメソッドを以下に記述します
Else
                `4D userかどうかチェックする
        GET USER LIST($users;$nums)
        $upos:=Find in array($users;$user)
        If ($upos > 0)
                $4Duser:=Not(Is user deleted($nums{$upos}))
        Else
                $4Duser:=False
        End if
        If (Not($4Duser))
```

```
                `4Dが定義したユーザではありません。Webユーザのテーブルを見ます。
                QUERY([WebUsers];[WebUsers]User=$user;*)
                QUERY([WebUsers]; & [WebUsers]Password=$password)
                $0:=(Records in selection([WebUsers]) = 1)
            Else
                $0:=True
            End if
        End if
            `これはイントラネット接続ですか？
        If (Substring($BrowserIP;1;7) # "192.100.")
            $0:=False
        End if
```

▼ WithWildcard メソッド：

```
        C_INTEGER ($i)
        C_BOOLEAN ($0)
        C_TEXT ($1)
        $0:=False
        For ($i;1;Length($1))
            If (Ascii (Substring ($1;$i;1)) = Ascii ("@"))
                $0:=True
            End if
        End for
```

参照

On Web Connection データベースメソッド、Web サービス：接続セキュリティ、Web サービス：特別な URL とフォームアクション

「On Web Connection」データベースメソッド

「On Web Connection」データベースメソッドは、次の 3 つの状況で呼びだされます。

■ Web サーバが「4DCGI」URL で始まるリクエストを受信した場合。

■ Web サーバが無効なリクエストを受信した場合。

■ また、Web ブラウザがコンテキストモードでデータベースへの接続を開始するか、あるいは Web サーバがコンテキストの作成を必要とするリクエストを受信するたびに、4th Dimension や 4D Server から呼び出されます（このケースは、4D Client では対処されません。4D Client はコンテキストモードをサポートしません。）。

詳細については、「On Web Connectionデータベースメソッドの呼び出し」の節を参照してください。

リクエストは「On Web Authentication」データベースメソッド（存在する場合）によって事前に許可されていなければなりません。また、データベースはWebサーバとして公開されていなければなりません。

On Web Connectionデータベースメソッドは、4Dから渡される6つのテキスト引数を受け取ります。

これら引数の内容は以下の通りです。

パラメータ	タイプ	内容
$1	テキスト	URL
$2	テキスト	HTTPヘッダ＋HTTPボディ（上限32kb）
$3	テキスト	Webクライアント（ブラウザ）のIPアドレス
$4	テキスト	サーバのIPアドレス
$5	テキスト	ユーザ名
$6	テキスト	パスワード

これらの2つの引数は、以下のように宣言できます。

```
`「On Web Connection」データベースメソッド
C_TEXT ($1;$2;$3;$4;$5;$6)
` メソッドのコード
```

URLエクストラデータ

最初の引数（$1）は、ブラウザのロケーションエリアでユーザが入力したURLであり、ホストアドレスが削除されたものです。

イントラネット接続を例に説明しましょう。4D WebサーバマシンのIPアドレスが「192.168.0.99」であると仮定します。以下の表は、Webブラウザに入力されたURLに従った$1の値を示しています。

Webブラウザロケーションエリアに入力されたURL	引数$1の値
192.168.0.99	/
http://192.168.0.99	/
192.168.0.99/Customers	/Customers
http://192.168.0.99/Customers	/Customers
http://192.168.0.99/Customers/Add	/Customers/Add
192.168.0.99/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

この引数は、必要に応じて自由に使用してください。4Dは URL のホスト部分以外に渡される値は単に無視します。

例えば、値"/Customers/Add"で「[顧客]テーブルに新しいレコードを追加するために直接移動せよ」という意味の表記を設定することができます。データベースの Web ユーザに対して利用可能な値のリストやデフォルトのブックマークを提示することで、アプリケーションのさまざまな部分へのショートカットを提供できます。この方法により、Web ユーザは、データベースに新しく接続するたびにナビゲーションパス全体を通らずに、即座に Web サイトのリソースにアクセスできます。

特別なケース：非コンテキストモードでは、要求された URL が存在するページか 4D の特別な URL（"/4DCGI"のような）に一致しない場合、$1 の値は"/"で始めることはできません。たとえば、要求された URL が"http://192.168.0.99/Clients/Add"で、"Add"ページが"Clients"フォルダに存在しないとき、"Clients/Add"を $1 の値として、On Web Authentication Database Method に続けて On Web Connection Database Method が呼び出されます。

警告：前回のセッションで作成されたブックマークでデータベースにユーザが再度入るのを避けるために、4D は標準 4D URL のいずれかに対応する URL があればそれを途中で処理します。

HTTP リクエストのヘッダとボディ

2 番目の引数（$2）は、Web ブラウザから送信された HTTP リクエストのヘッダとボディです。このヘッダが**On Web Connection** データベースメソッドにそのまま渡されていることに注意してください。そのコンテンツは接続を試行する Web ブラウザの特徴によって変わります。

MacOS 上で動作する Netscape 4.5 を使用している場合、次のようなヘッダを受け取ります。

```
GET/ HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.5(Macintosh; I; PPC)
Host: 192.168.0.99
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: us
Accept-Charset: iso-8
```

Windows 上で動作する Microsoft Internet Explorer 6 を使用している場合、次のようなヘッダを受け取ります。

```
229=1406
GET/ HTTP/1.0
```

Connection: Keep-Alive
User-Agent: Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.1)
Host: 192.168.0.99
Accept: image/gif, image/x-xbitmap, image/pjpeg, */*
Accept-Language: us

アプリケーションがこの情報を扱う場合には、ヘッダとボディの解析はプログラマの責任になります。

■ Webクライアントの IPアドレス

3番目の引数（$3）には、ブラウザが動作しているマシンの IPアドレスが渡されます。この情報は、イントラネット接続とインターネット接続を区別できるようにするものです。

■ Webサーバの IPアドレス

4番目の引数（$4）には、4D Webサーバの IPアドレスが渡されます。4Dは複数の IPアドレスを持つマシンでマルチホーミングを可能にしています（より詳しい情報は「Webサーバセッティング」を参照してください）。

■ ユーザ名とパスワード

5、6番目の引数$5および$6には、ユーザによって入力されたユーザ名とパスワードが渡されます。「パスワード入力」ダイアログは、「環境設定」ダイアログボックスで「パスワード使用」オプションが選択された場合、各接続時に表示されます（「接続セキュリティ」を参照してください）。

On Web Connectionデータベースメソッドの呼び出し

特別な「4DCGI」URL、またはカスタマイズしたコマンドの URLのいずれかを使用する場合、「On Web Connection」データベースメソッドは、4D Webサーバの入り口点として使用されます。また、コンテキストモードにおいても入口点としての役割を果たします（4th Dimensionと 4D Serverの使用時）。

警告：インタフェース要素（ALERT、DIALOG等）を表示する4Dコマンドを呼びだすと、メソッドの処理が終了します。

On Web connectionデータベースメソッドは以下の場合に呼ばれます。

■ ブラウザから非コンテキストモードで動作する4D Webサーバに接続する場合。データベースメソッドは「/<action>...」という URLを使用して呼び出されます。

■ 4Dが「/4DMETHOD」という URLを受け取った場合。Webサーバはコンテキストモードへ切り替わり、データベースメソッドは$1に代入された「/4DMETHOD/メソッド名」という URLを使用して呼び出されます。

■ 4Dが「/4DCGI」というURLを受け取った場合。データベースメソッドは $1 に代入された「/4DCGI/<action>」というURLを使用して呼び出されます。

■ Webページが「<path>/<file>」タイプのURLを使用して呼び出され、指定されたファイルが見つからない場合。こうした特別の場合、$1 に受け取られるURLは"/"で始められません。

■ タイプ<file>のURLでWebページが呼ばれ、デフォルトのホームページが定義されていない場合、データベースメソッドはURLでコールされます。こうした特別の場合、$1 に受け取られるURLは"/"で始められません。

On Web Connectionデータベースメソッドがコンテキストから、または非コンテキストでの接続から呼び出されたかをを知るには、**Web Context**関数を使うことができます。コンテキストモードで呼ばれた場合はTrueが戻り、そうでなければFalseが戻ります。

したがって、**On Web Connection**データベースメソッドを以下の構造にすることをおすすめします。

```
` On Web connection データベースメソッド
C_TEXT ($1;$2;$3;$4;$5;$6)
If (Web Context) `もしコンテキストモードなら
	WithContext ($1;$2;$3;$4;$5;$6)
		` WithContext は 4D 6.0.x の On Web connection
		`データベースメソッドのすべてを含む
Else
	NoContext ($1;$2;$3;$4;$5;$6)
		`NoContext メソッドは非コンテキストモードで
		'リクエストの処理を実行する（通常は短い）
End if
```

例：コンテキストモードにおけるクライアントローカルホームページの実装

以下の例では、**On Web Connection**データベースメソッドに送られる引数$1は、ある組織の中でクライアントホームページを実装するために使用されます。イントラネットサーバはコンテキストモードで動作します。

データベースには、[顧客]と[商品]という2つのテーブルがあります。次に示す**On Startup**データベースメソッドは、後で**On Web Connection**データベースメソッドで使用されるインタープロセス配列を初期化します。

```
`「On Startup」データベースメソッド
` テーブルリスト
ARRAY STRING (31 ; <>asTables ; Count tables)
For ($vlTable ; 1 ; Size of array (<>asTables))
```

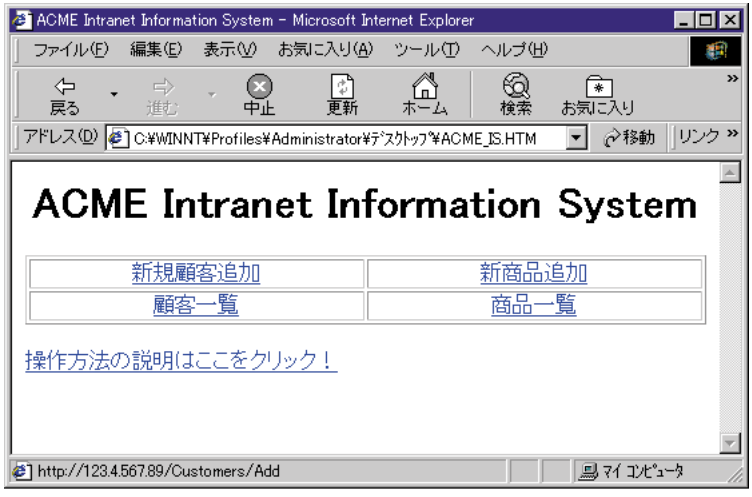
```
        <>asTables{$vlTable}:=Table name ($vlTable)
    End for
        `ログイン時の標準Webアクション
    ARRAY STRING (31 ; <>asActions ; 2)
    <>asActions{1}:="Add"
    <>asActions{2}:="List"
```

On Web Connectionデータベースメソッドの主な仕事は、アドレスのホスト部分の後ろにあるURLに渡されたエクストラデータを解読して、それに従ってアクションを実行することです。このメソッドは、以下の通りです。

```
    `「On Web Connection」データベースメソッド
    C_TEXT ($1;$2;$3;$4;$5;$6)
    C_TEXT ($vtURL)
    If (Web context) `もしコンテキストモードなら
        `万一に備えて$1が"/"または"/..."と同じかどうかをチェックする
        If ($1="/@")
            `URLから最初の"/"を引いたものをローカル変数にコピーする
        $vtURL:=Substring ($1 ; 2)
            `URLを解析し、URLのトークンでローカル配列を登録する
            `例えば、、URLのエクストラデータが"aaa/bbb/ccc"の場合には、
            `結果の配列は"aaa"、"bbb"、"ccc"の順の3つの要素になる
        $vlElem:=0
        ARRAY TEXT ($atTokens ; $vlElem)
        While ($vtURL # "")
            $vlElem:=$vlElem+1
            INSERT ELEMENT ($atTokens ; $vlElem)
            $vlPos:=Position ("/" ; $vtURL)
            If ($vlPos>0)
                $atTokens{$vlElem}:=Substring ($vtURL ; 1;$vlPos-1)
                $vtURL:=Substring ($vtURL ; $vlPos+1)
            Else
                $atTokens{$vlElem}:=$vtURL
                $vtURL:=""
            End if
        End while
        `URLのHOST部分の後にエクストラデータが渡された場合
        If ($vlElem>0)
            `On Startupデータベースメソッドで初期化されたインタープロセス配列を
                使用して、1番目のトークンがテーブルの名前であるかどうかを
                                                チェックする
            $vlTableNumber:=Find in array (<>asTables ; $atTokens{1})
```

```
                    If ($vlTableNumber>0)
                        `その場合には、このテーブルに対するポインタを取得する
                        $vpTable:=Table ($vlTableNumber)
                        `入力フォームと出力フォームを設定する
                        INPUT FORM ($vpTable-> ; "Web入力")
                        OUTPUT FORM ($vpTable-> ; "Web出力")
                        `On Startupデータベースメソッドで初期化されたインター
                                プロセス配列を使用して2番目のトークンが既知の
                                    標準アクションであるかどうかをチェックする
                        $vlAction:=Find in array (<>asActions ; $atTokens{2})
                        Case of
                            `レコードの追加
                            ¥ ($vlAction=1)
                                Repeat
                                    ADD RECORD ($vpTable-> ; *)
                                Until (OK=0)
                                `レコードの一覧表示
                            ¥ ($vlAction=2)
                                READ ONLY ($vpTable->)
                                ALL RECORDS ($vpTable->)
                                DISPLAY SELECTION ($vpTable-> ; *)
                                READ WRITE ($vpTable->)
                        Else
                        `ここで追加の標準テーブルアクションを実装できる
                        End case
                    Else
                    `ここでその他の標準アクションを実装できる
                    End if
                End if
            End if
            `これまでのコードで何が起こっても、通常のLog Onプロセスで追跡する
            WWW NORMAL LOG ON
            Else
            ... ` ここは非コンテキストモードが管理するコードが
                `実行されます。
            End if
```

この時点で、この組織の人々は、データベースに接続し、リストされたメソッドで設定された表記に従ったURLを入力できます。また、ユーザは毎回URLを再入力したくない場合には、お気に入りを作成することもできます。最終的なソリューションは、組織のメンバ全員に対してデータベースをアクセスするためにローカルで使用するHTMLページを提供することです。このHTMLページは、以下のようになります。



つまり、HTMLページ「ACME_IS.HTM」は、組織内の4Dベースの情報システムに対するクライアントのローカルホームページなのです。ユーザが「新規顧客追加」リンクをクリックすると、Webブラウザは URLが「http://192.168.0.99/Products/Add」であるホストに接続します。データベースコンピュータのIPアドレスが「192.168.0.99」である場合、**On Web Connection**データベースメソッドは$1にURLのエクストラデータ"/Products/Add"を受け取り、このため[商品]テーブルへのレコード追加処理に進みます。

最後には、ユーザはそのページからデスクトップ上へリンクをドラッグ&ドロップして、次に示す「新規顧客追加」アイコンのようなInternet Shortcutアイコンを作成できます。これらのアイコンをダブルクリックするだけで、4D Webデータベースの任意の場所に直接移動できます。

次は、このHTMLページのソースコードです。

```
<HTML>
<HEAD>
 <TITLE>ACME Intranet Information System</TITLE>
</HEAD>
<BODY>
<H1><B>ACME Intranet Information System</B></H1>
<P ALIGN=CENTER><TABLE BORDER=1 CELLPADDING=1 WIDTH="100%">
 <TR>
 <TD>
  <P ALIGN=CENTER><A HREF="http://123.4.567.89/Customers/Add">新規顧客追加</A>
 </TD>
 <TD>
  <P ALIGN=CENTER><A HREF="http://123.4.567.89/Products/Add">新商品追加</A>
 </TD>
 </TR>
 <TR>
 <TD>
  <P ALIGN=CENTER><A HREF="http://123.4.567.89/Customers/List">顧客一覧</A>
 </TD>
 <TD>
  <P ALIGN=CENTER><A HREF="http://123.4.567.89/Products/List">商品一覧</A>
 </TD>
 </TR>
</TABLE></P>
<P><B><A HREF="Help.HTM">If you need Help click Here!</A></B></P>
</BODY>
</HTML>
```

参照

データベースメソッド、On Web Authentication データベースメソッド、Web サービス：非コンテキストモード、Web サービス：特別な URL とフォームアクション

HTML オブジェクトと 4D オブジェクトのバインド

この節では、Web経由での情報のやり取り、つまり動的に値を送受信するために4D Webサーバで利用できる方法について説明します。

次の事柄が処理されます。

- 4D変数に保存された値を動的に送信。
- Webフォームを介してダイナミックな値を受信。
- COMPILER_WEB プロジェクトメソッドの使用。
- サーバ側のイメージマッピングの管理。
- JavaScriptの埋め込み。

注：ダイナミックな値の送受信は、変換された4Dフォームを使用し、コンテキストモードで自動的に実行されます。この件に関する詳細は、「コンテキストモードの使用」の節を参照してください。

ダイナミックな値の送信

HTMLページに4D変数への参照を挿入することができます。これらの参照を任意のタイプのHTMLオブジェクトにバインドすることができます。Webページがブラウザに送信されると、4Dはこれらの参照を変数の現在値で置き換えます。この結果、受信したページにはスタティック要素と4Dの値の両方が含まれます。このタイプのページはセミダイナミックページと呼ばれます。

注意：

・作業にはプロセス変数を使用します。

・HTMLは言語に関連づけられた単語処理をするので、普通テキスト変数を伴って使用するでしょう。しかし、BLOB変数を使うこともできます（これにより、テキスト変数の32000字の上限を回避することができます）。ただし、“text without length”フォーマットのBLOB変数を生成する必要があります。

まず最初に、HTMLオブジェクトは4D変数の値を使用してその値を初期化することができます。

次に、Webフォームがサブミットして返された後に、HTMLオブジェクトの値は4D変数に納めて返すことができます。これを実行するには、フォームのHTMLソースの中で、バインドしたい4Dプロセス変数の名前と同じ名前のHTMLオブジェクトを作成します。この詳細については、「ダイナミックな値の受信」の節で更に詳しく検討します。

注：非コンテキストモードでは、4Dピクチャ変数を参照することはできません。

HTMLオブジェクト値は4D変数値で初期化できるため、プログラムでHTMLオブジェクトのvalueフィールドに<!--4DVAR 変数名-->を含めることにより、デフォルト値を提供できます。この“VarName”は、現在のWebプロセスで定義された4Dプロセス変数の名前です。この変数名を標準のHTML表記<!--...-->で囲みます。

注意：いくつかのHTMLエディタはHTMLオブジェクトの変数フィールド中の<!--4DVAR 変数名-->を受け付けないかもしれません。その場合、HTMLコード中に記述する必要があります。

また、<!--4DVAR-->タグを使用し、送信するページに4D表記を挿入することができます（フィールド、配列要素等）。このタイプのデータを使用したこのタグの処理は、変数の場合と同じです。詳細については「4DのHTMLタグ」の節を参照してください。

実際、<!--4DVAR 変数名-->構文を使用すると、HTMLページの任意の場所に4Dデータを挿入できます。例えば、以下のように書いた場合に、

Welcome to <!--vtSiteName-->!

4D変数「vtSiteName」の値が、HTMLページ内に挿入されます。

以下に例を示します。

```
` 以下の4Dコードは、"4D4D"をプロセス変数「vs4D」に割り当てます
vs4D:="4D4D"
` 次に、HTMLページ"AnyPage.HTM"を送信します
SEND HTML FILE ("AnyPage.HTM")
```

HTMLページAnyPage.HTMのソースは、以下の通りです。

```
<HTML>
<HEAD>
   <TITLE>AnyPage</TITLE>
   <SCRIPT LANGUAGE="JavaScript"><!--

       function Is4DWebServer(){
          return (document.frm.vs4D.value=="4D4D")
       }

       function HandleButton(){
          if (Is4DWebServer()){
             alert("あなたは4D Webサーバに接続しています!")
          } else {
             alert("あなたは4D Webサーバに接続していません!")
          }
       }


   //--></SCRIPT>
</HEAD>
<BODY>
<FORM action="/4DMETHOD/WWW_STD_FORM_POST" method="POST" name="frm">

<P><INPUT TYPE="hidden" NAME="vs4D" VALUE="<!--4DVAR/vs4D-->"></P>

<P><A HREF="JavaScript:HandleButton()"><IMG SRC="AnyGIF.GIF" BORDER=0
ALIGN=bottom></A></P>

<P><INPUT TYPE="submit" NAME="bOK" VALUE="OK"></P>

</FORM>
</BODY>
</HTML>
```

非コンテクストモードでは、いくつかの条件下でのみHTMLソースコードを解析します。

前述のHTMLソースコードのvs4Dという名前のhidden入力オブジェクトを見てください。このオブジェクトの値はテキスト値“<!--4DVAR vs4D-->”に設定されています。HTMLファイルを送信しているプロジェクトメソッドには、以前に定義された4Dプロセス変数vs4Dがあるため、4DはこのHTMLオブジェクトの値を置き換えて4D変数の値である"4D4D"に設定します。

埋め込みJavaScript関数「Is4DWebServer」は、vs4D HTMLオブジェクトの値をテストします。以下にその手順を示します。HTMLページが4Dから提供される場合には、オブジェクトの値は"4D4D"に変更されます。ただし、HTMLページが別のアプリケーション（つまり、Macintosh上の4D WebSTAR）から提供される場合には、オブジェクトはHTMLページ内で定義された値“<!--4DVAR vs4D-->”のままになります。つまり、Webブラウザ側にあるページ内からJavaScriptを使ってそのオブジェクトの値をテストすることで、ページが4Dから提供されたものかどうかが検出できるのです。

この最初の例は、4Dで提供される際に他のWebサーバとの互換性を維持しながら追加の機能を提供する「インテリジェントな」HTMLページを構築する方法を示しています。

重要：ユーザがバインドできるのはプロセス変数だけです。また、現行のバージョンでは、4D配列でHTML SELECTオブジェクトをバインドすることはできません。一方、SELECTオブジェクトの各要素は独立した4D変数を参照できます（最初の要素をV1へ、2番目の要素をV2へなど）。

4DからWebブラウザへ向けてのバインド設定は、任意のカプセル化メソッド（**SEND HTML FILE**、**SEND HTML BLOB**コマンド、ならびにコンテキストモードにおける4Dフォームのスタティックテキストやテキスト変数、またはBLOB変数）を使用して行われます。

サーバから送信されたページの解析

■ コンテキストモードでHTMLページ（HTMLドキュメントや変換された4Dフォーム）を送信する前に、常に4DはHTMLソースコードを解析し、4D変数を参照するオブジェクトを探します。

■ 非コンテキストモードでは最適化のため、HTMLページが“.HTML”や“.HTM”で終わるシンプルなURLを用いて呼び出される場合には、4D Webサーバによる HTMLソースコード解析が行われません。もちろん、4Dでは必要に応じてページの解析を“強制的に”行える仕組みが提供されています（「4DのHTMLタグ」の節を参照）。

4D 変数への HTML コードの挿入

4Dの変数にはHTMLコードを挿入することができます。HTMLスタティックページがウェブサーバー上に表示されているとき、変数の値はHTMLコードに置き換えられ、ブラウザによって変換されます。

4D変数にHTMLコードを挿入するためには、2つの方法があります。

■ 4 Dの変数を、ASCIIコードの1で始まるようにして(例えば、vtHTML:=Char(1)+"...HTMLコード...")、それを<!--4DVAR vtHTML-->タグを使うHTMLページに加える。

■ 直接4D変数を(例えば、vtHTML:="...HTMLコード...")、<!--4DHTMLVAR vtHTML-->タグを使うHTMLページに挿入する。

テキスト変数かBLOB変数(Text Without Lengthモードで生成されたもの)を使うことができます。

詳細は「4D HTMLタグ」の節を参照してください。

ダイナミックな値の受信

SEND HTML FILE、**SEND HTML BLOB**コマンドを使用してHTMLページを送信する場合、「Webブラウザから4Dへ」の方向で4D変数をHTMLオブジェクトにバインドすることもできます。このバインドは双方向に作動します。つまり、HTMLフォームがサブミットされると、4DはHTMLオブジェクトの値を4Dプロセス変数にコピーして戻します。データベースをコンパイルする場合を考慮して、これらの変数はCOMPILER_WEBメソッド内で必ず宣言してください（後述する節を参照）。

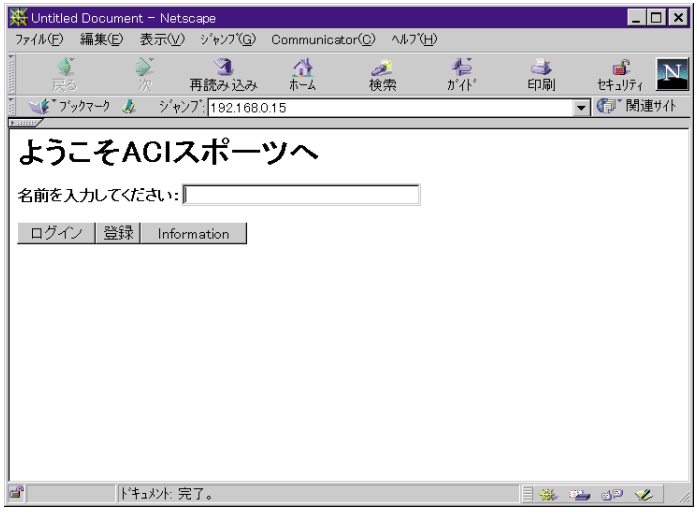
さらに、**GET WEB FORM VARIABLES**コマンドを使用すると、Webフォーム内に含まれているフィールドが事前に分からなくても、4Dへ送信されたWebフォームの値を取得することができます。

警告：値を4Dプロセス変数に取り戻すことは、HTMLページがSEND HTML FILEかSEND HTML BLOBコマンドを使用して送信されたときにのみ可能です。コンテキストモードにおいてHTMLを4Dフォームに組み込んだ場合、値の取得はフォームに実際に存在する4Dのオブジェクトに制限されます。

以下のHTMLページソースコードを参照してください。

```
<HTML>
<HEAD>
   <TITLE>Welcome</TITLE>
   <SCRIPT LANGUAGE="JavaScript"><!--

         function GetBrowserInformation(formObj){
            formObj.vtNav_appName.value = navigator.appName
            formObj.vtNav_appVersion.value = navigator.appVersion
            formObj.vtNav_appCodeName.value = navigator.appCodeName
            formObj.vtNav_userAgent.value = navigator.userAgent
            return true
         }

         function LogOn(formObj){
            if(formObj.vtUserName.value!=""){
               return true
            } else {
               alert("Enter your name, then try again.")
               return false
            }
         }
    //--></SCRIPT>
</HEAD>
<BODY>
<FORM action="/4DMETHOD/WWW_STD_FORM_POST" method="POST" name="frmWelcome"
                        onsubmit="return GetBrowserInformation(frmWelcome)">

<H1>ようこそACIスポーツへ</H1>

<P><B>名前を入力してください:</B>
   <INPUT TYPE="text" NAME="vtUserName" VALUE="<!--4DVARmvtUserName-->" SIZE=30 </P>
<P>
   <INPUT TYPE="submit' NAME="vsbLogOn" VALUE="ログイン" onclick="return LogOn(frmWelcome)">
   <INPUT TYPE="submit' NAME="vsbRegister" VALUE="登録">
   <INPUT TYPE="submit' NAME="vsbInformation" VALUE="Information">
</P>

<P>
   <INPUT TYPE="hidden' NAME="vtNav_appName" VALUE="">
   <INPUT TYPE="hidden' NAME="vtNav_appVersion" VALUE="">
   <INPUT TYPE="hidden' NAME="vtNav_appCodeName" VALUE="">
   <INPUT TYPE="hidden' NAME="vtNav_userAgent" VALUE="">
</P>
</FORM>
</BODY>
</HTML>
```

4Dがこのページを Web ブラウザに送ると、以下のように表示されます。



このページの主な機能は、以下のようになります。

■ vsbLogOn、vsbRegister、vsbInformationという3つのサブミットボタンがあります。

■ ユーザが「ログイン」ボタンをクリックすると、フォームのサブミットは最初にJavaScript関数「LogOn」で処理されます。名前が入力されていないと、フォームは4Dへのサブミットを行わないで、JavaScriptの警告を表示します。

■ フォームには4Dメソッドを Postする Submitスクリプト（GetBrowserInformation）があります。このスクリプトは、名前が「vtNav_App」で始まる4つの隠しオブジェクトに対して Navigatorプロパティをコピーします。

■ オブジェクト「vtUserName」の初期値は、<!--4DVAR vtUserName-->です。

このHTMLページを**SEND HTML FILE**コマンドを使用して送信する4Dメソッド「WWW Welcome」の内容を見てみましょう。このメソッドは、「On Web Connection」データベースメソッドから呼び出されます。

```
`「WWW Welcome 」プロジェクトメソッド
` WWW Welcome → ブール
` WWW Welcome → Yes = セッションを開始できる
C_BOOLEAN ($0)
$0:=False
`ブラウザ情報を返す隠し INPUT HTML オブジェクト
C_TEXT(vtNav_appName ; vtNav_appVersion ; vtNav_appCodeName ;
                                                    vtNav_userAgent)
vtNav_appName:=""
vtNav_appVersion:=""
vtNav_appCodeName:=""
vtNav_userAgent:=""
`ユーザ名が入力されるテキスト INPUT HTML オブジェクト
C_TEXT (vtUserName)
vtUserName:=""
` HTML サブミットボタンの値
C_STRING (31 ; vsbLogOn ; vsbRegister ; vsbInformation)
Repeat
    `サブミットボタンの値をリセットするのを忘れずに！
    vsbLogOn:=""
    vsbRegister:=""
    vsbInformation:=""
    `Web ページを送信する
    SEND HTML FILE ("Welcome.HTM")
```

```
`どのサブミットボタンがクリックされたのかを検出するためにボタンの値
                                                    をテストする
Case of
    `「Log On 」ボタンがクリックされた場合
    ¥(vsbLogOn  #  "")
        QUERY ([WWWユーザ];[WWWユーザ]ユーザ名
                                                    =vtUserName)
        $0:=(Records in selection ([WWWユーザ])>0)
        If ($0)
            WWW POST EVENT ("Log On" ; WWWログ情報)
            `WWW POST EVENTメソッドはデータベーステーブル
                                                    に情報を保存する
        Else
            CONFIRM ("このユーザ名は登録されていません。
                                                    登録しますか?")
            $0:=(OK=1)
            If ($0)
                $0:= WWW Register
                `WWW Registerメソッドで新しいWebユーザ
                                                    の登録ができる
            End if
        End if
    `「Register」ボタンがクリックされた場合
    ¥(vsbRegister  #  "")
        $0:= WWW Register
        `「Information  」ボタンがクリックされた場合
    ¥(vsbInformation  #  "")
        DIALOG ([ユーザインタフェース] ; "WWW情報")
End case
Until (Not (<>vbWebServicesOn)  |  $0)
```

このメソッドの機能は、以下の通りです。

■4D変数「vtNav_appName」、「vtNav_appVersion」、「vtNav_appCodeName」、「vtNav_userAgent」（同じ名前のHTMLオブジェクトにバインドされている）は、JavaScriptスクリプト「GetBrowserInformation」を使用してHTMLオブジェクトに割り当てられた値を戻します。単純、かつ直接的にメソッドは変数を文字列として初期化してから、Webページがサブミットされた後で値を戻します。

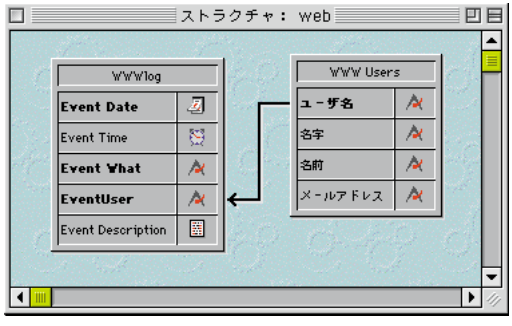
■ 4D変数「vsbLogOn」、「vsbRegister」、「vsbInformation」は、3つのサブミットボタンにバインドされます。これらの変数は、ページがブラウザに送られるたびにリセットされることに注意してください。これらのボタンのどれかでサブミットが実行されると、ブラウザはクリックされたボタンの値を4Dに返します。4D変数はそのたびにリセットされるため、空の文字列ではなくなった変数によって、どのボタンがクリックされたのかがわかります。それ以外の2つの変数は空の文字列であり、これはブラウザが空の文字列を返したためではなく、ブラウザがそれらの変数について何も「言わなかった」ためです。その結果、4Dは変数をそのままにしていたのです。このため、ページがブラウザに送信されるたびに、これらの変数を空の文字列でリセットする必要があるのです。

これが、複数のサブミットボタンがWebページに存在する時に、どのサブミットボタンがクリックされたのかを見分ける方法です。4Dフォーム内の4Dボタンは、数値変数です。ただし、HTMLでは、すべてのオブジェクトはテキストオブジェクトになります。

4D変数をSELECTオブジェクトでバインドする場合には、テキスト変数もバインドします。4Dでは、ドロップダウンリストのどの要素が選択されたのかをテストするには、4D配列の数値をテストします。HTMLでは、これはHTMLオブジェクトにバインドされた4D変数に返される選択された項目の値になります。

4D変数でどのオブジェクトをバインドするかに関係なく、返される値はテキストタイプであり、文字列またはテキストの4Dプロセス変数をバインドすることになります。

この例で注目すべき点は、ブラウザについての情報を取得した後、Webとデータベースの機能をもう一度組み合せることで、これらの値を4Dテーブルに格納できるということです。これが、(一覧されていない)「WWW POST EVENT」プロジェクトメソッドが実行する作業です。このメソッドは「イベントをポストする」わけではなく、Webセッション情報を次に示すテーブルに保存します。



情報をテーブルに保存した後、他のプロジェクトメソッドを使用してその情報をWebユーザに送り返すことができます。これを実行するには、単に**QUERY**コマンドを使用して適切な情報を検索し、**DISPLAY SELECTION**コマンドを使用して[WWW Log]レコードを表示します。以下の図は、Webサイトに登録されたユーザが利用できるログ情報を示しています。

この例で示すバインド機能を利用すると、HTML ダイアログや 4D フォームを使って提示する、またはユーザから収集した情報をすべて組み合わせ、データベースの Web サイトに非常に役立つ管理機能を追加することができます。

COMPILER_WEB プロジェクトメソッド

4D Web サーバはポストされたフォームを受け取ると、**COMPILER_WEB** という名前のプロジェクトメソッド（存在する場合）を自動的に呼び出します。このメソッドには、ポストされたフォーム内のフィールドと同じ名前を持つ全変数のタイプ指定や、変数の初期化指定が含まれていなければなりません。これはデータベースのコンパイル時に、コンパイラにより使用されます。

COMPILER_WEB メソッドはすべての Web フォームに共通です。デフォルトでは **COMPILER_WEB** メソッドが存在していません。したがって、これを明示的に作成しなければなりません。

注：GET WEB FORM VARIABLES コマンドを使用することもできます。このコマンドは、サブミットされた HTML ページに含まれるすべての変数の値を取得します。

Web サービス：SOAP リクエストが受け入れられるたびに、COMPILER_WEB メソッドが呼びだされます（存在する場合）。Web サービスとして公開されるすべてのメソッドに対し、COMPILER_WEB メソッドを使用して、SOAP 入力引数に関連付けた 4D 変数をすべて宣言しなければなりません。実際は、Web サービスメソッドでプロセス変数を使用する場合には、メソッド呼び出しの前にそれらの変数が宣言されている必要があります。この件に関する詳細は、SOAP DECLARATION コマンドの説明を参照してください。

HTMLオブジェクトと4D変数とのバインド設定・イメージマッピング

「コンテキストモードの使用」の節で説明したように、Webページとして4Dフォームを使用すると、4Dは静止ピクチャに重なる非表示系のボタンを使って、サーバ側のイメージマッピングを提供します。

SEND HTML FILEまたは**SEND HTML BLOB**コマンドを使用してHTMLドキュメントを送信する場合には、4D変数をイメージマップのHTMLオブジェクト（INPUT TYPE="IMAGE"）でバインドし、情報を取り出すことができます。例えば、「bImageMap」という名前のイメージマップのHTMLオブジェクトを作成することができます(実際には任意の名前を付けることができます)。ブラウザ側にあるイメージをクリックするたびに、クリック位置でのサブミットが4D Webサーバに送り返されます。クリックの座標（イメージの左上端から相対的に表される）を取り出すには、4Dプロセス変数bImageMapと、bImageMap_X変数およびbImageMap_Y変数（倍長整数タイプ）をリードする必要があるだけです。これらの変数はクリックの縦方向の座標と横方向の座標を含んでいます。これらの変数はCOMPILER_WEBプロジェクトメソッドで定義する必要があります。

HTMLページでは、以下のようなコードを作成します。

```
<P><INPUT TYPE="image" SRC="MonImage.GIF" NAME="bImageMap"
                                                    BORDER=0></P>
```

HTMLページを送信する4Dメソッドは次の通りです。

```
SEND HTML FILE("ThisPage.HTM")
```

COMPILER_WEBプロジェクトメソッドにおいて、以下のようなコードを作成します。

```
C_LONGINT(bImageMap_X;bImageMap_Y)
bImageMap_X:=-1 ` 変数の初期化
bImageMap_Y:=-1 ` 変数の初期化
```

次に、POSTアクション4Dメソッドまたはカレントメソッドで、POSTアクションメソッドが**SEND HMTL FILE**("")呼び出しを行った後に、変数bImageMap_XとbImageMap_Y におけるクリックの座標を取り出します。

```
If (($bImageMap_X#-1)&($bImageMap_Y#-1))
      ` 座標に対応した処理を行う
End if
```

JavaScriptカプセル化

4Dは、HTMLドキュメントに埋め込まれたJavaScriptソースコードをサポートします。また、JavaScriptの「.js」ファイルのHTMLドキュメントの埋め込みをサポートします（例えば、<SCRIPT SRC="...">）。

標準モードで**SEND HTML FILE**または**SEND HTML BLOB**コマンドを使用すると、HTMLソースエディタで準備したページや、4Dを使用してプログラムで構築し、ディスク上に保存していたページを送信できます。どちらの場合でも、開発者がページを全面的に制御できます。FORMマークアップでスクリプトを使用するのと同様に、ドキュメントのHEADセクションにJavaScriptスクリプトを挿入できます。前の例では、開発者がフォームに名前を付けることができたため、スクリプトはフォーム"frm"を参照します。FORMマークアップレベルで、フォームのサブミッションのトリガ、受け付け、拒否もできます。

コンテキストモードにおいて、4DフォームにHTMLをカプセル化する場合には、HEADセクションやFORM宣言に対する制御は行えません。このため、スクリプトの有効範囲が異なります。例えば、HTMLフォームに対してはその名前でアクセスすることはできません。ただし、下記のものと上記の例とで、JavaScript関数「Is4DWebServer」を比べてみると、

```
function Is4DWebServer ( ) {
        return (document.form[0].vs4D.value=="4D4D")
}
```

どちらの関数も同じことをしていますが、2番目の例はHTMLドキュメントオブジェクトのフォームプロパティを使用してforms[0]要素からオブジェクトにアクセスしています。結果として、4Dが変換されたHTMLページ（フォーム）に名前を設定してもしなくても、その名前を知らないときでも作動します。

注：4DはJavaアプレットのトランスポートをサポートしています。

参照

SEND HTML FILE、SEND HTML FILE、Webサービス：非コンテキストモード

URLとフォームアクション

4D Webサーバでは、各種URLとフォームアクションが提供されます。これによりコンテキストモードおよび非コンテキストモードの両方において、さまざまなアクションをデータベースに実装することができます。

これらのURLは以下の通りです。

■ 4DMETHOD/：コンテキストモードにおいて、任意のHTMLオブジェクトをデータベースのプロジェクトメソッドにリンクすることができます。

■ 4DACTION/：非コンテキストモードにおいて、任意のHTMLオブジェクトをデータベースのプロジェクトメソッドにリンクすることができます。

■ 4DCGI/：任意のHTMLオブジェクトから「**On Web Connection**」データベースメソッドを呼び出すことができます。

注：さらに、4D Webサーバは4つの特殊なURL、/4DSTATS、/4DHTMLSTATS、/4DCACHECLEAR、/4DWEBTESTを受け付け、これにより4D Webサイトの動作状況に関する情報を取得することができます。これらのURLについては、「Webサイトの情報」の節で説明しています。

URL 4D ACTION/

シンタックス：4DACTION/MyMethod{/Param}

モード：非コンテキストモード。コンテキストモードから呼び出されると、そのコンテキストのプロセスは中止され、非コンテキストモードに切り替わります。

使用方法：URLまたはフォームアクション

このURLを使用すると、コンテキストモードでHTMLオブジェクト（テキスト、ボタン等）を4Dプロジェクトメソッドにリンク付けることができます。リンクは、/4DMETHOD/メソッド名/引数と指定します。「メソッド名」は4Dプロジェクトメソッドの名前で、ユーザがそのリンクをクリックすると実行されます。

4Dが/4DACTION/MyMethod/Paramリクエストを受け取ると、「On Web Authentication」データベースメソッド（存在する場合）が呼び出されます。Trueが返された場合、メソッドMyMethodが実行され、オプションの/Param文字列は引数として$1に納められます（後述の「URLでコールした4Dメソッドに渡されたテキスト引数」の節を参照してください）。

スタティックなWebページにおいて、4DACTION/をURLに関連付けることができます。URLのシンタックスは、以下の形式でなければなりません。

```
<A HREF="/4DACTION/MyMeth/Param">処理を行う</A>
```

通常、プロジェクトメソッドMyMethは“応答”を返します（**SEND HTML FILE**や**SENT HTML BLOB**コマンドを使用してHTMLページを送信する等）。ブラウザをブロックしないように、必ずこの処理はできるだけ短時間で行ってください。

注：/4DACTIONで呼び出されたメソッドでは、インタフェースエレメント（DIALOG、ALERT等）を呼び出してはいけません。

警告：「4DACTION/」URLを用いて4Dメソッドを実行できるように、メソッドプロパティで「4DACTIONで利用可能」オプション（デフォルトでは未選択）を指定しなくてはなりません。この件に関する詳細は、「接続セキュリティ」の節を参照してください。

例題

この例題は、「4DACTION/」URLをHTMLのピクチャオブジェクトに関連付け、ページ内のピクチャを動的に表示する方法について説明しています。以下の指示をスタティックなHTMLページに挿入します。

<IMG SRC="/4DACTION/PICTFROMLIB/1000">

PICTFROMLIBメソッドは以下の通りです。

```
C_TEXT($1)`常にこの引数は宣言されていなければなりません
C_PICTURE($PictVar)
C_BLOB($BlobVar)
C_LONGINT($Number)
`文字列$1 からピクチャの番号を取得します
$Number:=Num(Substring($1;2;99))
GET PICTURE FROM LIBRARY($Number;$PictVar)
PICTURE TO GIF($PictVar;$BlobVar)
SEND HTML BLOB ($BlobVar;"Pict/gif")
```

フォームをPOSTする4DACTION

“ポストされた”フォームを使用したい場合、4D Webサーバでは追加機能が提供されます。これらのフォームはスタティックなHTMLページであり、Webサーバにデータを送信します。これらのフォームには必ずPOSTタイプを関連付けなければならず、フォームのアクションは/4DACTION/メソッド名で始まることが必須条件です。

注：フォームは2種類のメソッドを使用してサブミットすることができます（ともに4Dで使用されます）。

POST：通常は、Webサーバにデータを追加するために使用されるーデータベースへ

GET：通常は、Webサーバへのリクエストに使用されるーデータベースから

この場合、WebサーバがPOSTされたフォームを受け取ると、COMPILER_WEBプロジェクトメソッド（存在する場合、下記参照）が呼び出され、この後に**On Web Authentication**データベースメソッド（存在する場合）が呼び出されます。メソッドよりTrueが返された場合、メソッド「メソッド名」が実行されます。4Dは、フォーム内に存在するHTMLフィールドを解析して値を取り出し、自動的にその内容を4D変数へ代入します。フォームのフィールドと4D変数は、同じ名前でなければなりません。

注：詳細については、「HTMLオブジェクトと4Dオブジェクトのバインド」の節を参照してください。

フォームで適用するHTML構文は、下記のタイプになります。

■ フォーム内のアクションを定義するには：

```
<FORM ACTION="/4DACTION/メソッド名" METHOD=POST>
```

■ フォーム内のフィールドを定義するには：

```
<INPUT TYPE= フィールドタイプ NAME= フィールド名 VALUE="デフォルト値">
```

フォームの各フィールドに対し、4Dはフィールド値を同じ名前を持つ変数の値にセットします。フォームのオプション（例えばチェックボックス）に対しては、4Dは関連する変数を、選択されていれば1に、そうでなければ0にセットします。

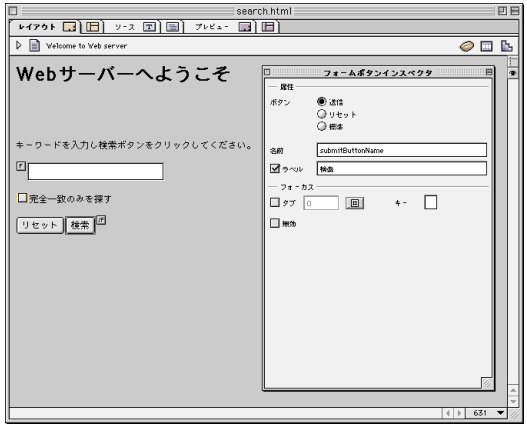
数値データの入力に関し、4Dはフィールドの値を文字から実数へと変換します。

注：フォームのフィールドにOKという名前が付けられている場合（例えばサブミットボタン）、フィールドの値が空でなければ、システム変数OKに 1 が代入されます。それ以外の場合は0が代入されます。

例題

非コンテキストモードで開始され、使用されている4D Webデータベースにおいて、ブラウザではスタティックなHTMLページを使用したレコードの検索を行いたいとします。このページを“search.htm”とします。さらに、このデータベースには他にもスタティックページがあり、例えば、検索結果を表示することができます（“results.htm”）。このページには、POSTタイプとともに/4DACTION/PROCESSFORMアクションが関連付けられています。

下図は、Adobe GoLiveのHTMLエディタに表示されるページです。



このページに対応するHTMLコードを以下に示します。

```
<FORM ACTION="/4DACTION/PROCESSFORM" METHOD=POST>
```

```
<INPUT TYPE=TEXT NAME=VNAME VALUE=""><BR>
<!-- 通常は、VALUEにボタン名を入れますが、インタプリタ上の理由により、
                                VALUEには番号を入れなければなりません-->
<INPUT TYPE=CHECKBOX NAME=EXACT VALUE="1">完全に一致する語句を
                                検索する<BR>
<!-- OK は特殊なケースです-->
<INPUT TYPE=SUBMIT NAME=OK VALUE="検索">
</FORM>
```

データ入力中に、データ入力エリアに“ABCD”と入力して、オプションをチェックし、検索ボタンをクリックして確定します。

次に、4Dは以下のようなCOMPILER_WEBプロジェクトメソッドを呼び出します。

```
C_TEXT(VNAME)
VNAME:=""
C_LONGINT(vEXACT)
vEXACT:=0
OK:=0 `特殊なケース
```

この例では、VNAMEには文字列“ABCD”が入り、vEXACTは1となり、OKは1になります（ボタン名がOKであるためです）。

4Dは「**On Web Authentication**」データベースメソッド（存在する場合）を呼び出した後、以下に示す**PROCESSFORM**プロジェクトメソッドを呼び出します。

```
If (OK=1)
    If (vEXACT=0) `オプションが選択されなかった場合
        vNAME:=VNAME+"@"
    End if
    QUERY([Jockeys];[Jockeys]Name=vNAME)
    vLIST:=Char(1) `リストをHTMLで返す
    FIRST RECORD([Jockeys])
    While (Not(End selection([Jockeys])))
        vLIST:=vLIST+[Jockeys]Name+" "+[Jockeys]Tel+¥<BR>¥
        NEXT RECORD([Jockeys])
    End while
    SEND HTML FILE("results.htm") `リストをresults.htmフォームに送信する
    `このフォームには変数vLISTへの参照が含まれる
                                    （つまり、 <!--4DVAR vLIST-->）
    ...
End if
```

URL 4DMETHOD/

シンタックス：4DMETHOD/MyMethod{/Param}

モード：コンテキストモード。非コンテキストモードから呼び出されると、コンテキストモードに切り替わります。

使用方法：URLまたはフォームアクション

このURLを使用すると、コンテキストモードでHTMLオブジェクト（テキスト、ボタン等）を4Dプロジェクトメソッドにリンク付けることができます。リンクは、/4DMETHOD/メソッド名/引数と指定します。「メソッド名」は4Dプロジェクトメソッドの名前で、そのHTMLオブジェクトがクリックされると実行されます。また、オプションのテキスト引数である＜引数＞が$1に納められメソッドへと渡されます（後述の「URLでコールした4Dメソッドに渡されたテキスト引数」の節を参照してください）。リンクされたアイテムは、そのURLを介して4Dプロジェクトメソッドの実行を開始します。プロジェクトメソッドでは、4Dフォームや他のHTMLページ等を表示することができます。

4Dが/4DMETHODリクエストを受け取ると、「**On Web Authentication**」データベースメソッド（存在する場合）が呼び出されます。このメソッドから“True（真）”が返されると、「**On Web Connection**」データベースメソッド（存在する場合）が呼び出され、この後に文字列/Paramを引数として使用して（$1に代入される）「メソッド名」メソッドが実行されます。

/4DMETHOD/メソッド名をフォームアクションとしてHTMLスタティックページに割り当てた場合、このメソッドはフォームの「サブミット」ボタンがクリックされた時に実行されます。4D側でこのHTMLフォームをサブミットするには、フォームのサブミット後に4Dで実行されるPOSTアクション4Dメソッドを指定する必要があります。**SEND HTML FILE**コマンドの例題を参照してください。

HTMLページを4Dに統合する際に、最も多く使用するのは「ノーマル」と「サブミット」タイプのボタンです。

フォームで適用するHTMLのシンタックスは、次のタイプのものです。

```
<FORM ACTION="/4DMETHOD/メソッド名" METHOD=POST>
```

ポストされたフォームに関する詳細は、前述した節を参照してください。

警告：「4DMETHOD/」URLを使用して4Dメソッドを実行できるように、メソッドプロパティで「4DACTIONで利用可能」オプション（デフォルトでは未選択）を指定しなくてはなりません。この件に関する詳細は、「接続セキュリティ」の節を参照してください。

URL 4DCGI/<アクション>

シンタックス：4DCGI/<アクション>

モード：両モード

使用方法：URL

4D Webサーバが/4DCGI/<アクション>のURLを受け取ると、「**On Web Authentication**」データベースメソッド（存在する場合）が呼び出されます。このメソッドから“True（真）”が返されると、Webサーバは“現状のまま”このURLを$1に送信して「**On Web Connection**」データベースメソッドを呼び出します。

4DCGI/のURLは、いずれのファイルにも対応しません。その役割は4Dを呼び出すことです。このURLの役割は、「**On Web Connection**」データベースメソッドを使用して4Dを呼び出すことです。引数<アクション>には、あらゆるタイプの情報を納めることができます

このURLを使用して、あらゆるタイプのアクションを実行することができます。必要となる作業は、「**On Web Connection**」データベースメソッドまたはそのサブメソッドのひとつにおいて$1の値を検証し、4D に適切な動作を実行させるだけです。例えば、完全にカスタムなHTMLページを構築して、レコードの追加や検索、ソートを行ったり、GIFイメージをオンザフライで作成することができます。このURLの使用方法に関する例題は、**PICTURE TO GIF**コマンドおよび**SEND HTTP REDIRECT**コマンドの説明を参照してください。

アクションを発行する際は、データを送信するコマンド（**SEND HTML FILE**、**SEND HTML BLOB**等）を使用して、必ず“応答”を返さなければなりません。

警告：ブラウザを停止させないように、可能な限り短い動作を実行することを心掛けてください。

URLでコールした4Dメソッドに渡されたテキスト引数

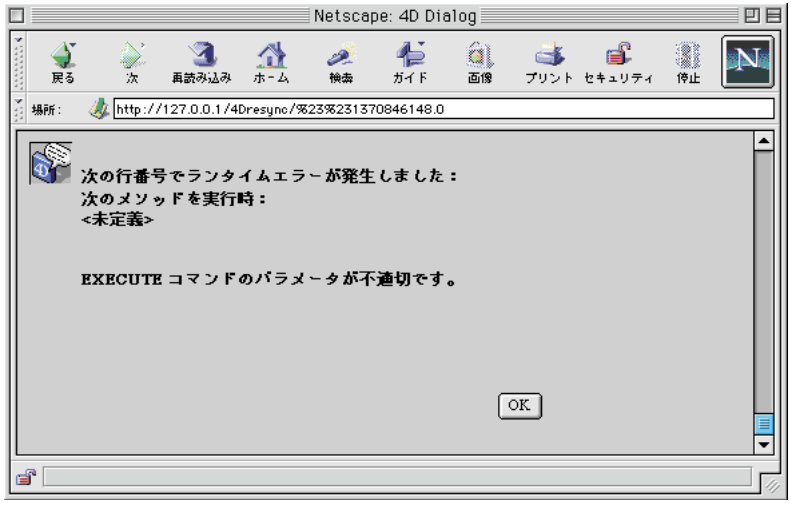
4th Dimensionは、URLでコールされた任意の4Dメソッドに対してテキスト引数を送信します（4DMETHOD/, 4DACTION/...）。コンテキストでも非コンテキストモードでも、このテキスト引数に関する説明は以下の通りです。

■ この引数を使用しない場合でも、「**C_TEXT**（$1）」というコマンドを使用して明示的に宣言しなければなりません。これを行わないと、Webからコンパイルモードで稼働しているデータベースへアクセスする際に、ランタイムエラーが発生します。

■ この引数$1により、URLの終わりの部分にあるエクストラデータが返されます。このエキストラデータは、HTML環境から4D環境へ値を受け渡す際のプレースホルダとして利用することができます。

コンパイルモードにおけるランタイムエラー

以下の例題を見てみましょう。リンクを使用して、HTMLオブジェクトにバインドしたメソッドを実行し、Webブラウザ上に以下の画面を表示します。



このランタイムエラーは、4Dメソッドで$1テキスト引数が宣言されていないことに関連しています。このメソッドは、それを参照しているHTMLリンクがクリックされた際に呼び出されます。カレントHTMLページから実行されているため、このエラーは、実際にそのページをWebブラウザに送信したメソッドの行番号“0”を参照します。

次は、次は、「Webサーバ入門」の例題です。の例題です。メソッドM_ADD_RECORDSおよびM_LIST_RECORDS内で$1テキスト引数を明示的に宣言し、問題を解消します。

```
`「M_ADD_RECORDS」プロジェクトメソッド
C_TEXT ($1) ` この引数は必ず明示的に宣言する
Repeat
        ADD RECORD ([Customers])
Until(OK=0)
`「M_LIST_RECORDS」プロジェクトメソッド
C_TEXT ($1) ` この引数は必ず明示的に宣言する
ALL RECORDS ([Customers])
MODIFY SELECTION ([Customers])
```

上記の変更を行った後は、コンパイルモードでのランタイムエラーは発生しなくなります。

4Dメソッド内で明示的に宣言する引数

4Dメソッドを呼び出す起源と性質によって違った引数を宣言しなければなりません。

On Web Authenticationデータベースメソッドと**On Web Connection**データベースメソッド接続には6つの引数を宣言しなければなりません。

```
` On Web Connection データベースメソッド
C_TEXT ($1;$2;$3;$4;$5;$6)
```

（4DMETHOD/）URLより呼び出されたメソッド引数$1を宣言しなければなりません。

```
`（4DMETHOD/）URLより呼び出されたメソッド
C_TEXT ($1)
```

（4DACTION/）タグよりURLとして呼び出されたメソッドは、引数$1を宣言しなければなりません。

```
` 4DACTION/ URLで呼び出されたメソッド
C_TEXT ($1)
```

ドキュメント中のHTMLコメントとして（4DACTION/）タグより呼び出されたメソッドは$0に値を返します。$0と$1引数を宣言しなければなりません。

```
` HTMLコメントとして（4DACTION/）タグより呼び出されたメソッド
C_TEXT ($0; $1)
```

参照

Webサービス：入門編（パートII）

Webサービス：4D HTMLタグ

4D Webサーバでは、4D特定のさまざまなHTMLタグが提供されます。これらのタグにより、例えば**SEND HTML FILE**コマンドや**SEND HTML BLOB**コマンドを使用してWebサーバから送信されるスタティックなHTMLページに4D変数や4D表記、または各種処理への参照を挿入することができます。これらのページは、セミダイナミックページと呼ばれます。

使用できる4D HTMLタグは以下の通りです。

■ 4DVAR：4D変数や4D表記を挿入します。

■ 4DHTMLVAR：4DVARと似ていますが、HTMLコードを挿入します。

■ 4DSCRIPT：4Dメソッドを実行します。

■ 4DINCLUDE：ページを他のページに挿入します。

■ 4DIF、4DELSE、4DENDIF：HTMLコードに条件式を挿入します。

■ 4DLOOPと4DENDLOOP：HTMLコードにループを作成します。

互換性に関する注意：バージョン6.0.xの4Dでは、スタティックページへの4D変数挿入のための表記として、角括弧 [VarName]が使用されていました。変換後のデータベースにおいて、標準のHTML構文（<!--4DVAR MAVAR-->）を使用可能にするには、「環境設定」ダイアログボックスの「ブラケットの代りに4DVARコメントを使用する」オプションがチェックされていることを確認してください（「Webサーバセッティング」の節を参照）。

4D HTMLタグについて

4Dにより送られたセミダイナミックページの内容の解析は、**SEND HTML FILE**コマンド（.htm、.html、.shtm、.shtml）または**SEND HTML BLOB**コマンド（テキストまたはhtmlタイプのBLOB）のコール時、およびURLを使用して呼び出されたページの送信時に実行されます。

しかし、非コンテキストモードでは最適化のため、“.HTML”や“.HTM”で終わるHTMLページは解析されません。この場合、“強制的に”HTMLページの解析を行うには、“.shtm”や“.shtml”という接尾辞を必ず付加してください（例：http://www.server.com/dir/page.shtm）。

このタイプのページの使用例は、**WEB CACHE STATISTICS**コマンドの解説に示されています。

Webブラウザに送信したHTMLページ内にあるタグを4Dが解析するケースを以下に示します。

送信条件	送信ページの内容分析	
	コンテキストモード	非コンテキストモード
・ページの拡張子（一般的なケース）：		
.htm、.html、.shtm、.shtml（HTMLページ）	X	X
.xml、.xsl（XMLページ）	X	X
.wml（WMLページ）	X	X
・URL経由で呼び出されたページ	X	X 拡張子が.htmまたはであるページを除く
・SEND HTML FILEコマンド呼び出し	X	X
・SEND HTML BLOBコマンド呼び出し（BLOBが“text/html”タイプの場合）	X	X
・<!--4DINCLUDE-->タグによる包含	X	X
・{mypage.htm}タグによる包含	X	-

4Dが処理を行うには、HTMLコメントが<!-- 4D...-->という形式でなくてはなりません。HTMLエディタの中には、コメントに他の情報を自動追加するものもあるため、注意が必要です。この場合、正しく解釈されなくなる可能性があります。

しかし<!--リストの始まり-->のような、その他のHTMLコメントは問題ありません。

<!--4D... というコメントが -->で終わっていない場合、“<!--4D... : --> が必要です”というメッセージが挿入され、この段階で解析が中断されます（エラーを示すためにそのページが送信されます）。

複数タイプのコメントを混在させることができます。例えば、次のようなHTML構文を作成できます。

```
<HTML>
...
<BODY>
<!--4DSCRIPT/PRE_PROCESS--> (Methodの呼び出し)
<!--4DIF (myvar=1)--> (If 条件)
        <!--4DINCLUDE banner1.html--> (サブページの挿入)
<!--ENDIF--> (End if)
<!--4DIF (myvar=2)-->
        <!--4DINCLUDE banner2.html-->
<!--ENDIF-->
<!--4DLOOP [TABLE]--> (カレントセレクションをループ)
<!--4DIF ([TABLE]ValNum>10)--> (If [TABLE]ValNum>10)
        <!--4DINCLUDE subpage.html--> (サブページの挿入)
<!--4DELSE--> (Else)
        <B>Value: <!--4DVAR [TABLE]ValNum--></B><BR>
        (フィールドの表示)
<!--ENDLOOP--> (End for)
</BODY>
</HTML>
```

4DVAR

シンタックス：<!--4DVAR VarName-->

<!--4DVAR VarName-->タグを使用して、変数、配列要素、フィールドへの参照をHTMLページの任意の場所に挿入することができます。例えば、以下のように記述すると

```
<P>Welcome to <!--4D VAR vtSiteName-->!</P>
```

4D変数であるvtSiteNameの値がHTMLページに挿入されます。

最初の桁がASCIIコードの1である場合（つまり、vtHTML:=Char(1)+"...HTMLコード..."）、4Dテキスト変数をHTMLコードに挿入することができます。また、4DHTMLVARタグを使用することもできます。

さらに、4DVARタグを使い、（変数だけではなく）4D言語表現を4D HTMLコメントに挿入することもできます。フィールド内容や（例：<!--4DVAR [テーブル名]フィールド名-->）、配列項目の内容（例：<!--4DVAR 配列{1}-->）を直接挿入することができます。この表記の変換には、変数の時と同じルールが使用されます。さらに、表記は4Dの構文ルールに従わなければなりません。

表記には4D関数のダイレクト呼び出しを含めることができますが、ローカライズの観点からすると、これはお勧めできません。例えば、<!--4DVAR Current date-->の場合、英語版の4Dでは正しく解釈されますが、フランス語版では理解されません。同様の問題が実数に関しても存在します（言語によって、小数点の位置が異なります）。両ケースとも変数への割り当てはプログラムから行うよう強くお勧めします。

インタープリタ上のエラーが発生すると、挿入されたテキストは“<!-4DVAR myvar--> : ##エラー#エラーコード”と表示されます。

注：

・プロセス変数を使用した作業を行えます。

・ピクチャフィールドの内容を表示できます。さらに（コンテキストモードのみ）、ピクチャ変数の内容を表示することもできます。両モードとも、ピクチャ配列項目の内容は表示できません。

・HTMLは言語処理指向のアプリケーションなので、通常はテキスト変数を使用して作業を行います。しかし、BLOB変数を使用することも可能で（これにより、テキストタイプの変数に関する32,000バイトの制限を回避できます）、その方法は長さ属性なしテキスト（Text without length）モードでBLOBを生成するだけです。

・4DVARの使用例は、「HTMLオブジェクトと4Dオブジェクトのバインド」の節に示されています。

互換性に関する注意：バージョン6.0.xの4Dでは、スタティックページへの4D変数挿入のための表記として、角括弧 [VarName]が使用されていました。変換後のデータベースにおいて、標準のHTML構文（<!--4DVAR MAVAR-->）を使用可能にするには、「環境設定」ダイアログボックスの「ブラケットの代りに4DVARコメントを使用する」オプションがチェックされていることを確認してください（「Webサーバセッティング」の節を参照）。

4DHTMLVAR

シンタックス：<!--4DHTMLVAR VarName-->

このタグを使用して、変数や4D表記の評価、およびこれをHTML表記としてページ内に挿入することができます。実際、VarNameがASCIIコードの1で始まる場合、このタグは<!--4DVAR VarName-->タグとまったく同じように作用します（前述の説明を参照）。

例として、使用可能なタグを用いた4Dテキスト変数myvarの挿入結果を以下に示します。

myvarの値	タグ	Web Pageへの挿入
myvar:="<B>"	<!--4DVAR myvar-->	<B>
myvar:=Char(1)+"<B>"	<!--4DVAR myvar-->	<B>
myvar:="<B>"	<!--4DHTMLVAR myvar-->	<B>

インタプリタ上のエラーが発生すると、挿入されたテキストは“<!--4DHTMLVAR myvar--> : ##エラー#エラーコード”と表示されます。

注：テキスト変数は、ISO Latin-1文字セットを使用して表わしてください（詳細は、Mac to ISO関数の説明を参照してください）。

4DSCRIPT/

シンタックス：<!--4DSCRIPT/MethodName/MyParam-->

スタティックなHTMLページを送信する際に4DSCRIPTタグを使用して、4Dメソッドを実行することができます。<!--4DSCRIPT/MyMethod/MyParam-->タグがHTMLコメントとしてスタティックページに存在すると、引数MyParamが文字列として$1にセットされて、メソッドMyMethodが強制的に実行されます。ホームページのロード時に、4Dは**On Web Authentication**データベースメソッド（存在する場合）を呼び出します。データベースメソッドよりTrueが返されると、4Dはメソッドを実行します。メソッドからは、$0にテキストが返されます。返された文字列がASCIIコードの1で始まる場合、HTMLであるとみなされます（変数に関しても同じ原則が適用されます）。

注：4DSCRIPTタグを用いたメソッドの実行は、メソッドプロパティで指定された「4DACTIONで利用可能」オプションの値に影響されません。

SEND HTML FILE（.htm、.html、.shtm、.shtml）または**SEND HTML BLOB**（テキスト／HTMLタイプのBLOB）が呼び出されると、ページコンテンツの解析が行われます。非コンテキストモードにおいては、URLの指すファイルの拡張子が“.shtm”または“.shtml”のいずれかである場合にも解析が行われる点に注意してください（例：http://www.server.com/dir/page.shtm）。

注：コンテキストモードでは、メソッドはコンテキスト内で実行されます。

例えば、“Today is <!-- 4DSCRIPT/MYMETH/MYPARAM-->”というコメントをスタティックページに挿入するとします。4Dはページをロードする時に、**On Web Authentication**データベースメソッド（存在する場合）を呼び出し、次にメソッドMYMETHを呼び出して、文字列“/MYPARAM”を引数$1として渡します。

このメソッドは$0 にテキストを返し（例：“12/31/99”）、したがって“Today is <!--4DSCRIPT/MYMETH/MYPARAM-->”という表記は、“Today is 12/31/99”になります。

メソッドMYMETHは次の通りです。

```
C_TEXT($0) `この引数は常に宣言しなければならない
C_TEXT($1) `この引数は常に宣言しなければならない
$0:=String(Current date)
```

警告：呼び出されるメソッドにおいて、引数$0 および$1 は常に宣言しなければなりません。

注：4DSCRIPTによって呼び出されるメソッドでは、インタフェースエレメント（DIALOG、ALERT等）を呼び出してはいけません。

4Dはメソッドを出現順に実行するため、使用するモードに関わらず、数々の変数がドキュメント内で更に参照されている場合でも、これら変数の値をセットするメソッドを呼び出すことができます。

注：<!--4DSCRIPT...-->コメントは、スタティックページにいくつでも挿入することができます。

注：以前のバージョンの4Dでは、同じタグである4DACTIONは、URL（例：http://myserver/4DACTION/meth）やスタティックページ内のHTMLコメント（<!--4DACTION/meth-->）として使用できました。このようにすると間違いやすいため、バージョン6.7 の4Dでは、4DACTIONに代わり4DSCRIPTというタグが提供されます。このタグは、HTMLコメント（<!--4DSCRIPT/meth-->）としてのみ使用され、<!--4DACTION/meth-->と同じ結果になります。4DACTIONタグは、URLに対してのみ使用します。

4DINCLUDE

シンタックス：<!--4DINCLUDE パス-->

このコメントを使用して、HTMLページに別のHTMLページ（引数<パス>で指定）

のボディを挿入できます。HTMLページのボディは<BODY>タグと</BODY>タグの間に納められます（タグそのものは含まれません）。

<!--4DINCLUDE -->コメントは、判定式（<!--4DIF-->）やループ（<!-- 4DLOOP-->）を使用する際に大変役立ちます。条件に応じて、またはランダムにタグを含める場合、非常に便利です。

含める際に、モードやファイル名の拡張子に関わらず、4Dは呼び出されたページを解析し、その内容（変更された、またはされていない）を4DINCLUDE呼び出しを行なったページ内に挿入します。

<!--4DINCLUDE -->コメントを使用して含められるページは、URLを介して呼び出されたページや**SEND HTML FILE**コマンドで送信されたページと同様に、Webサーバのキャッシュ内へロードされます。

＜パス＞には、含めようとするドキュメントへのパスを指定します。このパスは解析されるドキュメントへの相対位置です。フォルダ区切りにはスラッシュ記号（/）、1階層上がる（HTML構文）場合にはドット2つ（..）を使用してください。

1ページ内に使用できる<!--4DINCLUDE パス-->の数には制限がありません。

しかし、<!--4DINCLUDE パス-->の呼び出しは、1つのレベルでしか行うことができません。例えば、mydoc2.htmlがmydoc1.htmlに挿入されたタグ<!--4DINCLUDE mydoc2-->で呼び出されている場合、mydoc2.htmlのボディページで<!--4DINCLUDE mydoc3.html-->を挿入できないということです。

さらに、4Dはその包含式が再帰的ではないかを検証します。

エラーが発生すると、挿入されるテキストは“<!--4DINCLUDE パス--> :ドキュメントは開かれませんでした”となります。

注：コンテキストモードにおいて、スタティックテキストエリアに挿入した{mypage.html}タグを介してページがフォームに挿入されると、4DINCLUDEコメント（存在する場合）は無視されます。

例題

```
<!--4DINCLUDE subpage.html-->
<!--4DINCLUDE folder/subpage.html-->
<!--4DINCLUDE ../folder/subpage.html-->
```

4DIF、4DELSE、4DENDIF

シンタックス：<!--4DIF 判定式--> <!--4DELSE--> <!--4DENDIF-->

<!--4DIF 判定式-->コメントを、<!--4DELSE-->（オプション）および<!--4DENDIF-->コメントとともに使用すると、条件付きでHTMLコードを実行できるようになります。

引数＜判定式＞には、ブール値を返す有効な任意の4D表記を納めます。4D表記は括弧内に記載し、4Dの構文ルールに従わなければなりません。

<!--4DIF 判定式-->から<!--4DENDIF-->までのブロックは、複数のレベルで入れ子状態にして指定できます。4Dと同様に、それぞれの<!--4DIF 判定式-->は<!--4DENDIF-->と一対になっていなければなりません。

インタープリタ上のエラーが発生すると、<!--4DIF-->と<!--4DENDIF-->の間に入れられる内容の代わりに、“<!--4DIF 判定式-->:ブール式が必要です”というテキストが挿入されます。

同様に、<!--4DIF-->と<!--4DENDIF-->の数が合わない場合、<!--4DIF -->と<!--4DENDIF-->の間に入れられる内容の代わりに、“<!--4DIF 判定式-->:4DENDIFが必要です”というテキストが挿入されます。

例題

この例題コードは、スタティックなHTMLページに挿入されており、“vname#""”という判定式の結果に応じて異なるラベルを表示します。

```
<BODY>
...
<!--4DIF (vname#"")-->
<!--4DVAR vname--> で始まる名前
<!--4DELSE-->
名前が見つかりません。
<!--4DENDIF-->
...
</BODY>
```

4DLOOPと4DENDLOOP

シンタックス：<!--4DLOOP 条件式--> <!--4DENDLOOP-->

このコメントを使用して、条件を満たすかぎりHTMLコードの一部を繰り返すことができます。繰り返される部分は、<!--4DLOOP-->と<!--4DENDLOOP-->で指定されます。

<!--4DLOOP 条件式-->から<!--4DENDLOOP-->までのブロックは、入れ子状態にして指定できます。4Dと同様に、それぞれの<!--4DLOOP 条件式-->は<!--4DENDLOOP-->と一対になっていなければなりません。

条件式には3通りあります。

■ <!--4DLOOP [テーブル]-->

このシンタックスでは、カレントプロセス内のテーブルのカレントセレクションの各レコードをループします。2つのコメントの間に挟まれたHTMLコード部分が、カレントセレクションのレコード毎に繰り返されます。

注：4DLOOPタグをテーブルと共に使用すると、レコードはリードオンリーモードでロードされます。

以下のHTMLコードは、

```
<!--4DLOOP [People]-->
<!--4DVAR [People]Name--> <!--4DVAR [People]Surname--><BR>
<!--4DENDLOOP-->
```

... 4D言語で記述すると以下のようになります。

```
FIRST RECORD([People])
While(Not(End selection([People])))
    ...
    NEXT RECORD([People])
End while
```

■ <!--4DLOOP 配列-->

このシンタックスでは、各配列項目をループします。配列のカレントアイテム番号はHTMLコード部分が繰り返されるたびに増加します。

注：このシンタックスは、2次元配列には使用できません。2次元配列の場合には、ネストしたループをメソッドに組み合わせるとよいでしょう。

以下のHTMLコードの例は、

```
<!--4DLOOP arr_names-->
<!--4DVAR arr_names{arr_names}--><BR>
<!--4DENDLOOP-->
```

... 4D言語で記述すると以下のようになります。

```
For ($Elem;1;Size of array(arr_names))
    arr_names:=$Elem
    ...
End for
```

■ <!--4DLOOP method-->

このシンタックスは、メソッドが“True（真）”を返す限りループを行います。メソッドは倍長整数タイプの引数を使用します。まず初期化できるように（必要な場合）値0を使用してメソッドが呼び出され、次に値1、そして2、3というように、メソッドが“True”を返す限り呼び出されます。

セキュリティ上の理由から、この初期化ステージ（引数として0を使用してメソッドを実行）の直前に、**On Web Authentication**データベースメソッドを1度呼び出すことができます。認証がOKであれば、初期化ステージが進められます。

注：4DLOOPタグを用いたメソッドの実行は、メソッドプロパティで指定された「4DACTIONで利用可能」オプションの値に影響されません。

警告：コンパイルする場合は、C_BOOLEAN($0)およびC_LONGINT($1)をメソッド内で必ず宣言してください。

例題

以下のHTMLコードの例は、

```
<!--4DLOOP my_method-->
<!--4DVAR var--> <BR>
<!--4DENDLOOP-->
```

... 4D言語で記述すると以下のようになります。

```
If(AuthenticationWebOK)
    If(my_method(0))
        $counter:=1
        While(my_method($counter))
            ...
            $counter:=$counter+1
        End while
    End if
End if
```

メソッドmy_methodは次の通りです。

```
C_LONGINT($1)
C_BOOLEAN($0)
If($1=0)
    `初期化
    $0:=True
Else
    If($1<50)
        ...
        var:= ...
        $0:=True
    Else
        $0:=False `ループ中止
    End if
```

End if

インタープリタ上のエラーが発生すると、<!--4DLOOP -->と<!--4DENDLOOP-->の間に入れられる内容の代わりに、“<!--4DLOOP 条件式-->:説明”というテキストが挿入されます。

説明として、以下のメッセージが表示されます。

■ 予期しない式のタイプです（標準エラー）。

■ テーブル名が正しくありません（テーブル名に関するエラー）。

■ 配列が必要です（変数が配列ではない、または 2 次元配列である）。

■ メソッドが存在しません。

■ シンタックスエラー（メソッドの実行中）。

■ アクセスエラー（テーブルやメソッドへアクセスするための十分なアクセス権がない）。

■ 4DENDLOOPが必要です（<!--4DENDLOOP-->と<!--4DLOOP -->の数が一致しない）。

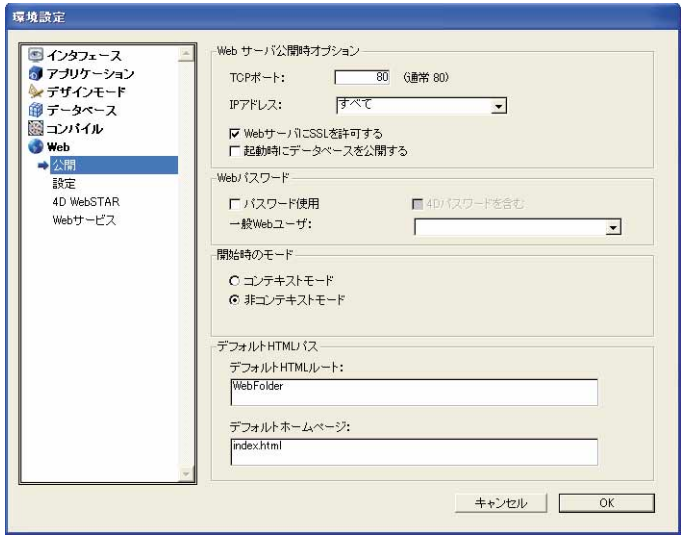
参照

Webサービス：HTMLとJavaScriptのカプセル化、Webサービス：非コンテキストモード、Webサービス：特殊なURLとフォームアクション

Web サーバセッテイング

データベースの「環境設定」の「Web」テーマで指定したパラメータを使用して、4D Webサーバの運用設定を行うことができます。この節では、「Web」テーマ内の公開、設定、および4D WebSTARページについて説明します。

「公開」ページ



デフォルトでは、4Dは通常のWeb TCPポートにWebデータベースを公開します。このポートはポート80です。ポート80が他のWebサービスで既に使用されている場合には、データベース用に4Dが使用するTCPポートを変更する必要があります。TCPポートを変更すると、そのマシンのルートユーザでなくてもMacOS X上で4D Webサーバを開始することができます（「Webサーバのシステム設定と接続管理」の節を参照）。変更するには、「TCPポート」入力エリアに移動し、適切な値（同一のマシン上で稼働している別のTCP／IPサービスで使用されていないTCPポート番号）を指定します。

注：0を指定すると、4DはデフォルトのTCPポート番号80を使用します。

Webブラウザでは、デフォルト以外のTCPポート番号の場合、この番号をWebデータベースへの接続用に入力するアドレスに指定する必要があります。そのアドレスにはコロンとポート番号で構成される接尾辞を指定します。例えば、TCPポート番号8080を使用している場合には、「192.168.0.99:8080」と指定します。

警告：デフォルトの80以外のTCPポート番号を使用する場合には、同時に使用する予定の他のサービス用のデフォルトのポート番号は使用しないように注意してください。例えば、Webサーバマシンで FTP プロトコルも使用する予定の場合には（その影響がわからない限り）、TCP ポートの20と21は使用しないでください。これらのポートはFTP プロトコルのデフォルトポートです。443のポートはTCP ポートでSSL接続に使用します。デフォルトのTCPポート番号とプロトコルの詳細については、TCP／IP プロトコルの解説書でRFC 1700標準の割り当て番号表を参照してください。256未満のポート番号は、既知のサービス用に予約されており、256 から1024までのポート番号はUNIX プラットフォームから提供される特定のサービス用に予約されています。最大限の安全対策として、ポート番号には2000番台や3000番台など、上記の値を超える番号を指定してください。

HTML リクエストの IP アドレスを定義する

WebサーバがHTTPリクエストを受信するIPアドレスを定義することができます。

デフォルトで、サーバはすべてのIPアドレス（あらゆるIPアドレスオプション）に応答します。

ドロップダウンリストには、そのマシン上で利用できるすべてのIPアドレスが自動的に一覧表示されます。特定のアドレスを選択すると、サーバはこのアドレスに送信されるリクエストにのみ応答します

この設定は、複数のTCP/IPアドレスでマシンに置かれる4D Webサーバのためにあります。例えば、ほとんどのインターネットホストプロバイダの場合です。

マルチホーミングしている、そのようなシステムを実装することは、Webサーバマシンの上で特定の構成を必要とします。

セカンダリ IP アドレスのインストール

マルチホーミングシステムの導入には、OSに応じた特別な設定が必要になります。

Macintosh 上の設定

▼ Macintosh上でマルチホーミングの設定をするには

1. この機能を使用するには、バージョン1.3以降のOpen Transportを使用する。

2.「コントロールパネル」の「TCP/IP」を開く。

3.「設定方法：」のポップアップメニューから「手入力」を選択する。

4. テキストファイルを作成し、"IP Secondary Addresses"という名前を付け、システムフォルダ内の「初期設定」フォルダに入れる。

IP Secondary Addressesの各行にシステムで使用するセカンダリIPアドレス用のIPアドレス、サブネットマスクおよびルータアドレスを記述します。

WindowsNT、Windows2000上の設定

▼ WindowsNTまたはWindows2000上でマルチホーミングの設定をするには

1. 次のコマンドシーケンスを選択する。

■ Windows NT：スタートメニュー＞設定＞コントロールパネル＞ネットワーク＞プロトコルタブ＞TCP/IPプロトコル＞プロパティボタン＞詳細ボタン

■ Windows 2000：スタートメニュー>設定>ネットワークとダイヤルアップ接続>ローカルエリア接続>プロパティボタン > インターネットプロトコル（TCP/IP）>プロパティボタン>詳細設定...ボタン

「TCP/IP詳細設定」ダイアログが表示されます。

■ Windows XP：スタートメニュー > コントロールパネル>ネットワークとインターネット接続>ネットワーク接続>ローカルエリア接続（プロパティ）>インターネットプロトコル（TCP/IP）>プロパティボタン > 詳細設定...ボタン

「TCP/IP詳細設定」ダイアログが表示されます。

2.「IPアドレス」エリアの「追加」ボタンをクリックし、追加するIPアドレスを入力する。

最高で5種類のIPアドレスを指定することができます。この作業にはネットワーク管理者のサポートが必要になることがあります。より詳しい情報は、Windowsドキュメントをご覧ください。

Webサーバに SSL を許可する

Webサーバで暗号化モードでの接続を許可するかどうかを表わします。このオプションについては、「SSLプロトコルの使用」の節で説明しています。

起動時にデータベースを公開する

4Dアプリケーションの起動時に、Webサーバを開始するかどうかを表わします。このオプションについては、「Webサービス：システム設定」の節で説明しています。

「Webパスワード」エリア

パスワードを使用して、Webサイトのアクセス保護に関する設定を行います。このオプションについては、「接続セキュリティ」の節で説明しています。

開始時のモード

Webサーバが開始するモードを指定することができます。このオプションについては、「コンテキストモードの使用」の節で説明しています。

一時的なコンテキストを再利用する（4D Clientでのみ表示される）

前回のWebリクエストの処理で作成されたWebプロセスを再利用することにより、4D ClientのWebサーバ操作を最適化することができます。実際のところ、4D ClientのWebサーバには、各Webリクエスト処理のために特定のWebプロセスが必要になります。つまり、必要に応じてこのプロセスが4D Serverマシンに接続し、データとデータベースエンジンにアクセスします。この後、独自の変数やセレクション等を使用して一時的なコンテキストが生成されます。リクエストの処理が終わると、このプロセスは終了します。

「一時的なコンテキストを再利用する」オプションが選択されると、4Dは4D Client上で作成された特定のWebプロセスを保持しておき、次回のリクエストでこのプロセスを再利用します。プロセス作成段階が省かれることにより、Webサーバのパフォーマンスが向上します。

その代わりこの場合は、誤った結果を取得しないように、4Dメソッドを使用して意識的に各変数を初期化しなくてはなりません。同様に、前回のリクエスト中に定義されたカレントセレクションやレコードを消去する必要があります。

デフォルトHTMLルート

Webサイトの各ファイルのデフォルト位置を定義し、ディスク上のこれより上位の階層にあるファイルへはアクセスできないことを示します。このオプションについては、「接続セキュリティ」の節で説明しています。

デフォルトのホームページを定義する

そのWebセッションに対して指定されたモード（コンテキストまたは非コンテキスト）に関係なく、データベースに接続するすべてのブラウザに対してデフォルトのホームページを定義することができます。

デフォルトでは、Webサーバの初回起動時に、4Dは“index.html”という名前のホームページを作成し、それをHTMLルートフォルダに納めます。この設定を変更しない場合、Webサーバに接続している任意のブラウザは、次のようなページを取得します。

デフォルトホームページを変更するには、データベースのルートフォルダ内のホームページを独自の“index.html”ページで置き換えるか、あるいは「デフォルトホームページ」入力エリアに定義したいページの相対アクセスパスを入力します。

アクセスパスは、デフォルトのHTMLルートフォルダに対して相対的に設定しなくてはなりません。

ユーザのデータベースのマルチプラットフォームの互換性を確実にするために、4D Webサーバは、アクセスパスを定義するのに特定の表記法を使用します。

- フォルダは"/"で区切る
- アクセスパスの最後は"/"で終了しない
- 上の階層のフォルダを定義する場合は、".."（2つのピリオド）"をフォルダの前に入力する
- アクセスパスは、"/"で始めない

例えば、デフォルトホームページを"Web"フォルダ"（それ自体がそのデータベースのデフォルトHTMLルートフォルダ内に配置されている）の中の"MyHome.htm"にしたい場合、"Web/MyHome.htm"と入力します。

注：Webプロセスごとに、デフォルトのホームページを設定するには、「SET HOME PAGE」を使用することもできます。

デフォルトのカスタムホームページを指定しない場合、Webサーバの動作は開始時のモードによって異なります。

- Webサーバがコンテキストモード（デフォルトで）で開始する場合、4Dの前バージョンの場合のように、カレントのメニューバー（デフォルトで、メニューバー#1）が送られます。
- Webサーバが非コンテキストモード（標準モード）上で開始する場合、**On Web Connection**データベースメソッドがコールされます。メソッドを用いてリクエストを処理するかどうかはユーザ次第です。

設定ページ

4D Webサーバのキャッシュを使用し、リクエストに応じてスタティックページやGIFイメージ、JPEG画像（<128 kb）およびスタイルシート（.cssファイル）をメモリにロードすることができます。

スタティックなページを送る時、キャッシュを使用すると、かなりWebサーバパフォーマンスを向上させます。

キャッシュは、すべてのWebプロセスで共有されます。キャッシュのサイズは、「環境設定」で設定することができます。デフォルトとして、スタティックページにはキャッシュを使用できません（サイズは0です）。キャッシュを使用可能にするには、「ページキャッシュサイズ」エリアに値を入力してください（KB単位）。

セットした値はホストマシンの処理能力だけでなく、ユーザのWebサイトのスタティックなページの数とサイズに依存します。

注：ユーザのWebデータベースを使用する間、ユーザはWEB CACHE STATISTICSルーチンを使用することによってキャッシュのパフォーマンスをチェックすることができます。例えば、ユーザが使用するキャッシュの率が約100%であると気がついた場合、それに割り当てられたサイズを増やすことを考えることがあります。4DSTATS、そして、4DHTMLSTATS URLは、さらにキャッシュの状態に関する情報を得ることができます。詳しくは、「Web サイトに関する情報」を参照してください。

一度キャッシュが割込み可能であると、4D Webサーバはキャッシュの中で最初のブラウザによって要求されるページを探します。それがページを見つけた場合、すぐにそれを送ります。そうでない場合、4Dはディスクからのページをロードして、キャッシュでそれを設定します。キャッシュがいっぱいで、追加のスペースが必要な場合、最も少なく要求されたものの中で、4Dは最も古いページを「アンロードします」。

キャッシュのクリア

いつでも、ユーザはページのキャッシュとそれが含むイメージをクリアすることができます（例えば、ユーザがスタティックなページを変更して、キャッシュでそれを再ロードしたい場合）。そのために、「キャッシュクリア」ボタンをクリックしなければなりません。キャッシュは、その時すぐにクリアされます。

Web プロセスタイムアウト

Web接続プロセスのタイムアウトを指定することができます（コンテキストモードのみ）。このオプションについては、「コンテキストモードの使用」の節で説明しています。

Web プロセスの最大数を定義する

このオプションを使用して、サーバ上で同時に開始できる任意のタイプ（コンテキストモード、非コンテキストモード、またはプロセスの“再利用”に属するタイプ）の「最大同時Webプロセス」の上限数を正確に指定することができます。このパラメータを使用すると、リクエストが多すぎたり、コンテキスト作成要求が多すぎるために、4D Serverが限界に達してしまう危険性を回避することができます。

デフォルトでは、この値は32,000ですが、10から32,000までの値を自由に設定できます。

同時Webプロセスの最大数（マイナス1）に達すると、4Dは新しくプロセスを作成しなくなり、新しい各リクエストに対して「サーバは使用できません」（ステータスHTTP 503 - Service Unavailable）というメッセージを送信します。

適切な値の決定方法は？

理論上、Webプロセスの最大数は次の計算式の結果になります。

使用可能メモリ／Webプロセスのスタックサイズ

別の解決策は、ランタイムエクスプローラに表示されるWebプロセス情報を示す方法です。つまり現在のWebプロセス数およびWebサーバの開始以降に達した最大数が示されている情報です。

Webプロセスの再利用

Webプロセスの“再利用”により、非コンテキストモードにおけるWebサーバの反応度を上げることができます。この保存分のサイズは、再利用されるプロセスの最小数（デフォルトでは0）および最大数（デフォルトでは10）で決まります。これらのプロセスは、**SET DATABASE PARAMETER**コマンドを使用して変更することができます。Webプロセスの最大数が一度変更されると、この最大数が“再利用”の上限数より小さい場合、上限数はWebプロセスの最大数にまで減らされます。また、Webプロセスの最大数は、**SET DATABASE PARAMETER**コマンドを使用して定義することもできます。

拡張ASCII文字を直接送る

デフォルトとして、4D Webサーバは、ダイナミックWebページやスタティックWebページを送信する前に、HTML規格に従ってページ内にある拡張ASCII文字を変換します。この後、これらの文字はブラウザにより翻訳されます。

拡張ASCII文字をHTMLエンティティに変換せずに、“現状のまま”送信されるようにWebサーバを設定することができます。このオプションにより、外国語ＯＳ（特に日本のOS）上での処理速度が向上します。

これを行うには、「拡張文字を直接送信」オプションを選択します。

文字セット

「スタンダードセット」ドロップダウンリストを使用し、4D Webサーバで使用する一連の文字セットを指定することができます。

また、入力データと出力データ双方に対してASCII文字変換テーブル（Webフィルタ）を変更すると、カスタマイズした文字セットを指定することもできます。

これを行うには、「ユーザ定義」ラジオボタンを選択します。このパラメータは、“x-user-defined”文字セットを選択した場合に相当します。

すると、「入力フィルタを編集」および「出力フィルタを編集」項目に関連するボタンが有効になります。入力フィルタはブラウザから4D Webサーバへ送信される文字を解釈し、出力フィルタは4D Webサーバからブラウザへ送信される文字を解釈します。

ユーザが変更したいフィルタと一致するボタンをクリックします。

「入力フィルタ編集」は4D Webサーバにブラウザによって送られる文字を翻訳し、「出力フィルタ編集」はブラウザに4D Webサーバで送る文字を翻訳します。

以下のダイアログボックスが、表示されます。

スクロールエリアでは、フィルタしたいマックキャラクタを探して、クリックします。「ASCII Code」入力エリアでは、文字の新しいASCIIコードを入力します。フィルタしたいすべての文字のために、この操作を繰り返します。

フィルタを保存するには「保存...」ボタンをクリックします。「読み込み...」ボタンを使用し、続けてフィルタをロードすることができます。

Webの入力フィルタや出力フィルタを有効にするには、「テーブル更新」ボタンをクリックします。

ブラケットの代りに4DVARコメントを使用する

このオプションを使用して、スタティックページ上で4D変数を挿入する際に使用するコメントを定義することができます。

■ このオプションを選択した場合（デフォルト値）、使用されるシンタックスは標準のHTMLコメントです。

<！--4DVAR 変数名-->

（4DVARと変数名の間にスペースキャラクタを挿入しなくてはなりません）。

■ オプションを選択しない場合、使用されるシンタックスは角カッコによるコメントです（[MYVAR]）。これは以前のバージョンの4D Webサーバで使用された独自のソリューションです。

新しいコンテキスト参照モードを使用

このオプションを選択すると（デフォルト値）、コンテキストモードにおいてWebサーバは、ブラウザに送信したドキュメントの基本のURLに現在のコンテキスト番号を設定します。

以前のシステムでは（オプションは未選択）、4D Webサーバはページの各要素をブラウザへ送信するたびにコンテキスト番号を送信します。この方法では処理速度が低下してしまいます。

互換性上の理由から、このオプションの選択を解除することができます。ただし、このオプションを変更した後は、新しい設定を有効にするためにデータベースを再起動しなくてはならない点に留意してください。

Javascriptを入力制御に使用する

このオプションを選択すると、コンテキストモードで自動的なJavascriptを使用することにより、データ入力制御のある部分をブラウザ側で処理することができます。

ブラウザで、それらが適用されることができるデータ入力制御とデータ型（フィールド、または、変数）は、次の通りです。

■ 最小値（数値）

■ 最大値（数値）

■ 必須入力

生成されたJavascriptは小さいサイズですが、ユーザをデータ入力（それは、まだ4Dの責任です）を受けるのを妨げることなく警告ダイアログボックスを表示します。実際に、データ入力エリアが適当でない値が入力された場合、ユーザがボタン（OK、Cancel、その他）をクリックすると、警告メッセージをブラウザに表示します。

ファイルにリクエストを保存する（logweb.txt）

このオプションにより、Webサーバに送信したリクエストのログをCLFテキストファイル形式で生成することができます。このオプションについては、「Webサイトに関する情報」の節で説明しています。

「4D WebSTAR」ページ

4D Connect を経由した 4D WebSTAR の接続を許可する

このオプションは、4D WebSTAR プラグインの 4D Web サーバへの接続を許可（チェック）、または禁止（チェックなし）する目的のために設定されています。

4D WebSTAR は、4D WebSTAR の Web サーバのためのプラグインで、4D の Web サーバとの通信を行います。

セキュリティ上の理由から、デフォルトでは「4D Connect を経由した 4D WebSTAR の接続を許可する」オプションが選択されていません。お使いの Web の環境設定に応じ、4D 社では以下のような設定をお勧めします。

■ お使いの 4D Web サーバが、4D WebSTAR プラグインを使用して 4D WebSTAR サーバに接続していない場合、このオプションはチェックされていない状態のままにしてください。

■ お使いの4D Webサーバが、4D WebSTAR プラグインを使用して4D WebSTARサーバに接続している場合、接続が正常に行われるよう、このオプションをチェックしてください。

この設定の場合、4D Webサーバをファイアーウォールの下で稼働させ、そのファイアーウォールを利用して4Dへのリクエストをフィルタリングすることをお勧めします。

参照

SET DATABASE PARAMETER、 SET HOME PAGE、「接続セキュリティ」、「非コンテキストモード」

Webサイトに関する情報

4DではWebサイトに関する情報を得ることができます。

■ 特定のURL（/4DSTATS、/4DHTMLSTATS、/4DCACHECLEAR、/4D WEB TEST）を使ってサイトの管理ができます。

■ すべてのリクエストのログを生成することができます。

■ ランタイムエクスプローラーウインドウのウォッチページ中でWebサーバに関する情報を取得することができます。

Webサーバ管理URL

4D Webサーバは、4つの特定のURL（/4DSTATS、4DHTMLSTATS、/4DCACHECLEAR、/4D WEB TEST）を受け入れます。

/4DSTATS、4DHTMLSTATS、/4DCACHECLEARのこれらのURLは、4Dパスワードシステムが設定されている場合はデザイナーおよび管理者のみ使用できます。もし、4Dパスワードシステムが設定されていない場合は、すべてのユーザが使用可能になります。

/4D WEB TESTは常に使用可能です。

/4DSTATS

/4DSTATSのURLは、純粋なテキストフォームで下記の情報を返します。

■ ヒット数（低レベル接続）

■ 作成されたコンテキストの数

■ 作成されなかったコンテキストの数

■ パスワードエラーとなった数

■ キャッシュ内に保存されているページの数

■ キャッシュの使用率（%）

■ スタティックホームページのキャッシュ内に、保存されてるページおよびJPEGまたはGIFファイルのリスト（*）

(*) スタティックホームページおよびピクチャに関するより詳しい情報は、「Webサーバセッティング」を参照してください。

この情報は、サーバ機能をチェックし、段階的に各パラメータを最適化することを可能にします。

注：WEB CACHE STATISTICSコマンドは、キャッシュがスタティックホームページにどのように使用されているのかに関する情報を得られるようにします。

/4DHTMLSTATS

/4DHTMLSTATSのURLも、/4DSTATSのURLと同じ情報を純粋なテキストフォームで返します。違いは、最後のフィールドでキャッシュ内に存在するHTMLページのリストだけが返されることです(キャッシュされているJPEGとGIFファイルのリストは含まない)。

/4DCACHECLEAR

/4DCACHECLEARのURLは、スタティックホームページとイメージのキャッシュを即座にクリアします。したがって、変更されたページを「強制的」に更新させることができます。

/4DWEBTEST

/4DWEBTESTのURLはWebサーバのステータスをチェックするように設計されています。このURLが呼び出されると、4Dは次のHTTPフィールドとともにテキストファイルを返します。

Date: current date at the RFC 822 format

例："Date: Wed, 26 Jan 2000 13:12:50 GMT"

Server: 4D WebStar_D/internal version number

例："4D WebStar_D/7.0"

User-Agent: name and version @ IP client address

例："Mozilla/4.08 (Macintosh; I; PPC, Nav) @ 192.193.00.00"

接続ログファイル

4Dでは、リクエストのログを取ることができます。ログは、ストラクチャファイルと同じ階層に、"weblog.txt"ファイルとして自動的に作成されます。このファイルは、ほとんどのWebサイト分析ツールで認識できる、CLF（Common Log File）フォーマットまたはNCSAフォーマットになります。

“weblog.txt”ファイルは次の場所へ自動的に配置されます。

■ 4th Dimensionおよび4D Serverでは、データベースストラクチャファイルと同じ階層。

■ 4D Clientでは、アプリケーションの「.exe」ファイル（Windows）またはソフトウェアパッケージ（MacOS）と同じ階層。

ファイルの各行は、以下のようなリクエストを表わします。

host rfc931 user[DD/MMM/YYYY:HH:MM:SS] "request" state length

各フィールドは、スペースで分離され、各行はCR/LF（文字コード13/文字コード10）で終わります。

■ host：クライアントのIPアドレス（例192.100.100.10）

■ rfc931：4Dでは生成しない情報で、常に「-（マイナス記号）」です。

■ user：認証されているユーザ名または「-（マイナス記号）」。ユーザ名にスペースがあると「_（下線）」に置き換えられます。

■ DD：日付、MMM：月の名前の3文字の略号（Jan、Feb、...）、YYYY：年HH :時間、MM：分、SS：秒

日付と時間はサーバマシン上の値です。

■ request：クライアントから来たリクエスト（例えば、GET/index.htmHTTP/1.0）

■ state：サーバからの返答

■ length：返答データのサイズ（HTTPヘッダを除く）または0

注：性能上の理由から、ディスクに書き込まれる前にサイズ1KBのパケットとしてメモリ上に保存され、5秒間リクエストが発生しなければディスクに書き込まれます。

stateとして取り得る値は下記の通りです。

200:OK
204:No contents
302:Redirection
304:not modfied
400:Incorrect request
401:Authentication required

404:Not found
500:Internal error

▼リクエストログで生成される行の例

■ 192.100.100.10 - - [25/Jan/1998:12:54:06] "GET /index.htm" 200 6524

アドレスが192.100.100.10のWebクライアントが認証されなかった。ページ"index.htm"が要求され、送信した（6,524バイト）。

■ 192.100.101.25 - - [25/Jan/1998:12:54:09] "GET /123456.htm" 404 125

アドレスが192.100.101.25のWebクライアントが認証されなかった。ページ"123456.htm"を要求されたが見つけられなかった（4Dは125バイトのメッセージを送りました）。

■ 192.100.101.31 - - [25/Jan/1998:12:54:10] "GET /secret.htm" 401 0

アドレスが192.100.101.31のWebクライアントが認証されなかった。ページ"secret.htm"を要求され、サーバは認証要求をした。

■ 192.100.101.31 - ZZZZ [25/Jan/1998:12:54:11] "GET /secret.htm" 401 0

アドレスが192.100.101.31のWebクライアントが"ZZZZ"として認証された。ページ"secret.htm"を要求され、ユーザ名が不明である。

■ 192.100.101.31 - 4D [25/Jan/1998:12:54:12] "GET /secret.htm" 200 2543

アドレスが192.100.101.31のWebクライアントが"4D"として認証された。ページ"secret.htm"を要求され、送信した（2,543バイト）。

警告：ログファイルはスプレッドシートまたは直接4Dへ読み込み可能です。しかし、データを読み込む前に必ずWebサーバを停止させなければなりません。

デフォルトではリクエストのログファイルは生成されません。すべてのWebリクエストのログファイルの作成を要求するには、データベースの「環境設定」で「Web」テーマの「設定」ページにある「ファイルにリクエストを保存する（logweb.txt）」オプションを選択しなくてはなりません。

ランタイムエクスプローラの情報

ランタイムエクスプローラの「ウォッチ」ページ（“情報”見出し）には、Webサーバに関連する3種類の情報が表示されます。

■ **Webキャッシュ使用率：** Webキャッシュ内にあるページ数とともにその使用率を示します。この情報は、Webサーバがアクティブで、かつキャッシュサイズが0より大きい場合にのみ表示されます。

■ **Webサーバ経過時間：** Webサーバの継続使用時間（時:分:秒形式）を示します。この情報は、Webサーバがアクティブである場合にのみ表示されます。

■ **Webヒット数：** Webサーバの開始以降に受信したHTTPリクエストの総数、および1秒毎の瞬間リクエスト数（2 回のランタイムエクスプローラ更新の間で計測）を示します。この情報は、Webサーバがアクティブである場合にのみ表示されます。

注：ランタイムエクスプローラに関する詳細は、『4Dデザインリファレンスマニュアル』を参照してください。

参照

WEB CACHE STATISTICS、Webサービス：Webサーバセッティング

コンテキストモードの使用

4D Webサーバは、非コンテキストモード（標準モード）とコンテキストモードという2種類のモードで動作することができます。この節では、これら2つのモードを説明し、更にコンテキストモードの特性について詳しく述べています。

警告：コンテキストモードは4th Dimensionならびに4D Serverでのみ使用することができます。4D ClientのWebサーバではこのモードがサポートされていません。

注：「Webサーバ入門」の節には、コンテキストモードでデータベースを公開するための例題が提供されています。

非コンテキストモードとコンテキストモード

バージョン2003の4th Dimensionより、デフォルトとして4D Webサーバでは非コンテキストモード（非接続モード）が使用されます。このモードの4D Webサーバの動作は、標準のWebサーバの動作と変わりません。つまり、ブラウザからのHTTPリクエストを受信すると（URL、ポストされたフォーム等）、サーバがリクエストを処理し、必要に応じて応答を返します（例えば、Webページを送信する）。この後、サーバとブラウザの間には特定の接続が維持されません。

非コンテキストモードにおいて、Webサーバはスタティックページまたはセミダイナミックページを送信することができます。セミダイナミックページを使用すると、特別な4Dタグを用いてデータベースのデータへのアクセスやあらゆるタイプの処理の実行が可能になります。この4Dタグはページを送信した時に評価されます。セミダイナミックページを使用することにより、コンテンツのすべて、またはその一部が4Dによる処理を元にしたWebページの作成、管理、送信を行うことができます。通常は、非コンテキストモードによりWebサイト開発における大半の要求に応えることができます。

コンテキストモードでWebブラウザへ接続するとコンテキストが作成され、このコンテキスト内にカレントセレクションや変数等が配置されます。ある意味では、各ブラウザは「カスタム」モードでデータベースに接続している4D Clientとして扱われます。コンテキストは特定のWeb接続プロセスによって処理されます。

このモードを使用すると、Webページを作成しなくても、簡単に4Dデータベースを Web上に公開することができます。つまり、4Dはデータベースのメニューバーやフォームを自動的にHTMLへ変換し、これを元に作成されたダイナミックページを管理してブラウザ側へ送信します。コンテキストモードでも、セミダイナミックページやスタティックページを送信することができます。また、Web上で表示されるページに機能を追加するために、HTMLコードやJavascriptを4Dフォームに挿入することも可能です。

さらに、このモードで4Dはデータへの同時アクセスを自動的に処理します。ブラウザまたは4D Clientマシンがレコードをロードすると、それがブラウザや他の4D Clientマシンであっても、4Dは他のユーザに対してこのレコードを透過的にロックします。また、4Dでは、4th Dimensionや4D Clientと同じ方法で、トランザクション中にWebブラウザを使用したデータ入力を行うことができます。このシステムにより、4D Webサーバ側でブラウザの動作を制御したり、データの整合性を保証できるようになります。

公開が容易である代わりに、コンテキストモードにはいくつかの制約があります。

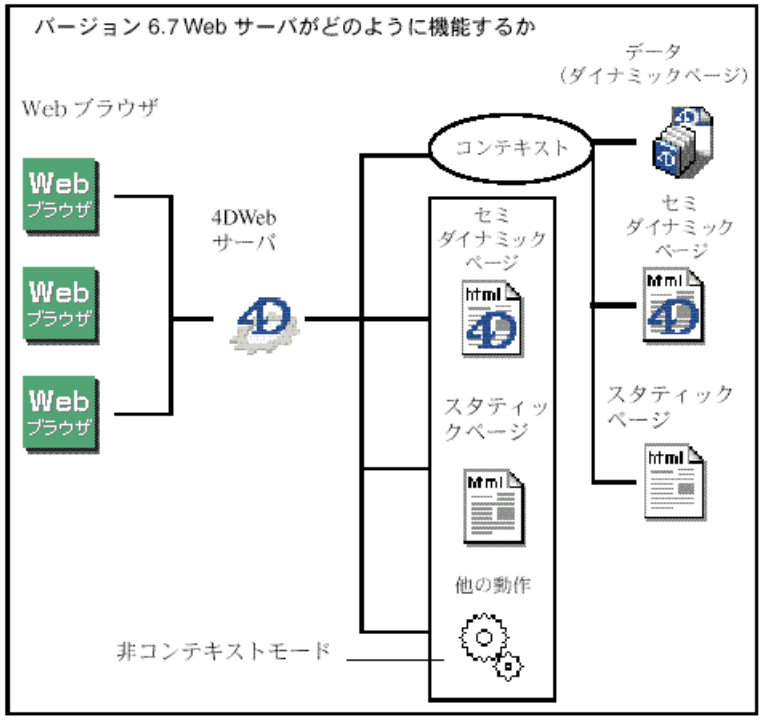
■ Webブラウザでは、あるWebページから別のページへ、あるサイトから別のサイトへと“サーフィン”することができます。クライアント／サーバタイプのデータベースを使用する場合には、データベーストランザクションのロジックを守るため、このナビゲーションを制御しなくてはなりません。ユーザが実行したレコードへの各入力は、不確定な状態のままにならないよう、必ず確定するかキャンセルしなくてはなりません。

4D Webサーバエンジンには、データベースセッションとコンテキストを管理する自動メカニズムが導入されています。これらのメカニズムにより、一部の標準的なブラウザ機能（再読み込み、戻る等。次の節を参照）が使用できなくなります。

■ コンテキスト管理を行う接続プロセスは、データベースの「環境設定」で指定したブラウザのタイムアウトに達するまで有効になります。例えば、ブラウザ側で一時的にそのサイトから抜けると、そのコンテキストは“無駄”になります。

これらの制約が意味するのは、コンテキストモードはどちらかと言えば、イントラネットや特定のインターネットアプリケーションのフレームワーク内での使用を目的としているということです。

4D Webサーバの機能方法を要約すると次の図のようになります。



モードの選択

4D Webサーバの動作モードの選択は、次のように行われます。

■ サーバの開始時に、データベースの「環境設定」の「開始時のモード」オプションを使用する。

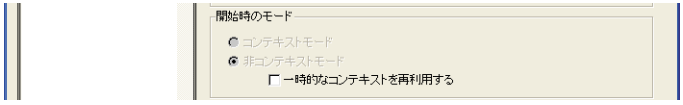
■ Webサーバの使用中に、送信されたURLや実行されたコマンドに従う。

実際には、いくつかの特別なURLや特定の4Dコマンドを使用して、モードを変更することができます。原則として、URLや4Dコマンドによりモードが変更されるまでの間は、現在のモードが使用されます。

■ 開始時にコンテキストモードを指定する。

デフォルトでは、Webサーバは非コンテキストモードで開始します。しかし、Webサーバを直接コンテキストモードで開始することができます。つまり、ユーザがデータベースに接続する際に、コンテキストが自動的に生成されます。

開始時に非コンテキストモードを指定するには、データベースの「環境設定」ダイアログで「Web」テーマの「設定」ページにある「コンテキストモード（永続的コンテキスト）」オプションを選択します。



■ モード変更を行うコマンドとURL

データベースの操作中に、次の要素を呼び出してモードを変更することができます。

■ 非コンテキストモードへの変更

SEND HTML BLOB：オプションの引数＜非コンテキスト＞に“True”を渡す。
SEND HTML TEXT：オプションの引数＜非コンテキスト＞に“True”を渡す。
SEND HTTP REDIRECT
/4DACTIONで始まるURL

■ コンテキストモードへの変更

/4DMETHOD/MyMethodで始まるURL
URL“/4DMETHOD/MyMethod”が送信されると、4D Webサーバは新しいコンテキストを作成し、次の処理を実行します。
・「Web Authentication」データベースメソッドを実行します（存在する場合）
・「On Web Connection」データベースメソッドを実行します（存在する場合）。この場合、 $1には“/（スラッシュ）”ではなく“/4DMETHOD/MyMethod”が代入されます。
・最後に、新しく作成したコンテキスト内でリクエストされたメソッドを実行します。

生成されたコンテキストの数を調べる

実行する動作によって、Webコンテキストを使用するWebプロセスもあれば、使用しないものもあります。

PROCESS PROPERTIESコマンドを使用すると、生成されたコンテキストの数を調べることができます。このコマンドは任意のWebプロセスに関して、コンテキストが使用されているか（-11：Web Process with Contex）または使用されていないか（-3：Web Process with no Context）を示す値を引数＜派生元＞に代入します。

「Web 接続」プロセス

「Web 接続」プロセスと Web セッション

ユーザの立場からすると、Webブラウザ側でのユーザのアクションがWebセッションを誘導します。

プログラム作成の立場からすると、「Web接続」プロセスがWebセッションを誘導するのであって、Webセッションが「Web接続」プロセスを誘導するわけではありません。Webブラウザは「Web接続」プロセスが送信したページを表示します。それは、以下のいずれかを行います。

■ 4Dコードを実行する、または

■ カレントWebページのブラウザからのサブミッションを待機する。

設計者の立場からすると、「Web接続」プロセスは実行のドメインが4th Dimensionまたは4D Serverである4Dプロセスとして見えるべきですが、そのユーザインタフェースは接続されたWebブラウザ上でエコーされます。

このようなことから、コンテキストモードのWebデータベースアプリケーションの設計を実行する場合には、この「Web接続」プロセスの二重性を常に考慮してください。以下の例を参照してください。

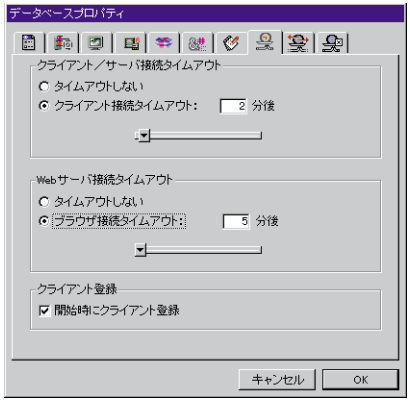
■ どんな種類のデータ入力でも、その間、メインメニューバーはブラウザのものであり、4Dのメニューバーではありません。フォーム内では、4Dメニューバーを頼りにしないでください。4DメニューバーはWebサーバマシン上にあり、Webブラウザマシン上にはありません。

■ Webブラウザで使用されるフォームを設計する場合には、4Dフォームの機能はHTMLのフォーム上で制限されていることを考慮してください（ただし、時には4Dの追加機能もあります）。4Dフォームの機能をすべて使用できるとは限りません（例：オブジェクトタイプやフォームイベント）。この件に関する詳細は、後述の「HTMLの自動変換」の節を参照してください。

■ プロセス間通信において、**CALL PROCESS**コマンドを「Web接続」プロセスに適用した場合には、現在アクティブなフォームはWebブラウザに表示されているため、何の動作も行いません。一方、「Web接続」プロセスは別の4Dプロセスに対して**CALL PROCESS**コマンドを発行できます。さらに、プロセス間通信は**GET PROCESS VARIABLE**コマンドおよび**SET PROCESS VARIABLE**コマンドを使用することで、双方向に実行することができます。これらのコマンドは、ユーザインタフェースを持つ必要がないからです。

非アクティブな Web プロセスのタイムアウト

前述したように、コンテキストモードにおける「Web接続」プロセスは4Dコードを実行しているか、またはブラウザ側で現在表示されているWebページのサブミッションを待機しているかのいずれかです。後者の場合には、Web接続プロセスは、「環境設定」ウインドウ（下図参照）で、または**SET WEB TIMEOUT**コマンドをプログラムで使用して設定された「Webプロセスタイムアウト」と同じ遅延時間だけ待機します。



「Webサーバ接続タイムアウト」設定の有効範囲は、データベースセッションです。

すべての「Web接続」プロセスは、タイムアウト時間の値に従います。値の設定が変更されると、プロセスは即座に影響を受けます。デフォルト値は、5分間です。

注：SET WEB TIMEOUT関数はWebプロセスごとにタイムアウト値を指定できます。

必要に応じてこのタイムアウトを長くする、あるいは短くすることができます。例えば、Webユーザがデータベースから提供されるページ内のHTMLリンクを経由して、他のWebサイトに移動することをアプリケーションが許可している場合には、タイムアウトを長くすることができます。タイムアウトを長くすることで、ユーザはデータベースへの接続がクローズされることなく、他のWebサイトをさらに長い時間ナビゲートできます。

警告：「Web接続」プロセスをプログラムで停止する方法はありません。長いタイムアウトを指定した場合には、Webユーザがかなり前にWeb接続での作業を終っていても、プロセスはその遅延を待つことになります。「タイムアウトなし」オプションを指定すると、「Web接続」プロセスはデータベースが終了した時にだけ停止します。しかし、Web接続プロセスはWebサーバが非コンテキストモードに切り替えられると同時に自動的にアボートされます。

Tips：「Webサーバ」プロセスとは違って、「Web接続」プロセスは「アボート」コマンドを使用してアボートすることができます（「プロセスページ」が表示されているときに「ランタイムエクスプローラ」で利用できます）。

「HTMLの自動変換」

この節では、コンテキストモードにおいて4th DimensionがデータベースをHTMLに変換する際、自動的に処理される要素やオブジェクト、ならびにメカニズムについて説明します。

HTMLサポート

メニューバー

■ 各メニュータイトルはテキストとしてのみ表示され、4Dメソッドに関連付けられたメニューコマンドは4Dメソッドへのリンクとして表示されます。自動アクションだけに関連付けられたメニューコマンドは、テキストとしてのみ表示されます。

■ Webブラウザ側でメニュー項目をクリックすると、「Web接続」プロセス側で関連する4Dメソッドの実行が開始されます。

注：メニューコマンドに対して「新規プロセス開始」プロパティが指定されている場合、関連するメソッドは、「4DMETHOD」URLを使用して、4D Webサーバにより新しいWeb接続プロセス内で実行されます。この場合、そのメニューのメソッドには「4DACTIONで利用可能」オプション（デフォルトでは新規データベースに対して未選択）が指定されていなくてはなりません。詳細については「接続セキュリティ」の節を参照してください。

■ ピクチャはブラウザのメニューの下に位置するメニューバーと結び付けられています。

HTMLの埋め込み

HTMLコード（またはJavascript）をフォーム内に埋め込むことにより、HTMLに変換された4Dフォームの内容をカスタマイズすることができます。この結果Webブラウザ側に表示されるフォームには、HTMLと4Dオブジェクトが混在します。

スタティックテキストオブジェクトを使用したHTMLページの挿入

例えば、“{page.HTM}”という文字列を含む4Dフォームのスタティックテキストオブジェクトは、4Dフォーム内にあるそのテキストオブジェクトの場所にHTMLドキュメント“page.HTM”を挿入します。この場合、ドキュメント全体（実際には<BODY>から</BODY>タグの間に含まれるものすべて）が挿入されます。既存のHTMLドキュメントを使用するか、あるいはランゲージを使用してディスクに保存するドキュメントを作成し、この後にそれを参照することもできます。

注：時には、バージョン6.0.xで作成された4Dフォームに HTMLドキュメント（{mypage.htm}）への参照が含まれている場合に、4D 2003ではこのフォームのHTMLへの変換が予期しない結果になることがあります。この場合、SET DATABASE PARAMETERコマンドを使用して、フォームの変換モードを変更することができます。

HTMLコードの挿入

4Dのテキスト変数の1桁目がASCIIコード1である場合、テキスト変数を使用してHTMLコードを4Dフォームに埋め込むことができます（例えば、vtHTML:=Character(1)+"...HTMLコード..."）。

このようにして、コードを挿入することができます。また、この場合はHTMLコードをメモリ上に作成することができます。

参照

「On Web Authentication」データベースメソッド、「On Web Connection」データベースメソッド、SET DATABASE PARAMETER

SSLプロトコルの使用

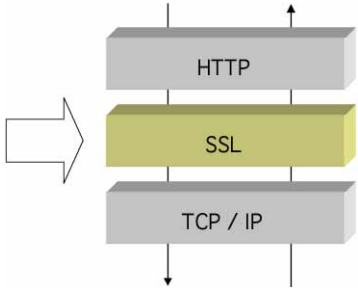
4D WebサーバではSSL（Secured Socket Layer）プロトコルを使用した暗号化モードでの通信が行えるようになりました。

SSLプロトコルの定義

SSLプロトコルは、2つのアプリケーション間、主にWebサーバとブラウザとのデータのやり取りを保護する目的で設計されました。このプロトコルは広く利用されており、また大部分のWebブラウザと互換性があります。

ネットワークレベルでは、SSLプロトコルはTCP/IPレイヤ（ローレベル）とHTTPハイレベルプロトコルの間に挿入されます。SSLは、主にHTTPと共に作業を行うように設計されています。

SSLを使用したネットワーク構成



注：SSLプロトコルは、標準の4D Serverのクライアント／サーバ接続を保護するためにも使用できます。詳細は、『4D Serverリファレンス』マニュアルの「クライアント／サーバ接続の暗号化」の節を参照してください。

SSLプロトコルは、送信者と受信者の認証を行い、やり取りする情報の秘匿性および整合性を保証する目的で作られています。

■ **認証：**送信者と受信者の身元を確認します。

■ **秘匿性：**送信データは暗号化されるため、第三者はメッセージを理解することができません。

■ **整合性：**受信したデータは、偶発的であれ作為的であれ、変更されることはありません。

SSLは、暗号化および復号化のために、公開鍵と秘密鍵という一対の非対称型鍵をベースとした公開鍵暗号化技術を使用します。

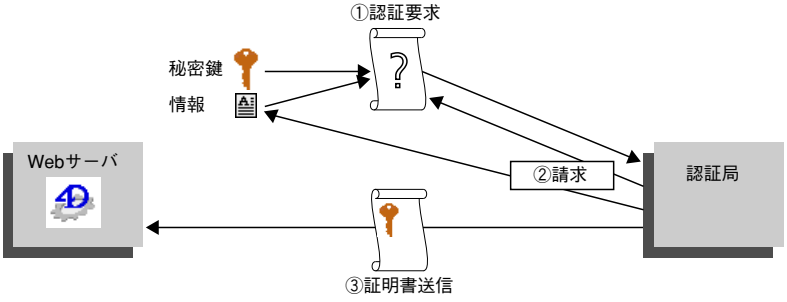
秘密鍵はデータの暗号化に使用されます。送信者（Webサイト）はこの鍵を誰にも与えません。公開鍵は情報の解読に使用され、証明書を介して送信者（Webブラウザ）に送られます。SSLをインターネットで使用する際、この証明書はVerisign®（ベリサイン社）のような認証局を通して届けられます。Webサイトは証明書配達のために認証局へ料金を支払いますが、この証明書によってサーバの認証は保証され、また暗号化モードでのデータのやり取りを行える公開鍵がこれに納められています。

注：暗号化メソッド、および公開鍵と秘密鍵の使用に関する詳細は、ENCRYPT BLOBコマンドの説明を参照してください。

証明書の取得方法

暗号化モードで動作する 4D Web サーバには、認証局発行の電子証明書が必要となります。証明書には、サイト ID およびそのサイトとの通信に使用する公開鍵などの各種情報が納められます。この証明書は、そのサイトに接続する Web サーバへ転送されます。証明書が確認され、受け付けられると、暗号化モードで通信が行われます。

注：ブラウザでは、そのプロパティで参照される認証局発行の証明書しか承認されません。



認証局の選定は、各種条件に応じて行われます。知名度が高い認証局であれば、その証明書は大部分のブラウザで承認されますが、料金は高くなります。

SSL 証明書を取得するには、

1 **GENERATE ENCRYPTION KEYPAIR** コマンドを使用して、秘密鍵を生成する。

警告：セキュリティ上の理由から、常に秘密鍵は人に知られないようにしてください。実際上、この鍵はいつも Web サーバマシンのもとに置いてください。また、Key.pem ファイルはデータベースストラクチャフォルダ内に配置しなければなりません。

2 **GENERATE CERTIFICATE REQUEST** コマンドを使用して、証明書リクエストを発行する。

3 選択した認証局へ証明書リクエストを送信する。

証明書リクエストの必要事項を満たすため、認証局との連絡が必要な場合もあります。認証局では、転送された情報が正しいかを確認します。また、証明書リクエストは PEM（Privacy Enhanced Mail）フォーマットを使用して BLOB 内に生成されます。このフォーマットを使用することにより、鍵をテキストとしてコピー＆ペーストし、その内容を変更せずに鍵を電子メールで送信することができます。例えば、証明書リクエストを納めた BLOB をテキストドキュメントに保存（**BLOB TO DOCUMENT** コマンドを使用）した後、これを開いてその内容をメールや Web フォームにコピー＆ペーストし、認証局へ送信することができます。

4 証明書を取得したら、“cert.pem”という名前のテキストファイルを作成し、このファイルに証明書の内容をペーストする。

証明書は、さまざまな方法で受信できます（通常は電子メールまたはHTMLフォーム）。4D Webサーバは、証明書用にすべてのプラットフォーム関連のテキストフォーマットを受け入れます（MacOS、PC、Linux等）。ただし、証明書は必ずPEMフォーマットでなくてはなりません。

5 “cert.pem”ファイルをデータベースストラクチャフォルダに配置する。

これで、Webサーバは暗号化モードで動作します。証明書は6ヶ月から1年の間有効です。

4Dにおける SSL のインストールとアクティブ化

4D WebサーバでSSLプロトコルを使用したい場合には、次のコンポーネントをサーバ上のそれぞれの場所にインストールしてください。

■ 4DSLI.DLL：SSL管理専用のSecured Layer Interface

このファイルは、データベースを発行する4Dアプリケーションの[4D Extensions]フォルダ内に配置してください。

■ key.pem：暗号化秘密鍵を含むドキュメント。

■ 4th Dimensionまたは4D Serverでは、このファイルは必ずデータベースフォルダ内に配置してください。

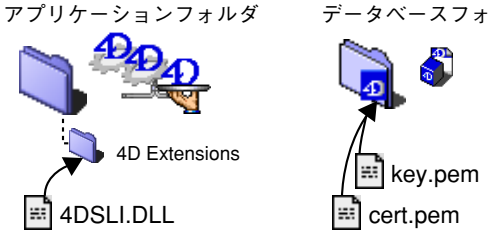
■ 4D Clientでは、このファイルは必ず4D Clientのアプリケーションフォルダ内に配置してください。

■ cert.pem：“証明書”を納めたドキュメント。

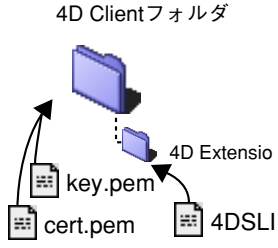
■ 4th Dimensionまたは4D Serverでは、このファイルは必ずデータベースフォルダ内に配置してください。

■ 4D Clientでは、このファイルは必ず4D Clientのアプリケーションフォルダ内に配置してください。

4D Web サーバで SSL を実装するために必要なファイル（4th Dimension および 4D Server）



4D Web サーバ（4D Client）で SSL を実装するために必要なファイル：



注：暗号化コマンドである ENCRYPT BLOB および DECRYPT BLOB を使用する際にも、4DSLI.DLL が必要となります。

これらのファイルがインストールされると、SSL を使用して 4D Web サーバへ接続することができます。ただし、4D Web サーバが SSL 接続を受け入れるには、SSL を“アクティブ”にしなくてはなりません。この設定は、データベースの「環境設定」で「Web」テーマの「公開」ページにあるパラメータを使用して行うことができます。

デフォルトでは、SSL 接続が許可されています。Web サーバで SSL 機能を使用したくない場合や、同一マシン上で暗号化接続を許可する他の Web サーバが動作している場合には、このオプションの選択を解除することができます。

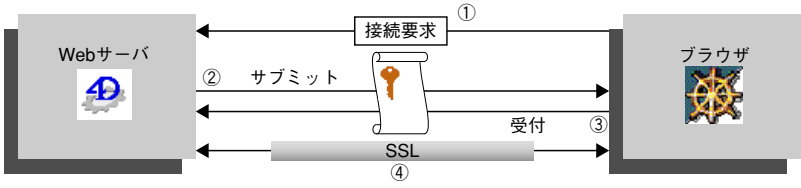
SSL データのやり取り専用の TCP ポートは 443 です。したがって、SSL を使用する Web サーバは、1 台のマシンに 1 つしかインストールできません。「環境設定」のこのページで定義した TCP ポートは、標準モードの Web サーバ接続に対して使用されます。

注：4D Web サーバの管理のために設定した「環境設定」のその他パラメータ（パスワード、タイムアウト、キャッシュサイズ等）は、サーバが SSL モードで動作しているかどうかに関わらず適用されます。

SSL を使用したブラウザ接続

Web 接続を暗号化モードで実行するには、ブラウザから送信する URL を（“http”ではなく）“https”で開始します。

この場合、ブラウザには警告ダイアログが表示されます。ユーザが「OK」をクリックすると、Webサーバは証明書をブラウザへ送信します。



次に、接続に使用される暗号化アルゴリズムがブラウザとWebサーバ側で決定されます。デフォルトとして、サーバはRC4（128ビット）暗号化アルゴリズムを提供し、このアルゴリズムは40ビットで暗号化した鍵を使って行われる接続に対しても使用されます。その国の法律や使用するブラウザのバージョンによっては、128ビットのアルゴリズムが使用できない場合があります。最も強力で一般的なアルゴリズムが使用されます。

接続モードの管理

4D Webサーバで SSL を使用する場合に、特別なシステム構成は必要ありません。しかし、SSL Webサーバは非暗号化モードでも動作できるという点に注意してください。また、接続モードは、ブラウザ側の要求があれば（例えば、ブラウザのURLエリアでユーザが“HTTPS”を“HTTP”で置き換えた場合）、もう一方のモードへ切り替えることができます。開発者は、非暗号化モードで行われたリクエストを禁止したり、リダイレクトすることが可能です。**Secured Web connection**関数を使用すると、現在の接続モードを取得できます。

参照

DECRYPT BLOB、ENCRYPT BLOB、GENERATE CERTIFICATE REQUEST、GENERATE ENCRYPTION KEYPAIR、Secured Web connection、Webサービス：Webサーバセッティング

START WEB SERVER

START WEB SERVER

説明

START WEB SERVERコマンドは、アプリケーション（4th Dimension、4D Server、4D Client）が実行されるマシン上で、4th DimensionアプリケーションのWebサーバを開始します。この結果、イントラネットネットワークまたはインターネット上にデータベースが公開されます。

Webサーバが正常に起動された場合には、システム変数OKに1が設定され、そうでなければシステム変数OKは0（ゼロ）が設定されます。例えば、TCP/IPネットワークプロトコルが正しく設定されている場合には、システム変数OKに0が代入されます。

参照

STOP WEB SERVER

システム変数とシステムセット

Webサーバが正常に開始された場合はシステム変数OKに1、そうでない場合はシステム変数OKに0が設定されます。

STOP WEB SERVER

STOP WEB SERVER

説明

STOP WEB SERVERコマンドは、アプリケーション（4th Dimension、4D Server、4D Client）が実行されるマシン上で、4th DimensionアプリケーションのWebサーバを終了します。Webサーバが開始している場合は、すべてのWeb接続が停止され、すべてのWebプロセスは終了します。

Webサーバが開始していない場合、このコマンドは何も行いません。

参照

START WEB SERVER

SET WEB TIMEOUT

SET WEB TIMEOUT (タイムアウト)

引数	タイプ		説明
timeout	数値	→	Web接続タイムアウトに設定する秒数

説明

SET WEB TIMEOUTコマンドは、コンテキストモードにおけるWeb接続プロセスへのタイムアウトを設定します。タイムアウトのデフォルトは5分です。

引数<パラメータ>に時間を渡すことにより、値を増減することができます。新しいタイムアウトのデフォルトは秒で表されます。

■ **SET WEB TIMEOUT**がWebプロセスから呼ばれると、<タイムアウト>引数はそのプロセスだけ適用されます。

■ **SET WEB TIMEOUT**がWebプロセス以外から呼ばれると、すべての「Web接続」プロセスが影響を受けます。

参照

「Webサービス：「Web接続」プロセス」

SET HTML ROOT

SET HTML ROOT (パス名HTML)

引数	タイプ		説明
パス名HTML	数値	→	HTMLファイル用のデフォルトディレクトリに対するHTMLパス名

説明

SET HTML ROOTコマンドを使用すると、**SEND HTML FILE**コマンドの引数としてHTMLファイルを渡す場合に4Dがこのファイルを検索するデフォルトのディレクトリやフォルダを変更することができます。

警告：SET HTML ROOTコマンドはコンテクストモードでのみ機能します。非コンテクストモードでデフォルトのHTMLのルートフォルダを設定するためには、「環境設定」ダイアログボックスの「デフォルトHTMLルート」エリアを使用します。パフォーマンス上の理由から、Webサーバの実行モードに関わらず、通常は「環境設定」でデフォルトのHTMLルートフォルダを設定することをお勧めします。

指定するパス名はHTMLパス名である必要があり、プラットフォームに関係なく、そのパス名はディレクトリまたはフォルダ名をスラッシュ（"/"）文字で区切ります。HTMLパス名の詳細については、書店にあるHTMLに関する書籍のランゲージリファレンスを参照してください。

無効なパス名を指定した場合には、OSのファイルマネージャエラーが生成されます。**ON ERR CALL**メソッドでこのエラーを処理することができます。エラーメソッド内部から警告やメッセージを表示する場合には、ブラウザ側に表示されます。

注意：SET HTML ROOTコマンドは、データベースの「環境設定」で定義されたデフォルトのHTMLルートフォルを考慮します。

▼ **SEND HTML FILE**コマンドの例を参照してください。

参照

ON ERR CALL

エラー処理

無効なパス名を指定すると、OSのファイルマネージャエラーが生成されます。**ON ERR CALL**メソッドでこのエラーを処理することができます。

SET WEB DISPLAY LIMIT

SET WEB DISPLAY LIMIT (レコード数 { ; ページ数 { ; ピクチャ参照番号}})

引数	タイプ		説明
レコード数	数値	→	各HTMLページに表示する最大レコード数
ページ数	数値	→	各HTMLページの下部にあるページ参照の最大数
ピクチャ参照番号	数値	→	フルページレコードボタン用のピクチャ参照番号

説明

SET WEB DISPLAY LIMITコマンドは、ユーザが**DISPLAY SELECTION**コマンドまたは**MODIFY SELECTION**コマンドを呼び出した際に、Webブラウザ側におけるレコードセレクションの4th Dimensionによる表示方法を変更します。このコマンドはコンテキストモードでのみ動作します。

4th Dimensionを使用してレコードセレクションを表示するとき、プログラムはセレクション中のすべてのレコードをロードするわけではなく、一度にウインドウに表示できる数のレコードを（ディスクから）ロードするだけです。そうすることによって、何千件ものレコードからなるセレクションを作成した場合でも、レコードの表示は非常に高速に行われます。その後、ウインドウのスクロールやサイズ変更をすると、4Dはそれに対応してレコードをロードします。

Webでは、4Dはページに表示されるレコードセレクションを分割します。ページング体系がなければ、数千レコードからなるセレクションの場合、結果としてインターネット上またはイントラネット上で1つのWebページだけに数千のレコードを表示することになります。また、これらのレコードのダウンロードにはかなりの時間を要し、Webブラウザのメモリ不足につながります。

デフォルトでは、4th Dimensionはセレクションの最初の20レコードを表示し、また、各HTMLページの最後に、最初の20ページのセレクションへのリンクを20個含みます。つまり、デフォルトでは、各セレクションページの最後にあるページリンク上をクリックすることにより、セレクションの最初の400レコードをブラウズできるということです。このページングシステムはコーディングに対して透過的であることに注意してください。すべての作業は、**DISPLAY SELECTION**コマンドや**MODIFY SELECTION**コマンドへの呼び出しの内部で行われます。

SET WEB DISPLAY LIMITコマンドでこれらの設定を変更できます。引数<レコード数>には、各セレクションページに表示したい最大レコード数を指定します。引数<ページ数>には、各セレクションページの最後に配置したい、セレクションページの最大リンク数を指定します。

例えば、10,000件のレコードセレクションがあり、1回のセレクション表示ですべてをブラウズしたい場合には、<レコード数>=100、<ページ数>=100を渡すことができます。ただし、このデータがネットワークやインターネットを経由するということに注意してください。インターネットの場合は、セレクションの表示設定を変更する際に、スピード要因を考慮する必要があります。

さらに、**SET WEB DISPLAY LIMIT**コマンドはオプションとして、フルページレコードボタンのデフォルトアイコンを変更できます。引数<ピクチャ参照番号>に、データベースのピクチャライブラリに格納されている、新しいアイコンとして使用したいピクチャのピクチャ参照番号を指定します。

SET WEB DISPLAY LIMITコマンドは、その後の**DISPLAY SELECTION**コマンドまたは**MODIFY SELECTION**コマンドへの呼び出しだけに影響を与え、その有効範囲はカレントプロセス内です。

SET HOME PAGE

SET HOME PAGE (ホームページ)

引数	タイプ		説明
ホームページ	文字列	→	ページ名またはHTMLアクセスパス、またはカスタムホームページを送らないようにする空白文字列

説明

SET HOME PAGEコマンドを使用すると、現在のWebプロセス用のカスタムホームページを変更することができます。

定義されたページはWebプロセスに関連付けられているため、接続されたユーザごとに違うホームページを定義することもできます。このページはスタティックにもセミダイナミックにもどちらにでもなることができます。

HTMLホームページの名前またはページのHTMLアクセスパスをホームページ引数に渡します。デフォルトホームページを有効にするには、空白をホームページに渡します。

注：4Dでは、「環境設定」ダイアログボックスでデフォルトホームページを定義することができます。この場合、Webサーバのスタートアップモード（コンテキストモードまたは非コンテキストモード）に関わらず、デフォルトホームページはすべてのWeb接続に適用されます。

参照

Web Server設定

SEND HTML FILE

SEND HTML FILE (HTMLファイル)

引数	タイプ		説明
HTMLファイル		→	HTMLファイルへのHTMLアクセスパス、または**SEND HTML FILE**を終了させる空白文字列

説明

SEND HTML FILEコマンドは、HTMLドキュメントに保存されたWebページをWebブラウザへ送信します。このドキュメントのパス名は＜HTMLファイル＞に渡します。

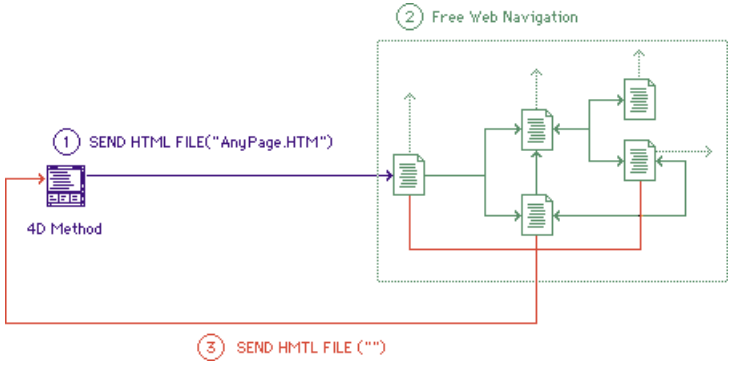
デフォルトとして4th Dimensionは、アプリケーションの「環境設定」で指定したHTMLルートフォルダ内でこのHTMLドキュメントを探します。

このコマンドは、HTML構文で表わされたパス名だけを引数として受け入れます。つまり、プラットフォームに関わらず、ディレクトリ名やフォルダ名はスラッシュ（“/”）で区切らなくてはなりません。

無効なHTMLパス名を指定した場合、4Dは「リクエストされたHTMLページが見つかりません」というメッセージをWebブラウザに送信します。

＜HTMLファイル＞に空の文字列を渡すもう一つのシンタックス、**SEND HTML FILE**("")は、コンテキストモードでHTMLモードを開始した**SEND HTML FILE**コマンドの呼び出しを終了します。

これを次の図で説明します。



1. コンテキストモードにおいて、4Dメソッド（プロジェクト、オブジェクト、またはデータベース）は**SEND HTML FILE**をコールして、HTMLドキュメントをブラウザへ送信します。

2. ブラウザへ送られた最初のWebページは、別のWebページへのHTMLリンクを含むか、またはそのページ内には別のWebページを送信する**SEND HTML FILE**をコールする4Dメソッドへの参照を含みます。さらに、これらの別のページには、別のページをアクセスする4Dメソッドのリンクや参照を含む、というように続きます。Webページを参照し続けている間でも、戻るボタン等でブラウザのナビゲーションコントロールを使用することができます。

3. 任意のWebページに**SEND HTML FILE**（""）を呼び出す4Dメソッドへの参照を組み込むことができます。この呼び出しによって、すべての始まりである**SEND HTML FILE**のコールを終了して戻り、自由なWebナビゲーションの開始元である4Dメソッドの実行を続けます。

SEND HTML FILEコマンドは、システム変数OKをセットするようになりました。送るファイルが存在していて、タイムアウトになっていなければ、システム変数OKは1になります。そうでなければ0になります。

注：Webプロセスではないプロセスから SEND HTML FILE を実行すると、コマンドは何もせずエラーが戻ります。呼び出しは無視されます。

いずれのモードでも、ページ中の4D変数および4DSCRIPTタイプのタグへの参照は、常に解析されます。

1. データベースのHTMLルートフォルダは「WebDocs」フォルダです。このフォルダには次の要素があります。

```
..\WebDocs\HTM\MyPage.HTM
```

Webページ“MyPage.HTM”の送信は、次の方法で実行しなくてはなりません。

```
SEND HTML FILE("HTM/MyPage.HTM")
```

2. コンテキストモードの例題：4D Webセッション中に、4Dフォームを使用してレコードを追加します。このフォームには「bHelp」ボタンがあり、そのオブジェクトメソッドは次の通りです。

```
'bHelpボタンのオブジェクトメソッド
 SEND HTML FILE ("Help.HTM")
```

Help.HTMドキュメントから始めた場合、非常に多くのWebサイトからのデータベースのヘルプシステムのあるHTMLページの間でも自由にナビゲートできます。

Submitボタンで、データ入力に戻られます。

このようなHTMLドキュメントはsubmitボタンの定義を含まなければなりません。

```
<!-- bDone submit button ->
<P><INPUT TYPE="submit" NAME="bDone" VALUE="Done"></P>
```

post actionフォームの定義と同様

```
<!-- Execute the 4D htm_Help_Done when a submit button is hit ->
<FORM action="/4DMETHOD/htm_Help_Done" method="POST"
```

4D側では、htm_Help_Doneプロジェクトメソッドが**SEND HTML FILE**の開始をbHelpによってを終了させます。

```
` htm_Help_Doneプロジェクトメソッド
SEND HTML FILE ("")
```

BHelpボタンのオブジェクトメソッドでの**SEND HTML FILE**の呼び出しはメソッドの最終行で行います。メソッドが完成した時、データ入力に戻ります。

システム変数とセット

送られたファイルが存在し、タイムアウトが実行されていない場合は、システム変数OKは1がセットされ､それ以外は0がセットされます。

参照

SEND HTML BLOB、「HTMLとJavaScriptのカプセル化」、「Webサービス：入門編(パートII)」

GET WEB FORM VARIABLES

GET WEB FORM VARIABLES (名前配列;値配列)

引数	タイプ		説明
名前配列	テキスト配列	←	Webフォーム変数の名前
値配列	テキスト配列	←	Webフォーム変数の値

説明

GET WEB FORM VARIABLESコマンドは、“サブミットされた”（つまり、Webサーバに送信された）Webフォームの変数の名前および値を、テキスト配列である＜名前配列＞と＜値配列＞に代入します。

このコマンドは、HTMLページに納めることができるすべての変数の値を取得します。つまり、テキストエリア、ボタン、チェックボックス、ラジオボタン、ポップアップメニュー、選択リスト等の値です。

注：チェックボックスに関しては、チェックボックスが実際にチェックされている場合にのみ、変数の名前と値（“On”）が返されます。

次の状況で呼び出されると、このコマンドは非コンテキストモード、またはコンテキストモード（4D Clientを除く）のいずれかで有効になります。

- フォームが「POST」メソッドでサブミットされる場合（/4DACTIONまたは/4DMETHODまたは/4DCGIで開始するアクション）
- フォームが「GET」メソッドでサブミットされる場合（/4DACTIONまたは/4DMETHODまたは/4DCGIで開始するアクション）
- リクエストの文字列を含むURLがWebサーバに送信される場合

必要があれば、このコマンドを**On Web Connection**データベースメソッドや、フォームのサブミットの結果として発生するその他の4Dメソッド内で呼び出すことができます。

Webフォームと関連するアクションについての詳細

各フォームには、名前の付いたデータ入力エリア（テキストエリア、ボタン、チェックボックスが含まれます。

フォームがサブミットされると（リクエストがWebサーバに送信されると)、そのリクエストには(その他の項目の中に）データ入力エリアのリストとこれに関連する値が納められます。

フォームは、次の2種類のメソッドを使用してサブミットできます（両方とも4Dと共に使用できます)。

■ POST：通常はWebサーバへデータを追加するために使用される－データベースへの追加。

■ GET：通常はWebサーバのリクエストに使用される

例題

▼ フォームにはvNameとvCityという2つのフィールドがあり、それぞれ“ROBERT”と“DALLAS”という値が納められています。このフォームに関連付けられたアクションは、“/4DACTION/WEBFORM”です。

■ フォームメソッドが「POST」（最も頻繁に使用される）の場合、入力データはURL上には表示されません（http://127.0.0.1/4DACTION/WEBFORM)。

■ フォームメソッドが「GET」の場合、入力データはURL上に表示されます

(http://127.0.0.1/4DACTION/WEBFORM?vNAME=ROBERT&vCITY=DALLAS)。

WEBFORMメソッドは次のようになります。

```
ARRAY TEXT($anames;0)
ARRAY TEXT($avalues;0)
GET WEB FORM VARIABLES($anames;$avalues)
```

結果は次の通りです。

```
$anames{1} = "vNAME"
$anames{2} = "vCITY"
$avalues{1} = "ROBERT"
$avalues{2} = "DALLAS"
```

変数vNAMEには“ROBERT”が、変数vCITYには“DALLAS”が納められます。

参照

Webサービス：HTMLとJavaScriptのカプセル化、Webサービス：特殊なURLとフォームアクション

Web Context

Web Context → ブール

引数	タイプ		説明
このコマンドには、引数はありません。			
戻り値	ブール	←	True=コンテキストモード False=非コンテキストモード

説明

この関数はWebプロセスから呼び出されなければなりません。これは、Webの接続がコンテキストモード（True）で実行されているのか、それとも非コンテキストモード（False）で実行されているのかを返します。

注：次の場合、Web Context関数は常に“False”を返します。

・Webプロセスではないプロセスから呼び出された場合

・4D Clientマシン上で実行された場合

この関数の使用は、**On Web Connection**データベースメソッド内で行なうことをお勧めします。

▼ **On Web Connection**データベースメソッドの例を示します。

```
If (Web Context)
    WithContext ($1;$2;$3;$4;$5;$6)
Else
    WithoutContext ($1;$2;$3;$4;$5;$6)
End if
```

参照

PROCESS PROPERTIES

SEND HTTP REDIRECT

SEND HTTP REDIRECT (url {; *})

引数	タイプ		説明
url	文字列	→	新しいURL
*	*	→	指定されている場合=4Dは URLをエンコードしない 省略されている場合=4Dは URLをエンコードする

説明

このコマンドは、URLのリダイレクトを可能にします。

引数＜url＞には、リクエストを転送する新しいURLを渡します。この引数がファイルへのURLである場合、このファイルへの参照を含まなくてはなりません。

例：**SEND HTTP REDIRECT**（"/MyPage.HTM"）

このコマンドがコンテキストモードで呼び出されると、実行された直後にWebプロセスはアボートされ、Webライセンスは開放されます。このコマンドは、同じメソッド内にあるデータを送るコマンド（**SEND HTML FILE**、**SEND HTML BLOB**等）より優先されます。

さらに、このコマンドを使用すると、他のWebサーバへリクエストを転送することができます。

4DはURLを自動的にエンコードしますが、*を渡すとエンコードしません。

例題

コマンドを使用して、スタティックページを用いて4D内でカスタムリクエストを実行することができます。スタティックHTMLページ内に下記のエレメントを置いたと仮定します。

注：POST アクション"/4dcgi/rech"はテキストエリアと OK ボタン、CANCEL ボタンに関連づけられています。

非コンテキストモードを管理する **On Web Connection** データベースメソッド部分（またはサブパート）では、下記のコードを挿入します。

```
Case of
	￥ ($1="/4dcgi/rech")`4D はこの URL を受け取ると
		`OK ボタンが使用され、'name'フィールドに avalue がある場合
	If ((bOK="OK") & (name # ""))
		`同じメソッド内のはるか下に置かれた
		`リクエストコードを実行する URL を変更する
		SEND HTTP REDIRECT ("/4dcgi/rech?"+name)
	Else
		`そうでなければ、始めのページに戻る
		SEND HTTP REDIRECT ("/page1.htm")
	End if
:
￥($1="/4dcgi/rech?@") `URL がリダイレクトされている場合
	... `リクエストコードはここに挿入する
END case
```

参照

なし

ウインドウコマンド

Open form window

Open form window ({テーブル ;} フォーム名 {; タイプ {; 水平位置 {; 垂直位置{; *}}}})→ ウインドウ参照番号

引数	タイプ		説明
テーブル	テーブル	→	フォームのテーブル 省略されている場合デフォルトテーブル
フォーム名	文字列	→	フォームの名前
タイプ	倍長整数	→	ウインドウのタイプ
水平位置	倍長整数	→	ウインドウの水平位置
垂直位置	倍長整数	→	ウインドウの垂直位置
*	*	→	ウインドウの現在位置とサイズをセーブ
ウインドウ参照番号	倍長整数	←	ウインドウの参照番号

説明

Open form windowコマンドは、フォーム名引数で指定したフォームのサイズおよびサイズ変更プロパティを使用して新しいウインドウを開きます。

フォームの内容はウインドウに表示されないことに注意してください。フォームを表示したい場合には、フォームをロードするコマンドを呼び出さなければなりません（例えば、**ADD RECORD**）。

デフォルトでは（タイプ引数が渡されていない場合）クローズボックス付きの標準ウインドウが開かれます。**Open window**コマンドとは異なり、ウインドウのクローズボックスには何のメソッドも定義されません。このクローズボックスをクリックすると、**On Close Box**フォームイベントがフォーム用に起動されている場合を除き、ウインドウをキャンセルして閉じます。この場合、**On Close Box**イベントに定義されたコードが実行されます。

フォームのサイズ変更が可能であれば、開かれたウインドウはズームボックスならびにグローボックスを持ちます。

注：フォームの主なプロパティを知るには、GET FORM PROPERTIESコマンドを使用します。

オプションのタイプ引数はウインドウのタイプが指定でき、下記の“Open window”テーマ内にある定数の内から1つを渡します。

定数	タイプ	値
Standard form window	倍長整数	8
Modal form dialog box	倍長整数	1
Movable form dialog box	倍長整数	5
Palette window	倍長整数	1984

オプションの引数水平位置は、ウインドウの水平位置を定義します。ポイント単位で位置を指定します（**Open window**コマンドを参照してください）。または、下記の“Open form window”テーマ内にある定数の内から1つを渡します。

定数	タイプ	値
Horizontally Centered	倍長整数	65536
On the Left	倍長整数	131072
On the Right	倍長整数	196608

オプションの引数垂直位置は、ウインドウの垂直位置を定義します。ポイント単位で位置を指定します（**Open window**コマンドを参照してください）。または、下記の“Open form window”テーマ内にある定数の内から1つを渡します。

定数	タイプ	値
Vertically Centered	倍長整数	262144
At the Top	倍長整数	327680
At the Bottom	倍長整数	393216

これらの引数は、ツールバーおよびメニューバーの存在ならびにアプリケーションウインドウの現在のサイズ（Windowsの場合）を考慮に入れます。

オプションの引数*を渡すと、ウインドウをクローズした時の位置およびサイズが記憶されます。ウインドウが再度開かれる時に、その前の位置とサイズが優先されます。この場合、垂直位置と水平位置の引数は最初にウインドウが開かれる時のみに使用されます。

▼ 下記のステートメントは標準のウインドウをクローズボックス付きで開き、自動的にそれを「入力」フォームと同じサイズになるように調整します。フォームは、サイズ変更可能なものとして定義されているので、ウインドウもグローボックスおよびズームボックスを持ちます。

```
$winRef := Open form window ([Table1];"Enter")
```

▼ 下記のステートメントは、画面の左上部にあるフローティングパレットを開くものです。このパレットは、開かれるたびに前回ユーザが閉じた時の位置に表示されます。

```
$winRef:= Open form window([Table1]; "Tools"; Palette form window; On the Left;
                                                                  At the Top;*)
```

参照

Open window、GET FORMS PROPERTIES

シンタックスエラー

コード	エラーの起きた理由
69	外部ウィンドウ参照が必要です。

データベースエンジンエラー

コード	エラーの起きた理由
-9910	Soap fault
-9911	パーサー fault
-9912	HTTP fault
-9913	ネットワーク fault
-9914	内部 fault