DataStore プロバイダ

汎用データ共有

Version 1.0.5

ユーザーズ ガイド

October 31, 2018

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2006-02-23	初版.
1.0.1.0	2006-12-15	2.2.4.DataBase 保存機能→@VARS 変数に修正
1.0.2.0	2007-03-07	@VARS 変数で固定インデックス機能, HELP 書込み機能追加.
1.0.3.0	2008-02-06	Database オプションの追加
1.0.4.0	2008-04-17	@LVARS_MAX 変数追加.
1.0.4.1	2010-02-10	エラーコード追加
1.0.4	2012-07-17	ドキュメントのバージョンルールを変更
1.0.4	2018-05-21	誤植を修正. (PutMacro, GetMacro→PutVarsMacro, GetVarsMacro)
1.0.5	2018-10-31	メモリリーク バグ修正

【対応機器】

機種	バージョン	注意事項

目次

1. はじめに

本書は，ORiN2 アプリケーション間でデータ共有を簡単にする DataStore プロバイダのユーザガイドです．

DataStore プロバイダは内部に変数テーブルを格納し，その変数テーブルを共有することで ORiN2 アプリケーション間でのデータ共有を実現します．このプロバイダを用いれば，同一マシンでのアプリケーション間データ共有だけでなく，別のマシンで稼働しているアプリケーションとのデータ共有も簡単に実現できます．

本書は，この DataStore プロバイダの機能と実装されているメソッドについて説明します．

2. プロバイダの概要

2.1. 概要

ORiN2 の CAO(Controller Access Object)はロボットコントローラを抽象化したオブジェクトモデルです. よって, その対象とするリソースには, タスクや変数, プログラムファイル, ロボットなど様々でありますが, DataStore プロバイダは“変数”リソースだけを持つ特定用途のプロバイダです. 但し, 内部的には, コントローラごとに変数データを管理しているので, コントローラ名が違えば, 同じ変数名でも違う値を保持できます[1].

格納できるデータ型は VARIANT 型データで, その中で配列も含めて様々な型のデータを格納することができるので, アプリケーション間で複雑なデータの共有でも簡単に実現することができます.

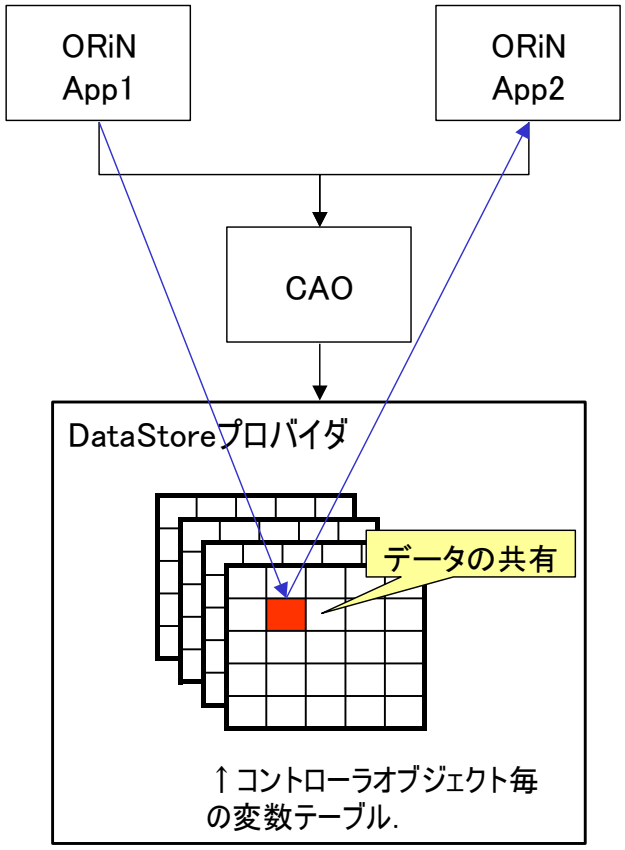

図 2-1 概要

[1] 但し, 同じプロセス空間で実行された場合に限られます. 違うプロセス空間では, 同じコントローラ名で同じ変数名でも各々の値を保持します.

表 2-1 DataStore プロバイダ

ファイル名	CaoProvData.dll
ProgID	CaoProv.DataStore
レジストリ登録[2]	regsvr32 CaoProvData.dll
レジストリ登録の抹消	regsvr32 /u CaoProvData.dll

DataStore プロバイダには, MDAC(Microsoft Data Access Components)2.7 が必要です.

[2] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません.

2.2. 動作概要

2.2.1. スタック機能

変数名“@STACK”を使用すると，コントローラ毎に変数値を複数格納できるバッファを用意します．このときのバッファは先入れ後出し型となります．

バッファの内容はコントローラが削除されるまで保持されます．

2.2.2. キュー機能

変数名“@QUEUE”を使用すると，コントローラ毎に変数値を複数格納できるバッファを用意します．このときのバッファは先入れ先出し型となります．

バッファの内容はコントローラが削除されるまで保持されます．

2.2.3. ランダム取得

変数名“@RANDOM”を使用すると，値を取得したときの値がランダムで変化します．

2.2.4. @VARS 変数

変数名“@VARS[.<固定インデックス番号>]”を使用すると，ID プロパティを変更することで複数の値を保持することができます．<固定インデックス番号>は省略可能で，指定した場合は ID プロパティで動的にインデックスを変更することはできません．

“@VARS”変数の値はコントローラ名の影響を受けません．このため，すべてのコントローラで値を共有することができます．

値の格納先は，データベースと一時変数があります．データベース領域に値を格納した場合のみ，DataStore プロバイダを終了してもデータの内容は保存されます．

“@VARS”変数の領域及びデータベース領域のサイズは，CaoConfig を使用して DataStore プロバイダの Parameter に以下の文字列を指定することで変更できます．

表 2-2 CaoConfig の Parameter 文字列

パラメータ	意味
DataBase=<データベースファイルのパス>	データベースファイルの絶対パスを指定します． (デフォルト:デフォルトデータベース) データベースファイルは，以下のファイルをコピーして使用してください． <DataStore_Root>¥Bin¥datastore_master_en.mdb
ItemMax=<“@VARS”変数のサイズ>	“@VARS”変数で使用できる ID プロパティのサイズを指定します． (デフォルト:200)

ItemDBMax=<データベース領域のサイズ>	“@VARS”変数のデータベース領域のサイズを指定します. (デフォルト:100)

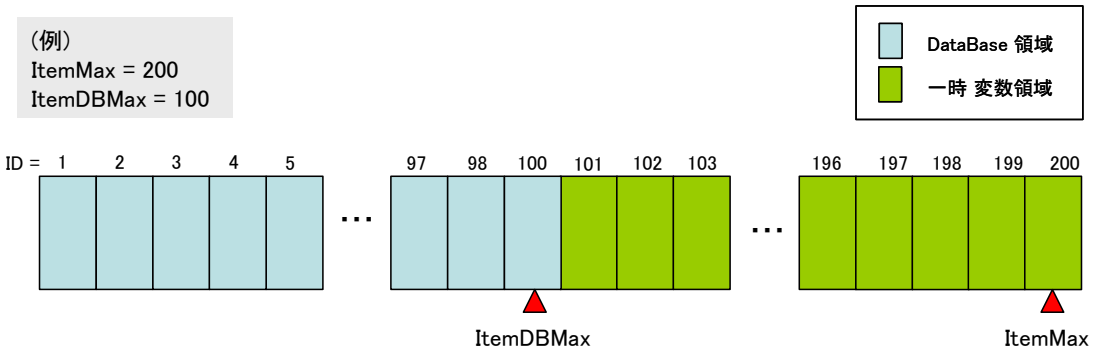

図 2-2 VARS 変数の構造

2.2.5. @LVARS 変数

変数名“@LVARS[.<固定インデックス番号>]”を使用すると, ID プロパティを変更することで複数の値を保持することができます. <固定インデックス番号>は省略可能で, 指定した場合は ID プロパティで動的にインデックスを変更することはできません.

“@LVARS”変数の値は同一のコントローラ名の場合のみ, 共有することができます.

値はメモリ空間上に生成されるため, 値は保存されません. 格納された値のクリアは, 同一のコントローラ名のオブジェクトが消滅したときに行われます.

“@LVARS”変数の領域のサイズは, CaoConfig を使用して DataStore プロバイダの Parameter に以下の文字列を指定することで変更できます

表 2-3 CaoConfig の Parameter 文字列

パラメータ	意味
LItemMax=<“@LVARS”変数のサイズ>	“@LVARS”変数で使用できる ID プロパティのサイズを指定します. (デフォルト:200)

以下に“@LVARS”変数と“@VARS”変数の機能比較を示します.

表 2-4 “@LVARS”変数と“@VARS”変数 機能比較

	@LVARS	@VARS
スコープ	同名コントローラのみ	全コントローラ
確保領域	メモリ領域のみ	メモリ領域とデータベース領域
値の保存	保存されません.	データベース領域のみ
値の消滅タイミング	同名のコントローラが全て削除されたとき	DataStore プロバイダが終了したとき

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

DataStore プロバイダでは AddController 時に仮想のコントローラ名を指定します. 前述したように, 変数テーブルはこのコントローラごとに管理されます. 指定したコントローラ名が既に存在する場合[3]はその変数テーブルが共有され, 逆に存在しない場合は, 新規に変数テーブルが作成されます. また, コントローラ名は大文字・小文字が区別されるので注意して下さい. コントローラオブジェクトが削除(Disconnect)された時点で, その変数テーブルも破棄されます. 一度破棄されたデータは復元できないので注意して下さい.

以下に AddController の引数仕様を示します.

```
AddController
(
    “<コントローラ名>”,          // 仮想コントローラ名
    “CaoProv.DataStore”,        // プロバイダ名. 固定.
    “<マシン名>”,                    // プロバイダの実行マシン名4
    “<オプション>”                   // オプション文字列
)
```

以下に AddController メソッドを実行するときの例を示します.

```
AddController
(
    “App1”,                                 // 仮想コントローラ名 ＝ App1
    “CaoProv. DataStore”,
    “”                                      // CAO エンジンプロセスで実行
    “”'
);
```

(仮想コントローラ名の命名規則)

各 ORiN アプリケーションが勝手に仮想コントローラ名を付けると競合する可能性があるので, 次のルールに従って作成することを推奨します.

[3] CAO のコレクション機能により 1 つのワークスペース内では, 同じコントローラ名を指定することはできません. 同じコントローラ名を指定するときは異なるワークスペースを使用して下さい.

[4] (注) 前述したように, <マシン名>が各アプリケーションで違っていると DataStore プロバイダは違うプロセス空間にロードされるので, 同じ<コントローラ名>を指定しても変数テーブルを共有することはできません.

(1) <アプリケーション名>を使います. ここでいう<アプリケーション名>とは, 実行可能ファイルの拡張子を除いたファイル名をすべて大文字で表記したものを指します. 例えば, “CaoSQL.exe”のアプリケーション名は, “CAOSQL”となります.

(2) 複数のインスタンスができるアプリケーションの場合は, “<アプリケーション名>.<連番>”とします. <連番>は 1 から開始することを推奨します. 例えば, “CAOSQL.1”, “CAOSQL.2”, … のようにして下さい.

2.3.2. CaoController::AddVariable メソッド

この CaoController クラスの AddVariable メソッドは，CAO の一般的意味では，変数にアクセスするためのメソッドです．DataStore プロバイダでは，このメソッドで変数テーブルに変数を登録します．指定された変数は，このメソッドを提供する CaoController オブジェクトの管理テーブルに登録されます．指定した変数名が既に存在する場合はその変数が共有され，逆に存在しない場合は新規に登録されます．また，変数名もコントローラ名と同じく大文字・小文字が区別されるので注意して下さい．変数値は VARIANT 型で，単純変数はもちろん配列データなども格納することができます．

以下に，AddVariable の引数仕様を示します．

```
AddVariable
(
    “<変数名>”,          // 変数テーブルに登録する変数名.
    “<オプション>”       // (未使用)
)
```

以下に AddVariable メソッドを実行するときの例を示します．

```
AddVariable
(
    “ABC”,               // ABC 変数を指定.
    “”
)
```

本来 DataStore プロバイダは，データを共有するために使われることを想定しているので，データの共有元アプリケーションは変数の一覧表を公開して格納データを明確にする必要があります．

また，変数名にはユーザ変数，システム変数のどちらでも指定することができます．システム変数を指定するときは，変数名の最初に“@”を付けます．変数名の最初に“@”がないものはすべてユーザ変数として扱われます．

DataStore プロバイダで実装されているシステム変数は 2.4.1 を参照して下さい．

2.3.3. CaoController::Execute メソッド

コマンドを実行します．コマンドの実行に必要なパラメータおよび取得する結果は表 2-5 を参照して下さい．

表 2-5 CaoController::Execute メソッドのコマンド実装一覧

コマンド	パラメータ	返り値	動作
PutHelp <Index>, <Help>	<Index:VT_I4>= @VARS 変数のインデックス番号 <Help:VT_BSTR>= @VARS 変数のヘルプ文字列	なし	@VARS 変数のヘルプ文字列を設定します．取得は変数クラスの Help プロパティを使います．

PutVarsMacro <Index>, <Macro>	<Index:VT_I4>= @VARS 変数のインデックス番号 <Macro:VT_BSTR>= @VARS 変数のマクロ名	なし	@VARS 変数のマクロ名を設定します. マクロ名には数値及び“¥!#$:.,”の記号は使用することができません.
GetVarsMacro <Index>	<Index:VT_I4>= @VARS 変数のインデックス番号	<Macro:VT_BSTR>= @VARS 変数のマクロ名	@VARS 変数のマクロ名を取得します.

2.3.4. CaoController::get_VariableNames プロパティ

get_VariableNames プロパティでは, AddVariable で追加したユーザ変数の一覧を文字列型の配列を格納した VARIANT 型で取得します.

2.3.5. CaoVariable::put_ID プロパティ

変数名“@VARS”のときのみ使用することができます. これ以外の変数でこのプロパティを使用した時はエラーを返します.

データベース保存機能でアクセスするデータの ID を指定します. この値は, 最小値が 1, 最大値は AddController の ItemMax オプションの値になります. この範囲外の値を指定した時はエラーを返します.

2.3.6. CaoVariable::get_ID プロパティ

変数名“@VARS”のときのみ使用することができます. これ以外の変数でこのプロパティを使用した時はエラーを返します.

データベース保存機能で現在アクセスしているデータの ID を取得します.

2.3.7. CaoVariable::get_Help プロパティ

変数名“@VARS”のときのみ使用することができます. これ以外の変数でこのプロパティを使用した時はエラーを返します.

データベース保存機能で現在アクセスしているデータから説明情報を取得します.

2.3.8. CaoVariable::get_Value プロパティ

変数テーブルに登録されている値を VARIANT 型で取得します. システム変数の場合は 2.4.1 の内容の値を取得します. ユーザ変数の場合は, 2.3.9 で設定した値を取得します.

2.3.9. CaoVariable::put_Value プロパティ

変数テーブルに値を VARIANT 型で登録します. システム変数の場合は 2.4.1 の“Put”の項目が“○”になっているもののみ put_Value プロパティを使用することができます.

2.3.10. CaoVariable::get_DateTime プロパティ

値が更新された日時を返します. 起動時に現在時刻で初期化されます. 変数名“@VARS”のときのみ使用することができます. これ以外の変数でこのプロパティを使用した時はエラーを返します.

2.3.11. CaoVariable::get_Microsecond プロパティ

値が更新された日時のマイクロ秒の要素を返します. 起動時に現在時刻で初期化されます. 変数名“@VARS”のときのみ使用することができます. これ以外の変数でこのプロパティを使用した時はエラーを返します.

2.4. 変数一覧

2.4.1. コントローラクラス

DataStore プロバイダでは, システム変数のみが予約されています. ユーザ変数には, 任意の名前を使用することができます.

表 2-6 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@COUNT	VT_I4	ユーザ変数の数を返します. システム変数は含みません.	○	-
@CURRENT_TIME	VT_DATE	プロバイダが実行されているマシンのローカルタイム.	○	-
@QUEUE	VT_VARIANT	キュー型変数. 1コントローラオブジェクトに1つ. 空になったときは, VT_EMPTY を返します. 逆に, VT_EMPTY を書き込むとバッファはクリアされ空になります.	○	○
@QUEUE_SIZE	VT_I4	キュー型変数(@U_QUEUE)の現在の要素数を返します.	○	-
@RANDOM	VT_I4	乱数を生成します.	○	-
@STACK	VT_VARIANT	スタック型変数. 1 コントローラオブジェクトに 1 つ. 空になったときは, VT_EMPTY を返します. 逆に, VT_EMPTY を書き込むとバッファはクリアされ空になります.	○	○
@STACK_SIZE	VT_I4	スタック型変数(@U_STACK)の現在の要素数を返します.	○	-

@STRESS	VT_I4	CPU 負荷を生成します. 値の書き込みでストレスレベルを設定します. 読み込むたびにストレスレベルに応じて CPU 負荷を生成します.	○	○
@TICK_COUNT	VT_I4	システムを起動した後の経過時間をミリ秒（ms）単位で返します. GetTickCount 関数.	○	-
@VERSION	VT_BSTR	バージョン.	○	-
@VARS[.<固定インデックス>]	VT_VARIANT	共有変数. <固定インデックス>は省略可能です. <固定インデックス>にはインデックス番号またはマクロ名を指定することができます. Variable::ID プロパティで指定されているデータに対して値の取得/設定を行います. <固定インデックス>を指定した場合は ID プロパティを設定することはできません. 取得時にデータが設定されていないときは, VT_EMPTY を取得します.	○	○
@VARS_MAX	VT_I4	データベースアクセス用変数(@VARS)で指定可能な ID の最大値を取得します.	○	-
@LVARS[.<固定インデックス番号>]	VT_VARIANT	コントローラ毎の共有変数. <固定インデックス番号>は省略可能です. Variable::ID プロパティで指定されているデータに対して値の取得/設定を行います. <固定インデックス番号>を指定した場合は ID プロパティを設定することはできません. 取得時にデータが設定されていないときは, VT_EMPTY を取得します.	○	○
@LVARS_MAX	VT_I4	@LVARS で指定可能な ID の最大値を取得します. LItemMax パラメータの値と一致します.	○	-

2.5. エラーコード

DataStore プロバイダでは，以下の固有エラーコードが定義されています．ORiN2 共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

表 2-7 独自エラーコード一覧

エラー名	エラー番号	説明
E_FAILED_CREATE	0x80100000	生成に失敗しました．
E_DOUBLE_DEFINED	0x80100001	既に定義されています．

3. サンプルプログラム

以下に App1.frm(1exe)で変数値を設定し, App2.frm(別の exe)でその値を取得するサンプルを示します.

List 3-1 SampleApp1.frm

```
Private Eng As CaoEngine
Private Ctrl As CaoController
Private Var As CaoVariable

' フォームのロード
Private Sub Form_Load()
    Set Eng = New CaoEngine

    '変数オブジェクトを取得します.
    Set Ctrl = Eng.Workspaces(0).AddController("RC1", "CaoProv.DataStore")
    Set Var = Ctrl.AddVariable("Sample")

End Sub

' Ctrl の変数に値を設定
Private Sub Command1_Click ()
    Var = Text1.Text
End Sub
```

List 3-2 SampleApp2.frm

```
Private Eng As CaoEngine
Private Ctrl As CaoController
Private Var As CaoVariable

' フォームのロード
Private Sub Form_Load()
    Set Eng = New CaoEngine

    '変数オブジェクトを取得します.
    Set Ctrl = Eng.Workspaces(0).AddController("RC1", "CaoProv.DataStore")
    Set Var = Ctrl.AddVariable("Sample")

End Sub

' Ctrl2 の変数から値を取得
Private Sub Command2_Click ()
    Text2.Text = Var
End Sub
```