iAnywhere SOLUTIONS™
A SYBASE COMPANY

Ultra Light.NET
ユーザーズ・ガイド

パート番号：DC50046-01-0902-01
改訂：2005 年 3 月

版権

Copyright © 2005 iAnywhere Solutions, Inc., Sybase, Inc. All rights reserved.

ここに記載されている内容を iAnywhere Solutions, Inc.、Sybase, Inc. またはその関連会社の書面による事前許可を得ずに電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても複製、転載、翻訳することを禁じます。

Sybase、SYBASE のロゴ、Adaptive Server、AnswerBase、Anywhere、EIP、Embedded SQL、Enterprise Connect、Enterprise Portal、GainMomentum、iAnywhere、jConnect MASS DEPLOYMENT、Netimpact、ObjectConnect、ObjectCycle、OmniConnect、Open ClientConnect、Open ServerConnect、PowerBuilder、PowerDynamo、Powersoft、Quickstart Datamart、Replication Agent、Replication Driver、SQL Anywhere、SQL Central、SQL Remote、Support Plus、SWAT、Sybase IQ、Sybase System 11、Sybase WAREHOUSE、SyBooks、XA-Library は米国法人 Sybase, Inc. の登録商標です。Backup Server、Client-Library、jConnect for JDBC、MainframeConnect、Net-Gateway、Net-Library、Open Client、Open Client/Server、S-Designor、SQL Advantage、SQL Debug、SQL Server、SQL Server Manager、Sybase Central、Watcom、Web.SQL、XP Server は米国法人 Sybase, Inc. の商標です。

Certicom、MobileTrust、および SSL Plus は Certicom Corp. の商標です。Security Builder は Certicom Corp. の登録商標です。

ここに記載されている上記以外の社名および製品名は、各社の商標または登録商標の場合があります。

目次

はじめに vii
SQL Anywhere Studio のマニュアル viii
表記の規則 xii
CustDB サンプル・データベース xv
詳細情報の検索／フィードバックの提供 xvi

1 Ultra Light.NET の概要 1
Ultra Light と .NET 2
システムの稼働条件とサポートされるプラットフォーム 3
Ultra Light.NET のアーキテクチャ 5

2 チュートリアル：Ultra Light.NET アプリケーションの構築 7
概要 8
レッスン 1：Visual Studio プロジェクトの作成 9
レッスン 2：Ultra Light スキーマ・ファイルの作成 14
レッスン 3：データベースへの接続 16
レッスン 4：データの挿入、更新、削除 20
レッスン 5：アプリケーションの構築と配置 27

3 Ultra Light.NET 開発の概要 29
Ultra Light データベース・スキーマ 30
データベースへの接続 32
暗号化と難読化 37
動的 SQL を使用したデータへのアクセスと操作 38
テーブル API によるデータ・アクセスと操作 46
Ultra Light でのトランザクション処理 54
スキーマ情報へのアクセス 55
エラー処理 57
ユーザ認証 58
Ultra Light アプリケーションの同期 59

4 iAnywhere.Data.UltraLite ネームスペース 63
ULActiveSyncListener インタフェース 64
ULAuthStatusCode 列挙 68
ULCommand クラス 69
ULConnection クラス 89
ULConnectionParms クラス 124
ULConnectionParms.UnusedEventHandler 委譲 139
ULCursorSchema クラス 140
ULDataAdapter クラス 149
ULDatabaseManager クラス 173
ULDatabaseSchema クラス 179
ULDataReader クラス 190
ULDbType 列挙 231
ULException クラス 235
ULIndexSchema クラス 239
ULInfoMessageEventArgs クラス 248
ULInfoMessageEventHandler 委譲 251
ULParameter クラス 252
ULParameterCollection クラス 268
ULPublicationSchema クラス 286
ULResultSetSchema クラス 290
ULRowUpdatedEventArgs クラス 292
ULRowUpdatedEventHandler 委譲 295
ULRowUpdatingEventArgs クラス 296
ULRowUpdatingEventHandler 委譲 299
ULRuntimeType 列挙 300
ULSchemaUpgradeData クラス 301
ULSchemaUpgradeListener インタフェース 304
ULSchemaUpgradeState 列挙 306
ULServerSyncListener インタフェース 307
ULSQLCode 列挙 311
ULStreamErrorCode 列挙 321
ULStreamErrorContext 列挙 327
ULStreamErrorID 列挙 329
ULStreamType 列挙 331
ULSyncParms クラス 333
ULSyncProgressData クラス 348
ULSyncProgressListener インタフェース 357

ULSyncProgressState 列挙 359
ULSyncResult クラス 362
ULTable クラス 368
ULTableSchema クラス 413
ULTransaction クラス 427

5 iAnywhere.UltraLite ネームスペース 431
ActiveSyncListener インタフェース 432
AuthStatusCode 列挙 436
Connection クラス 437
ConnectionParms クラス 459
CreateParms クラス 465
Cursor クラス 469
CursorSchema クラス 498
DatabaseManager クラス 505
DatabaseNameParms クラス 514
DatabaseNameParms.UnusedEventHandler 委譲 523
DatabaseSchema クラス 524
IndexSchema クラス 535
PreparedStatement クラス 543
PublicationSchema クラス 570
ResultSet クラス 574
ResultSetSchema クラス 579
RuntimeType 列挙 581
SchemaParms クラス 582
SchemaUpgradeData クラス 587
SchemaUpgradeListener インタフェース 590
SchemaUpgradeState 列挙 592
ServerSyncListener インタフェース 594
SQLCode 列挙 598
SQLException クラス 606
SQLType 列挙 610
StreamErrorCode 列挙 614
StreamErrorContext 列挙 620
StreamErrorID 列挙 622
StreamType 列挙 624
SyncParms クラス 626
SyncProgressData クラス 640

SyncProgressDialog クラス 649
SyncProgressListener インタフェース 659
SyncProgressState 列挙 661
SyncResult クラス 664
Table クラス 670
TableSchema クラス 719

索引 733

はじめに

このマニュアルの内容 このマニュアルでは、Ultra Light.NET について説明します。Ultra Light.NET を使用すると、PC、ハンドヘルド、モバイル、埋め込みデバイスのデータベース・アプリケーションを開発し、これらのデバイスに配備できます。

対象読者 このマニュアルは、Ultra Light リレーショナル・データベースのパフォーマンス、リソース効率、堅牢性、セキュリティを利用してデータを格納、同期することを目的とする .NET アプリケーション開発者を対象にしています。

SQL Anywhere Studio のマニュアル

このマニュアルは、SQL Anywhere のマニュアル・セットの一部です。この項では、マニュアル・セットに含まれる各マニュアルと使用法について説明します。

SQL Anywhere Studio のマニュアル

SQL Anywhere Studio のマニュアルは、各マニュアルを 1 つの大きなヘルプ・ファイルにまとめたオンライン形式、マニュアル別の PDF ファイル、および有料の製本版マニュアルで提供されます。SQL Anywhere Studio のマニュアルは、次の分冊マニュアルで構成されています。

- **『SQL Anywhere Studio の紹介』** このマニュアルでは、SQL Anywhere Studio のデータベース管理と同期テクノロジの概要について説明します。また、SQL Anywhere Studio を構成する各部分について説明するチュートリアルも含まれています。
- **『SQL Anywhere Studio 新機能ガイド』** このマニュアルは、SQL Anywhere Studio のこれまでのリリースのユーザを対象としています。ここでは、製品の今回のリリースと以前のリリースで導入された新機能をリストし、アップグレード手順を説明しています。
- **『Adaptive Server Anywhere データベース管理ガイド』** このマニュアルでは、データベースおよびデータベース・サーバの実行、管理、設定について説明しています。
- **『Adaptive Server Anywhere SQL ユーザーズ・ガイド』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『Adaptive Server Anywhere SQL リファレンス・マニュアル』** このマニュアルは、Adaptive Server Anywhere で使用する SQL 言語の完全なリファレンスです。また、Adaptive Server Anywhere のシステム・テーブルとシステム・プロシージャについても説明しています。
- **『Adaptive Server Anywhere プログラミング・ガイド』** このマニュアルでは、C、C++、Java プログラミング言語を使用してデータベース・アプリケーションを構築、配備する方法につい

て説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。また、Adaptive Server Anywhere ADO.NET データ・プロバイダについても説明します。

- **『Adaptive Server Anywhere SNMP Extension Agent ユーザーズ・ガイド』** このマニュアルでは、Adaptive Server Anywhere SNMP Extension Agent を SNMP 管理アプリケーションとともに使用できるように設定して、Adaptive Server Anywhere データベースを管理できるようにする方法を説明します。

- **『Adaptive Server Anywhere エラー・メッセージ』** このマニュアルでは、Adaptive Server Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。

- **『SQL Anywhere Studio セキュリティ・ガイド』** このマニュアルでは、Adaptive Server Anywhere データベースのセキュリティ機能について説明します。Adaptive Server Anywhere 7.0 は、米国政府から TCSEC (Trusted Computer System Evaluation Criteria) の C2 セキュリティ評価を授与されています。このマニュアルには、Adaptive Server Anywhere の現在のバージョンを、C2 基準を満たした環境と同等の方法で実行することを望んでいるユーザにとって役に立つ情報が含まれています。

- **『Mobile Link 管理ガイド』** このマニュアルでは、モバイル・コンピューティング用の Mobile Link データ同期システムについてあらゆる角度から説明します。このシステムによって、Oracle、Sybase、Microsoft、IBM の単一データベースと、Adaptive Server Anywhere や Ultra Light の複数データベースの間でのデータ共有が可能になります。

- **『Mobile Link クライアント』** このマニュアルでは、Adaptive Server Anywhere リモート・データベースと Ultra Light リモート・データベースの設定を行い、これらを同期させる方法について説明します。

- **『Mobile Link サーバ起動同期ユーザーズ・ガイド』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期の開始を可能にする Mobile Link の機能です。

- **『Mobile Link チュートリアル』** このマニュアルには、Mobile Link アプリケーションの設定と実行を行う方法を説明するチュートリアルがいくつか用意されています。
- **『QAnywhere ユーザーズ・ガイド』** このマニュアルでは、Mobile Link QAnywhere について説明します。Mobile Link QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアントだけでなく、モバイル・クライアントや無線クライアント用のメッセージング・アプリケーションの開発と展開を可能にするメッセージング・プラットフォームです。
- **『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』** このマニュアルでは、Mobile Link 同期サーバから、または Adaptive Server Anywhere リモート・データ・アクセスによって、Adaptive Server Anywhere 以外の統合データベースにアクセスするための ODBC ドライバの設定方法について説明します。
- **『SQL Remote ユーザーズ・ガイド』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere Studio ヘルプ』** このマニュアルには、Sybase Central や Interactive SQL、その他のグラフィカル・ツールに関するコンテキスト別のヘルプが含まれています。これは、製本版マニュアル・セットには含まれていません。
- **『Ultra Light データベース・ユーザーズ・ガイド』** このマニュアルは、Ultra Light 開発者を対象としています。ここでは、Ultra Light データベース・システムの概要について説明します。また、すべての Ultra Light プログラミング・インタフェースに共通する情報を提供します。
- **Ultra Light のインタフェースに関するマニュアル** 各 Ultra Light プログラミング・インタフェースには、それぞれに対応するマニュアルを用意しています。これらのインタフェースは、RAD (

ラピッド・アプリケーション開発) 用の Ultra Light コンポーネントとして提供されているものと、C、C++、Java 開発用の静的インタフェースとして提供されているものがあります。

このマニュアル・セットの他に、PowerDesigner と InfoMaker には、独自のオンライン・マニュアル (英語版) がそれぞれ用意されています。

マニュアルの形式

SQL Anywhere Studio のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル** オンライン・マニュアルには、SQL Anywhere Studio の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

  Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

  UNIX オペレーティング・システムでオンライン・マニュアルにアクセスするには、SQL Anywhere のインストール・ディレクトリに保存されている HTML マニュアルを参照してください。

- **PDF 版マニュアル** SQL Anywhere の各マニュアルは、Adobe Acrobat Reader で表示できる PDF ファイルで提供されています。

  PDF 版マニュアルは、オンライン・マニュアルまたは Windows の [スタート] メニューから利用できます。

- **製本版マニュアル** 製本版マニュアルをご希望の方は、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。

表記の規則

この項では、このマニュアルで使用されている書体およびグラフィック表現の規則について説明します。

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

- **キーワード** SQL キーワードはすべて次の例に示す ALTER TABLE のように大文字で表記します。

  ALTER TABLE [*owner*.]*table-name*

- **プレースホルダ** 適切な識別子または式で置き換えられる項目は、次の例に示す *owner* や *table-name* のように表記します。

  ALTER TABLE [*owner*.]*table-name*

- **繰り返し項目** 繰り返し項目のリストは、次の例に示す *column-constraint* のように、リストの要素の後ろに省略記号 (ピリオド 3 つ …) を付けて表します。

  ADD *column-definition* [*column-constraint*, …]

  複数の要素を指定できます。複数の要素を指定する場合は、各要素間をカンマで区切る必要があります。

- **オプション部分** 文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

  この例では、角カッコで囲まれた *savepoint-name* がオプション部分です。角カッコは入力しないでください。

- **オプション** 項目リストから 1 つだけ選択するか、何も選択しなくてもよい場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。

  [**ASC** | **DESC**]

  この例では、ASC と DESC のどちらか 1 つを選択しても、どちらも選択しなくてもかまいません。角カッコは入力しないでください。

- **選択肢**　オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を中カッコで囲み、縦棒で区切ります。

  [**QUOTES** { **ON** | **OFF** }]

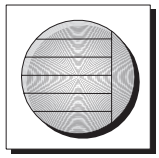
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

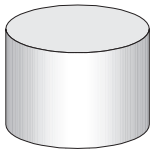
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

- Sybase Adaptive Server Anywhere などのデータベース・サーバ



- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- レプリケーションまたは同期のミドルウェア。ソフトウェアのこれらの部分は、データベース間のデータ共有を支援します。たとえば、Mobile Link 同期サーバ、SQL Remote Message Agent などがあげられます。

- プログラミング・インタフェース

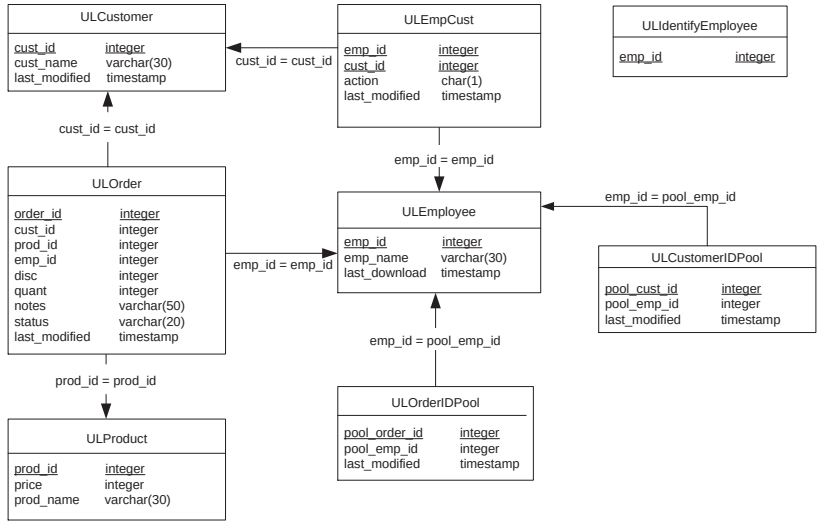
CustDB サンプル・データベース

Mobile Link と Ultra Light のマニュアルでは、多くの例で Ultra Light のサンプル・データベースが使用されています。

Ultra Light サンプル・データベースのリファレンス・データベースは、*custdb.db* という名前のファイルに保存され、SQL Anywhere ディレクトリのサブディレクトリ *Samples¥UltraLite¥CustDB* に置かれています。全面的にこのデータベースを使用して構築したアプリケーションも提供されています。

サンプル・データベースは、あるハードウェア販売会社の販売管理データベースです。データベースには、この販売会社の顧客、製品、営業戦力に関する情報が入っています。

次の図は、CustDB データベース内のテーブルと、各テーブル間の関係を示しています。



詳細情報の検索／フィードバックの提供

詳細情報の検索

詳しい情報やリソース (コード交換など) については、iAnywhere Developer Network (http://www.ianywhere.com/developer/) を参照してください。

ご質問がある場合や支援が必要な場合は、次に示す iAnywhere Solutions ニュースグループのいずれかにメッセージをお寄せください。

ニュースグループにメッセージをお送りいただく際には、ご使用の SQL Anywhere Studio バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng9 -v** と入力して確認できます。

ニュースグループは、ニュース・サーバ *forums.sybase.com* にあります (ニュースグループにおけるサービスは英語でのみの提供となります)。以下のニュースグループがあります。

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

ニュースグループに関するお断り

iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。

iAnywhere Solutions のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

マニュアルに関するご意見、ご提案は、SQL Anywhere ドキュメンテーション・チームの iasdoc@ianywhere.com 宛てに電子メールでお寄せください。このアドレスに送信された電子メールに返信はいたしませんが、お寄せいただいたご意見、ご提案は必ず読ませていただきます。

マニュアルまたはソフトウェアについてのフィードバックは、上記のニュースグループを通してお寄せいただいてもかまいません。

第 1 章

Ultra Light.NET の概要

この章の内容

この章では、Ultra Light.NET について説明します。ここでは、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light へようこそ」で説明した Ultra Light の機能について理解していることを前提としています。

Ultra Light.NET を使用したアプリケーションの作成の詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「Ultra Light.NET 開発の概要」を参照してください。

Ultra Light.NET の概要のチュートリアルについては、『Ultra Light.NET ユーザーズ・ガイド』> 「チュートリアル：Ultra Light.NET アプリケーションの構築」を参照してください。

Ultra Light と .NET

.NET Compact Framework は、Windows CE 用の Microsoft .NET ランタイム・コンポーネントです。複数のプログラミング言語をサポートします。Visual Basic.NET または C# を使用して、Ultra Light.NET を使用したアプリケーションを構築できます。

Ultra Light.NET には、次のネームスペースが用意されています。

- **iAnywhere.Data.UltraLite** このネームスペースには、Ultra Light への ADO.NET インタフェースが用意されています。業界標準モデルに基づいて構築でき、非常によく似た Adaptive Server Anywhere ADO.NET インタフェースへのマイグレーション・パスが提供されるという利点があります。
- **iAnywhere.UltraLite** このネームスペースは、オリジナルの Ultra Light.NET ネームスペースです。リリース 9.0.2 では推奨されなくなり、SQL Anywhere の次のメジャー・リリースでは使用できなくなります。

Ultra Light.NET ランタイムは、この両方のネームスペースをサポートしています。

Ultra Light.NET アプリケーションは、Windows CE および Windows XP に配備できます。Windows CE に配備する場合、Ultra Light.NET は .NET Compact Framework を必要とします。Windows XP に配備する場合は、.NET Framework を必要とします。また、ActiveSync 同期もサポートしています。

システムの稼働条件とサポートされるプラットフォーム

開発プラットフォーム

Ultra Light.NET を使用してアプリケーションを開発するには、以下が必要です。

- Microsoft Windows NT/2000/XP
- Visual Studio.NET または Visual Studio.NET 2003
- Windows CE デバイスでは、.NET Compact Framework バージョン 1.0.5000 以降

ターゲット・プラットフォーム

Ultra Light.NET は、次のターゲット・プラットフォームをサポートしています。

- Windows CE 3.0 以降で動作する Microsoft .NET Compact Framework 1.0.3705 以降 (Arm または MIPS プロセッサ上で動作する Pocket PC も含む)。Pocket PC 2002 エミュレータもサポートされています。
- Windows XP 上で動作する Microsoft .NET Compact Framework バージョン 1.0.5000 以降。

アプリケーション・コードと .NET Compact Framework に加えて、次のファイルを Windows CE デバイスまたは Windows XP がインストールされているコンピュータに配備してください。

- **iAnywhere.Data.UltraLite.dll**　iAnywhere.Data.UltraLite ADO.NET ネームスペースを含むアセンブリ。このファイルは、アプリケーションで iAnywhere.Data.UltraLite ネームスペースを使用する場合のみ必要です。
- **iAnywhere.Data.UltraLite.resources.dll**　iAnywhere.Data.UltraLite ADO.NET ネームスペースで必要とされるリソース。このファイルは、アプリケーションで iAnywhere.Data.UltraLite ネームスペースを使用する場合のみ必要です。

- **iAnywhere.UltraLite.dll** iAnywhere.UltraLite ネームスペースを含むアセンブリ。このファイルは、アプリケーションで iAnywhere.UltraLite ネームスペースを使用する場合のみ必要です。
- **iAnywhere.UltraLite.resources.dll** iAnywhere.UltraLite ネームスペースで必要とされるリソース。このファイルは、アプリケーションで iAnywhere.UltraLite ネームスペースを使用する場合のみ必要です。
- **ulnet9.dll** このファイルには、Ultra Light ランタイムが含まれます。このランタイムの異なるバージョンが、ターゲット・プラットフォームごとに用意されています。
- **Ultra Light スキーマ・ファイル** Ultra Light ランタイム・ライブラリは、スキーマ・ファイルの情報を使用してデータベース・スキーマを設定します。データベース・ファイルが作成されると、スキーマ・ファイルは不要になります。

詳細については、『SQL Anywhere Studio の紹介』> 「Ultra Light 開発プラットフォーム」と『SQL Anywhere Studio の紹介』> 「Ultra Light ターゲット・プラットフォーム」を参照してください。

Ultra Light.NET のアーキテクチャ

Ultra Light.NET ネームスペースの名前は、iAnywhere.UltraLite (ADO.NET インタフェース) と iAnywhere.UltraLite です。新しいアプリケーションでは、iAnywhere.Data.UltraLite ネームスペースを使用することをおすすめします。

次のリストは、iAnywhere.Data.UltraLite ADO.NET ネームスペースでよく使用される高レベル・クラスの一部を説明しています。

- **ULConnection** 各 ULConnection オブジェクトは、Ultra Light データベースとの接続を表します。ULConnection オブジェクトは 1 つまたは複数作成できます。
- **ULTable** 各 ULTable オブジェクトを使用して、単一テーブル内のデータにアクセスできます。
- **ULCommand オブジェクト** 各 ULCommand オブジェクトは、データベースに対して実行される SQL 文を格納します。
- **ULDataReader オブジェクト** 各 ULDataReader オブジェクトは、単一のクエリの結果セットを格納します。
- **ULSyncParms** ULSyncParms オブジェクトを使用して、Ultra Light データベースを Mobile Link 同期サーバと同期させます。

次のリストは、iAnywhere.UltraLite ネームスペースでよく使用される高レベル・クラスの一部を説明しています。

- **DatabaseManager**　アプリケーションごとに 1 つの DatabaseManager オブジェクトを作成します。
- **Connection**　各 Connection オブジェクトは、Ultra Light データベースとの接続を表します。Connection オブジェクトは 1 つまたは複数作成できます。
- **Table**　Table オブジェクトは、データベース内のデータへのアクセスを提供します。

- **PreparedStatement、ResultSet、ResultSetSchema** これらの動的 SQL オブジェクトを使って、動的 SQL 文の作成、クエリの記述、INSERT、UPDATE、および DELETE 文の実行、およびプログラムによるデータベースの結果セットの制御ができます。
- **SyncParms** SyncParms オブジェクトを使用して、Ultra Light データベースを Mobile Link 同期サーバと同期させます。

詳細については、ADO.NET インタフェースの『Ultra Light.NET ユーザーズ・ガイド』> 「iAnywhere.Data.UltraLite ネームスペース」およびコンポーネント・インタフェースの『Ultra Light.NET ユーザーズ・ガイド』> 「iAnywhere.UltraLite ネームスペース」を参照してください。

第 2 章

チュートリアル：Ultra Light.NET アプリケーションの構築

この章の内容

この章はチュートリアル形式で、Visual Studio .NET 2003 を使用して Ultra Light.NET アプリケーションを構築するプロセスを解説します。

概要

このチュートリアルでは、Visual Studio で Ultra Light.NET アプリケーションを構築するプロセスについて説明します。また、このチュートリアルでは iAnywhere.Data.UltraLite ネームスペースに用意されている ADO.NET インタフェースを使用します。

このチュートリアルには、Visual Basic .NET アプリケーションと Visual C# アプリケーションのコードが含まれています。

所要時間

このチュートリアルは、コードをカット・アンド・ペーストした場合、約 30 分で終了します。コードを自分で入力する場合、これよりかなり長い時間が必要です。

能力と経験

このチュートリアルは、次のことを前提にしています。

- C# プログラミング言語または Visual Basic .NET プログラミング言語に精通している
- Microsoft Visual Studio .NET 2003 がコンピュータにインストールされている
- Ultra Light スキーマ・ペインタを使用して、Ultra Light スキーマを作成する方法を知っている

  詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「レッスン 1：Ultra Light データベース・スキーマの作成」を参照してください。

目的

このチュートリアルの目的は、Visual Studio 環境での Ultra Light.NET アプリケーションの開発プロセスについて、知識と経験を得ることです。

レッスン 1：Visual Studio プロジェクトの作成

次の手順では、新しい Visual Studio アプリケーションを作成および設定します。プログラミング言語として Visual Basic.NET と C# のどちらも使用できます。

❖ Visual Studio プロジェクトを作成するには、次の手順に従います。

1　Visual Studio プロジェクトを作成します。

- Visual Studio .NET 2003 の [ファイル] メニューで、[新規作成] ー [プロジェクト] を選択して新規プロジェクトを作成します。

  [新しいプロジェクト] ウィンドウが表示されます。

- 左ウィンドウ枠で、[Visual Basic プロジェクト] フォルダまたは [Visual C# プロジェクト] フォルダを選択します。

- 右ウィンドウ枠で、[スマート デバイス アプリケーション] を選択し、プログラミング言語として Visual Basic と C# のどちらを選択したかに応じて、プロジェクトに **VBApp** または **CSApp** という名前を付けます。

- *c:¥tutorial¥uldotnet* の場所を入力し、[OK] をクリックします。

  [スマート デバイス アプリケーション] ウィザードが表示されます。

- Pocket PC をターゲット・プラットフォームとして選択し、プロジェクト・タイプとして [Windows アプリケーション] を選択します。[OK] をクリックします。

  [デザイン] 作業領域が表示され、フォームが表示されます。

2　プロジェクトに参照を追加します。

- iAnywhere.Data.UltraLite アセンブリと、関連するリソースをプロジェクトに追加します。

1. [プロジェクト] メニューで [参照の追加] を選択します。

   [参照の追加] ウィンドウが表示されます。

2. 利用可能な参照のリストから [**iAnywhere.Data.UltraLite (CE)**] を選択します。[選択] をクリックして選択済みコンポーネントのリストに追加します。

   この参照がリストに表示されない場合は、[参照] をクリックし、SQL Anywhere インストール環境の *ultralite¥UltraLite.NET¥ce* サブディレクトリで探します。*iAnywhere.Data.UltraLite.dll* を選択し、[開く] をクリックします。

3. 利用可能な参照のリストから [**iAnywhere.Data.UltraLite (CE) JA**] を選択します。[選択] をクリックして選択済みコンポーネントのリストに追加します。

   この参照がリストに表示されない場合は、[参照] をクリックし、SQL Anywhere インストール環境の *ultralite¥UltraLite.NET¥ce¥xx* サブディレクトリで探します。この場合、xx は言語を示す 2 文字の略語です。*iAnywhere.Data.UltraLite.resources.dll* を選択し、[開く] をクリックします。

4. [OK] をクリックし、アセンブリとリソースをプロジェクトに追加します。

- Ultra Light コンポーネントをプロジェクトにリンクします。

  ここで、リンクをコンポーネントに追加したことと、コンポーネントを開いていないことを確認します。

1. [プロジェクト] ー [既存項目の追加] をクリックし、SQL Anywhere インストール環境の *ultralite¥UltraLite.NET¥ce* サブディレクトリを参照します。
2. [ファイルの種類] ドロップダウンを [すべてのファイル] に設定し、DLL ファイルが表示されるようにします。

3. 使用している CE デバイスのプロセッサに対応するフォルダ (Pocket PC エミュレータの場合は x86 フォルダ) を開き、*ulnet9.dll* を選択します。*[開く] はクリックしません。*

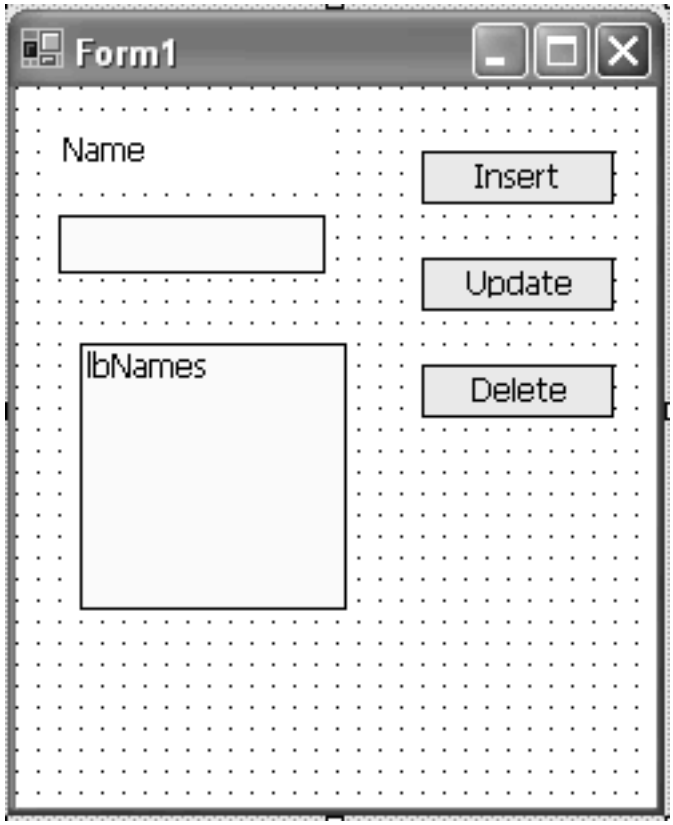
4. [開く] ボタンの矢印をクリックし、[**リンク ファイル**] を選択してプロジェクトにリンクします。

3　アプリケーションのフォームを作成します。

次のビジュアル・コンポーネントをフォームに追加します。

Type	**Name**	**Text**
Button	btnInsert	Insert
Button	btnUpdate	Update
Button	btnDelete	Delete
TextBox	txtName	
ListBox	lbNames	
Label	laName	Name

フォームは、次の図のようになります。



4　ソリューションを構築および配置します。

ソリューションを構築および配置し、Visual Studio.NET プロジェクトが正しく設定されたことを確認します。

a. [ビルド] メニューから [ソリューションのビルド] を選択します。プロジェクトが正常に構築されたことを確認します。

b. [デバッグ] メニューから [開始] を選択します。これによって、アプリケーションがデバイスまたはエミュレータに配置され、起動されます。プロジェクト名に応じて、アプリケーションは *¥Program Files¥VBApp* または *¥Program Files¥CSApp* に配置されます。

配置には時間がかかる場合があります。

c. アプリケーションがエミュレータまたはターゲット・デバイスに配置されたことを確認します。

d. [デバッグ] メニューから [デバッグの停止] を選択し、アプリケーションを閉じます。

レッスン 2：Ultra Light スキーマ・ファイルの作成

次の手順では、Ultra Light スキーマ・ペインタを使用して Ultra Light データベース・スキーマを作成します。

Ultra Light スキーマ・ペインタの使い方については、『Ultra Light データベース・ユーザーズ・ガイド』>「レッスン 1：Ultra Light データベース・スキーマの作成」を参照してください。

❖ スキーマ・ファイルを作成するには、次の手順に従います。

1 Ultra Light スキーマ・ペインタを起動します。

[スタート] メニューで、[プログラム] － [SQL Anywhere 9] － [Ultra Light] － [Ultra Light Schema Painter] を選択します。

2 Ultra Light Schema Painter を使用して、アプリケーションと同じディレクトリに次の特性を持つデータベース・スキーマを作成します。

- **スキーマ・ファイル名** *vbapp.usm* または *csapp.usm* (アプリケーションによって異なります)
- **照合順** デフォルト
- **大文字と小文字の区別** チェックしない
- **テーブル名** Names
- **カラム** 次の属性を持つカラムを Names テーブルに作成します。

カラム名	データ型 (サイズ)	Null 入力可	デフォルト値
ID	integer	いいえ	グローバル・オートインクリメント
Name	char(30)	いいえ	なし

- **プライマリ・キー** 昇順 ID

3　スキーマ・ファイルをアプリケーションにリンクします。

- Visual Studio .NET 2003 インタフェースから、[プロジェクト] ー [既存項目の追加] を選択します。
- チュートリアル・ディレクトリを参照し、必要に応じて *csapp.usm* または *vbapp.usm* を選択します。*[開く] はクリックしません。*
- [開く] ボタンの矢印をクリックし、[リンク ファイル] を選択します。
- ソリューション・エクスプローラでスキーマ・ファイルを右クリックし、[プロパティ] を選択します。プロパティのウィンドウ枠で、[ビルド アクション] プロパティを [**コンテンツ**] に設定します。

レッスン 3：データベースへの接続

次の手順では、Ultra Light データベースへの接続を確立する Ultra Light.NET アプリケーションにコントロールを追加します。

Ultra Light アプリケーションがデータベースに接続しようとしたとき、指定されたデータベース・ファイルが存在しない場合は、Ultra Light はスキーマ・ファイルを使用してデータベースを作成します。

このチュートリアルでは、C# アプリケーションを設計している場合はファイルがディレクトリ *c:¥tutorial¥uldotnet¥CSApp* にあり、Visual Basic アプリケーションを設計している場合はファイルがディレクトリ *c:¥tutorial¥uldotnet¥VBApp* にあることを前提とします。別の名前のディレクトリを作成した場合は、チュートリアルを通じてそのディレクトリを使用してください。

❖ アプリケーションに Ultra Light 接続を追加するには、次の手順に従います。

1 ConnectionParms コントロールをフォームに追加します。

ツールボックスの [UltraLite.NET CF] タブで [ULConnectionParms] を選択し、これをフォームに追加します。このコントロールの名前は、UlConnectionParms1 (Visual Basic) または ulConnectionParms1 (C#) になります。

SQL Anywhere Studio をインストールしてから再起動を行っていない場合、ここでエラー・メッセージが表示されることがあります。その場合は、マシンを再起動して問題を解決してください。

2 接続パラメータを指定します。

次のプロパティが設定されていることを確認します。

- **DatabaseOnCE** *¥Program Files¥CSApp¥CSApp.udb* (Visual C# アプリケーション) または *¥Program Files¥VBApp¥VBApp.udb* (Visual Basic アプリケーション)

- **SchmaOnCE** *¥Program Files¥CSApp¥CSApp.usm* (Visual C# アプリケーション) または *¥Program Files¥VBApp¥VBApp.usm* (Visual Basic アプリケーション)

3　フォームのソース・コードを開きます。

フォームをダブルクリックして、ソース・ファイルを開きます。

4　iAnywhere.Data.UltraLite ネームスペースをインポートします。

ファイルの先頭行に次の文を追加します。

```
//Visual C#
 using iAnywhere.Data.UltraLite;

'Visual Basic
 Imports iAnywhere.Data.UltraLite
```

5　グローバル変数をフォーム宣言に追加します。

Visual C# アプリケーションの場合は、フォーム・コンポーネントが記述されたコードの後、および最初のメソッド宣言の前に次のコードを追加します。

```
//Visual C#
 private ULConnection Conn;
 private int[] ids;
```

Visual Basic プロジェクトの場合は、フォーム・コンポーネントの宣言 (FriendsWithEvents 宣言) の後、および最初のメソッド宣言の前に次のコードを追加します。

```
'Visual Basic
 Dim Conn As ULConnection
 Dim ids() As Integer
```

これらの変数は次のように使用します。

- **ULConnection** Connection オブジェクトは、データベースへの接続時に実行されるすべてのアクションのルート・オブジェクトです。
- **ids** ids 配列を使用して、クエリの実行後に返される ID カラム値を格納します。

ListBox コントロール自体によって順序番号にアクセスできますが、ローが削除されると、これらの番号は ID カラムの値とは異なります。このため、ID カラム値を個別に格納してください。

6 フォームの空白領域をダブル・クリックして、Form1_Load メソッドを作成します。

このメソッドでは次のタスクを実行します。

- ULConnectionParms1 コントロールで設定されている接続パラメータを使用して、データベースへの接続を開きます。
- このチュートリアルで後ほど定義する RefreshListBox メソッドを呼び出します。
- エラーが発生した場合は、エラー・メッセージが出力されます。SQL エラーの場合は、エラー・コードも出力されます。エラー・コードの詳細については、『ASA エラー・メッセージ』> 「ASA エラー・メッセージ」を参照してください。

C# の場合は、次のコードをメソッドに追加します。

```
//Visual C#
 try {
     Conn = new ULConnection(
         ulConnectionParms1.ToString() );
     Conn.OpenWithCreate();
     Conn.DatabaseID = 1;
     RefreshListBox();
 }
 catch ( System.Exception t ) {
     MessageBox.Show( "Exception: " + t.Message);
 }
```

Visual Basic の場合は、次のコードをメソッドに追加します。

```
'Visual Basic
 Try
     Conn = New ULConnection( _
         UlConnectionParms1.ToString())
     Conn.OpenWithCreate()
```

```
    Conn.DatabaseID = 1
    RefreshListBox()
Catch
    MsgBox("Exception: " + err.Description)
End Try
```

7　プロジェクトをビルドします。

[ビルド] メニューから [ソリューションのビルド] を選択します。この段階で、RefreshListBox という名前が宣言されていないことを示す 1 つのエラーがレポートされます。このエラーを修正するには、次のレッスンで関数を追加します。

他のエラーが発生したら、それらを修正します。C# における大文字と小文字の違いなどの一般的なエラーをチェックします。たとえば、**Ultra Light**、**ULConnection** は大文字と小文字が正確に一致している必要があります。

レッスン 4：データの挿入、更新、削除

このレッスンでは、コードをアプリケーションに追加し、データベースのデータを変更します。次の手順では、動的 SQL を使用します。テーブル API を使用して同じテクニックを実行することもできます。

詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「コンポーネントと静的インタフェースの選択」または「テーブル API によるデータ・アクセスと操作」46 ページを参照してください。

次の手順では、リスト・ボックスを管理するサポート・メソッドを作成します。このメソッドは、残りの手順で作成されるデータ操作メソッドに必要です。

❖ リストボックスを管理するコードを追加するには、次の手順に従います。

1 フォームを右クリックし、[コードの表示] を選択します。

2 Form1 クラスのメソッドを追加し、リストボックスを更新してデータを移植します。このメソッドでは、次のタスクを実行します。

- リストボックスをクリアします。
- ULCommand オブジェクトをインスタンス化し、データベース内の Names テーブルを返す SELECT クエリに割り当てます。
- クエリを実行し、ULDataReader として結果セットを返します。
- 結果セットの行数と同じ長さの整数配列をインスタンス化します。
- ULDataReader で返される名前をリストボックスに移植し、ULDataReader で返される ID を整数配列に移植します。
- ULDataReader を閉じます。

- エラーが発生した場合は、エラー・メッセージが出力されます。SQL エラーの場合は、エラー・コードも出力されます。エラー・コードの詳細については、『ASA エラー・メッセージ』> 「ASA エラー・メッセージ」を参照してください。

C# の場合は、Form1 クラスのメソッドとして次のコードをアプリケーションに追加します。

```
//Visual C#
private void RefreshListBox(){
    try{
        long NumRows;
        int i = 0;
        lbNames.Items.Clear();
       using( ULCommand cmd = Conn.CreateCommand() {
            cmd.CommandText = "SELECT ID, Name FROM
Names";
             using( ULDataReader dr =
cmd.ExecuteReader()){
                dr.MoveBeforeFirst();
                NumRows = dr.RowCount;
                ids = new int[ NumRows ];
                while (dr.MoveNext())
                {
                    lbNames.Items.Add(
                    dr.GetString(1));
                    ids[ i ] = dr.GetInt32(0);
                    i++;
                }
            }
        }
    }
    catch( Exception err ){
        MessageBox.Show(
        "Exception in RefreshListBox: " +
err.Message );
    }
}
```

Visual Basic の場合は、Form1 クラスのメソッドとして次のコードをアプリケーションに追加します。

```
'Visual Basic
 Private Sub RefreshListBox()
     Try
         Dim cmd As ULCommand = Conn.CreateCommand()
         Dim i As Integer = 0
         lbNames.Items.Clear()
         cmd.CommandText = "SELECT ID, Name FROM
Names"
        Dim dr As ULDataReader = cmd.ExecuteReader()
         ReDim ids(dr.RowCount)
         While (dr.MoveNext)
             lbNames.Items.Add(dr.GetString(1))
             ids(i) = dr.GetInt32(0)
             i = i + 1
         End While
         dr.Close()
     Catch ex As Exception
         MsgBox(ex.ToString)
     End Try
 End Sub
```

3 プロジェクトをビルドします。

プロジェクトのビルドによってエラーが発生しないようにします。

❖ INSERT、UPDATE、DELETE を実装するには、次の手順に従います。

1 [挿入] ボタンをダブルクリックして、btnInsert_Click メソッドを作成します。このメソッドでは、次のタスクを実行します。

- ULCommand オブジェクトをインスタンス化し、テキストボックス内の値をデータベースに挿入する INSERT 文に割り当てます。
- 文を実行します。
- ULCommand オブジェクトを破棄します。
- リストボックスをリフレッシュします。

- エラーが発生した場合は、エラー・メッセージが出力されます。SQL エラーの場合は、エラー・コードも出力されます。エラー・コードの詳細については、『ASA エラー・メッセージ』> 「ASA エラー・メッセージ」を参照してください。

C# の場合は、次のコードをメソッドに追加します。

```
//Visual C#
 try {
     long RowsInserted;
     using( ULCommand cmd = Conn.CreateCommand() ) {
         cmd.CommandText =
             "INSERT INTO Names(name) VALUES (?)";
         cmd.Parameters.Add("", txtName.Text);
         RowsInserted = cmd.ExecuteNonQuery();
     }
     RefreshListBox();
 }
 catch( Exception err ) {
     MessageBox.Show("Exception: " + err.Message );
 }
```

Visual Basic の場合は、次のコードをメソッドに追加します。

```
'Visual Basic
 Try
     Dim RowsInserted As Long
     Dim cmd As ULCommand = Conn.CreateCommand()
     cmd.CommandText = "INSERT INTO Names(name)
VALUES (?)"
     cmd.Parameters.Add("", txtName.Text)
     RowsInserted = cmd.ExecuteNonQuery()
     cmd.Dispose()
     RefreshListBox()
 Catch
     MsgBox("Exception: " + Err.Description)
 End Try
```

2　[更新] ボタンをダブルクリックして、btnUpdate_Click メソッドを作成します。このメソッドでは、次のタスクを実行します。

- ULCommand オブジェクトをインスタンス化し、関連 ID に基づいて、テキストボックス内の値をデータベースに挿入する UPDATE 文に割り当てます。
- 文を実行します。
- ULCommand オブジェクトを破棄します。
- リストボックスをリフレッシュします。
- エラーが発生した場合は、エラー・メッセージが出力されます。SQL エラーの場合は、エラー・コードも出力されます。エラー・コードの詳細については、『ASA エラー・メッセージ』> 「ASA エラー・メッセージ」を参照してください。

C# の場合は、次のコードをメソッドに追加します。

```
//Visual C#
 try {
     long RowsUpdated;
     int updateID = ids[ lbNames.SelectedIndex ];
     using( ULCommand cmd = Conn.CreateCommand() ){
         cmd.CommandText =
            "UPDATE Names SET name = ? WHERE id = ?"
;
         cmd.Parameters.Add("", txtName.Text );
         cmd.Parameters.Add("", updateID);
         RowsUpdated = cmd.ExecuteNonQuery();
     }
     RefreshListBox();
 }
 catch( Exception err ) {
     MessageBox.Show(
         "Exception: " + err.Message);
 }
```

Visual Basic の場合は、次のコードをメソッドに追加します。

```
'Visual Basic
 Try
     Dim RowsUpdated As Long
     Dim updateID As Integer =
ids(lbNames.SelectedIndex)
     Dim cmd As ULCommand = Conn.CreateCommand()
```

```
        cmd.CommandText = "UPDATE Names SET name = ?
WHERE id = ?"
        cmd.Parameters.Add("", txtName.Text)
        cmd.Parameters.Add("", updateID)
        RowsUpdated = cmd.ExecuteNonQuery()
        cmd.Dispose()
        RefreshListBox()
    Catch
        MsgBox("Exception: " + Err.Description)
    End Try
```

3　[削除] ボタンをダブルクリックして、btnDelete_Click メソッドを作成します。コードを追加し、次のタスクを実行します。

- ULCommand オブジェクトをインスタンス化し、DELETE 文に割り当てます。DELETE 文は、整数配列 ids からの関連 ID に基づいて、選択した行をデータベースから削除します。
- 文を実行します。
- ULCommand オブジェクトを破棄します。
- リストボックスをリフレッシュします。
- エラーが発生した場合は、エラー・メッセージが出力されます。SQL エラーの場合は、エラー・コードも出力されます。エラー・コードの詳細については、『ASA エラー・メッセージ』> 「ASA エラー・メッセージ」を参照してください。

C# の場合は、次のコードをメソッドに追加します。

```
//Visual C#
 try{
        int deleteID = ids[lbNames.SelectedIndex];
        long RowsDeleted;
        using( ULCommand cmd = Conn.CreateCommand() ){
            cmd.CommandText =
                "DELETE From Names WHERE id = ?" ;
            cmd.Parameters.Add("", deleteID);
            RowsDeleted = cmd.ExecuteNonQuery ();
        }
```

```
        RefreshListBox();
    }
    catch( Exception err ) {
        MessageBox.Show("Exception: " + err.Message );
    }
```

Visual Basic の場合は、次のコードをメソッドに追加します。

```
'Visual Basic
 Try
     Dim deleteID As Integer =
ids(lbNames.SelectedIndex)
     Dim RowsDeleted As Long
     Dim cmd As ULCommand = Conn.CreateCommand()
    cmd.CommandText = "DELETE From Names WHERE id =
?"
     cmd.Parameters.Add("", deleteID)
     RowsDeleted = cmd.ExecuteNonQuery()
     cmd.Dispose()
     RefreshListBox()
 Catch
     MsgBox("Exception: " + Err.Description)
 End Try
```

4 アプリケーションを構築し、正常にコンパイルされることを確認します。

レッスン 5：アプリケーションの構築と配置

次の手順では、アプリケーションを構築し、リモート・デバイスに配置します。

❖ アプリケーションを配置するには、次の手順に従います。

1　ソリューションをビルドします。

アプリケーションがエラーなくビルドされることを確認します。

2　[デバッグ] ー [開始] を選択します。

これにより、アプリケーションが格納された実行可能ファイルがビルドされ、Pocket PC エミュレータに配置されます。このプロセスには時間がかかることがあります。アプリケーションを実行する前に .NET Compact Framework を配置する必要がある場合は、特に時間がかかります。

配置のトラブルシューティングのチェックリスト

エラーがレポートされた場合は、次のチェックリストを使用して、配置が正常に完了したかどうかをチェックします。

- アプリケーションが *¥Program Files¥appname* に配置されていることを確認します。*appname* は、レッスン 1 でアプリケーションに付けた名前 (CSApp または VBApp) です。
- スキーマ・ファイルがアプリケーションと同じディレクトリに配置されていることを確認します。
- アプリケーション・コード内のスキーマ・ファイルへのパスが正しいことを確認します。詳細については、「レッスン 3：データベースへの接続」16 ページを参照してください。
- スキーマ・ファイルをプロジェクトに追加するときに [リンクファイル] を選択し、[ビルド アクション] を [コンテンツ] に設定したことを確認します。これらの操作を行っていない場合は、ファイルがデバイスに配置されません。

- ターゲット・プラットフォームに対して適切なバージョンの *ulnet9.dll* への参照を追加したこと、または Windows CE インストーラを実行したことを確認します。エミュレータと実際のデバイス間で切り替える場合は、使用するこのライブラリのバージョンを変更します。詳細については、「レッスン 1：Visual Studio プロジェクトの作成」9 ページを参照してください。

- ステータスを保存しないでエミュレータを終了できます。再配置を行うと、必要なすべてのファイルがエミュレータにコピーされ、バージョン上の問題がないことが確認されます。

❖ アプリケーションをテストするには、次の手順に従います。

1 データベースにデータを挿入します。

テキスト・ボックスに名前を入力し、[挿入] をクリックします。名前がリストボックスに表示されます。

2 データベース内のデータを更新します。

リストボックスから名前を選択します。テキスト・ボックスに新しい名前を入力します。[更新] をクリックします。新しい名前が、リストボックスの古い名前の代わりに表示されます。

3 データベースからデータを削除します。

リストボックスから名前を選択します。[削除] をクリックします。名前はリストボックスに表示されなくなります。

これでチュートリアルが完了しました。

第 3 章

Ultra Light.NET 開発の概要

この章の内容

この章では、Ultra Light.NET を使用してアプリケーションを開発する方法について説明します。

チュートリアルについては、『Ultra Light.NET ユーザーズ・ガイド』>「チュートリアル：Ultra Light.NET アプリケーションの構築」を参照してください。

C# のコード・サンプル

この章のサンプル・コードは Microsoft C# で記述されています。サポートされている他の開発ツールを使用している場合は、このマニュアルの記述を適切に読み替えてください。

Ultra Light データベース・スキーマ

データベース・スキーマは、データベースに関する記述です。データベース・スキーマは、データベース内のテーブル、インデックス、キー、パブリケーションの集合であり、それらの間のすべての関係を示します。

Ultra Light データベースのスキーマを直接変更しないでください。代わりに、アプリケーションに組み込まれた Ultra Light 機能を使用して、スキーマ・ファイル (*.usm*) を作成し、作成したファイルからデータベース・スキーマをアップグレードします。

スキーマ・ファイルは、データベースを最初に作成するときにも使用され、データベースの構造を指定します。

Ultra Light データベース・スキーマ・ファイルの作成

Ultra Light スキーマ・ペインタまたは *ulinit* ユーティリティを使用して、Ultra Light スキーマ・ファイルを作成できます。

- **Ultra Light スキーマ・ペインタ**　Ultra Light スキーマ・ペインタは、Ultra Light スキーマ・ファイルの作成と編集に使用するグラフィカル・ユーティリティです。

  スキーマ・ペインタを起動するには、[スタート] － [プログラム] － [SQL Anywhere 9] － [Ultra Light] － [Ultra Light Schema Painter] を選択するか、Windows エクスプローラでスキーマ・ファイル (*.usm*) をダブルクリックします。

  Ultra Light スキーマ・ペインタの使用方法の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「レッスン 1：Ultra Light データベース・スキーマの作成」を参照してください。

- **ulinit ユーティリティ**　Adaptive Server Anywhere データベース管理システムがある場合は、*ulinit* コマンド・ライン・ユーティリティを使用して Ultra Light スキーマ・ファイルを生成できます。

ulinit ユーティリティの使用の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「ulinit ユーティリティ」を参照してください。

データベース・スキーマのアップグレード

既存のデータベース構造を修正するには、ULDatabaseSchema.ApplyFile メソッドを使用します。ほとんどの場合、データ損失は起こりません。ただし、カラムを削除したり、カラムのデータ型が互換性のない型に変更された場合などは、データ損失が起こる場合があります。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「メンバ ULDatabaseSchema」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』> 「メンバ DatabaseSchema」(iAnywhere.UltraLite ネームスペース) を参照してください。

例

次のコードは、ApplyFile メソッドを使用して新しいスキーマ・ファイルを適用します。ここでは、すでにデータベースに接続していることを前提としています (「データベースへの接続」32 ページを参照)。

```
// iAnywhere.Data.UltraLite namespace
 ULDatabaseSchema schema = conn.Schema;
 ULConnectionParms parms = new ULConnectionParms();
 parms.SchemaOnDesktop =
@"c:¥tutorial¥tutcustomer.usm";
 try {
   schema.ApplyFile(parms );
 } catch {
   // handle error
 }

// iAnywhere.UltraLite namespace
 DatabaseSchema schema = conn.Schema;
 SchemaParms parms = new SchemaParms();
 parms.SchemaOnDesktop =
@"c:¥tutorial¥tutcustomer.usm";
 try {
   schema.ApplyFile( parms );
 } catch {
   // handle error
 }
```

データベースへの接続

Ultra Light アプリケーションをデータベースに接続しないと、データベースのデータを操作できません。この項では、Ultra Light データベースに接続する方法について説明します。

Connection オブジェクトの使用

次に示す Connection オブジェクトのプロパティは、アプリケーションのグローバルな動作を管理します。

- **コミット動作** デフォルトでは、Ultra Light.NET アプリケーションは AutoCommit モードに設定されています。insert 文、update 文、delete 文はすべて、すぐにデータベースにコミットされます。ULConnection.BeginTransaction() を使用して、アプリケーションにトランザクションを構築できます。

  詳細については、「Ultra Light でのトランザクション処理」54 ページを参照してください。

- **ユーザ認証** Grant と Revoke の接続許可メソッドを使用すると、アプリケーションのユーザ ID とパスワードをデフォルト値の DBA と SQL から別の値に変更できます。各アプリケーションは、最大 4 つのユーザ ID を保持できます。

  詳細については、「ユーザ認証」58 ページを参照してください。

- **同期** Connection オブジェクトから、同期を管理するオブジェクトのセットにアクセスできます。

  詳細については、「Ultra Light アプリケーションの同期」59 ページを参照してください。

- **テーブル** Ultra Light テーブルには、Connection オブジェクトのメソッドを使用してアクセスします。

  詳細については、「テーブル API によるデータ・アクセスと操作」46 ページを参照してください。

- **コマンド** 動的 SQL 文の実行を処理し、結果セットをナビゲートするために提供されるオプジェクトのセット。

詳細については、「動的 SQL を使用したデータへのアクセスと操作」38 ページを参照してください。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULConnection クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「Connection クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

マルチスレッド・アプリケーション

Connection オブジェクトと、このオブジェクトから作成されるすべてのオブジェクトは、それぞれ単一のスレッドで使用してください。アプリケーションが Ultra Light データベースにアクセスするのに複数のスレッドを必要とする場合は、スレッドごとに個別の接続が必要です。

❖ Ultra Light データベースに接続するには、次の手順に従います。

1. DatabaseManager オブジェクトを作成および初期化します。

   iAnywhere.Data.UltraLite ネームスペースを使用する場合は、この手順は必要ありません。

   DatabaseManager オブジェクトは、オブジェクト階層のルートにあります。DatabaseManager オブジェクトは、1 つのアプリケーションに 1 つだけ作成します。多くの場合、DatabaseManager オブジェクトは、アプリケーションに対してグローバルに宣言するのが最も効果的です。

   ```
   // iAnywhere.UltraLite namespace
   DatabaseManager dbMgr = new DatabaseManager();
   ```

   詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「DatabaseManager クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

2. Connection オブジェクトを宣言します。

   ほとんどのアプリケーションは、Ultra Light データベースへの単一接続を使用し、接続を開いたままにしておきます。複数の接続が必要になるのは、マルチスレッド・データ・アク

セスの場合だけです。そのため、Connection オブジェクトは、アプリケーションに対してグローバルに宣言するのが最も効果的です。

```
// iAnywhere.Data.UltraLite namespace
ULConnection conn;

// iAnywhere.UltraLite namespace
Connection conn;
```

3 既存のデータベースへの接続を開きます。指定されたデータベース・ファイルが存在しない場合は、新しいデータベースを作成します。

ほとんどの Ultra Light アプリケーションは、データベース・ファイルではなくスキーマ・ファイルを配備し、最初の接続時に Ultra Light にデータベース・ファイルを作成させます。次のコードは、既存のデータベースへの接続を試みます。データベース・ファイルが存在しない場合は、アプリケーションがデータベース・ファイルを作成します。

```
// iAnywhere.Data.UltraLite namespace
ULConnectionParms parms = new ULConnectionParms();
// specify the location of the database file
parms.DatabaseOnDesktop = "mydata.udb";
// specify the location of the schema file
parms.SchemaOnDesktop = "mydata.usm";
conn = new ULConnection( parms.ToString() );
conn.OpenWithCreate();

// iAnywhere.UltraLite namespace
ConnectionParms parms = new ConnectionParms();
// specify the location of the database file
parms.DatabaseOnDesktop = "mydata.udb";
try {
  conn = dbMgr.OpenConnection( parms );
} catch( SQLException econn ) {
  if( econn.GetErrorCode() ==
    SQLCode.SQLE_ULTRALITE_DATABASE_NOT_FOUND ){
    CreateParms parms = new CreateParms();
    parms.DatabaseOnDesktop = "mydata.udb";
```

```
        parms.Schema.SchemaOnDesktop = "mydata.usm";
        try {
          conn = dbMgr.CreateDatabase( parms );
      }
    }
```

例

次のコードは、Ultra Light データベース *mydata.udb* への接続を開きます。データベースが存在しない場合は、*mydata.usm* という名前の Ultra Light スキーマ・ファイルを使用して作成されます。

```
// iAnywhere.Data.UltraLite namespace
 ULConnectionParms parms = new ULConnectionParms();
 parms.DatabaseOnDesktop = "mydata.udb";
 parms.SchemaOnDesktop = "mydata.usm";
 conn = new ULConnection( parms.ToString() );
 conn.InfoMessage += new ULInfoMessageEventHandler(
     InfoMessageEventHandler );
 conn.OpenWithCreate();

// iAnywhere.UltraLite namespace
 CreateParms parms = new CreateParms();
 parms.DatabaseOnDesktop = "mydata.udb";
 parms.Schema.SchemaOnDesktop = "mydata.usm";
 try {
   conn = dbMgr.OpenConnection( parms );
   System.Console.WriteLine(
     "Connected to an existing database." );
 }
 catch( SQLException econn ) {
   if( econn.GetErrorCode() ==
       SQLCode.SQLE_ULTRALITE_DATABASE_NOT_FOUND ){
     conn = dbMgr.CreateDatabase( parms );
     System.Console.WriteLine(
       "Connected to a new database." );
   } else {
     throw econn;
   }
 }
```

上記のコードでは、データベースが新しいかを検出するために、以下のように定義される InfoMessage ハンドラを使用します。

```
// iAnywhere.Data.UltraLite namespace
 protected void InfoMessageHandler(
     object obj, ULInfoMessageEventArgs args )
 {
   if( args.NativeError ==
ULSQLCode.SQLE_DATABASE_CREATED ){
     System.Console.WriteLine( args.Message );
   }
 }
```

暗号化と難読化

Ultra Light.NET を使用して Ultra Light データベースを暗号化または難読化できます。

暗号化

暗号化を使用してデータベースを作成するには、ULConnectionParms.EncryptionKey プロパティを使用して暗号化キーを指定し、次に OpenWithCreate メソッドを呼び出します。OpenWithCreate メソッドを呼び出すと、データベースが作成され、指定されたキーを使用して暗号化されます。

詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「EncryptionKey 接続パラメータ」を参照してください。

暗号化キーを変更するには、Connection.ChangeEncryptionKey メソッドを使用して新しい暗号化キーを指定します。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「ULConnection クラス」と 『Ultra Light.NET ユーザーズ・ガイド』> 「ULConnectionParms クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』> 「Connection クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

データベースが暗号化された後は、データベースへの接続で正しい暗号化キーを指定する必要があります。そうしないと、接続は失敗します。

難読化

データベースを難読化するには、作成パラメータとして obfuscate=1 を指定します。

データベースの暗号化の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light データベースの暗号化」を参照してください。

動的 SQL を使用したデータへのアクセスと操作

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、動的 SQL を使用したデータ・アクセスについて説明します。

テーブル API の使用方法については、「テーブル API によるデータ・アクセスと操作」46 ページを参照してください。

この項では、動的 SQL を使用して次の操作を行う方法について説明します。

- ローの挿入、削除、更新
- 結果セットのローの取得
- 結果セットのローのスクロール

この項では、SQL 言語そのものについては説明しません。動的 SQL 機能の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「動的 SQL」を参照してください。

SQL 操作に必要な一連の操作の概要については、『Ultra Light データベース・ユーザーズ・ガイド』>「動的 SQL の使用」を参照してください。

データ操作 : INSERT、UPDATE、DELETE

Ultra Light では、SQL データ操作言語の操作を実行できます。この操作は、ULCommand.ExecuteNonQuery メソッド (iAnywhere.Data.UltraLite ネームスペース) または PreparedStatement.executeStatement メソッド (iAnywhere.UltraLite ネームスペース) を使用して実行されます。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULCommand クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「PreparedStatement クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

iAnywhere.Data.UltraLite コマンドで使用するパラメータ
SQL 文のパラメータのプレースホルダは、? 文字を使用して指定します。INSERT、UPDATE、DELETE では必ず、コマンドの**パラメータ**での並び順に従ってそれぞれの ? が参照されます。たとえば、最初の ? は 0、2 番目の ? は 1 のようになります。

iAnywhere.UltraLite 準備文で使用するパラメータ
SQL 文のパラメータのプレースホルダは、? 文字を使用して指定します。INSERT、UPDATE、DELETE で、準備文での並び順に従ってそれぞれの ? が参照されます。たとえば、最初の ? は 1、2 番目の ? は 2 のようになります。

❖ ローを挿入するには、次の手順に従います。

1　ULCommand を宣言します。

```
// iAnywhere.Data.UltraLite namespace
ULCommand cmd;

// iAnywhere.UltraLite namespace
PreparedStatement prepStmt;
```

2　SQL 文を ULCommand オブジェクトに割り当てます。

```
// iAnywhere.Data.UltraLite namespace
cmd = conn.CreateCommand();
cmd.Command =
   "INSERT INTO MyTable(MyColumn) values (?)";

// iAnywhere.UltraLite namespace
prepStmt = conn.PrepareStatement(
   "INSERT INTO MyTable(MyColumn) values (?)");
```

3　文の入力パラメータ値を割り当てます。

次のコードは、文字列パラメータを示します。

```
// iAnywhere.Data.UltraLite namespace
String newValue;
// assign value
cmd.add("", newValue);

// iAnywhere.UltraLite namespace
String newValue;
// assign value
prepStmt.SetStringParameter(1, newValue);
```

4 文を実行します。

戻り値は、文に影響されたローの数を示します。

```
// iAnywhere.Data.UltraLite namespace
int rowsInserted = cmd.ExecuteNonQuery();

// iAnywhere.UltraLite namespace
long rowsInserted = prepStmt.ExecuteStatement();
```

5 トランザクションを使用している場合 (iAnywhere.UltraLite ネームスペースの AutoCommit が false) は、変更内容をコミットします。

```
// iAnywhere.Data.UltraLite namespace
cmd.Transaction.Commit();

// iAnywhere.UltraLite namespace
conn.Commit();
```

❖ ローを更新するには、次の手順に従います。

1 ULCommand を宣言します。

```
// iAnywhere.Data.UltraLite namespace
ULCommand cmd;

// iAnywhere.UltraLite namespace
PreparedStatement prepStmt;
```

2 文を ULCommand オブジェクトに割り当てます。

```
// iAnywhere.Data.UltraLite namespace
cmd = conn.CreateCommand();
cmd.Command =
    "UPDATE MyTable SET MyColumn1 = ? WHERE
MyColumn2 = ?";

// iAnywhere.UltraLite namespace
prepStmt = conn.PrepareStatement(
    "UPDATE MyTable SET MyColumn1 = ? WHERE
MyColumn2 = ?");
```

3　文の入力パラメータ値を割り当てます。

```
// iAnywhere.Data.UltraLite namespace
String newValue;
String oldValue;
// assign values
cmd.add("", newValue);
cmd.add("", oldValue);

// iAnywhere.UltraLite namespace
String newValue;
String oldValue;
// assign values
prepStmt.SetStringParameter(1, newValue);
prepStmt.SetStringParameter(2, oldValue);
```

4　文を実行します。

```
// iAnywhere.Data.UltraLite namespace
int rowsUpdated = cmd.ExecuteNonQuery();

// iAnywhere.UltraLite namespace
long rowsUpdated = prepStmt.ExecuteStatement();
```

5　トランザクションを使用している場合 (iAnywhere.UltraLite ネームスペースの AutoCommit が false) は、変更内容をコミットします。

```
// iAnywhere.Data.UltraLite namespace
cmd.Transaction.Commit();

// iAnywhere.UltraLite namespace
conn.Commit();
```

❖ ローを削除するには、次の手順に従います。

1 ULCommand を宣言します。

```
// iAnywhere.Data.UltraLite namespace
ULCommand cmd;

// iAnywhere.UltraLite namespace
PreparedStatement prepStmt;
```

2 文を ULCommand オブジェクトに割り当てます。

```
// iAnywhere.Data.UltraLite namespace
cmd = conn.CreateCommand();
cmd.Command =
   "DELETE FROM MyTable WHERE MyColumn = ?";

// iAnywhere.UltraLite namespace
prepStmt = conn.PrepareStatement(
   "DELETE FROM MyTable WHERE MyColumn = ?");
```

3 文の入力パラメータ値を割り当てます。

```
// iAnywhere.Data.UltraLite namespace
String deleteValue;
// assign value
cmd.add("", deleteValue);

// iAnywhere.UltraLite namespace
String deleteValue;
prepStmt.SetStringParameter(1, deleteValue);
```

4 文を実行します。

```
// iAnywhere.Data.UltraLite namespace
int rowsDeleted = cmd.ExecuteNonQuery();

// iAnywhere.UltraLite namespace
long rowsDeleted = prepStmt.ExecuteStatement();
```

5 トランザクションを使用している場合 (iAnywhere.UltraLite ネームスペースの AutoCommit が false) は、変更内容をコミットします。

```
// iAnywhere.Data.UltraLite namespace
cmd.Transaction.Commit();
```

```
// iAnywhere.UltraLite namespace
 conn.Commit();
```

データの取得 : SELECT

SELECT 文を使用すると、データベースから情報を取り出すことができます。この項では、SELECT 文の実行方法と返される結果セットの処理方法について説明します。

❖ SELECT 文を実行するには、次の手順に従います。

1　クエリを保持する ULCommand オブジェクトを宣言します。

```
// iAnywhere.Data.UltraLite namespace
ULCommand cmd;

// iAnywhere.UltraLite namespace
PreparedStatement prepStmt;
```

2　このオブジェクトに文を割り当てます。

```
// iAnywhere.Data.UltraLite namespace
cmd = conn.CreateCommand();
cmd.Command =
    "SELECT MyColumn FROM MyTable";

// iAnywhere.UltraLite namespace
prepStmt = conn.PrepareStatement(
"SELECT MyColumn FROM MyTable");
```

3　文を実行します。

以下のコードでは、SELECT のクエリ結果に文字列が含まれています。これはコマンド・プロンプトへ出力されます。

```
// iAnywhere.Data.UltraLite namespace
ULDataReader customerNames =
prepStmt.ExecuteReader();
 while( customerNames.MoveNext() ) {
   for ( int i = 0;
     i < customerNames.GetFieldCount();
     i++ ) {
```

```
      System.Console.Write(
      customerNames.GetString( i ) + " " );
    }
    System.Console.WriteLine();
  }

// iAnywhere.UltraLite namespace
 ResultSet customerNames = prepStmt.ExecuteQuery();
 customerNames.MoveBeforeFirst();
 while( customerNames.MoveNext() ) {
   for ( int i = 1;
     i <= customerNames.Schema.GetColumnCount();
     i++ ) {
     System.Console.Write(
     customerNames.GetString( i ) + " " );
   }
   System.Console.WriteLine();
 }
```

動的 SQL 結果セットのナビゲーション

ULDataReader (iAnywhere.Data.UltraLite ネームスペース) または ResultSet (iAnywhere.UltraLite ネームスペース) オブジェクトに関連したメソッドを使用して、結果セット内をナビゲートすることができます。

結果セット・オブジェクトは、結果セットをナビゲーションする次のメソッドを提供します。

- **MoveAfterLast()** 最後のローの後に移動します。
- **MoveBeforeFirst()** 最初のローの前に移動します。
- **MoveFirst()** 最初のローに移動します。
- **MoveLast()** 最後のローに移動します。
- **MoveNext()** 次のローに移動します。
- **MovePrevious()** 前のローに移動します。
- **MoveRelative(offset)** 現在のローを基準にして、オフセットが指定した数だけローを移動します。オフセット値を正の値で指定すると、現在の結果セットのカーソル位置から前方に

移動します。負の値で指定すると後方に移動します。オフセット値が 0 の場合、カーソルは移動しませんが、ロー・バッファが再配置されます。

結果セット・スキーマの説明

ULDataReader.GetSchemaTable メソッドと ULDataReader.Schema プロパティ (iAnywhere.Data.UltraLite ネームスペース) または ResultSet.Schema プロパティ (iAnywhere.UltraLite ネームスペース) を使用して、カラム名、カラムの総数、カラム・スケール、カラム・サイズ、カラム SQL 型など、結果セットに関する情報を取得できます。

例

次のサンプル・コードは、ULDataReader.Schema と ResultSet.Schema プロパティを使用して、スキーマ情報をコンソール・ウィンドウに表示する方法を示しています。

```
// iAnywhere.Data.UltraLite namespace
 for ( int i = 0;
       i < MyResultSet.Schema.GetColumnCount();
       i++ ) {
    System.Console.WriteLine(
       MyResultSet.Schema.GetColumnName(i) + " " +
       MyResultSet.Schema.GetColumnSQLType(i)
    );
 }

// iAnywhere.UltraLite namespace
 for ( int i = 1;
       i <= MyResultSet.Schema.GetColumnCount();
       i++ ) {
    System.Console.WriteLine(
       MyResultSet.Schema.GetColumnName(i) + " " +
       MyResultSet.Schema.GetColumnSQLType(i)
    );
 }
```

テーブル API によるデータ・アクセスと操作

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、テーブル API を使用したデータ・アクセスについて説明します。

動的 SQL については、「動的 SQL を使用したデータへのアクセスと操作」38 ページを参照してください。

この項では、テーブル API を使用して次の操作を行う方法について説明します。

- テーブルのローのスクロール
- 現在のローの値へのアクセス
- find メソッドと lookup メソッドを使用したテーブルのローの検索
- ローの挿入、削除、更新

テーブルのローのナビゲーション

Ultra Light.NET は、幅広いナビゲーション作業を行うため、テーブルをナビゲーションする方法を多数提供します。

テーブル・オブジェクトは、テーブルをナビゲーションする次のメソッドを提供します。

- **MoveAfterLast()** 最後のローの後に移動します。
- **MoveBeforeFirst()** 最初のローの前に移動します。
- **MoveFirst()** 最初のローに移動します。
- **MoveLast()** 最後のローに移動します。
- **MoveNext()** 次のローに移動します。
- **MovePrevious()** 前のローに移動します。

- **MoveRelative(offset)** 現在のローを基準にして、オフセットが指定した数だけローを移動します。オフセット値を正の値で指定すると、現在のテーブルのカーソル位置から前方に移動します。負の値で指定すると後方に移動します。オフセット値が 0 の場合、カーソルは移動しませんが、ロー・バッファが再配置されます。

例

次のサンプル・コードは、MyTable テーブルを開き、各ローの MyColumn カラムの値を表示します。

```
// iAnywhere.Data.UltraLite namespace
 ULTable t = conn.ExecuteTable( "MyTable" );
 int colID = t.GetOrdinal( "MyColumn" );
 while ( t.MoveNext() ){
    System.Console.WriteLine( t.GetString( colID ) );
 }

// iAnywhere.UltraLite namespace
 Table t = conn.GetTable( "MyTable" );
 short colID = t.Schema.GetColumnID( "MyColumn" );
 t.Open();
 t.MoveBeforeFirst();
 while ( t.MoveNext() ){
    System.Console.WriteLine( t.GetString( colID ) );
 }
```

テーブル・オブジェクトを開くと、テーブルのローがアプリケーションに公開されます。デフォルトでは、ローはプライマリ・キー値の順に並んでいますが、テーブルを開くときにインデックスを指定すると特定の順序でローにアクセスできます。

例

次のコードは、ix_col インデックスで順序付けられた MyTable テーブルの最初のローに移動します。

```
// iAnywhere.Data.UltraLite namespace
 ULTable t = conn.ExecuteTable( "MyTable", "ix_col" );
 t.MoveFirst();

 // iAnywhere.UltraLite namespace
 Table t = conn.GetTable( "MyTable" );
 t.Open( "ix_col" );
 t.MoveFirst();
```

詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「ULTable クラス」と 『Ultra Light.NET ユーザーズ・ガイド』> 「ULTableSchema クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』> 「Table クラス」と 『Ultra Light.NET ユーザーズ・ガイド』> 「TableSchema クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

Ultra Light のモードの使用

Ultra Light モードは、バッファ内の値の使用目的を指定します。Ultra Light には、デフォルト・モードに加えて、次の 4 つの操作モードがあります。

- **挿入モード** Insert メソッドを呼び出すと、バッファ内のデータが新しいローとしてテーブルに追加されます。
- **更新モード** Update メソッドを呼び出すと、現在のローがバッファ内のデータに置き換えられます。
- **検索モード** find メソッドの 1 つが呼び出されたときに、値がバッファ内のデータに正確に一致するローの検索に使用されます。
- **ルックアップ・モード** いずれかの lookup メソッドが呼び出されたときに、バッファ内のデータと一致するか、それより大きい値のローを検索します。

現在のローの値へのアクセス

Table オブジェクトは、次のいずれかの位置に常に置かれています。

- テーブルの最初のローの前
- テーブルのいずれかのローの上
- テーブルの最後のローの後ろ

Table オブジェクトがローの上に置かれている場合は、そのデータ型に適したメソッド・セットを使用して、各カラムの値を取得したり、変更したりできます。

カラム値の取得

Table オブジェクトは、カラム値を取得するメソッド・セットを提供します。これらのメソッドは、カラム ID を引数として取ります。

例

次のコードは、lname カラムの値を取得します。このカラムの値は文字列です。

```
// iAnywhere.Data.UltraLite namespace
 int lname = t.GetOrdinal( "lname" );
 string lastname = t.GetString( lname );

// iAnywhere.UltraLite namespace
 short lname = t.Schema.GetColumnID( "lname" );
 string lastname = t.GetString( lname );
```

次のコードは、cust_id カラムの値を取得します。このカラムの値は整数です。

```
// iAnywhere.Data.UltraLite namespace
 int cust_id = t.GetOrdinal( "cust_id" );
 int id = t.GetInt( cust_id );

// iAnywhere.UltraLite namespace
 short cust_id = t.Schema.GetColumnID( "cust_id" );
 int id = t.GetInt( cust_id );
```

カラム値の変更

値を取り出すメソッド以外に、値を設定するメソッドもあります。値を設定するメソッドは、カラム ID と値を引数として取ります。

例

たとえば、次のコードは、lname カラムの値を Kaminski に設定します。

```
t.SetString( lname, "Kaminski" );
```

これらのプロパティへの値の割り当てによって、データベース内のデータの値が変更されることはありません。現在の位置がテーブルの最初のローの前でも、テーブルの最後のローの後ろであっても、プロパティに値を割り当てることができます。ただし、これらの位置に現在のローが置かれている場合に、たとえば、変数へのプロパティの割り当てなどによって、データにアクセスしようとするとエラーになります。

```
// This code is incorrect
 t.MoveBeforeFirst();
 id = t.GetInt( cust_id );
```

値のキャスト

選択するメソッドは、割り当てるデータ型に一致させてください。データ型に互換性がある場合は、Ultra Light が自動的にデータベースのデータ型をキャストするため、getString メソッドを使用して整数値を文字列変数にフェッチしたりできます。

find と lookup を使用したローの検索

Ultra Light には、データを操作するための操作モードがいくつかあります。これらのモードのうちの 2 つ (検索モードとルックアップ・モード) は、検索に使用されます。Table オブジェクトには、テーブル内の特定のローを検索するために、これらのモードに対応するメソッドがあります。

注意
find メソッドや lookup メソッドを使用して検索されるカラムは、テーブルを開くのに使用されたインデックスにあることが必要です。

- **find メソッド** Table オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と正確に一致する最初のローに移動します。検索値が見つからない場合は、最初のローの前、または最後のローの後ろに位置設定されます。
- **lookup メソッド** Table オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と一致するか、それより大きい値の最初のローに移動します。

❖ ローを検索するには、次の手順に従います。

1 検索モードまたはルックアップ・モードを開始します。

モードを開始するには、テーブル・オブジェクトのメソッドを呼び出します。たとえば、次のコードは検索モードを開始します。

```
t.FindBegin();
```

2　検索値を設定します。

検索値は、現在のローの値を設定することで設定します。これらの値の設定は、データベースではなく、現在のローを保持しているバッファにのみ影響します。たとえば、次のコードは、バッファの値を Kaminski に設定します。

```
// iAnywhere.Data.UltraLite namespace
 int lname = t.GetOrdinal( "lname" );
 t.SetString( lname, "Kaminski" );

// iAnywhere.UltraLite namespace
 short lname = t.Schema.GetColumnID( "lname" );
 t.SetString( lname, "Kaminski" );
```

3　ローを検索します。

適切なメソッドを使用して検索を実行します。たとえば、次の指示は、現在のインデックスで指定された値と正確に一致する最初のローを検索します。

マルチカラム・インデックスの場合、最初のカラムの値が常に使用され、ほかのカラムは省略できます。

```
tCustomer.FindFirst();
```

4　ローの次のインスタンスを検索します。

適切なメソッドを使用して検索を実行します。検索操作では、FindNext() で、インデックス内の次のインスタンスを検索します。ルックアップ操作では、MoveNext() で次のインスタンスを検索します。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULTable クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「Table クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

ローの更新

次の手順では、ローを更新します。

❖ **ローを更新するには、次の手順に従います。**

1 更新するローに移動します。

テーブルをスクロールするか、find メソッドと lookup メソッドを使用してテーブルを検索し、ローに移動できます。

2 更新モードを開始します。

たとえば、次の指示は、t 上で更新モードを開始します。

```
t.BeginUpdate();
```

3 更新するローの新しい値を設定します。たとえば、次の指示は、バッファ内の id カラムを 3 に設定します。

```
t.SetInt( id , 3);
```

4 Update を実行します。

```
t.Update();
```

更新操作が終了すると、更新したローが現在のローになります。Table オブジェクトを開いたときに指定したインデックスのカラム値を変更した場合は、現在のローの定義が解除されます。

デフォルトでは、Ultra Light.NET は AutoCommit モードで動作するため、更新は永続的な記憶領域のローに即時適用されます。AutoCommit モードを無効にした場合は、コミット操作を実行するまで、更新は適用されません。詳細については、「Ultra Light でのトランザクション処理」54 ページを参照してください。

警告
ローのプライマリ・キーは更新できません。代わりに、ローを削除して新しいローを追加してください。

ローの挿入

ローの挿入手順は、ローの更新手順とほぼ同じです。ただし、挿入操作の場合は、テーブル内のローをあらかじめ指定する必要はありません。テーブルへのローの挿入順序に意味はありません。

例

次のコードでは、新しいローが挿入されます。

```
t.InsertBegin();
t.SetInt( id, 3 );
t.SetString( lname, "Carlo" );
t.Insert();
```

カラムの値を設定しない場合、そのカラムにデフォルト値があるときはデフォルト値が使用されます。カラムにデフォルトがない場合は、次のエントリが使用されます。

- NULL 入力可能なカラムの場合は NULL
- NULL 入力不可の数値カラムの場合は 0
- NULL 入力不可の文字カラムの場合は空の文字列
- 明示的に値を NULL に設定するには、setNull メソッドを使用します。

更新操作の場合は、コミットを実行したときに、永続的な記憶領域のデータベースに挿入が適用されます。AutoCommit モードでは、insert メソッドの一部としてコミットが実行されます。

ローの削除

ローの削除手順は、ローの挿入や更新よりも簡単です。挿入モードや更新モードに対応する削除モードはありません。

次の手順は、ローを削除します。

❖ ローを削除するには、次の手順に従います。

1　削除するローに移動します。

2　Table.Delete() メソッドを実行します。

```
t.Delete();
```

Ultra Light でのトランザクション処理

Ultra Light のトランザクション処理は、データベース内のデータの整合性を保証します。トランザクションは、作業の論理単位です。トランザクション全体が実行されるか、トランザクション内の文がどれも実行されないかのいずれかです。

デフォルトでは、Ultra Light.NET は AutoCommit モードで動作するため、挿入、更新、削除はそれぞれ独立したトランザクションとして実行されます。操作が完了すると、データベースに変更が加えられます。

複数文のトランザクションを使用するには、ULConnection.BeginTransaction() を呼び出すことによって『Ultra Light.NET ユーザーズ・ガイド』>「ULTransaction クラス」オブジェクトを作成する必要があります。たとえば、2 つの口座間で資金を移動するアプリケーションでは、振込み元の口座からの引き落としと、振り込み先口座への振り込みを、両方とも完了するか、どちらも完了しないかのいずれかです。

接続に有効なトランザクションがある場合は、ULTransaction.Commit() 文を実行してトランザクションを完了し、データベースへの変更を永続的なものにするか、ULTransaction.Rollback() 文を実行してトランザクションのすべての処理をキャンセルしてください。トランザクションがコミットまたはロールバックされると、次に ULConnection.BeginTransaction() を呼び出すまで、接続は AutoCommit モードに戻ります。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULConnection クラス」と 『Ultra Light.NET ユーザーズ・ガイド』>「ULTransaction クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「Connection クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

スキーマ情報へのアクセス

API のオブジェクトは、テーブル、カラム、インデックス、同期パブリケーションを表します。各オブジェクトには、そのオブジェクトの構造情報へアクセスするための Schema プロパティがあります。

API によるスキーマの変更はできません。スキーマに関する情報の取得のみが可能です。

スキーマの修正については、「データベース・スキーマのアップグレード」31 ページを参照してください。

次のスキーマ・オブジェクトと情報にアクセスできます。

- **DatabaseSchema**　データベース内のテーブルの数と名前、日付と時刻のフォーマットなどのグローバル・プロパティを公開します。

  ULDatabaseSchema オブジェクトを取得するには、ULConnection.Schema (iAnywhere.Data.UltraLite ネームスペース) または Connection.Schema (iAnywhere.UltraLite ネームスペース) にアクセスします。

  詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULConnection クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「Connection クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

- **TableSchema**　このテーブル内のカラムとインデックスの数と名前。

  ULTableSchema オブジェクトを取得するには、ULTable.Schema. (iAnywhere.Data.UltraLite ネームスペース) または Table.Schema (iAnywhere.UltraLite ネームスペース) にアクセスします。

- **IndexSchema**　インデックス内のカラムに関する情報。インデックスには直接に対応するデータがないため、個別の Index クラスはなく、ULIndexSchema クラスのみが存在します。

ULIndexSchema オブジェクトを取得するには、ULTableSchema.GetIndex、ULTableSchema.GetOptimalIndex、または ULTableSchema.GetPrimaryKey メソッドを呼び出します。

- **PublicationSchema** パブリケーションに含まれるテーブルとカラムのリスト。パブリケーションもスキーマのみで構成されているため、Publication オブジェクトは存在しません。

  ULPublicationSchema オブジェクトを取得するには、ULDatabaseSchema.GetPublicationSchema メソッドを呼び出します。

詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULTableSchema クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「TableSchema クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

エラー処理

.NET の標準エラー処理機能を使用して、エラーを処理できます。iAnywhere.Data.UltraLite ネームスペースの大部分のメソッドは、ULException エラーをスローします。ULException.NativeError を使用して、エラーに割り当てられた ULSQLCode 値を取得できます。iAnywhere.UltraLite ネームスペースの大部分のメソッドは、SQLException エラーをスローします。SQLException.ErrorCode を使用して、エラーに割り当てられた SQLCode 値を取得できます。ULException と SQLException のどちらにも、エラーの説明文を取得するために使用できる Message プロパティがあります。ULSQLCode エラーと SQLCode エラーは、エラー・タイプを示す負の数字です。

エラー・コードのリストについては、『ASA エラー・メッセージ』>「ASA エラー・メッセージ」を参照してください。

同期後は、接続の SyncResult プロパティを使用して詳細なエラー情報を取得できます。

詳細については、次の項目を参照してください。

- 『Ultra Light.NET ユーザーズ・ガイド』>「ULSyncProgressListener インタフェース」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「SyncProgressListener インタフェース」(iAnywhere.UltraLite ネームスペース)
- 『Ultra Light.NET ユーザーズ・ガイド』>「ULSyncResult クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「SyncResult クラス」(iAnywhere.UltraLite ネームスペース)

ユーザ認証

新しいユーザは既存の接続から追加します。Ultra Light のすべてのデータベースは、デフォルトのユーザ ID DBA とパスワード SQL を使用して作成されるため、最初はこの初期ユーザとして接続します。

ユーザ ID の変更はできません。ユーザを 1 人追加して既存のユーザを削除します。Ultra Light ではデータベースごとにユーザ ID が 4 つまで許可されます。

詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「Ultra Light のユーザ認証」を参照してください。

❖ ユーザを追加する、または既存のユーザのパスワードを変更するには、次の手順に従います。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. ULConnection.GrantConnectTo メソッドを使用して、希望するパスワードによるデータベースへのアクセスをユーザに付与します。

   新規ユーザを追加する場合も、既存のユーザのパスワードを変更する場合も、この手順は同じです。

   詳細については、『Ultra Light.NET ユーザーズ・ガイド』>「ULConnection クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「Connection クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。

❖ 既存のユーザを削除するには、次の手順に従います。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. Connection.RevokeConnectFrom メソッドを使用して、ユーザの接続権限を取り消します。

Ultra Light アプリケーションの同期

Ultra Light アプリケーションを中央の統合データベースと同期します。同期には、SQL Anywhere Studio に付属の Mobile Link 同期ソフトウェアが必要です。

この項では、同期の概要について簡単に説明し、Ultra Light.NET のユーザにとって特に関心があるいくつかの機能について説明します。

同期の詳細な説明については、『Mobile Link クライアント』> 「Ultra Light クライアント」を参照してください。

また、CustDB サンプル・アプリケーションには、同期の実例もあります。詳細については、SQL Anywhere 9 インストール環境の *Samples¥UltraLite.NET¥CustDB* サブディレクトリを参照してください。

Ultra Light.NET は、TCP/IP、HTTP、HTTPS 通信による同期をサポートします。同期は、Ultra Light アプリケーションによって開始されます。いずれの場合でも、SyncParms オブジェクトのプロパティを使用して同期を制御します。

注意
HTTPS を使用して同期するには、別途セキュリティ・オプション・ライセンスが必要です。このオプションのご注文については、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。

詳細については、『SQL Anywhere Studio の紹介』> 「SQL Anywhere Studio へようこそ」を参照してください。

アプリケーションへの ActiveSync 同期の追加

この項では、ActiveSync を Ultra Light.NET アプリケーションに追加する方法について説明します。また、エンド・ユーザのコンピュータ上の ActiveSync で使用するアプリケーションの登録方法についても説明します。

ActiveSync 同期を設定する概要については、『Mobile Link クライアント』> 「ActiveSync を使用するアプリケーションの配備」を参照してください。アプリケーションに同期を追加する概要については、『Mobile Link クライアント』> 「Ultra Light クライアント」を参照してください。

ActiveSync 同期を開始できるのは、ActiveSync 自体だけです。デバイスがクレードルにある場合や、[ActiveSync] ウィンドウで同期コマンドが選択された場合に、ActiveSync は同期を開始します。

ActiveSync が同期を開始すると、Ultra Light アプリケーションがまだ実行されていない場合は Mobile Link ActiveSync プロバイダが Ultra Light アプリケーションを起動し、メッセージを送信します。Mobile Link プロバイダからのメッセージを受信および処理するには、ULActiveSyncListener をアプリケーションに実装します。アプリケーションは、SetActiveSyncListener メソッドを使用してリスナ・オブジェクトを指定する必要があります。ここで、MyAppClassName は、アプリケーションのユニークな Windows クラス名です。

```
dbMgr.SetActiveSyncListener(
    "MyAppClassName", listener );
```

サンプル・コードを含む詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「ULActiveSyncListener インタフェース」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』> 「ActiveSyncListener インタフェース」(iAnywhere.UltraLite ネームスペース) を参照してください。

Ultra Light は、ActiveSync メッセージを受信すると、指定されたリスナの ActiveSyncInvoked() メソッドを別のスレッド上で呼び出します。マルチスレッドの問題を回避するには、ActiveSyncInvoked() メソッドでユーザ・インタフェースにイベントを通知してください。

マルチスレッド・アプリケーションを使用する場合は、スレッドごとに別々の接続を使用し、C# の **lock** キーワードまたは Visual Basic .NET の **SyncLock** キーワードを使用して、アプリケーションの共有オブジェクトにアクセスします。ActiveSyncInvoked() メソッドは、使用する接続の SyncParms.Stream に ULStreamType.ACTIVE_SYNC を指定してから ULConnection.Synchronize() を呼び出します。

アプリケーションを登録するときは、次のパラメータを設定します。

- **クラス名**　Connection.SetActiveSyncListener メソッドでアプリケーションが使用したクラス名と同じクラス名

第 4 章

iAnywhere.Data.UltraLite ネームスペース

この章について

iAnywhere.Data.UltraLite ネームスペースには、ADO.NET 用 Ultra Light.NET データ・プロバイダのクラス、インタフェース、列挙が含まれています。

Ultra Light.NET では、ADO.NET 標準を使用する Ultra Light データベース・アプリケーションを開発し、Adaptive Server Anywhere へのマイグレーション・パスを提供するために、C# または Visual Basic .NET のコードを作成できます。Ultra Light と Adaptive Server Anywhere のどちらを使用するかを決定していない場合、最初は iAnywhere.Data.UltraLite ネームスペースを使用し、Adaptive Server Anywhere の高度な機能が必要になったときに Adaptive Server Anywhere (iAnywhere.Data.AsaClient ネームスペース) に切り替えます。Adaptive Server Anywhere に移行する可能性がある場合は、iAnywhere.Data.UltraLite ネームスペースに含まれている Ultra Light 専用の拡張機能はできるだけ使用しないでください。

この API リファレンスでは、Adaptive Server Anywhere の ADO.NET 用データ・プロバイダで使用できない Ultra Light.NET 拡張機能には「UL 拡張：」と表記されています。

Ultra Light.NET の Ultra Light エンジンのランタイムを使用するには、RuntimeType プロパティ を適切な値に設定してから、他の Ultra Light.NET API を使用します。

アプリケーションは接続を開いて、データベースの操作を行う必要があります。接続は、ULConnection クラスを使用して開きます。

iAnywhere.Data.UltraLite アセンブリは、**iAnywhere.Data.UltraLite.resources** というサテライト・リソース・アセンブリを使用します。メイン・アセンブリは、このリソース・アセンブリを国別に System.Globalization.CultureInfo.CurrentUICulture、System.Globalization.CultureInfo.CurrentCulture、国 "EN" の順に検索します。

ULActiveSyncListener インタフェース

UL 拡張：ActiveSync イベントを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ULActiveSyncListener

// C#
    public interface ULActiveSyncListener
```

ULActiveSyncListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
ActiveSyncInvoked メソッド	ActiveSync 用の Mobile Link プロバイダがアプリケーションを呼び出して同期を実行するときに起動されます。

ActiveSyncInvoked メソッド

ActiveSync 用の Mobile Link プロバイダがアプリケーションを呼び出して同期を実行するときに起動されます。

プロトタイプ

```
' Visual Basic
    Public Sub ActiveSyncInvoked( _
    ByVal launchedByProvider As Boolean _
    )

// C#
    public void ActiveSyncInvoked(
    bool launchedByProvider
    );
```

パラメータ

- **launchedByProvider** ActiveSync 同期を実行するために、Mobile Link プロバイダがアプリケーションを起動した場合は true。同期の完了後、アプリケーションは自動的に停止しなけ

ればなりません。ActiveSync 用 Mobile Link プロバイダが呼び出したときに、すでにアプリケーションが実行されていた場合は false。

説明

このメソッドは、別のスレッドによって呼び出されます。マルチスレッドの問題を回避するには、このメソッドがユーザ・インタフェースにイベントを通知する必要があります。マルチスレッドを使用する場合は、スレッドごとに別々の接続を使用し、lock キーワードを使用して、アプリケーションの共有オブジェクトにアクセスすることをおすすめします。ActiveSyncInvoked への呼び出しによって発生した同期では、その接続の Stream プロパティに ULStreamType 列挙 値を指定する必要があります。

例

次のコード・フラグメントは、ActiveSync 要求を受信して、UI スレッドで同期を実行する方法を示してします。

```
' Visual Basic Imports iAnywhere.Data.UltraLite

 Public Class MainWindow
   Inherits System.Windows.Forms.Form
   Implements ULActiveSyncListener
   Private conn As ULConnection


  Public Sub New(ByVal args() As String)

     MyBase.New()

     'This call is required by the Windows Form
Designer.
     InitializeComponent()

     'Add any initialization after the
InitializeComponent() call

ULConnection.DatabaseManager.SetActiveSyncListener( _
         "myCompany.myapp", Me _
       )
     'Create Connection
     ...
   End Sub
```

```
  Protected Overrides Sub OnClosing( _
         ByVal e As
System.ComponentModel.CancelEventArgs _
       )

ULConnection.DatabaseManager.SetActiveSyncListener( _
         Nothing, Nothing _
       )
     MyBase.OnClosing(e)
   End Sub


  Public Sub ActiveSyncInvoked( _
         ByVal launchedByProvider As Boolean _
       ) Implements
ULActiveSyncListener.ActiveSyncInvoked
     Me.Invoke(New EventHandler(AddressOf
Me.ActiveSyncAction))
   End Sub


  Public Sub ActiveSyncAction( _
         ByVal sender As Object, ByVal e As EventArgs _
       )
     ' Do active sync
     conn.SyncParms.Stream = ULStreamType.ACTIVE_SYNC
     conn.Synchronize()
   End Sub
 End Class


// C# using iAnywhere.Data.UltraLite;
 public class Form1 : System.Windows.Forms.Form,
ULActiveSyncListener
 {
   private System.Windows.Forms.MainMenu mainMenu1;
   private ULConnection conn;


  public Form1()
   {
     //
     // Required for Windows Form Designer support
     //
     InitializeComponent();

     //
```

```
      // TODO: Add any constructor code after
      // InitializeComponent call
      //

ULConnection.DatabaseManager.SetActiveSyncListener(
          "myCompany.myapp", this
        );
      // Create connection
      ...
    }


  protected override void Dispose( bool disposing )
    {
      base.Dispose( disposing );
    }


  protected override void OnClosing(
       System.ComponentModel.CancelEventArgs e
      )
    {

ULConnection.DatabaseManager.SetActiveSyncListener(
          null, null
        );
      base.OnClosing(e);
    }


  public void ActiveSyncInvoked(bool
launchedByProvider)
    {
      this.Invoke( new EventHandler( ActiveSyncHandler )
);
    }


  internal void ActiveSyncHandler(object sender,
EventArgs e)
    {
      conn.SyncParms.Stream = ULStreamType.ACTIVE_SYNC;
      conn.Synchronize();
    }
 }
```

ULAuthStatusCode 列挙

UL 拡張：Mobile Link ユーザ認証の実行中にレポートされる可能性のあるステータス・コードを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULAuthStatusCode
      Inherits Short

// C#
    public enum ULAuthStatusCode :
      short
```

メンバ

メンバ	説明
EXPIRED	ユーザ ID またはパスワードの有効期限が切れているため、認証に失敗しました (EXPIRED = 3000)。
IN_USE	ユーザ ID がすでに使用されているため、認証に失敗しました (IN_USE = 5000)。
INVALID	ユーザ ID またはパスワードが正しくないため、認証に失敗しました (INVALID = 4000)。
UNKNOWN	認証ステータスが不明です。接続がまだ同期を実行していない可能性があります (UNKNOWN = 0)。
VALID	ユーザ ID とパスワードは、同期時には有効でした (VALID = 1000)。
VALID_BUT_EXPIRES_SOON	ユーザ ID とパスワードは、同期時には有効でしたが、まもなく有効期限が切れます (VALID_BUT_EXPIRES_SOON = 2000)。

参照

- AuthStatus プロパティ

ULCommand クラス

IN パラメータ付きまたは IN パラメータなしで事前にコンパイルされた SQL 文またはクエリを表します。このオブジェクトを使用すると、文またはクエリを効率的に何回も実行できます。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULCommand
      Inherits Component

// C#
    public sealed class ULCommand :
      Component
```

説明

ULCommand オブジェクトは、直接作成したり、CreateCommand メソッドを使用して作成したりできます。このメソッドによって、コマンドは特定の接続で文を実行するための適切なトランザクションを使用できます。

Transaction プロパティ、現在のトランザクションがコミットまたはロールバックされた後でリセットする必要があります。

ULCommand には、Ultra Light.NET データベースでコマンドを実行するための次のメソッドがあります。

メソッド	説明
ExecuteNonQuery メソッド	SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。
ExecuteReader メソッド	SQL SELECT 文を実行し、ULDataReader クラスで結果セットを返します。
ExecuteScalar メソッド	SQL SELECT 文を実行し、単一の値を返します。

メソッド	説明
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。CommandText プロパティ はテーブルの名前として解釈され、IndexName プロパティ はテーブルのソート順の指定に使用できます。CommandType プロパティ は CommandType.TableDirect でなければなりません。

CommandText プロパティ を含め、ほとんどのプロパティをリセットできるほか、ULCommand オブジェクトを再利用できます。

リソース管理の理由により、コマンドを使用し終わったら、そのコマンドを明示的に閉じることをおすすめします。

実装：IDbCommand、IDisposable

ULCommand メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULCommand コンストラクタ	ULCommand オブジェクトを初期化します。
ULCommand コンストラクタ	ULCommand オブジェクトを、指定したコマンド・テキストで初期化します。
ULCommand コンストラクタ	ULCommand オブジェクトを、指定したコマンド・テキストと接続で初期化します。
ULCommand コンストラクタ	ULCommand オブジェクトを、指定したコマンド・テキスト、接続、トランザクションで初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
CommandText プロパティ	CommandType プロパティ が CommandType.TableDirect である場合の、SQL 文のテキストまたはテーブルの名前を指定します。パラメータ化された文の場合、疑問符 (?) プレースホルダを使用してパラメータを渡します。
CommandTimeout プロパティ	この機能は Ultra Light.NET ではサポートされていません。
CommandType プロパティ	実行されるコマンドのタイプを指定します。
Connection プロパティ	ULCommand オブジェクトを実行する接続オブジェクトです。
IndexName プロパティ	**UL 拡張 :** CommandType プロパティ が CommandType.TableDirect であるときに、テーブルを開く (ソートする) インデックスの名前を指定します。
Parameters プロパティ	現在の文のパラメータを指定します。
Plan プロパティ	**UL 拡張 :** Ultra Light.NET がクエリの実行に使用するアクセス・プランを返します。このプロパティは、主に開発中の使用を目的とします。
Transaction プロパティ	ULCommand が実行される ULTransaction クラスを指定します。
UpdatedRowSource プロパティ	ULDataAdapter の Update メソッドによって使用されるときにコマンドの結果が DataRow に適用される方法を指定します。

パブリック・インスタンス・メソッド

メンバ	説明
Cancel メソッド	このメソッドは Ultra Light.NET ではサポートされていません。

メンバ	説明
CreateParameter メソッド	ULCommand オブジェクトにパラメータを指定するために ULParameter クラス オブジェクトを提供します。
ExecuteNonQuery メソッド	SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。
ExecuteReader メソッド	SQL SELECT 文を実行し、結果セットを返します。
ExecuteReader メソッド	コマンドの動作を指定して SQL SELECT 文を実行し、結果セットを返します。
ExecuteScalar メソッド	SQL SELECT 文を実行し、単一の値を返します。
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。CommandText プロパティ はテーブルの名前として解釈され、IndexName プロパティ はテーブルのソート順の指定に使用できます。
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、コマンド動作を指定してデータベース・テーブルを取り出します。CommandText プロパティ はテーブルの名前として解釈され、IndexName プロパティ はテーブルのソート順の指定に使用できます。
Prepare メソッド	このコマンドの SQL 文を事前にコンパイルして格納します。

ULCommand コンストラクタ

ULCommand オブジェクトを初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public ULCommand();
```

説明

ULCommand オブジェクトに CommandText プロパティ、Connection プロパティ、Transaction プロパティが設定されていないと、文を実行できません。

参照

- ULCommand クラス
- ULCommand メンバ
- CreateCommand メソッド
- ULCommand コンストラクタ
- ULCommand コンストラクタ
- ULCommand コンストラクタ

ULCommand コンストラクタ

ULCommand オブジェクトを、指定したコマンド・テキストで初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal cmdText As String _
    )

// C#
    public ULCommand(
      string  cmdText
    );
```

パラメータ

- **cmdText** CommandType プロパティ が CommandType.TableDirect である場合の、SQL 文のテキストまたはテーブルの名前。パラメータ化された文の場合、疑問符 (?) プレースホルダを使用してパラメータを渡します。

説明

ULCommand オブジェクトに Connection プロパティ、Transaction プロパティが設定されていないと、文を実行できません。

参照

- ULCommand クラス
- ULCommand メンバ
- CreateCommand メソッド
- ULCommand コンストラクタ
- ULCommand コンストラクタ

- ULCommand コンストラクタ

ULCommand コンストラクタ

ULCommand オブジェクトを、指定したコマンド・テキストと接続で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal cmdText As String, _
      ByVal connection As ULConnection _
    )

// C#
    public ULCommand(
      string  cmdText,
      ULConnection connection
    );
```

パラメータ

- **cmdText** CommandType プロパティ が CommandType.TableDirect である場合の、SQL 文のテキストまたはテーブルの名前。パラメータ化された文の場合、疑問符 (?) プレースホルダを使用してパラメータを渡します。
- **connection** 現在の接続

説明

ULCommand オブジェクトに Transaction プロパティが設定されていないと、文を実行できないことがあります。

参照

- ULCommand クラス
- ULCommand メンバ
- CreateCommand メソッド
- ULCommand コンストラクタ
- ULCommand コンストラクタ
- ULCommand コンストラクタ

ULCommand コンストラクタ

ULCommand オブジェクトを、指定したコマンド・テキスト、接続、トランザクションで初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal cmdText As String, _
      ByVal connection As ULConnection, _
      ByVal transaction As ULTransaction _
    )

// C#
    public ULCommand(
      string  cmdText,
      ULConnection connection,
      ULTransaction transaction
    );
```

パラメータ

- **cmdText** CommandType プロパティ が CommandType.TableDirect である場合の、SQL 文のテキストまたはテーブルの名前。パラメータ化された文の場合、疑問符 (?) プレースホルダを使用してパラメータを渡します。
- **connection** 現在の接続
- **transaction** ULCommand が実行される ULTransaction クラス。

参照

- ULCommand クラス
- ULCommand メンバ
- CreateCommand メソッド
- ULCommand コンストラクタ
- ULCommand コンストラクタ
- ULCommand コンストラクタ

CommandText プロパティ

CommandType プロパティ が CommandType.TableDirect である場合の、SQL 文のテキストまたはテーブルの名前を指定します。パラメータ化された文の場合、疑問符 (?) プレースホルダを使用してパラメータを渡します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property CommandText As String _
      Implements IDbCommand.CommandText

// C#
    public string  CommandText {get;set;}
```

プロパティ値

SQL 文のテキストまたはテーブルの名前を指定する文字列。デフォルトは、空の文字列 (無効なコマンド) です。

実装

IDbCommand.CommandText

例

次の例は、パラメータ化されたプレースホルダの使用方法を示します。

```
' Visual Basic myCmd.CommandText = "SELECT * FROM
Customers WHERE CustomerID = ?"

 // C# myCmd.CommandText = "SELECT * FROM Customers
WHERE CustomerID = ?";
```

参照

- ULCommand クラス
- ULCommand メンバ
- ExecuteNonQuery メソッド
- ExecuteReader メソッド
- ExecuteScalar メソッド
- ExecuteTable メソッド

CommandTimeout プロパティ

この機能は Ultra Light.NET ではサポートされていません。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property CommandTimeout As Integer _
      Implements IDbCommand.CommandTimeout

// C#
    public int CommandTimeout {get;set;}
```

プロパティ値　値は常にゼロです。

例外

- ULException クラス - 値を設定することは、Ultra Light.NET ではサポートされていません。

実装　IDbCommand.CommandTimeout

CommandType プロパティ

実行されるコマンドのタイプを指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property CommandType As CommandType _
      Implements IDbCommand.CommandType

// C#
    public CommandType CommandType {get;set;}
```

プロパティ値　CommandType 値の 1 つ。デフォルトは CommandType.Text です。

説明　サポートされているコマンド・タイプは、次のとおりです。

- CommandType.TableDirect - **UL 拡張：** この CommandType を指定するときは、CommandText プロパティ はデータベース・テーブルの名前でなければなりません。また、IndexName プロパティ を使用して、テーブルを開く (ソートする) インデックスを指定することもできます。テーブルにアクセスするには、ExecuteTable メソッド または ExecuteReader メソッド を使用します。
- CommandType.Text - この CommandType を指定するときは、CommandText プロパティ は SQL 文またはクエリでなければなりません。クエリ以外の SQL 文を実行するには、

ExecuteNonQuery メソッド を使用します。また、クエリを実行するには、ExecuteReader メソッド または ExecuteScalar メソッド を使用します。

例外

- ArgumentException - CommandType.StoredProcedure は Ultra Light.NET ではサポートされていません。

実装 IDbCommand.CommandType

Connection プロパティ

ULCommand オブジェクトを実行する接続オブジェクトです。

プロトタイプ

```
' Visual Basic
    Public Property Connection As ULConnection

// C#
    public ULConnection Connection {get;set;}
```

プロパティ値 コマンドを実行する ULConnection クラス オブジェクト

説明

ULCommand オブジェクトは、開かれた接続がないと、実行することができません。

デフォルト値は null 参照 (Visual Basic の Nothing) です。

これは、IDbCommand.Connection が厳密に型指定されたものです。

IndexName プロパティ

UL 拡張： CommandType プロパティ が CommandType.TableDirect であるときに、テーブルを開く (ソートする) インデックスの名前を指定します。

プロトタイプ

```
' Visual Basic
    Public Property IndexName As String

// C#
    public string  IndexName {get;set;}
```

プロパティ値

インデックスの名前を指定する文字列。デフォルトは null 参照 (Visual Basic の Nothing) です。この場合、テーブルはプライマリ・キーを使用して開かれます。

参照

- ULCommand クラス
- ULCommand メンバ
- ExecuteTable メソッド
- ExecuteReader メソッド

Parameters プロパティ

現在の文のパラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Parameters As ULParameterCollection

// C#
    public ULParameterCollection Parameters {get;}
```

プロパティ値

SQL 文のパラメータを保持する ULParameterCollection クラス。デフォルト値は空のコレクションです。

説明

CommandText プロパティ で疑問符を使用してパラメータを示します。コレクション内のパラメータは、疑問符のプレースホルダと同じ順序で指定されます。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。

これは、IDbCommand.Parameters が厳密に型指定されたものです。

Plan プロパティ

UL 拡張 : Ultra Light.NET がクエリの実行に使用するアクセス・プランを返します。このプロパティは、主に開発中の使用を目的とします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Plan As String
```

```
// C#
    public string  Plan {get;}
```

プロパティ値　クエリ実行プランのテキストベースの記述が含まれる文字列

例外

- ULException クラス - SQL エラーが発生しました。

Transaction プロパティ

ULCommand が実行される ULTransaction クラス を指定します。

プロトタイプ

```
' Visual Basic
    Public Property Transaction As ULTransaction
```

```
// C#
    public ULTransaction Transaction {get;set;}
```

プロパティ値　ULCommand が実行される ULTransaction クラス。これは、Connection プロパティ によって指定された接続の現在のトランザクションでなければなりません。デフォルト値は null 参照 (Visual Basic の Nothing) です。

説明

トランザクションがコミットまたはロールバックされた後でコマンドを再使用する場合は、このプロパティをリセットする必要があります。

これは、IDbCommand.Transaction が厳密に型指定されたものです。

参照

- ULCommand クラス
- ULCommand メンバ
- BeginTransaction メソッド

UpdatedRowSource プロパティ

ULDataAdapter の Update メソッドによって使用されるときにコマンドの結果が DataRow に適用される方法を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property UpdatedRowSource As UpdateRowSource
    _
      Implements IDbCommand.UpdatedRowSource

// C#
    public UpdateRowSource UpdatedRowSource {get;set;}
```

プロパティ値

UpdateRowSource 値の 1 つ。デフォルト値は UpdateRowSource.Both です。

実装

IDbCommand.UpdatedRowSource

Cancel メソッド

このメソッドは Ultra Light.NET ではサポートされていません。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Cancel() _
      Implements IDbCommand.Cancel

// C#
    public void Cancel();
```

説明

このメソッドによって行われる処理はありません。Ultra Light.NET コマンドは、実行中に中断できません。

実装

IDbCommand.Cancel

CreateParameter メソッド

ULCommand オブジェクトにパラメータを指定するために ULParameter クラス オブジェクトを提供します。

プロトタイプ

```
' Visual Basic
    Public Function CreateParameter() As ULParameter

// C#
    public ULParameter CreateParameter();
```

戻り値

ULParameter クラス オブジェクトとしての新しいパラメータ

説明

一部の SQL 文では、疑問符 (?) によって文のテキストに示されているパラメータを使用できます。CreateParameter メソッドは、ULParameter クラス オブジェクトを提供します。ULParameter にプロパティを設定して、そのパラメータの値を指定できます。

これは、IDbCommand.CreateParameter が厳密に型指定されたものです。

ExecuteNonQuery メソッド

SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function ExecuteNonQuery() As Integer _
      Implements IDbCommand.ExecuteNonQuery

// C#
    public int ExecuteNonQuery();
```

戻り値

影響を受けたローの数

説明

この文は、必要に応じて CommandText プロパティ および Parameters プロパティ が指定された、現在の ULCommand オブジェクトです。

UPDATE、INSERT、DELETE 文の場合、戻り値はコマンドの影響を受けるローの数です。その他すべての文のタイプおよびロールバックの場合、戻り値は -1 です。

CommandType プロパティを CommandType.TableDirect にすることはできません。

例外

- ULException クラス - SQL エラーが発生しました。

- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

実装　IDbCommand.ExecuteNonQuery

ExecuteReader メソッド

SQL SELECT 文を実行し、結果セットを返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteReader() As ULDataReader

// C#
    public ULDataReader ExecuteReader();
```

戻り値　ULDataReader クラス オブジェクトとしての結果セット

説明

この文は、必要に応じて CommandText プロパティ および Parameters プロパティ が指定された、現在の ULCommand オブジェクトです。ULDataReader クラス オブジェクトは、読み込み専用の結果セットです。編集可能な結果セットには、ExecuteTable メソッド または ULDataAdapter クラス を使用します。

CommandType プロパティ が CommandType.TableDirect である場合、ExecuteReader は ExecuteTable メソッド を実行し、ULDataReader クラス として ULTable クラス のダウンキャストを返します。

これは、IDbCommand.ExecuteReader が厳密に型指定されたものです。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

参照

- ULCommand クラス
- ULCommand メンバ
- ExecuteReader メソッド

ExecuteReader メソッド

コマンドの動作を指定して SQL SELECT 文を実行し、結果セットを返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteReader( _
      ByVal cmdBehavior As CommandBehavior _
    ) As ULDataReader

// C#
    public ULDataReader ExecuteReader(
      CommandBehavior cmdBehavior
    );
```

パラメータ

- **cmdBehavior** クエリの結果の記述と、接続への影響の記述の CommandBehavior フラグのビット処理の組み合わせ。Ultra Light.NET で有効なのは、CommandBehavior.Default フラグ、CommandBehavior.CloseConnection フラグ、CommandBehavior.SchemaOnly フラグだけです。

戻り値

ULDataReader クラス オブジェクトとしての結果セット

説明

この文は、必要に応じて CommandText プロパティ および Parameters プロパティ が指定された、現在の ULCommand オブジェクトです。ULDataReader クラス オブジェクトは、読み込み専用の結果セットです。編集可能な結果セットには、ExecuteTable メソッド または ULDataAdapter クラス を使用します。

CommandType プロパティ が CommandType.TableDirect である場合、ExecuteReader は ExecuteTable メソッド を実行し、ULDataReader クラス として ULTable クラス のダウンキャストを返します。

これは、IDbCommand.ExecuteReader が厳密に型指定されたものです。

例外

- ULException クラス - SQL エラーが発生しました。

- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

参照

- ULCommand クラス
- ULCommand メンバ
- ExecuteReader メソッド

ExecuteScalar メソッド

SQL SELECT 文を実行し、単一の値を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function ExecuteScalar() As Object _
      Implements IDbCommand.ExecuteScalar

// C#
    public object ExecuteScalar();
```

戻り値

結果セットの最初のローの最初のカラム、または結果セットが空の場合は null 参照 (Visual Basic の Nothing)。

説明

この文は、必要に応じて CommandText プロパティ および Parameters プロパティ が指定された、現在の ULCommand オブジェクトです。

複数のローとカラムを返すクエリでこのメソッドが呼び出されると、最初のローの最初のカラムのみが返されます。

CommandType プロパティ が CommandType.TableDirect である場合、ExecuteScalar は ExecuteTable メソッド を実行し、最初のローの最初のカラムを返します。

例外

- ULException クラス - SQL エラーが発生しました。

- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

実装 IDbCommand.ExecuteScalar

ExecuteTable メソッド

UL 拡張：直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。CommandText プロパティ はテーブルの名前として解釈され、IndexName プロパティ はテーブルのソート順の指定に使用できます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteTable() As ULTable

// C#
    public ULTable ExecuteTable();
```

戻り値 ULTable クラス オブジェクトとしてのテーブル

説明

CommandType プロパティは、CommandType.TableDirect に設定する必要があります。

IndexName プロパティが null 参照 (Visual Basic の Nothing) であると、プライマリ・キーを使用してテーブルが開かれます。そうでない場合は、ソート基準となるインデックスの名前として IndexName プロパティ 値を使用してテーブルが開かれます。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

参照

- ULCommand クラス

- ULCommand メンバ
- ExecuteTable メソッド

ExecuteTable メソッド

UL 拡張：直接の操作用に、コマンド動作を指定してデータベース・テーブルを取り出します。CommandText プロパティ はテーブルの名前として解釈され、IndexName プロパティ はテーブルのソート順の指定に使用できます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteTable( _
      ByVal cmdBehavior As CommandBehavior _
    ) As ULTable

// C#
    public ULTable ExecuteTable(
      CommandBehavior cmdBehavior
    );
```

パラメータ

- **cmdBehavior** クエリの結果の記述と、接続への影響の記述の CommandBehavior フラグのビット処理の組み合わせ。Ultra Light.NET で有効なのは、CommandBehavior.Default フラグ、CommandBehavior.CloseConnection フラグ、CommandBehavior.SchemaOnly フラグだけです。

戻り値

ULTable クラス オブジェクトとしてのテーブル

説明

CommandType プロパティは、CommandType.TableDirect に設定する必要があります。

IndexName プロパティが null 参照 (Visual Basic の Nothing) であると、プライマリ・キーを使用してテーブルが開かれます。そうでない場合は、ソート基準となるインデックスの名前として IndexName プロパティ 値を使用してテーブルが開かれます。

例外

- ULException クラス - SQL エラーが発生しました。

- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

参照

- ULCommand クラス
- ULCommand メンバ
- ExecuteTable メソッド

Prepare メソッド

このコマンドの SQL 文を事前にコンパイルして格納します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Prepare() _
      Implements IDbCommand.Prepare

// C#
    public void Prepare();
```

説明

文を事前にコンパイルすると、パラメータ値のみが変更されているときに文を効率的に再使用できます。このコマンドの他のプロパティを変更すると、文の準備は解除されます。

準備が解除されたコマンドはいずれも、さまざまな Execute メソッドへの呼び出し時に準備が行われるため、Ultra Light.NET では明示的に文を準備する必要はありません。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException - コマンドのステータスは有効ではありません。Connection プロパティ が見つからないか閉じている、Transaction プロパティ 値が接続の現在のトランザクションのステータスに一致しない、または CommandText プロパティ が無効です。

実装

IDbCommand.Prepare

ULConnection クラス

Ultra Light.NET データベースへの接続を示します。

プロトタイプ

```
' Visual Basic
NotInheritable Public Class ULConnection
  Inherits Component

// C#
public sealed class ULConnection :
  Component
```

説明

Ultra Light.NET の Ultra Light エンジンのランタイムを使用するには、RuntimeType プロパティ を適切な値に設定してから、他の Ultra Light.NET API を使用します。

既存のデータベースへの接続は、Open メソッドを使用して開かれます。新しいデータベースを作成し、そのデータベースに接続するには、OpenWithCreate メソッドを使用します。

接続は、他の操作の実行前に開きます。また、その接続での操作がすべて終了したら、接続を閉じてからアプリケーションを終了します。また、接続で開いた結果セットとテーブルをすべて閉じてから、その接続を閉じます。

データベースのスキーマには、開いている接続の Schema プロパティを使用してアクセスできます。

実装：IDbConnection、IDisposable

ULConnection メンバ

パブリック静的フィールド (共有)

メンバ	説明
INVALID_DATABASE_ID フィールド	**UL 拡張：**DatabaseID プロパティ が設定されていないことを示すデータベース ID の定数です。

パブリック静的プロパティ (共有)

メンバ	説明
DatabaseManager プロパティ	**UL 拡張：**シングルトンの ULDatabaseManager クラス オブジェクトへのアクセスに使用します。

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULConnection コンストラクタ	ULConnection オブジェクトを初期化します。接続を開いてから、データベース操作を実行してください。
ULConnection コンストラクタ	ULConnection オブジェクトを、指定した接続文字列で初期化します。接続を開いてから、データベース操作を実行してください。

パブリック・インスタンス・プロパティ

メンバ	説明
ConnectionString プロパティ	Ultra Light.NET データベースへの接続を開くためのパラメータを指定します。接続文字列は、ULConnectionParms クラス オブジェクトを使用して指定できます。
ConnectionTimeout プロパティ	この機能は Ultra Light.NET ではサポートされていません。
Database プロパティ	接続を開くデータベースの名前を返します。
DatabaseID プロパティ	**UL 拡張：**グローバル・オートインクリメントのカラムに使用するデータベース ID の値を指定します。
GlobalAutoIncrementUsage プロパティ	**UL 拡張：**利用可能なグローバル・オートインクリメントの値の使用済み比率 (%) を返します。
LastIdentity プロパティ	**UL 拡張：**直前に使用した identity の値を返します。

メンバ	説明
Schema プロパティ	**UL 拡張：**この接続に関連付けられている現在のデータベースのスキーマへのアクセスに使用します。
State プロパティ	接続の現在のステータスを返します。
SyncParms プロパティ	**UL 拡張：**この接続の同期設定を指定します。
SyncResult プロパティ	**UL 拡張：**この接続の最後の同期結果を返します。

パブリック・インスタンス・メソッド

メンバ	説明
BeginTransaction メソッド	トランザクション・オブジェクトを返します。トランザクション・オブジェクトに関連付けられているコマンドは、単一のトランザクションとして実行されます。トランザクションは、Commit メソッド または Rollback メソッド で終了します。
BeginTransaction メソッド	指定した独立性レベルのトランザクション・オブジェクトを返します。トランザクション・オブジェクトに関連付けられているコマンドは、単一のトランザクションとして実行されます。トランザクションは、Commit メソッド または Rollback メソッド で終了します。
ChangeDatabase メソッド	開いている ULConnection の現在のデータベースを変更します。
ChangeEncryptionKey メソッド	**UL 拡張：**データベースの暗号化キーを、指定した新しいキーに変更します。
Close メソッド	データベース接続を閉じます。
CountUploadRows メソッド	**UL 拡張：**次回の同期でアップロードする必要のあるロー数を返します。
CountUploadRows メソッド	**UL 拡張：**次回の同期でアップロードする必要のあるロー数を返します。

メンバ	説明
CreateCommand メソッド	この接続と現在のトランザクションに関連付けられる ULCommand クラス オブジェクトを作成して初期化します。ULCommand のプロパティを使用して、その動作を制御できます。
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。テーブルのプライマリ・キーを使用して、テーブルが開かれます (ソートされます)。
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。テーブルは、指定したインデックスを使用して開かれます (ソートされます)。
ExecuteTable メソッド	**UL 拡張：**直接の操作用に、コマンド動作を指定してデータベース・テーブルを取り出します。テーブルは、指定したインデックスを使用して開かれます (ソートされます)。
GetLastDownloadTime メソッド	**UL 拡張：**指定されたパブリケーションの最後のダウンロードの時刻を返します。
GetNewUUID メソッド	**UL 拡張：**新しい UUID を生成します (Guid)。
GrantConnectTo メソッド	**UL 拡張：**指定されたパスワードを持つユーザ ID に、Ultra Light データベースへのアクセスを許可します。
Open メソッド	以前に指定した接続文字列を使用してデータベースへの接続を開きます。
OpenWithCreate メソッド	**UL 拡張：**以前に指定した接続文字列を使用してデータベースへの接続を開きます。データベースが存在しない場合、新しいデータベースが作成されます。
ResetLastDownloadTime メソッド	**UL 拡張：**最後のダウンロードの時刻をリセットします。
RevokeConnectFrom メソッド	**UL 拡張：**指定したユーザ ID から、Ultra Light データベースへのアクセス権を取り消します。

メンバ	説明
RollbackPartialDownload メソッド	**UL 拡張：**部分的なダウンロードから、未処理の変更をデータベースにロールバックします。
StartSynchronizationDelete メソッド	**UL 拡張：**同期用に、この接続によって行われる以後のすべての削除にマークを付けます。
StopSynchronizationDelete メソッド	**UL 拡張：**削除操作が同期されないようにします。
Synchronize メソッド	**UL 拡張：**現在の SyncParms プロパティ を使用してデータベースを同期します。
Synchronize メソッド	**UL 拡張：**指定したリスナに送信されるプログレス・イベントとともに、現在の SyncParms プロパティ を使用してデータベースを同期します。

パブリック・インスタンス・イベント

メンバ	説明
InfoMessage イベント	Ultra Light.NET が、この接続の警告または情報メッセージを送信するときに発生します。
StateChange イベント	この接続のステータスが変更されると発生します。

ULConnection コンストラクタ

ULConnection オブジェクトを初期化します。接続を開いてから、データベース操作を実行してください。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public ULConnection();
```

説明

Ultra Light.NET の Ultra Light エンジンのランタイムを使用するには、RuntimeType プロパティ を適切な値に設定してから、他の Ultra Light.NET API を使用します。

ULConnection オブジェクトは、ConnectionString プロパティが設定されていないと、開くことができません。

参照

- ULConnection クラス
- ULConnection メンバ
- Open メソッド
- OpenWithCreate メソッド
- ULConnection コンストラクタ

ULConnection コンストラクタ

ULConnection オブジェクトを、指定した接続文字列で初期化します。接続を開いてから、データベース操作を実行してください。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal connectionString As String _
    )

// C#
    public ULConnection(
      string  connectionString
    );
```

パラメータ

- **connectionString** Ultra Light.NET の接続文字列。接続文字列は、キーワード値の組がセミコロンで区切られたリストです。

  パラメータのリストについては、ConnectionString プロパティを参照してください。

説明

Ultra Light.NET の Ultra Light エンジンのランタイムを使用するには、RuntimeType プロパティ を適切な値に設定してから、他の Ultra Light.NET API を使用します。

接続文字列は、ULConnectionParms クラス オブジェクトを使用して指定できます。

例外

- ArgumentException - 指定された接続文字列が無効です。

例

次のコードでは、Windows CE デバイス上のデータベース ¥UltraLite¥MyDatabase.udb への接続を作成して開きます。

```
' Visual Basic Dim openParms As ULConnectionParms = New
ULConnectionParms
 openParms.DatabaseOnCE = "¥UltraLite¥MyDatabase.udb"
 Dim conn As ULConnection = _
   New ULConnection( openParms.ToString() )
 conn.Open()

 // C# ULConnectionParms openParms = new
ULConnectionParms();
 openParms.DatabaseOnCE = @"¥UltraLite¥MyDatabase.udb";
 ULConnection conn = new ULConnection(
openParms.ToString() );
 conn.Open();
```

参照

- ULConnection クラス
- ULConnection メンバ
- Open メソッド
- OpenWithCreate メソッド
- ULConnection コンストラクタ

INVALID_DATABASE_ID フィールド

UL 拡張：DatabaseID プロパティ が設定されていないことを示すデータベース ID の定数です。

プロトタイプ

' Visual Basic

Public Shared **INVALID_DATABASE_ID** As Long

// C#

public const long **INVALID_DATABASE_ID**;

ConnectionString プロパティ

Ultra Light.NET データベースへの接続を開くためのパラメータを指定します。接続文字列は、ULConnectionParms クラス オブジェクトを使用して指定できます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property ConnectionString As String _
      Implements IDbConnection.ConnectionString

// C#
    public string  ConnectionString {get;set;}
```

プロパティ値

キーワードと値の組み合わせがセミコロンで区切られたリスト形式の、この接続を開くためのパラメータ。デフォルトは、空の文字列 (無効な接続文字列) です。

説明

UL 拡張：Ultra Light.NET によって使用されるパラメータは Ultra Light データベースに固有であるため、その接続文字列は Adaptive Server Anywhere の接続文字列との互換性がありません。パラメータのリストについては、『Ultra Light データベース・ユーザーズ・ガイド』>「接続パラメータ」を参照してください。

例外

- ArgumentException - 指定された接続文字列が無効です。
- InvalidOperationException - 接続が開いている間は、値を設定できません。

実装

IDbConnection.ConnectionString

例

次のコードでは、Windows CE デバイス上のデータベース ¥UltraLite¥MyDatabase.udb への接続を作成して開きます。

```
' Visual Basic Dim openParms As ULConnectionParms = New
ULConnectionParms
 openParms.DatabaseOnCE = "¥UltraLite¥MyDatabase.udb"
 Dim conn As ULConnection = New ULConnection
 conn.ConnectionString = openParms.ToString()
 conn.Open()

 // C# ULConnectionParms openParms = new
ULConnectionParms();
```

```
openParms.DatabaseOnCE = @"¥UltraLite¥MyDatabase.udb";
ULConnection conn = new ULConnection();
conn.ConnectionString = openParms.ToString();
conn.Open();
```

参照

- ULConnection クラス
- ULConnection メンバ
- Open メソッド
- OpenWithCreate メソッド

ConnectionTimeout プロパティ

この機能は Ultra Light.NET ではサポートされていません。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property ConnectionTimeout As Integer
    _
      Implements IDbConnection.ConnectionTimeout

// C#
    public int ConnectionTimeout {get;}
```

プロパティ値

値は常にゼロです。

例外

- ULException クラス - 値を設定することは、Ultra Light.NET ではサポートされていません。

実装

IDbConnection.ConnectionTimeout

Database プロパティ

接続を開くデータベースの名前を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property Database As String _
      Implements IDbConnection.Database

// C#
    public string  Database {get;}
```

プロパティ値 データベースの名前が含まれる文字列

説明 Windows CE デバイスでは、ULConnection は dbn、ce_file の順に接続文字列を参照します。

デスクトップ・マシンでは、ULConnection は dbn、nt_file の順に接続文字列を参照します。

実装 IDbConnection.Database

DatabaseID プロパティ

UL 拡張：グローバル・オートインクリメントのカラムに使用するデータベース ID の値を指定します。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseID As Long

// C#
    public long DatabaseID {get;set;}
```

プロパティ値 現在のデータベースのデータベース ID の値

説明 データベース ID の値は、[0,UInt32.MaxValue] の範囲内である必要があります。INVALID_DATABASE_ID フィールド の値は、現在のデータベースにデータベース ID が設定されていないことを示すのに使用されます。

例外

- ULException クラス - 指定された新しいデータベース ID が無効です。

DatabaseManager プロパティ

UL 拡張：シングルトンの ULDatabaseManager クラス オブジェクトへのアクセスに使用します。

プロトタイプ

```
' Visual Basic
    Public Shared Readonly Property DatabaseManager As
  ULDatabaseManager
```

// C#
public const ULDatabaseManager **DatabaseManager** {get;}

プロパティ値　シングルトンの ULDatabaseManager クラス オブジェクトへの参照

GlobalAutoIncrementUsage プロパティ

UL 拡張 : 利用可能なグローバル・オートインクリメントの値の使用済み比率 (%) を返します。

プロトタイプ

' Visual Basic
Public Readonly Property **GlobalAutoIncrementUsage** As Short

// C#
public short **GlobalAutoIncrementUsage** {get;}

プロパティ値　使用可能なグローバル・オートインクリメントの値の使用済み比率 (%)。これは、0 ～ 100 の範囲内の整数です。

説明　比率が 100% に近づいたら、DatabaseID プロパティ を使用して、使用中のアプリケーションに新しいグローバル・データベース ID を設定してください。

例外

- ULException クラス - SQL エラーが発生しました。

LastIdentity プロパティ

UL 拡張 : 直前に使用した identity の値を返します。

プロトタイプ

' Visual Basic
Public Readonly Property **LastIdentity** As UInt64

// C#
public ulong **LastIdentity** {get;}

プロパティ値　直前に使用した identity の値 (符号なし long)

説明　直前に使用した identity の値。このプロパティは、次の Adaptive Server Anywhere SQL 文と同義です。

```
SELECT @@identity
```

LastIdentity は、グローバル・オートインクリメント・カラムで使うと特に便利です。

このプロパティでは最後に割り当てられたデフォルト値がわかるだけなので、間違った結果を取らないために INSERT 文を実行した直後にこの値を取り出してください。

ときには、1 つの INSERT 文にグローバル・オートインクリメント型のカラムが複数含まれていることがあります。この場合、LastIdentity は生成されたデフォルト値の 1 つですが、その値の生成元のカラムを判別する信頼できる方法はありません。このため、このような状況を回避するようなデータベースの設計と INSERT 文の記述を行ってください。

例外

- ULException クラス - SQL エラーが発生しました。

Schema プロパティ

UL 拡張：この接続に関連付けられている現在のデータベースのスキーマへのアクセスに使用します。

プロトタイプ

' Visual Basic
Public Readonly Property **Schema** As ULDatabaseSchema

// C#
public ULDatabaseSchema **Schema** {get;}

プロパティ値

この接続が開かれているデータベースのスキーマを示す ULDatabaseSchema クラス オブジェクトへの参照

説明

このプロパティが有効なのは、接続が開かれている間だけです。

State プロパティ

接続の現在のステータスを返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property State As ConnectionState _
      Implements IDbConnection.State

// C#
    public ConnectionState State {get;}
```

プロパティ値

接続が開いている場合は ConnectionState.Open、閉じている場合は ConnectionState.Closed

実装

IDbConnection.State

参照

- ULConnection クラス
- ULConnection メンバ
- StateChange イベント

SyncParms プロパティ

UL 拡張：この接続の同期設定を指定します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncParms As ULSyncParms

// C#
    public ULSyncParms SyncParms {get;}
```

プロパティ値

この接続によって同期に使用されるパラメータへの参照。パラメータを修正すると、この接続で行われる次の同期に反映されます。

参照

- ULConnection クラス
- ULConnection メンバ
- Synchronize メソッド
- SyncResult プロパティ

SyncResult プロパティ

UL 拡張：この接続の最後の同期結果を返します。

プロトタイプ

' Visual Basic
Public Readonly Property **SyncResult** As ULSyncResult

// C#
public ULSyncResult **SyncResult** {get;}

プロパティ値

この接続の最後の同期結果に対する参照

参照

- ULConnection クラス
- ULConnection メンバ
- Synchronize メソッド
- SyncParms プロパティ

BeginTransaction メソッド

トランザクション・オブジェクトを返します。トランザクション・オブジェクトに関連付けられているコマンドは、単一のトランザクションとして実行されます。トランザクションは、Commit メソッド または Rollback メソッド で終了します。

プロトタイプ

' Visual Basic
Overloads Public Function **BeginTransaction()** As ULTransaction

// C#
public ULTransaction **BeginTransaction();**

戻り値

新しいトランザクションを示す ULTransaction クラス オブジェクト

説明

コマンドをトランザクション・オブジェクトに関連付けるには、Transaction プロパティを使用します。現在のトランザクションは、CreateCommand メソッド によって作成されたコマンドに自動的に関連付けられます。

デフォルトでは、接続はトランザクションを使用しません。また、すべてのコマンドは、実行されるときに自動的にコミットされます。現在のトランザクションがコミットまたはロールバックされると、次に BeginTransaction を呼び出すまで、接続はオートコミット・モードに戻ります。

これは、IDbConnection.BeginTransaction が厳密に型指定されたものです。

例外

- ULException クラス - 接続が閉じています。
- InvalidOperationException - ULConnection は並列トランザクションをサポートしていません。

参照

- ULConnection クラス
- ULConnection メンバ
- BeginTransaction メソッド

BeginTransaction メソッド

指定した独立性レベルのトランザクション・オブジェクトを返します。トランザクション・オブジェクトに関連付けられているコマンドは、単一のトランザクションとして実行されます。トランザクションは、Commit メソッド または Rollback メソッド で終了します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function BeginTransaction( _
      ByVal isolationLevel As IsolationLevel _
    ) As ULTransaction

// C#
    public ULTransaction BeginTransaction(
      IsolationLevel isolationLevel
    );
```

パラメータ

- **isolationLevel** 要求されたトランザクションの独立性レベル。Ultra Light.NET では、IsolationLevel.ReadUncommitted のみがサポートされています .。

戻り値

新しいトランザクションを示す ULTransaction クラス オブジェクト

説明

コマンドをトランザクション・オブジェクトに関連付けるには、Transaction プロパティを使用します。現在のトランザクションは、CreateCommand メソッド によって作成されたコマンドに自動的に関連付けられます。

デフォルトでは、接続はトランザクションを使用しません。また、すべてのコマンドは、実行されるときに自動的にコミットされます。現在のトランザクションがコミットまたはロールバックされると、次に BeginTransaction を呼び出すまで、接続はオートコミット・モードに戻ります。

これは、IDbConnection.BeginTransaction が厳密に型指定されたものです。

例外

- ULException クラス - 接続が閉じているか、サポートされていない独立性レベルが指定されました。
- InvalidOperationException - ULConnection は並列トランザクションをサポートしていません。

参照

- ULConnection クラス
- ULConnection メンバ
- BeginTransaction メソッド

ChangeDatabase メソッド

開いている ULConnection の現在のデータベースを変更します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub ChangeDatabase( _
      ByVal connectionString As String _
    ) _
      Implements IDbConnection.ChangeDatabase

// C#
    public void ChangeDatabase(
      string  connectionString
    );
```

パラメータ

- **connectionString** 新しいデータベースへの接続を開く完全な接続文字列

説明

パラメータのエラーがあった場合でも、現在のデータベースへの接続は閉じられます。

UL 拡張：*connectionString* は、dbn または dbf ではなく、完全な接続文字列です。

実装　IDbConnection.ChangeDatabase

参照

- ULConnection クラス
- ULConnection メンバ
- ConnectionString プロパティ

ChangeEncryptionKey メソッド

UL 拡張：データベースの暗号化キーを、指定した新しいキーに変更します。

プロトタイプ

```
' Visual Basic
    Public Sub ChangeEncryptionKey( _
      ByVal newKey As String _
    )

// C#
    public void ChangeEncryptionKey(
      string  newKey
    );
```

パラメータ

- **newKey** データベースの新しい暗号化キー

説明　暗号化キーが失われた場合は、データベースを開くことができません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- EncryptionKey プロパティ

Close メソッド

データベース接続を閉じます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Close() _
      Implements IDbConnection.Close

// C#
    public void Close();
```

説明

Close メソッドは、保留中のトランザクションをロールバックし、その接続を閉じます。アプリケーションは、Close を複数回呼び出すことができます。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDbConnection.Close

CountUploadRows メソッド

UL 拡張：次回の同期でアップロードする必要のあるロー数を返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function CountUploadRows( _
      ByVal mask As Integer, _
      ByVal threshold As UInt32 _
    ) As UInt32

// C#
    public uint CountUploadRows(
      int mask,
      uint threshold
    );
```

パラメータ

- **mask** ローのチェック対象となるパブリケーションのセット。詳細については、ULPublicationSchema クラスを参照してください。
- **threshold** カウントするローの最大数。CountUploadRows の所要時間を制限します。値 0 は制限が最大であることを示します。値 1 は、同期の必要なローがあるかどうかを判別する場合に使用します。

戻り値　指定したパブリケーションからアップロードする必要のあるロー数

例外
- ULException クラス - SQL エラーが発生しました。

参照
- ULConnection クラス
- ULConnection メンバ
- CountUploadRows メソッド

CountUploadRows メソッド

UL 拡張：次回の同期でアップロードする必要のあるロー数を返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function CountUploadRows( _
      ByVal mask As Integer, _
      ByVal threshold As Long _
    ) As Long

// C#
    public long CountUploadRows(
      int mask,
      long threshold
    );
```

パラメータ
- **mask** ローのチェック対象となるパブリケーションのセット。詳細については、ULPublicationSchema クラス を参照してください。
- **threshold** カウントするローの最大数。CountUploadRows の所要時間を制限します。値 0 は制限が最大であることを示します。値 1 は、同期の必要なローがあるかどうかを判別する場合に使用します。スレッショルド値は、[0,0x0fffffff] の範囲内である必要があります。

戻り値　指定したパブリケーションからアップロードする必要のあるロー数

説明　このメソッドは、System.UInt32 型をネイティブでサポートしていない言語を対象として用意されています。この型がアプリケーションでサポートされている場合は、このメソッドの別のフォームを使用します。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- CountUploadRows メソッド

CreateCommand メソッド

この接続と現在のトランザクションに関連付けられる ULCommand クラス オブジェクトを作成して初期化します。ULCommand のプロパティを使用して、その動作を制御できます。

プロトタイプ

```
' Visual Basic
    Public Function CreateCommand() As ULCommand

// C#
    public ULCommand CreateCommand();
```

戻り値

新しい ULCommand クラス オブジェクト

説明

CommandText プロパティを設定してから、コマンドを実行する必要があります。

これは、IDbConnection.CreateCommand が厳密に型指定されたものです。

ExecuteTable メソッド

UL 拡張：直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。テーブルのプライマリ・キーを使用して、テーブルが開かれます (ソートされます)。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteTable( _
      ByVal tableName As String _
    ) As ULTable
```

```
// C#
    public ULTable ExecuteTable(
      string  tableName
    );
```

パラメータ

- **tableName** 開くテーブルの名前

戻り値

ULTable クラス オブジェクトとしてのテーブル

説明

このメソッドは、ULCommand クラス インスタンスを必要としない ExecuteTable メソッドのショートカットです。また、以前のバージョンの Ultra Light.NET から移行できるようにするために用意されています (iAnywhere.UltraLite.Connection.GetTable() と iAnywhere.UltraLite.Table.Open() が置き換えられます)。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException*tableName* が無効です。

例

次のコードでは、テーブルのプライマリ・キーを使用して MyTable というテーブルが開かれます。また、conn という開いた ULConnection インスタンスが想定されています。

```
' Visual Basic Dim t As ULTable =
conn.ExecuteTable("MyTable")
 ' The line above is equivalent to
 ' Dim cmd As ULCommand = conn.CreateCommand()
 ' cmd.CommandText = "MyTable"
 ' cmd.CommandType = CommandType.TableDirect
 ' Dim t As ULTable = cmd.ExecuteTable()
 ' cmd.Dispose()


// C# ULTable t = conn.ExecuteTable("MyTable");
 // The line above is equivalent to
 // ULTable t;
 // using(ULCommand cmd = conn.CreateCommand())
 // {
 //     cmd.CommandText = "MyTable";
 //     cmd.CommandType = CommandType.TableDirect;
 //     t = cmd.ExecuteTable();
 // }
```

参照

- ULConnection クラス
- ULConnection メンバ
- ExecuteTable メソッド
- ExecuteTable メソッド

ExecuteTable メソッド

UL 拡張：直接の操作用に、ULTable クラス のデータベース・テーブルを取り出します。テーブルは、指定したインデックスを使用して開かれます (ソートされます)。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ExecuteTable( _
      ByVal tableName As String, _
      ByVal indexName As String _
    ) As ULTable

// C#
    public ULTable ExecuteTable(
      string  tableName,
      string  indexName
    );
```

パラメータ

- **tableName** 開くテーブルの名前
- **indexName** テーブルを開く (ソートする) インデックスの名前

戻り値

ULTable クラス オブジェクトとしてのテーブル

説明

このメソッドは、ULCommand クラス インスタンスを必要としない ExecuteTable メソッドのショートカットです。また、以前のバージョンの Ultra Light.NET から移行できるようにするために用意されています (iAnywhere.UltraLite.Connection.GetTable() と iAnywhere.UltraLite.Table.Open() が置き換えられます)。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException*tableName* が無効です。

例

次のコードでは、MyIndex というインデックスを使用して MyTable というテーブルが開かれます。また、conn という開いた ULConnection インスタンスが想定されています。

```
' Visual Basic Dim t As ULTable =
conn.ExecuteTable("MyTable", "MyIndex")
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
' cmd.CommandText = "MyTable"
' cmd.IndexName = "MyIndex"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable = cmd.ExecuteTable()
' cmd.Dispose()


// C# ULTable t = conn.ExecuteTable("MyTable",
"MyIndex");
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.IndexName = "MyIndex";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable();
// }
```

参照

- ULConnection クラス
- ULConnection メンバ
- ExecuteTable メソッド
- ExecuteTable メソッド

ExecuteTable メソッド

UL 拡張：直接の操作用に、コマンド動作を指定してデータベース・テーブルを取り出します。テーブルは、指定したインデックスを使用して開かれます (ソートされます)。

プロトタイプ

' Visual Basic

Overloads Public Function **ExecuteTable(_**
ByVal *tableName* As String, _

```
    ByVal indexName As String, _
    ByVal cmdBehavior As CommandBehavior _
  ) As ULTable

// C#
  public ULTable ExecuteTable(
    string  tableName,
    string  indexName,
    CommandBehavior cmdBehavior
  );
```

パラメータ

- **tableName** 開くテーブルの名前
- **indexName** テーブルを開く (ソートする) インデックスの名前
- **cmdBehavior** クエリの結果の記述と、接続への影響の記述の CommandBehavior フラグのビット処理の組み合わせ。Ultra Light.NET で有効なのは、CommandBehavior.Default フラグ、CommandBehavior.CloseConnection フラグ、CommandBehavior.SchemaOnly フラグだけです。

戻り値

ULTable クラス オブジェクトとしてのテーブル

説明

このメソッドは、ULCommand クラス インスタンスを必要としない ExecuteTable メソッドのショートカットです。また、以前のバージョンの Ultra Light.NET から移行できるようにするために用意されています (iAnywhere.UltraLite.Connection.GetTable() と iAnywhere.UltraLite.Table.Open() が置き換えられます)。

例外

- ULException クラス - SQL エラーが発生しました。
- InvalidOperationException*tableName* が無効です。

例

次のコードでは、MyIndex というインデックスを使用して MyTable というテーブルが開かれます。また、conn という開いた ULConnection インスタンスが想定されています。

```
' Visual Basic Dim t As ULTable = conn.ExecuteTable( _
     "MyTable", "MyIndex", CommandBehavior.Default _
   )
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
```

```
' cmd.CommandText = "MyTable"
' cmd.IndexName = "MyIndex"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable =
cmd.ExecuteTable(CommandBehavior.Default)
' cmd.Dispose()


// C# ULTable t = conn.ExecuteTable(
     "MyTable", "MyIndex", CommandBehavior.Default
   );
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.IndexName = "MyIndex";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable(CommandBehavior.Default);
// }
```

参照

- ULConnection クラス
- ULConnection メンバ
- ExecuteTable メソッド
- ExecuteTable メソッド

GetLastDownloadTime メソッド

UL 拡張：指定されたパブリケーションの最後のダウンロードの時刻を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetLastDownloadTime( _
      ByVal mask As Integer _
    ) As Date

// C#
    public DateTime GetLastDownloadTime(
      int mask
    );
```

パラメータ

- **mask** チェックするパブリケーションのマスク。詳細については、ULPublicationSchema クラスを参照してください。

戻り値

最後のダウンロードのタイムスタンプ

説明

パラメータ *mask* は、単一のパブリケーションを参照するか、データベース全体の最後のダウンロード時刻の特殊定数 SYNC_ALL_DB フィールド である必要があります。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- ResetLastDownloadTime メソッド

GetNewUUID メソッド

UL 拡張：新しい UUID を生成します (Guid)。

プロトタイプ

```
' Visual Basic
    Public Function GetNewUUID() As Guid

// C#
    public Guid GetNewUUID();
```

戻り値

Guid としての新しい UUID

説明

このメソッドは、.NET Compact Framework に含まれていないため、ここで提供されています。

例外

- ULException クラス - SQL エラーが発生しました。

GrantConnectTo メソッド

UL 拡張：指定されたパスワードを持つユーザ ID に、Ultra Light データベースへのアクセスを許可します。

プロトタイプ

```
' Visual Basic
    Public Sub GrantConnectTo( _
      ByVal uid As String, _
      ByVal pwd As String _
    )

// C#
    public void GrantConnectTo(
      string  uid,
      string  pwd
    );
```

パラメータ

- **uid** データベースへのアクセス権を受け取るユーザ ID。ユーザ ID の最大長は 16 文字です。
- **pwd** ユーザ ID に関連付けられるパスワード。最大長は 16 文字です。

説明

既存のユーザ ID が指定されていれば、この関数を使用してそのユーザのパスワードを更新します。Ultra Light では、最大で 4 人のユーザがサポートされます。このメソッドが有効になるのは、接続が開かれたときにユーザ認証が有効になっていた場合だけです。

参照

- ULConnection クラス
- ULConnection メンバ
- UserID プロパティ
- Password プロパティ
- ConnectionString プロパティ

Open メソッド

以前に指定した接続文字列を使用してデータベースへの接続を開きます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Open() _
      Implements IDbConnection.Open

// C#
    public void Open();
```

説明

接続は、使用し終わったら明示的に閉じるか、破棄する必要があります。

例外

- ULException クラス - データベースを開くときに SQL エラーが発生しました。
- InvalidOperationException - 接続がすでに開かれているか、接続文字列が ConnectionString プロパティ で指定されていません。

実装

IDbConnection.Open

参照

- ULConnection クラス
- ULConnection メンバ
- ConnectionString プロパティ
- OpenWithCreate メソッド
- State プロパティ

OpenWithCreate メソッド

UL 拡張：以前に指定した接続文字列を使用してデータベースへの接続を開きます。データベースが存在しない場合、新しいデータベースが作成されます。

プロトタイプ

```
' Visual Basic
    Public Sub OpenWithCreate()

// C#
    public void OpenWithCreate();
```

説明

新しいデータベースを作成するには、そのデータベースのスキーマ・ファイルの場所が接続文字列に含まれている必要があります。

接続は、使用し終わったら明示的に閉じるか、破棄する必要があります。

例外

- ULException クラス - データベースを開くとき、またはデータベースの作成中に SQL エラーが発生しました。
- InvalidOperationException - 接続がすでに開かれているか、接続文字列が ConnectionString プロパティ で指定されていません。

参照

- ULConnection クラス
- ULConnection メンバ
- ConnectionString プロパティ
- SchemaOnCE プロパティ
- SchemaOnDesktop プロパティ
- Open メソッド
- State プロパティ
- DropDatabase メソッド

ResetLastDownloadTime メソッド

UL 拡張：最後のダウンロードの時刻をリセットします。

プロトタイプ

```
' Visual Basic
    Public Sub ResetLastDownloadTime( _
      ByVal mask As Integer _
    )

// C#
    public void ResetLastDownloadTime(
      int mask
    );
```

パラメータ

- **mask** リセットするパブリケーションのセット。詳細については、ULPublicationSchema クラスを参照してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- GetLastDownloadTime メソッド

RevokeConnectFrom メソッド

UL 拡張：指定したユーザ ID から、Ultra Light データベースへのアクセス権を取り消します。

プロトタイプ

```
' Visual Basic
    Public Sub RevokeConnectFrom( _
      ByVal uid As String _
    )

// C#
    public void RevokeConnectFrom(
      string  uid
    );
```

パラメータ

- **uid** データベースへのアクセス権を取り消すユーザ ID

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- GrantConnectTo メソッド

RollbackPartialDownload メソッド

UL 拡張：部分的なダウンロードから、未処理の変更をデータベースにロールバックします。

プロトタイプ

```
' Visual Basic
    Public Sub RollbackPartialDownload()

// C#
    public void RollbackPartialDownload();
```

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- KeepPartialDownload プロパティ
- ResumePartialDownload プロパティ

StartSynchronizationDelete メソッド

UL 拡張：同期用に、この接続によって行われる以後のすべての削除にマークを付けます。

プロトタイプ

```
' Visual Basic
    Public Sub StartSynchronizationDelete()

// C#
    public void StartSynchronizationDelete();
```

説明

この関数が呼び出されると、すべての削除操作が再度同期され、Ultra Light データベースから削除されたローが統合データベースからも削除されます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- StopSynchronizationDelete メソッド
- Truncate メソッド

StopSynchronizationDelete メソッド

UL 拡張：削除操作が同期されないようにします。

プロトタイプ

```
' Visual Basic
    Public Sub StopSynchronizationDelete()

// C#
    public void StopSynchronizationDelete();
```

説明

領域を節約するために Ultra Light データベースの古い情報を削除して、統合データベースにはそれを残しておく場合に、このメソッドを使用すると便利です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ

- StartSynchronizationDelete メソッド

Synchronize メソッド

UL 拡張：現在の SyncParms プロパティ を使用してデータベースを同期します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Synchronize()

// C#
    public void Synchronize();
```

説明

結果の詳細なステータスは、この接続の SyncResult プロパティでレポートされます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- Synchronize メソッド

Synchronize メソッド

UL 拡張：指定したリスナに送信されるプログレス・イベントとともに、現在の SyncParms プロパティ を使用してデータベースを同期します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Synchronize( _
      ByVal listener As ULSyncProgressListener _
    )

// C#
    public void Synchronize(
      ULSyncProgressListener listener
    );
```

パラメータ

- **listener** 同期プログレス・イベントを受信するオブジェクト

説明

リスナに送信された最後のイベントのステータスは、ULSyncProgressState 列挙 です。

同期中のエラーは、ULSyncProgressState 列挙 イベントとして送信され、ULException クラス としてスローされます。

結果の詳細なステータスは、この接続の SyncResult プロパティでレポートされます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULConnection クラス
- ULConnection メンバ
- ULSyncProgressListener インタフェース
- Synchronize メソッド

InfoMessage イベント

Ultra Light.NET が、この接続の警告または情報メッセージを送信するときに発生します。

プロトタイプ

' Visual Basic

```
Public Event InfoMessage As ULInfoMessageEventHandler
```

// C#

```
public event ULInfoMessageEventHandler InfoMessage;
```

説明

Ultra Light.NET の警告または情報メッセージを処理するには、ULInfoMessageEventHandler 委譲 を作成し、このイベントにアタッチする必要があります。

例

次のコードでは、情報メッセージのイベント・ハンドラが定義されます。

```
' Visual Basic Private Sub MyInfoMessageHandler( _
       obj As Object, args As ULInfoMessageEventArgs _
    )
   System.Console.WriteLine( _
      "InfoMesageHandler: " + args.NativeError + ", " _
       + args.Message _
    )
```

```
End Sub

// C# private void MyInfoMessageHandler(
      object obj, ULInfoMessageEventArgs args
    )
{
  System.Console.WriteLine(
      "InfoMesageHandler: " + args.NativeError + ", "
      + args.Message
    );
}
```

次のコードでは、MyInfoMessageHandler が conn という接続に追加されます。

```
' Visual Basic AddHandler conn.InfoMessage, AddressOf
MyInfoMessageHandler

// C# conn.InfoMessage +=
  new ULInfoMessageEventHandler(MyInfoMessageHandler);
```

StateChange イベント

この接続のステータスが変更されると発生します。

プロトタイプ

' Visual Basic

Public Event **StateChange** As StateChangeEventHandler

// C#

public event StateChangeEventHandler **StateChange**;

説明

ステータス変更メッセージを処理するには、StateChangeEventHandler デリゲートを作成し、このイベントにアタッチする必要があります。

例

次のコードでは、ステータス変更のイベント・ハンドラが定義されます。

```
' Visual Basic Private Sub MyStateHandler( _
      obj As Object, args As StateChangeEventArgs _
    )
  System.Console.WriteLine( _
      "StateHandler: " + args.OriginalState + " to " _
```

```
        + args.CurrentState _
    )
End Sub

// C# private void MyStateHandler(
      object obj, StateChangeEventArgs args
    )
{
  System.Console.WriteLine(
      "StateHandler: " + args.OriginalState + " to "
      + args.CurrentState
    );
}
```

次のコードでは、MyStateHandler が conn という接続に追加されます。

```
' Visual Basic AddHandler conn.StateChange, AddressOf
MyStateHandler

// C# conn.StateChange += new
StateChangeEventHandler(MyStateHandler);
```

ULConnectionParms クラス

UL 拡張：Ultra Light データベースへの接続を開く接続文字列を作成します。頻繁に使用する接続パラメータは、ULConnectionParms オブジェクトの個々のプロパティです。

プロトタイプ

```
' Visual Basic
    Public Class ULConnectionParms
      Inherits Component

// C#
    public class ULConnectionParms :
      Component
```

説明

ULConnectionParms オブジェクトは、接続を開くパラメータ (Open メソッド)、データベースを作成するパラメータ (OpenWithCreate メソッド)、データベースを削除するパラメータ (DropDatabase メソッド)、スキーマ・ファイルを指定するパラメータ (ApplyFile メソッド) を指定するために使用します。

前後のスペースは無視されます。また、値にはセミコロン (;)、カッコ ()、中カッコ {} を使用しないでください。今後の互換性を考慮し、値には一重引用符 (') または二重引用符 (") を使用しないことを強くおすすめします。

接続文字列を作成するときは、データベースを識別し、オプションの接続設定を指定します。さらに、OpenWithCreate メソッド を使用してデータベースを作成する場合はデータベース・スキーマ・ファイルを指定します。ULConnectionParms オブジェクトで適切なプロパティを設定して、接続パラメータをすべて指定したら、ToString メソッド を使用して接続文字列を作成します。作成した文字列は、ULConnection コンストラクタ で新しい ULConnection クラス を作成したり、既存の ULConnection クラス オブジェクトの ConnectionString プロパティを設定したりするのに使用されます。

データベースの識別

各インスタンスには、データベースへのプラットフォーム固有のパスがあります。実行されているプラットフォームに対応する値だけが使用されます。たとえば、次のコードでは、パス ¥UltraLite¥mydb1.udb は Windows CE で使用され、mydb2.udb はその他のプラットフォームで使用されます。

```
' Visual Basic Dim dbName As ULConnectionParms = new
ULConnectionParms
 dbName.DatabaseOnCE = "¥UltraLite¥mydb1.udb"
 dbName.DatabaseOnDesktop = "mydb2.udb"

 // C# ULConnectionParms dbName = new
ULConnectionParms();
 dbName.DatabaseOnCE = "¥¥UltraLite¥¥mydb1.udb";
 dbName.DatabaseOnDesktop = "mydb2.udb";
```

Ultra Light データベース・ファイルに推奨されるファイル拡張子は .udb です。Windows CE デバイスでは、デフォルトのデータベースは ¥UltraLiteDB¥ulstore.udb です。その他の Windows プラットフォームでは、デフォルトのデータベースは ulstore.udb です。C# では、パス内の円記号をエスケープするか、アットマーク (@) で囲まれた文字列リテラルを使用する必要があります。

複数のデータベースを使用している場合は、各データベースのデータベース名を指定する必要があります。詳細については、AdditionalParms プロパティを参照してください。

オプションの接続設定

アプリケーションのニーズおよびデータベースの作成方法に応じて、デフォルト以外の UserID プロパティ と Password プロパティ、データベース EncryptionKey プロパティ、接続の CacheSize プロパティ を指定しなければならない場合があります。アプリケーションで複数の接続を使用している場合は、それぞれの接続に対してユニークな ConnectionName プロパティを指定する必要があります。

データベースは、1 人の認証済みユーザ DBA で作成されます。このユーザの最初のパスワードは SQL です。デフォルトでは、ユーザ ID DBA とパスワード SQL を使用して、接続が開かれます。デフォル

ト・ユーザを無効にするには、RevokeConnectFrom メソッド を使用します。ユーザを追加したり、ユーザのパスワードを変更したりするには、GrantConnectTo メソッドを使用します。

データベースを作成したときに暗号化キーを指定した場合は、そのデータベースへの以後の接続では、すべて同じ暗号化キーを使用する必要があります。データベースの暗号化キーを変更するには、ChangeEncryptionKey メソッドを使用します。

スキーマの識別

データベースを作成するときは、スキーマを指定する必要があります。スキーマ・ファイルへのパスを指定するには、プラットフォーム固有のプロパティ SchemaOnCE プロパティ および SchemaOnDesktop プロパティ を使用します。実行されているプラットフォームに対応する値だけが使用されます。

詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「接続パラメータ」を参照してください。

ULConnectionParms メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULConnectionParms コンストラクタ	ULConnectionParms インスタンスを、そのデフォルト値で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AdditionalParms プロパティ	「名前 = 値」の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。
CacheSize プロパティ	キャッシュのサイズを指定します。

メンバ	説明
ConnectionName プロパティ	接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。
DatabaseOnCE プロパティ	Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。
DatabaseOnDesktop プロパティ	Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。
EncryptionKey プロパティ	データベースを暗号化するためのキーを指定します。
Password プロパティ	認証済みユーザのパスワードを指定します。
SchemaOnCE プロパティ	Windows CE 上の Ultra Light スキーマのパスとファイル名を指定します。
SchemaOnDesktop プロパティ	Windows デスクトップ・プラットフォーム 上の Ultra Light スキーマのパスとファイル名を指定します。
UserID プロパティ	データベースで認証されるユーザを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド	このインスタンスの文字列表現を返します。

パブリック・インスタンス・イベント

メンバ	説明
UnusedEvent イベント	使われていません。

ULConnectionParms コンストラクタ

ULConnectionParms インスタンスを、そのデフォルト値で初期化します。

プロトタイプ

```
' Visual Basic
    Public Sub New()

// C#
    public ULConnectionParms();
```

AdditionalParms プロパティ

「名前 = 値」の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。

プロトタイプ

```
' Visual Basic
    Public Property AdditionalParms As String

// C#
    public string  AdditionalParms {get;set;}
```

プロパティ値

「キーワード = 値」をセミコロンで区切ったリストによる追加のパラメータ。デフォルト値は null 参照 (Visual Basic の Nothing) です。

説明

ページ・サイズや予約サイズのパラメータの値は、バイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス m または M を使用します。

追加のパラメータを次に示します。

キーワード	説明
dbn	接続する必要のある、ロードしたデータベースを識別します。 データベースが起動されると、データベース名が割り当てられます。データベース名は、dbn パラメータを使用して明示的に割り当てられるか、または拡張子とパスが削除されたベース・ファイル名を使用して Ultra Light によって割り当てられます。 接続が開かれると、Ultra Light は dbn が一致する実行中のデータベースをまず検索します。一致するデータベースが見つからない場合、Ultra Light は適切なデータベースのファイル名のパラメータ (DatabaseOnCE プロパティ または DatabaseOnDesktop プロパティ) を使用して、新しいデータベースを起動します。 このパラメータが必要なのは、同じベース・ファイル名を持つ 2 つの異なるデータベースにアプリケーション (または Ultra Light エンジン) がアクセスする場合です。 このパラメータが使用されるのは、Open メソッド または OpenWithCreate メソッド を使用して接続を開いたときだけです。
obfuscate	1 に設定すると、難読化されたデータベースを作成することが指定されます。次に例を示します。 `createParms.AdditionalParms = "obfuscate=1"` 難読化することによって、表示ツールでデータベース内のデータを直接表示しようとする単純な行為に対するセキュリティが提供されます。巧妙で容赦ないデータへのアクセスが試みられた場合は、このオプションでは防止できません。難読化は、パフォーマンスにほとんど影響しません。 このパラメータが使用されるのは、OpenWithCreate メソッド 使用して新しいデータベースを作成したときだけです。

キーワード	説明
page_size	Ultra Light データベースはページ単位で格納されます。I/O 操作は一度に 1 ページずつ実行されます。Ultra Light データベースのデフォルト・ページ・サイズは 4 KB です。次の保管パラメータ文字列を使用すると 2 KB のページを指定できます。 `connParms.AdditionalParms = "page_size=2k"` ページ・サイズを 2 KB に設定すると、最大テーブル数が約 500 に減少します。 このパラメータが使用されるのは、OpenWithCreate メソッド使用して新しいデータベースを作成したときだけです。

キーワード	説明
reserve_size	Ultra Light 継続データの保管に使用する、ファイル・システム領域を予約します。 reserve_size パラメータを使用すると、データを挿入することなく、Ultra Light データベースが必要とするファイル・システム領域を事前に割り付けることができます。ファイル・システムの領域を予約すると、パフォーマンスが多少向上し、メモリが不足するという障害を防ぐことができます。 デフォルトでは、永続ストレージ・ファイルのサイズは、アプリケーションがデータベースを更新して、サイズを大きくする必要が生じた場合にだけ大きくなります。 reserve_size で予約されるファイル・システム領域には、ロー・データだけでなく、継続保管ファイルのメタデータも含まれることに注意してください。データベースのデータ量から、必要なファイル・システム領域を計算する場合は、メタデータのオーバヘッドとデータの圧縮を考慮してください。テスト・データを入れてデータベースを実行し、継続保管ファイルのサイズを確認することをおすすめします。 reserve_size パラメータは、起動時に継続保管ファイルを設定された予約サイズまで大きくすることにより、領域を予約します。これは、そのファイルが以前に存在していたかどうかに関係なく行われます。このファイルはトランケートされません。 次のパラメータ文字列では、起動時に継続保管ファイルのサイズが最低 2 MB 確保されます。 `createParms.` **`AdditionalParms = "reserve_size=2m"`** このパラメータが使用されるのは、Open メソッド または OpenWithCreate メソッド を使用して接続を開いたときだけです。

例外

- ArgumentException - 値にカッコ () または中カッコ {} が含まれています。

CacheSize プロパティ

キャッシュのサイズを指定します。

プロトタイプ

```
' Visual Basic
    Public Property CacheSize As String

// C#
    public string  CacheSize {get;set;}
```

プロパティ値

キャッシュ・サイズを指定する文字列。デフォルトは null 参照 (Visual Basic の Nothing) です。この場合、デフォルトの 16 ページが使用されます。

説明

キャッシュ サイズの値は、バイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス m または M を使用します。

たとえば、次の例ではキャッシュ・サイズが 128 KB に設定されます。

```
connParms.CacheSize = "128k"
```

デフォルトのキャッシュ・サイズは 16 ページです。デフォルトのページ・サイズは 4 KB なので、デフォルトのキャッシュ・サイズは 64 KB です。キャッシュの最小サイズは、プラットフォームによって異なります。

デフォルトのキャッシュ・サイズは小さめの値です。テスト結果により、パフォーマンスの向上が必要と判断した場合は、キャッシュ・サイズを大きくしてください。

キャッシュ・サイズをデータベース自体のサイズよりも大きくすると、パフォーマンスは向上しません。また、キャッシュ・サイズが大きいと、その他の使用可能なアプリケーションの数が少なくなることがあります。

キャッシュ・サイズが未指定であるか、正しく指定されていない場合は、デフォルトのサイズが使用されます。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

ConnectionName プロパティ

接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。

プロトタイプ

```
' Visual Basic
    Public Property ConnectionName As String

// C#
    public string  ConnectionName {get;set;}
```

プロパティ値

接続の名前を指定する文字列。デフォルト値は null 参照 (Visual Basic の Nothing) です。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

DatabaseOnCE プロパティ

Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseOnCE As String

// C#
    public string  DatabaseOnCE {get;set;}
```

プロパティ値

データベースへのフル・パスを指定する文字列。値が null 参照 (Visual Basic の Nothing) の場合、デフォルトのデータベース ¥UltraLiteDB¥ulstore.udb が使用されます。C# では、パス内の円記号をエスケープするか、アットマーク (@) で囲まれた文字列リテラルを使用する必要があります。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

DatabaseOnDesktop プロパティ

Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseOnDesktop As String

// C#
    public string  DatabaseOnDesktop {get;set;}
```

プロパティ値

データベースへの絶対パスまたは相対パスを指定する文字列。値が null 参照 (Visual Basic の Nothing) の場合、デフォルトのデータベース ulstore.udb が使用されます。C# では、パス内の円記号をエスケープするか、アットマーク (@) で囲まれた文字列リテラルを使用する必要があります。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

EncryptionKey プロパティ

データベースを暗号化するためのキーを指定します。

プロトタイプ

```
' Visual Basic
    Public Property EncryptionKey As String

// C#
    public string  EncryptionKey {get;set;}
```

プロパティ値

暗号化キーを指定する文字列。デフォルト値は null 参照 (Visual Basic の Nothing) で、暗号化は行われません。

説明

すべての接続では、データベースが作成されたときに指定されたキーと同じキーを使用する必要があります。キーを忘れた場合はデータベースにまったくアクセスできなくなります。

すべてのパスワードに共通しますが、最善の方法は、簡単には推測できないキー値を選択することです。キーの長さは任意ですが、短いと推測されやすいため、一般的には長い方が適しています。また、数字、文字、特殊文字を組み合わせると、キーは推測されにくくなります。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

参照

- ULConnectionParms クラス
- ULConnectionParms メンバ
- ChangeEncryptionKey メソッド

Password プロパティ

認証済みユーザのパスワードを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Password As String

// C#
    public string  Password {get;set;}
```

プロパティ値

データベースのユーザ ID を指定する文字列。デフォルトは SQL です。

説明

データベースが作成されると、ユーザ ID DBA のパスワードは SQL に設定されます。データベースで大文字と小文字が区別されない場合は、パスワードの大文字と小文字は区別されません。データベースで大文字と小文字が区別される場合は、パスワードの大文字と小文字も区別されます。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

参照

- ULConnectionParms クラス
- ULConnectionParms メンバ
- UserID プロパティ

SchemaOnCE プロパティ

Windows CE 上の Ultra Light スキーマのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property SchemaOnCE As String

// C#
    public string  SchemaOnCE {get;set;}
```

プロパティ値 スキーマ・ファイルへのフル・パスを指定する文字列。デフォルトは null 参照 (Visual Basic の Nothing) です。この場合、このプラットフォームにはスキーマ・ファイルは指定されません。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

SchemaOnDesktop プロパティ

Windows デスクトップ・プラットフォーム 上の Ultra Light スキーマのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property SchemaOnDesktop As String

// C#
    public string  SchemaOnDesktop {get;set;}
```

プロパティ値 スキーマ・ファイルへの絶対パスまたは相対パスを指定する文字列。デフォルトは null 参照 (Visual Basic の Nothing) です。この場合、このプラットフォームにはスキーマ・ファイルは指定されません。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

UserID プロパティ

データベースで認証されるユーザを指定します。

プロトタイプ

```
' Visual Basic
    Public Property UserID As String

// C#
    public string  UserID {get;set;}
```

プロパティ値

データベースのユーザ ID を指定する文字列。デフォルト値は DBA です。

説明

データベースは最初、DBA という名前の 1 人の認証済みユーザで作成されます。データベースで大文字と小文字が区別されない場合は、ユーザ ID の大文字と小文字は区別されません。データベースで大文字と小文字が区別される場合は、ユーザ ID の大文字と小文字も区別されます。

例外

- ArgumentException - 値にセミコロン (;)、カッコ ()、中カッコ {} が含まれています。

参照

- ULConnectionParms クラス
- ULConnectionParms メンバ
- Password プロパティ
- GrantConnectTo メソッド
- RevokeConnectFrom メソッド
- IsCaseSensitive プロパティ

ToString メソッド

このインスタンスの文字列表現を返します。

プロトタイプ

```
' Visual Basic
    Overrides Public Function ToString() As String

// C#
    public override string  ToString();
```

戻り値

「キーワード = 値」の組み合わせがセミコロンで区切られたリスト形式の、このインスタンスの文字列表現

UnusedEvent イベント

使われていません。

プロトタイプ

```
' Visual Basic
    Public Event UnusedEvent As ULConnectionParms.UnusedEventHandler

// C#
    public event ULConnectionParms.UnusedEventHandler UnusedEvent;
```

説明

この public Event は、Visual Basic .NET プロジェクトでこのクラスを組み込む場合の Visual Studio .NET のバグを修正するために用意されています。実用的な用途はありません。

ULConnectionParms.UnusedEventHandler 委譲

UL 拡張：使われていません。

プロトタイプ

```
' Visual Basic
    Delegate Sub ULConnectionParms.UnusedEventHandler( _
      ByVal sender As Object, _
      ByVal args As System.EventArgs _
    )

// C#
    delegate void ULConnectionParms.UnusedEventHandler(
      object sender,
      System.EventArgs args
    );
```

パラメータ

- **sender** 送信側のオブジェクト
- **args** イベント引数

説明

この public Delegate は、Visual Basic .NET プロジェクトでこのクラスを組み込む場合の Visual Studio .NET のバグを修正するために用意されています。実用的な用途はありません。

ULCursorSchema クラス

UL 拡張：Ultra Light.NET カーソルのスキーマを示します。

プロトタイプ

```
' Visual Basic
    MustInherit Public Class ULCursorSchema

// C#
    public abstract class ULCursorSchema
```

説明

このクラスは、ULTableSchema クラスおよび ULResultSetSchema クラス の抽象基本クラスです。

iAnywhere.UltraLite ネームスペースから移行するユーザへの注意：カラム ID は、iAnywhere.UltraLite ネームスペース内にあるため、1 ではなく 0 から始まります。

ULCursorSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ	このカーソル内のカラム数を返します。
IsOpen プロパティ	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	カーソルの名前を返します。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnID メソッド	指定したカラムのカラム ID を返します。
GetColumnName メソッド	指定したカラム ID で識別されたカラムの名前を返します。

メンバ	説明
GetColumnPrecision メソッド	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの精度を返します。
GetColumnScale メソッド	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの位取りを返します。
GetColumnSize メソッド	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラム ID によって識別されたカラムのサイズを返します。
GetColumnULDbType メソッド	指定したカラム ID によって識別されたカラムの Ultra Light.NET データ型を返します。
GetSchemaTable メソッド	ULDataReader クラス のカラムのスキーマが記述された DataTable を返します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	ULCursorSchema によって使用されるアンマネージ・リソースを解放します。

ColumnCount プロパティ

このカーソル内のカラム数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ColumnCount As Short

// C#
    public short ColumnCount {get;}
```

プロパティ値

カーソル内のカラム数、またはカーソルのスキーマが閉じている場合は 0

説明

カラム ID の範囲は、0 ～ ColumnCount-1 です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

IsOpen プロパティ

カーソルのスキーマが現在開いているかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値 カーソルのスキーマが現在開いている場合は true、閉じている場合は false

Name プロパティ

カーソルの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

プロパティ値 文字列としてのカーソル名

Finalize メソッド

ULCursorSchema によって使用されるアンマネージ・リソースを解放します。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()
```

```
// C#
    protected override void Finalize();
```

GetColumnID メソッド

指定したカラムのカラム ID を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnID( _
      ByVal name As String _
    ) As Short

// C#
    public short GetColumnID(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

指定したカラムのカラム ID

説明

カラム ID の範囲は、0 ～ ColumnCount プロパティ -1 です。

結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。エイリアスを使用していない場合は、非計算カラムの名前には、そのカラムの元となるテーブルの名前がプレフィクスとして付けられます。たとえば、MyTable.ID は、クエリ "SELECT ID FROM MyTable" の結果セットに含まれる唯一のカラムの名前です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- ColumnCount プロパティ

GetColumnName メソッド

指定したカラム ID で識別されたカラムの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnName( _
      ByVal columnID As Integer _
    ) As String

// C#
    public string  GetColumnName(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。

戻り値

カラムの名前、またはカラムに名前がない場合は null 参照 (Visual Basic の Nothing)。SQL クエリでカラムのエイリアスが使用されている場合は、そのエイリアスが返されます。

説明

結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。エイリアスを使用していない場合は、非計算カラムの名前には、そのカラムの元となるテーブルの名前がプレフィクスとして付けられます。たとえば、MyTable.ID は、クエリ "SELECT ID FROM MyTable" の結果セットに含まれる唯一のカラムの名前です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- ColumnCount プロパティ

GetColumnPrecision メソッド

カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの精度を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnPrecision( _
      ByVal columnID As Integer _
    ) As Integer

// C#
    public int GetColumnPrecision(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。

戻り値

指定された数値カラムの精度

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- GetColumnULDbType メソッド

GetColumnScale メソッド

カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの位取りを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnScale( _
      ByVal columnID As Integer _
    ) As Integer

// C#
    public int GetColumnScale(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。

戻り値

指定された数値カラムの位取り

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- GetColumnULDbType メソッド

GetColumnSize メソッド

カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラム ID によって識別されたカラムのサイズを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnSize( _
      ByVal columnID As Integer _
    ) As Integer

// C#
    public int GetColumnSize(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。

戻り値

サイズ指定された、指定したカラムのサイズ

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- GetColumnULDbType メソッド

GetColumnULDbType メソッド

指定したカラム ID によって識別されたカラムの Ultra Light.NET データ型を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnULDbType( _
      ByVal columnID As Integer _
    ) As ULDbType

// C#
    public ULDbType GetColumnULDbType(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。

戻り値

ULDbType 列挙整数

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULCursorSchema クラス
- ULCursorSchema メンバ
- ULDbType 列挙
- ColumnCount プロパティ

GetSchemaTable メソッド

ULDataReader クラス のカラムのスキーマが記述された DataTable を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetSchemaTable() As DataTable

// C#
    public DataTable GetSchemaTable();
```

戻り値

カラムのスキーマが記述された DataTable

説明

詳細については、GetSchemaTable メソッド を参照してください。

ULDataAdapter クラス

DataSet に設定したりデータベースを更新したりするために使用する一連のコマンドとデータベース接続を示します。

プロトタイプ

' Visual Basic

NotInheritable Public Class **ULDataAdapter**

// C#

public sealed class **ULDataAdapter**

説明

DataSet は、Ultra Light データベースとは別に、データをオフラインで処理するための方法を提供します。ULDataAdapter は、DataSet を一連の SQL 文に関連付けるためのメソッドを提供します。

Ultra Light はローカルのデータベースであり、Mobile Link には競合解決があるため、ULDataAdapter の使用は制限されています。ほとんどの場合、ULDataReader クラスまたは ULTable クラスを使用すると、より効率的にデータにアクセスできます。

継承： DbDataAdapter

実装： IDbDataAdapter、IDataAdapter、IDisposable

ULDataAdapter メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULDataAdapter コンストラクタ	ULDataAdapter オブジェクトを初期化します。
ULDataAdapter コンストラクタ	ULDataAdapter オブジェクトを、指定した SELECT 文で初期化します。
ULDataAdapter コンストラクタ	ULDataAdapter オブジェクトを、指定した SELECT 文および接続で初期化します。

メンバ	説明
ULDataAdapter コンストラクタ	ULDataAdapter オブジェクトを、指定した SELECT 文および接続文字列で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AcceptChangesDuringFill プロパティ	DataTable に追加された後で、DataRow で DataRow.AcceptChanges を呼び出すかどうかを指定します。
ContinueUpdateOnError プロパティ	ローの更新時にエラーが検出されたときに例外を生成するかどうかを指定します。
DeleteCommand プロパティ	DataSet で削除されたローに該当するデータベース内のローを削除するために、DbDataAdapter.Update が呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。
InsertCommand プロパティ	DataSet に挿入されたローに該当するローをデータベースに挿入するために、DbDataAdapter.Update が呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。
MissingMappingAction プロパティ	受信データと一致するテーブルまたはカラムがない場合に実行するアクションを決定します。
MissingSchemaAction プロパティ	既存の DataSet スキーマが受信データと一致しない場合に実行するアクションを決定します。
SelectCommand プロパティ	結果セットをデータベースから取得して DataSet にコピーする Fill メソッド または FillSchema メソッド の実行中に使用される ULCommand クラス を指定します。
TableMappings プロパティ	ソース・テーブルと DataTable とのマスタ・マッピングを提供するコレクションを返します。

メンバ	説明
UpdateCommand プロパティ	DataSet で更新されたローに該当するデータベース内のローを更新するために、Update メソッドが呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
Dispose メソッド	ULDataAdapter によって使用されるアンマネージ・リソースを解放するほか、必要に応じてマネージ・リソースを解放します。
Fill メソッド	指定した DataSet の "Table" という DataTable でローの追加または再表示を行います。必要に応じて、"Table" という DataTable が作成され、DataSet に追加されます。
Fill メソッド	指定した DataSet の指定した DataTable でローの追加または再表示を行います。必要に応じて、指定した DataTable が作成され、DataSet に追加されます。
Fill メソッド	指定した DataSet の指定した DataTable で、指定した範囲のローの追加または再表示を行います。必要に応じて、指定した DataTable が作成され、DataSet に追加されます。
Fill メソッド	指定した DataTable でローの追加または再表示を行います。
FillSchema メソッド	"Table" という DataTable を DataSet に追加し、スキーマがデータ・ソースのスキーマと一致するように設定します。
FillSchema メソッド	"Table" という DataTable を DataSet に追加し、スキーマがデータ・ソースのスキーマと一致するように設定します。
FillSchema メソッド	DataTable のスキーマがデータ・ソースのスキーマと一致するように設定します。

メンバ	説明
GetFillParameters メソッド	SELECT 文の実行時にユーザによって設定されるパラメータを返します。
Update メソッド	指定した DataSet の "Table" という DataTable への変更内容を使用して、データベース内のローを更新します。
Update メソッド	指定した DataSet の指定した DataTable への変更内容を使用して、データベース内のローを更新します。
Update メソッド	指定した DataTable への変更内容を使用して、データベース内のローを更新します。
Update メソッド	指定した DataRow 配列への変更内容を使用して、データベース内のローを更新します。

パブリック・インスタンス・イベント

メンバ	説明
FillError イベント	設定オペレーションの実行時にエラーが検出されたときに発生します。
RowUpdated イベント	データ・ソースに対してコマンドが実行された後の更新時に発生します。更新が試みられると、イベントが発生します。
RowUpdating イベント	データ・ソースに対してコマンドが実行される前の更新時に発生します。更新が試みられると、イベントが発生します。

ULDataAdapter コンストラクタ

ULDataAdapter オブジェクトを初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public ULDataAdapter();
```

参照

- ULDataAdapter クラス
- ULDataAdapter メンバ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ

ULDataAdapter コンストラクタ

ULDataAdapter オブジェクトを、指定した SELECT 文で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal selectCommand As ULCommand _
    )

// C#
    public ULDataAdapter(
      ULCommand selectCommand
    );
```

パラメータ

- **selectCommand** DataSet に配置するためのレコードをデータ・ソースから選択するために DbDataAdapter.Fill の実行時に使用される ULCommand クラス オブジェクト

参照

- ULDataAdapter クラス
- ULDataAdapter メンバ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ

ULDataAdapter コンストラクタ

ULDataAdapter オブジェクトを、指定した SELECT 文および接続で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal selectCommandText As String, _
      ByVal selectConnection As ULConnection _
    )

// C#
    public ULDataAdapter(
      string  selectCommandText,
      ULConnection selectConnection
    );
```

パラメータ

- **selectCommandText** ULDataAdapter の SelectCommand プロパティによって使用される SELECT 文
- **selectConnection** データベースへの接続を定義する ULConnection クラス オブジェクト

参照

- ULDataAdapter クラス
- ULDataAdapter メンバ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ

ULDataAdapter コンストラクタ

ULDataAdapter オブジェクトを、指定した SELECT 文および接続文字列で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal selectCommandText As String, _
      ByVal selectConnectionString As String _
    )

// C#
    public ULDataAdapter(
      string  selectCommandText,
      string  selectConnectionString
    );
```

パラメータ

- **selectCommandText** ULDataAdapter の SelectCommand プロパティによって使用される SELECT 文
- **selectConnectionString** Ultra Light.NET データベースの接続文字列

参照

- ULDataAdapter クラス
- ULDataAdapter メンバ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ
- ULDataAdapter コンストラクタ

AcceptChangesDuringFill プロパティ

DataTable に追加された後で、DataRow で DataRow.AcceptChanges を呼び出すかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property AcceptChangesDuringFill As Boolean

// C#
    public bool AcceptChangesDuringFill {get;set;}
```

プロパティ値

ULDataAdapter が DataRow で DataRow.AcceptChanges 関数を呼び出すように指定する場合は true、AcceptChanges を呼び出さず、新しく追加されたローを挿入されたローとして処理する場合は false。デフォルトは true です。

ContinueUpdateOnError プロパティ

ローの更新時にエラーが検出されたときに例外を生成するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property ContinueUpdateOnError As Boolean

// C#
    public bool ContinueUpdateOnError {get;set;}
```

プロパティ値

例外を生成しないで更新を継続する場合は true、例外を生成する場合は false。デフォルトは false です。

説明

ContinueUpdateOnError が true の場合、ローの更新時にエラーが発生しても例外はスローされません。ローの更新はスキップされ、エラー情報はローの DataRow.RowError プロパティに格納されます。ULDataAdapter は後続のローの更新を継続します。

ContinueUpdateOnError が false の場合、エラーが発生すると例外がスローされます。

DeleteCommand プロパティ

DataSet で削除されたローに該当するデータベース内のローを削除するために、DbDataAdapter.Update が呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Property DeleteCommand As ULCommand

// C#
    public ULCommand DeleteCommand {get;set;}
```

プロパティ値

DataSet で削除されたローに該当するデータベース内のローを削除するために実行される ULCommand クラス オブジェクト

説明

DeleteCommand が既存の ULCommand クラス オブジェクトに割り当てられる場合、ULCommand クラス オブジェクトのクローンは作成されません。DeleteCommand は、既存の ULCommand クラス への参照を保持します。

これは、IDbDataAdapter.DeleteCommand が厳密に型指定されたものです。

InsertCommand プロパティ

DataSet に挿入されたローに該当するローをデータベースに挿入するために、DbDataAdapter.Update が呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Property InsertCommand As ULCommand

// C#
    public ULCommand InsertCommand {get;set;}
```

プロパティ値

DataSet に挿入されたローに該当するローをデータベースに挿入するために実行される ULCommand クラス オブジェクト

説明

InsertCommand が既存の ULCommand クラス オブジェクトに割り当てられる場合、ULCommand クラス オブジェクトのクローンは作成されません。InsertCommand は、既存の ULCommand クラス への参照を保持します。

これは、IDbDataAdapter.InsertCommand が厳密に型指定されたものです。

MissingMappingAction プロパティ

受信データと一致するテーブルまたはカラムがない場合に実行するアクションを決定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property MissingMappingAction As
   MissingMappingAction _
     Implements IDataAdapter.MissingMappingAction

// C#
    public MissingMappingAction MissingMappingAction {get;set;}
```

プロパティ値

MissingMappingAction 値の 1 つ。デフォルトは MissingMappingAction.Passthrough です。

実装

IDataAdapter.MissingMappingAction

MissingSchemaAction プロパティ

既存の DataSet スキーマが受信データと一致しない場合に実行するアクションを決定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property MissingSchemaAction As
   MissingSchemaAction _
     Implements IDataAdapter.MissingSchemaAction

// C#
    public MissingSchemaAction MissingSchemaAction {get;set;}
```

プロパティ値

MissingSchemaAction 値の 1 つ。デフォルトは MissingSchemaAction.Add です。

実装

IDataAdapter.MissingSchemaAction

SelectCommand プロパティ

結果セットをデータベースから取得して DataSet にコピーする Fill メソッド または FillSchema メソッド の実行中に使用される ULCommand クラス を指定します。

プロトタイプ

```
' Visual Basic
    Public Property SelectCommand As ULCommand

// C#
    public ULCommand SelectCommand {get;set;}
```

プロパティ値

DataSet を設定するために実行される ULCommand クラス オブジェクト

説明

SelectCommand が既存の ULCommand クラス オブジェクトに割り当てられる場合、ULCommand クラス オブジェクトのクローンは作成されません。SelectCommand は、既存の ULCommand クラス への参照を保持します。

SelectCommand がローを返さない場合、DataSet にテーブルが追加されず、例外も発生しません。SELECT 文は、ULDataAdapter コンストラクタ 、ULDataAdapter コンストラクタ、ULDataAdapter コンストラクタにも指定できます。

これは、IDbDataAdapter.SelectCommand が厳密に型指定されたものです。

TableMappings プロパティ

ソース・テーブルと DataTable とのマスタ・マッピングを提供するコレクションを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableMappings As DataTableMappingCollection

// C#
    public DataTableMappingCollection TableMappings {get;}
```

プロパティ値

ソース・テーブルと DataTable とのマスタ・マッピングを提供する DataTableMapping オブジェクトのコレクション。デフォルト値は空のコレクションです。

説明

変更を調整する場合、ULDataAdapter は DataTableMappingCollection コレクションを使用して、データ・ソースによって使用されるカラム名を、DataSet によって使用されるカラム名に関連付けます。

これは、IDataAdapter.TableMappings が厳密に型指定されたものです。

UpdateCommand プロパティ

DataSet で更新されたローに該当するデータベース内のローを更新するために、Update メソッド が呼び出されたときにデータベースに対して実行される ULCommand クラス オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Property UpdateCommand As ULCommand
```

```
// C#
    public ULCommand UpdateCommand {get;set;}
```

プロパティ値

DataSet で更新されたローに該当するデータベース内のローを更新するために実行される ULCommand クラス オブジェクト

説明

UpdateCommand が既存の ULCommand クラス オブジェクトに割り当てられる場合、ULCommand クラス オブジェクトのクローンは作成されません。UpdateCommand は、既存の ULCommand クラス への参照を保持します。

このコマンドを実行するとローが返される場合、ULCommand クラス オブジェクトの UpdatedRowSource プロパティの設定方法によっては、これらのローが DataSet にマージされることがあります。

これは、IDbDataAdapter.UpdateCommand が厳密に型指定されたものです。

Dispose メソッド

ULDataAdapter によって使用されるアンマネージ・リソースを解放するほか、必要に応じてマネージ・リソースを解放します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Dispose() _
      Implements IDisposable.Dispose

// C#
    public void Dispose();
```

実装

IDisposable.Dispose

Fill メソッド

指定した DataSet の "Table" という DataTable でローの追加または再表示を行います。必要に応じて、"Table" という DataTable が作成され、DataSet に追加されます。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function Fill( _
      ByVal dataSet As DataSet _
    ) As Integer _
      Implements IDataAdapter.Fill

// C#
    public int Fill(
      DataSet dataSet
    );
```

パラメータ

- **dataSet** レコード、および必要に応じてスキーマを使用して設定する DataSet

戻り値

DataSet で正常に追加または再表示されたローの数

説明

結果セットのサイズが大きい場合、パフォーマンスに重大な影響を及ぼす可能性があります。別の方法として、読み込み専用の結果セットで十分であれば、ULDataReader クラス (場合によっては SQL 文 (ExecuteNonQuery) とともに) を使用して修正を行います。さらに別の方法として、そのデータベースへの読み込みと書き込みアクセスが可能な ULTable クラス を使用できます。

SelectCommand プロパティ がローを返さない場合、DataSet にテーブルが追加されず、例外も発生しません。

詳細については、IDataAdapter.Fill を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上に MissingMappingAction プロパティ が MissingMappingAction.Error に設定されているか、DataTable が DataSet から見つからない上に MissingSchemaAction プロパティ が MissingSchemaAction.Error に設定されています。

実装

IDataAdapter.Fill

Fill メソッド

指定した DataSet の指定した DataTable でローの追加または再表示を行います。必要に応じて、指定した DataTable が作成され、DataSet に追加されます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Fill( _
      ByVal dataSet As DataSet, _
      ByVal srcTable As String _
    ) As Integer

// C#
    public int Fill(
      DataSet dataSet,
      string  srcTable
    );
```

パラメータ

- **dataSet** レコード、および必要に応じてスキーマを使用して設定する DataSet
- **srcTable** テーブルのマッピングに使用するソース・テーブルの名前

戻り値

DataSet で正常に追加または再表示されたローの数

説明

結果セットのサイズが大きい場合、パフォーマンスに重大な影響を及ぼす可能性があります。別の方法として、読み込み専用の結果セットで十分であれば、ULDataReader クラス (場合によっては SQL 文 (ExecuteNonQuery) とともに) を使用して修正を行います。さらに別の方法として、そのデータベースへの読み込みと書き込みアクセスが可能な ULTable クラス を使用できます。

SelectCommand プロパティ がローを返さない場合、DataSet にテーブルが追加されず、例外も発生しません。

詳細については、IDataAdapter.Fill を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータまたは *srcTable* パラメータが無効です。

- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上に MissingMappingAction プロパティ が MissingMappingAction.Error に設定されているか、DataTable が DataSet から見つからない上に MissingSchemaAction プロパティ が MissingSchemaAction.Error に設定されています。

Fill メソッド

指定した DataSet の指定した DataTable で、指定した範囲のローの追加または再表示を行います。必要に応じて、指定した DataTable が作成され、DataSet に追加されます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Fill( _
      ByVal dataSet As DataSet, _
      ByVal startRecord As Integer, _
      ByVal maxRecords As Integer, _
      ByVal srcTable As String _
    ) As Integer

// C#
    public int Fill(
      DataSet dataSet,
      int startRecord,
      int maxRecords,
      string  srcTable
    );
```

パラメータ

- **dataSet** レコード、および必要に応じてスキーマを使用して設定する DataSet
- **startRecord** 0 から始まるレコード番号
- **maxRecords** DataSet に読み込むレコードの最大数。値が 0 であると、先頭のレコードの後に見つかったすべてのレコードが取得されます。
- **srcTable** テーブルのマッピングに使用するソース・テーブルの名前

戻り値

DataSet で正常に追加または再表示されたローの数

説明

startRecord 引数を使用して、DataSet にコピーされるレコードの数を制限した場合でも、ULDataAdapter クエリ内のすべてのレコードがデータベースからクライアントにフェッチされます。結果セットのサイズが大きい場合、パフォーマンスに重大な影響を及ぼす可能性があります。別の方法として、読み込み専用の結果セットで十分であれば、ULDataReader クラス (場合によっては SQL 文 (ExecuteNonQuery) とともに) を使用して修正を行います。さらに別の方法として、そのデータベースへの読み込みと書き込みアクセスが可能な ULTable クラス を使用できます。

SelectCommand プロパティ がローを返さない場合、DataSet にテーブルが追加されず、例外も発生しません。

詳細については、IDataAdapter.Fill を参照してください。

例外

- ArgumentException*startRecord* パラメータまたは *maxRecords* パラメータがゼロ未満でした。
- System.ArgumentNullException*dataSet* パラメータまたは *srcTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上に MissingMappingAction プロパティ が MissingMappingAction.Error に設定されているか、DataTable が DataSet から見つからない上に MissingSchemaAction プロパティ が MissingSchemaAction.Error に設定されています。

Fill メソッド

指定した DataTable でローの追加または再表示を行います。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Fill( _
      ByVal dataTable As DataTable _
    ) As Integer
```

```
// C#
    public int Fill(
      DataTable dataTable
    );
```

パラメータ

- **dataTable** レコード、および必要に応じてスキーマを使用して設定する DataTable

戻り値

DataTable で正常に追加または再表示されたローの数

説明

結果セットのサイズが大きい場合、パフォーマンスに重大な影響を及ぼす可能性があります。別の方法として、読み込み専用の結果セットで十分であれば、ULDataReader クラス (場合によっては SQL 文 (ExecuteNonQuery) とともに) を使用して修正を行います。さらに別の方法として、そのデータベースへの読み込みと書き込みアクセスが可能な ULTable クラス を使用できます。

SelectCommand プロパティ がローを返さない場合、テーブルが追加されず、例外も発生しません。

詳細については、IDataAdapter.Fill を参照してください。

例外

- System.ArgumentNullException*dataTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上 MissingMappingAction プロパティ が MissingMappingAction.Error に設定されています。

FillSchema メソッド

"Table" という DataTable を DataSet に追加し、スキーマがデータ・ソースのスキーマと一致するように設定します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function FillSchema( _
      ByVal dataSet As DataSet, _
      ByVal schemaType As SchemaType _
    ) As DataTable() _
      Implements IDataAdapter.FillSchema
```

```
// C#
    public DataTable[] FillSchema(
      DataSet dataSet,
      SchemaType schemaType
    );
```

パラメータ

- **dataSet** スキーマを使用して設定する DataSet
- **schemaType** スキーマを挿入する方法を指定する SchemaType 値の 1 つ

戻り値

DataSet に追加された DataTable オブジェクトのコレクションへの参照

説明

詳細については、IDataAdapter.FillSchema を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上 MissingMappingAction プロパティ が MissingMappingAction.Error に設定されています。

実装

IDataAdapter.FillSchema

FillSchema メソッド

"Table" という DataTable を DataSet に追加し、スキーマがデータ・ソースのスキーマと一致するように設定します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FillSchema( _
      ByVal dataSet As DataSet, _
      ByVal schemaType As SchemaType, _
      ByVal srcTable As String _
    ) As DataTable()

// C#
    public DataTable[] FillSchema(
      DataSet dataSet,
```

```
    SchemaType schemaType,
    string  srcTable
);
```

パラメータ

- **dataSet** スキーマを使用して設定する DataSet
- **schemaType** スキーマを挿入する方法を指定する SchemaType 値の 1 つ
- **srcTable** テーブルのマッピングに使用するソース・テーブルの名前

戻り値

DataSet に追加された DataTable オブジェクトのコレクションへの参照

説明

詳細については、DbDataAdapter.FillSchema を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータまたは *srcTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上 MissingMappingAction プロパティ が MissingMappingAction.Error に設定されています。

FillSchema メソッド

DataTable のスキーマがデータ・ソースのスキーマと一致するように設定します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FillSchema( _
      ByVal dataTable As DataTable, _
      ByVal schemaType As SchemaType _
    ) As DataTable

// C#
    public DataTable FillSchema(
      DataTable dataTable,
      SchemaType schemaType
    );
```

パラメータ

- **dataTable** スキーマを使用して設定する DataTable
- **schemaType** スキーマを挿入する方法を指定する SchemaType 値の 1 つ

戻り値

スキーマが含まれている DataTable オブジェクトへの参照

説明

詳細については、DbDataAdapter.FillSchema を参照してください。

例外

- System.ArgumentNullException*dataTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効です。

GetFillParameters メソッド

SELECT 文の実行時にユーザによって設定されるパラメータを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetFillParameters() As ULParameter()

// C#
    public ULParameter[] GetFillParameters();
```

戻り値

ユーザによって設定されたパラメータが含まれる ULParameter クラス オブジェクトの配列

Update メソッド

指定した DataSet の "Table" という DataTable への変更内容を使用して、データベース内のローを更新します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function Update( _
      ByVal dataSet As DataSet _
    ) As Integer _
      Implements IDataAdapter.Update
```

```
// C#
    public int Update(
      DataSet dataSet
    );
```

パラメータ

- **dataSet** 更新に使用するレコードの DataSet

戻り値

DataSet から正常に更新されたローの数

説明

詳細については、IDataAdapter.Update を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上に MissingMappingAction プロパティ が MissingMappingAction.Error に設定されているか、DataTable が DataSet から見つかりません。

実装

IDataAdapter.Update

Update メソッド

指定した DataSet の指定した DataTable への変更内容を使用して、データベース内のローを更新します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Update( _
      ByVal dataSet As DataSet, _
      ByVal srcTable As String _
    ) As Integer

// C#
    public int Update(
      DataSet dataSet,
      string  srcTable
    );
```

パラメータ

- **dataSet** 更新に使用するレコードの DataSet
- **srcTable** テーブルのマッピングに使用するソース・テーブルの名前

戻り値

DataSet から正常に更新されたローの数

説明

詳細については、DbDataAdapter.Update を参照してください。

例外

- System.ArgumentNullException*dataSet* パラメータまたは *srcTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上に MissingMappingAction プロパティ が MissingMappingAction.Error に設定されているか、DataTable が DataSet から見つかりません。

Update メソッド

指定した DataTable への変更内容を使用して、データベース内のローを更新します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Update( _
      ByVal dataTable As DataTable _
    ) As Integer

// C#
    public int Update(
      DataTable dataTable
    );
```

パラメータ

- **dataTable** 更新の元になる DataTable

戻り値

DataTable から正常に更新されたローの数

説明

詳細については、DbDataAdapter.Update を参照してください。

例外

- System.ArgumentNullException*dataTable* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上 MissingMappingAction プロパティ が MissingMappingAction.Error に設定されています。

Update メソッド

指定した DataRow 配列への変更内容を使用して、データベース内のローを更新します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Update( _
      ByVal dataRows As DataRow() _
    ) As Integer

// C#
    public int Update(
      DataRow[] dataRows
    );
```

パラメータ

- **dataRows** 更新の元になる DataRow 配列

戻り値

DataRow 配列から正常に更新されたローの数

説明

詳細については、DbDataAdapter.Update を参照してください。

例外

- System.ArgumentNullException*dataRows* パラメータが無効です。
- InvalidOperationExceptionSelectCommand プロパティが無効であるか、テーブルのマッピングが見つからない上 MissingMappingAction プロパティ が MissingMappingAction.Error に設定されています。

FillError イベント

設定オペレーションの実行時にエラーが検出されたときに発生します。

プロトタイプ

```
' Visual Basic
    Public Event FillError As FillErrorEventHandler

// C#
    public event FillErrorEventHandler FillError;
```

説明

FillError イベントを使用して、エラーが発生した後に設定オペレーションを継続するかどうかを決定できます。

FillError イベントが発生する例は、次のとおりです。

- DataSet に追加されるデータを、精度を落とさないで共通言語ランタイム・タイプに変換できない。
- DataSet の DataColumn に適用する必要がある制約に違反するデータが、追加されるローに含まれている。

ローの設定エラー・イベントを処理するには、FillErrorEventHandler デリゲートを作成し、このイベントにアタッチする必要があります。

RowUpdated イベント

データ・ソースに対してコマンドが実行された後の更新時に発生します。更新が試みられると、イベントが発生します。

プロトタイプ

```
' Visual Basic
    Public Event RowUpdated As ULRowUpdatedEventHandler

// C#
    public event ULRowUpdatedEventHandler RowUpdated;
```

説明

ロー更新イベントを処理するには、ULRowUpdatedEventHandler 委譲デリゲートを作成し、このイベントにアタッチする必要があります。

RowUpdating イベント

データ・ソースに対してコマンドが実行される前の更新時に発生します。更新が試みられると、イベントが発生します。

プロトタイプ

```
' Visual Basic
    Public Event RowUpdating As ULRowUpdatingEventHandler

// C#
    public event ULRowUpdatingEventHandler RowUpdating;
```

説明

ロー更新イベントを処理するには、ULRowUpdatingEventHandler 委譲デリゲートを作成し、このイベントにアタッチする必要があります。

ULDatabaseManager クラス

UL 拡張 : 同期リスナと Ultra Light.NET のランタイム・タイプを管理します。ULDatabaseManager クラスを使用すると、Ultra Light.NET データベースを削除できます。

プロトタイプ

' Visual Basic

NotInheritable Public Class **ULDatabaseManager**

// C#

public sealed class **ULDatabaseManager**

説明

このクラスはシングルトン・クラスであり、その唯一のインスタンスには、静的な (Visual Basic の Shared) DatabaseManager プロパティ のみを通じてアクセスできます。

Ultra Light.NET の Ultra Light エンジンのランタイムを使用するには、RuntimeType プロパティ を適切な値に設定してから、他の Ultra Light.NET API を使用します。

ULDatabaseManager メンバ

パブリック静的プロパティ (共有)

メンバ	説明
RuntimeType プロパティ	Ultra Light.NET ランタイム・タイプを指定します。ランタイム・タイプを選択してから、その他の Ultra Light.NET API を使用してください。

パブリック・インスタンス・メソッド

メンバ	説明
DropDatabase メソッド	指定したデータベースを削除します。 接続が開かれているデータベースを削除することはできません。

メンバ	説明
SetActiveSyncListener メソッド	ActiveSync 用 Mobile Link プロバイダからの ActiveSync の呼び出しを処理するリスナ・オブジェクトを指定します。
SetServerSyncListener メソッド	指定したサーバ同期メッセージの処理に使用するリスナ・オブジェクトを指定します。

RuntimeType プロパティ

Ultra Light.NET ランタイム・タイプを指定します。ランタイム・タイプを選択してから、その他の Ultra Light.NET API を使用してください。

プロトタイプ

```
' Visual Basic
    Public Shared Property RuntimeType As ULRuntimeType

// C#
    public const ULRuntimeType RuntimeType {get;set;}
```

プロパティ値

アンマネージ Ultra Light.NET ランタイムのタイプを示す ULRuntimeType 列挙 値

例

次の例では、Ultra Light エンジンのランタイムが選択され、接続が作成されます。

```
' Visual Basic ULDatabaseManager.RuntimeType =
ULRuntimeType.UL_ENGINE_CLIENT
 Dim conn As ULConnection = new ULConnection
 ' The RuntimeType is now locked

 // C# ULDatabaseManager.RuntimeType =
ULRuntimeType.UL_ENGINE_CLIENT;
 ULConnection conn = new ULConnection();
 // The RuntimeType is now locked
```

DropDatabase メソッド

指定したデータベースを削除します。

接続が開かれているデータベースを削除することはできません。

プロトタイプ

```
' Visual Basic
    Public Sub DropDatabase( _
      ByVal parms As ULConnectionParms _
    )

// C#
    public void DropDatabase(
      ULConnectionParms parms
    );
```

パラメータ

- **parms** データベースを識別するためのパラメータ。詳細については、ULConnectionParms クラス を参照してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseManager クラス
- ULDatabaseManager メンバ
- OpenWithCreate メソッド

SetActiveSyncListener メソッド

ActiveSync 用 Mobile Link プロバイダからの ActiveSync の呼び出しを処理するリスナ・オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetActiveSyncListener( _
      ByVal appClassName As String, _
      ByVal listener As ULActiveSyncListener _
    )

// C#
    public void SetActiveSyncListener(
      string  appClassName,
      ULActiveSyncListener listener
    );
```

パラメータ

- **appClassName** アプリケーションのユニークなクラス名。これは、アプリケーションが ActiveSync で使用されるように登録されているときに使用されるクラス名です。
- **listener** ULActiveSyncListener インタフェース オブジェクト。前のリスナを削除するには、null (Visual Basic の Nothing) を使用します。

説明

パラメータ *appClassName* は、アプリケーションの識別に使用されるユニークな識別子です。アプリケーションは、一度に 1 つの *appClassName* のみを使用できます。リスナが特定の *appClassName* に登録されているときは、別の *appClassName* で SetServerSyncListener メソッド または SetActiveSyncListener メソッド を呼び出すと失敗します。

ActiveSync リスナを削除するには、*listener* パラメータとして null 参照 (Visual Basic の Nothing) を使用して SetActiveSyncListener メソッド を呼び出します。

すべてのリスナを削除するには、すべてのパラメータに null 参照 (Visual Basic の Nothing) を使用して SetServerSyncListener メソッド を呼び出します。

アプリケーションは、すべてのリスナを削除してから終了する必要があります。

例外

- ULException クラス - SQL エラーが発生しました。

例

SetActiveSyncListener メソッド の例については、ActiveSyncInvoked メソッド のマニュアルを参照してください。

参照

- ULDatabaseManager クラス
- ULDatabaseManager メンバ
- ColumnCount プロパティ

SetServerSyncListener メソッド

指定したサーバ同期メッセージの処理に使用するリスナ・オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetServerSyncListener( _
      ByVal messageName As String, _
      ByVal appClassName As String, _
      ByVal listener As ULServerSyncListener _
    )

// C#
    public void SetServerSyncListener(
      string  messageName,
      string  appClassName,
      ULServerSyncListener listener
    );
```

パラメータ

- **messageName** メッセージの名前
- **appClassName** アプリケーションのユニークなクラス名。これは、アプリケーションの識別に使用されるユニークな識別子です。
- **listener** ULServerSyncListener インタフェース オブジェクト。前のリスナを削除するには、null (Visual Basic の Nothing) を使用します。

説明

パラメータ *appClassName* は、アプリケーションの識別に使用されるユニークな識別子です。アプリケーションは、一度に 1 つの *appClassName* のみを使用できます。リスナが特定の *appClassName* に登録されているときは、別の *appClassName* で SetServerSyncListener メソッド または SetActiveSyncListener メソッド を呼び出すと失敗します。

特定のメッセージのリスナを削除するには、*listener* パラメータとして null 参照 (Visual Basic の Nothing) を使用して SetServerSyncListener メソッド を呼び出します。

すべてのリスナを削除するには、すべてのパラメータに null 参照 (Visual Basic の Nothing) を使用して SetServerSyncListener メソッド を呼び出します。

アプリケーションは、すべてのリスナを削除してから終了する必要があります。

例外

- ULException クラス - SQL エラーが発生しました。

例

SetServerSyncListener メソッド の例については、ServerSyncInvoked メソッド のマニュアルを参照してください。

ULDatabaseSchema クラス

UL 拡張：Ultra Light.NET データベースのスキーマを示します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **ULDatabaseSchema**

// C#
public sealed class **ULDatabaseSchema**

説明

このクラスを直接インスタンス化することはできません。
ULDatabaseSchema クラス オブジェクトは、Schema プロパティとして接続にアタッチされ、接続が開いている間のみ有効です。

ULDatabaseSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
CollationName プロパティ	データベースの照合順の名前です。
IsCaseSensitive プロパティ	データベースで大文字と小文字が区別されるかどうかをチェックします。
IsOpen プロパティ	データベース・スキーマが開いているかどうかを示します。
PublicationCount プロパティ	データベース内のパブリケーションの数です。
Signature プロパティ	このデータベースのシグニチャです。
TableCount プロパティ	データベース内のテーブルの数です。

パブリック・インスタンス・メソッド

メンバ	説明
ApplyFile メソッド	データベース・スキーマ・ファイルをデータベースに適用するとともに、リスナに進行状況をモニタさせます。
ApplyFile メソッド	データベース・スキーマ・ファイルをデータベースに適用します。
GetDatabaseProperty メソッド	指定したデータベースのプロパティの値を返します。
GetPublicationName メソッド	指定したパブリケーション ID で識別されたパブリケーションの名前を返します。パブリケーション ID は、パブリケーション・マスクではありません。
GetPublicationSchema メソッド	指定したパブリケーションに対応するパブリケーション・スキーマを返します。
GetTableCountInPublications メソッド	指定したパブリケーション・マスクに含まれるテーブルの数を返します。
GetTableName メソッド	指定したテーブル ID で識別されたテーブルの名前を返します。

CollationName プロパティ

データベースの照合順の名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property CollationName As String

// C#
    public string  CollationName {get;}
```

プロパティ値

データベースの照合順を示す文字列

説明

データベースの照合順は、テーブルのインデックスおよび結果セットのソート方法に影響します。

例外

- ULException クラス - SQL エラーが発生しました。

IsCaseSensitive プロパティ

データベースで大文字と小文字が区別されるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsCaseSensitive As Boolean

// C#
    public bool IsCaseSensitive {get;}
```

プロパティ値

データベースで大文字と小文字が区別される場合は true、区別されない場合は false

説明

データベースで大文字と小文字が区別されるかどうかは、テーブルのインデックスおよび結果セットのソート方法に影響します。また、大文字と小文字の区別は、UserID プロパティ と Password プロパティ の検証方法に影響します。

例外

- ULException クラス - SQL エラーが発生しました。

IsOpen プロパティ

データベース・スキーマが開いているかどうかを示します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

データベースのスキーマが現在開いている場合は true、現在閉じている場合は false

説明

ULDatabaseSchema オブジェクトが開いているのは、それがアタッチされている接続が開いている場合だけです。

PublicationCount プロパティ

データベース内のパブリケーションの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PublicationCount As Integer

// C#
    public int PublicationCount {get;}
```

プロパティ値

データベース内のパブリケーション数、または接続が開いていない場合は 0

説明

パブリケーション ID の範囲は、1 ～ PublicationCount です。パブリケーション ID は、パブリケーション・マスクではありません。

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

参照

- ULDatabaseSchema クラス
- ULDatabaseSchema メンバ
- GetPublicationName メソッド

Signature プロパティ

このデータベースのシグニチャです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Signature As String

// C#
    public string  Signature {get;}
```

プロパティ値

現在開かれているデータベースのシグニチャを示す文字列

例外

- ULException クラス - SQL エラーが発生しました。

TableCount プロパティ

データベース内のテーブルの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableCount As Integer

// C#
    public int TableCount {get;}
```

プロパティ値

データベース内のテーブル数、または接続が開いていない場合は 0

説明

テーブル ID の範囲は、1 ～ TableCount です。

注意：テーブルの ID とカウントは、スキーマのアップグレード中に変更されることがあります。テーブルを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

ApplyFile メソッド

データベース・スキーマ・ファイルをデータベースに適用するとともに、リスナに進行状況をモニタさせます。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub ApplyFile( _
      ByVal parms As ULConnectionParms, _
      ByVal listener As ULSchemaUpgradeListener _
    )

// C#
    public void ApplyFile(
      ULConnectionParms parms,
      ULSchemaUpgradeListener listener
    );
```

パラメータ

- **parms** データベースに適用されるスキーマを指定するためのパラメータ。詳細については、ULConnectionParms クラス を参照してください。

- **listener** スキーマのアップグレードのプログレス・イベントを受信する ULSchemaUpgradeListener インタフェース オブジェクト

説明

ULResultSetSchema クラス、ULTableSchema クラス、ULIndexSchema クラス、ULPublicationSchema クラス のすべてのインスタンスは無効になるため、置き換える必要があります。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseSchema クラス
- ULDatabaseSchema メンバ
- ApplyFile メソッド
- ULConnectionParms クラス

ApplyFile メソッド

データベース・スキーマ・ファイルをデータベースに適用します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub ApplyFile( _
      ByVal parms As ULConnectionParms _
    )

// C#
    public void ApplyFile(
      ULConnectionParms parms
    );
```

パラメータ

- **parms** データベースに適用されるスキーマを指定するためのパラメータ。詳細については、ULConnectionParms クラス を参照してください。

説明

ULResultSetSchema クラス、ULTableSchema クラス、ULIndexSchema クラス、ULPublicationSchema クラス のすべてのインスタンスは無効になるため、置き換える必要があります。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseSchema クラス

- ULDatabaseSchema メンバ
- ApplyFile メソッド
- ULConnectionParms クラス

GetDatabaseProperty メソッド

指定したデータベースのプロパティの値を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetDatabaseProperty( _
      ByVal name As String _
    ) As String

// C#
    public string  GetDatabaseProperty(
      string  name
    );
```

パラメータ

- **name** 問い合わせるデータベース・プロパティの名前

戻り値

文字列としてのプロパティの値

説明

識別されるプロパティは次のとおりです。

- "DATE_FORMAT" データベースによる文字列変換に使用される日付フォーマットです。

  このフォーマットは、DateTime によって使用されるフォーマットと同じであるとは限りません。

- "DATE_ORDER" データベースによる文字列変換に使用される日付順です。
- "NEAREST_CENTURY" データベースによる文字列変換に使用される最も近い世紀です。
- "PRECISION" データベースによる文字列変換に使用される浮動小数点の精度です。

- "SCALE" データベースによる文字列変換中に、計算結果が最大 PRECISION にトランケートされるときの小数点以下の最大桁数です。
- "TIME_FORMAT" データベースによる文字列変換に使用される時刻フォーマットです。

  このフォーマットは、TimeSpan によって使用されるフォーマットと同じであるとは限りません。
- "TIMESTAMP_FORMAT" データベースによる文字列変換に使用されるタイムスタンプのフォーマットです。

  このフォーマットは、DateTime によって使用されるフォーマットと同じであるとは限りません。
- "TIMESTAMP_INCREMENT" 2 つのユニークなタイムスタンプ間のナノ秒 (1,000,000 分の 1 秒) 単位の最小差です。

例外

- ULException クラス - SQL エラーが発生しました。

GetPublicationName メソッド

指定したパブリケーション ID で識別されたパブリケーションの名前を返します。パブリケーション ID は、パブリケーション・マスクではありません。

プロトタイプ

```
' Visual Basic
    Public Function GetPublicationName( _
      ByVal pubID As Integer _
    ) As String

// C#
    public string  GetPublicationName(
      int pubID
    );
```

パラメータ

- **pubID** パブリケーションの ID。値は、[1,PublicationCount プロパティ] の範囲内である必要があります。

戻り値

文字列としてのパブリケーション名

説明

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseSchema クラス
- ULDatabaseSchema メンバ
- PublicationCount プロパティ

GetPublicationSchema メソッド

指定したパブリケーションに対応するパブリケーション・スキーマを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetPublicationSchema( _
      ByVal name As String _
    ) As ULPublicationSchema

// C#
    public ULPublicationSchema GetPublicationSchema(
      string  name
    );
```

パラメータ

- **name** パブリケーションの名前

戻り値

指定したパブリケーションを示す ULPublicationSchema クラス オブジェクト

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseSchema クラス
- ULDatabaseSchema メンバ
- GetPublicationName メソッド
- ULPublicationSchema クラス

GetTableCountInPublications メソッド

指定したパブリケーション・マスクに含まれるテーブルの数を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTableCountInPublications( _
      ByVal mask As Integer _
    ) As Integer

// C#
    public int GetTableCountInPublications(
      int mask
    );
```

パラメータ

- **mask** チェックするパブリケーションのセット。パブリケーション・マスクの作成の詳細については、ULPublicationSchema クラス を参照してください。

戻り値

パブリケーションのセットに含まれるテーブル数

説明

カウントには、名前が _nosync で終わるテーブルは含まれません。

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

GetTableName メソッド

指定したテーブル ID で識別されたテーブルの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTableName( _
      ByVal tableID As Integer _
    ) As String
```

```
// C#
    public string  GetTableName(
      int tableID
    );
```

パラメータ

- **tableID** テーブルの ID。値は、[1,TableCount] の範囲内である必要があります。

戻り値

文字列としてのテーブル名

説明

テーブル ID は、スキーマのアップグレード中に変更されることがあります。テーブルを正しく識別するには、名前でアクセスするか、キャッシュされている ID をスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDatabaseSchema クラス
- ULDatabaseSchema メンバ
- TableCount プロパティ

ULDataReader クラス

Ultra Light データベースの読み込み専用の双方向カーソルを示します。カーソルとは、テーブルまたはクエリからの結果セットの一連のローです。

プロトタイプ

```
' Visual Basic
    Public Class ULDataReader
     Inherits MarshalByRefObject

// C#
    public class ULDataReader :
     MarshalByRefobject
```

説明

ULDataReader にはコンストラクタがありません。ULDataReader オブジェクトを取得するには、次のように ULCommand クラス を実行します。

```
' Visual Basic Dim cmd As ULCommand = new ULCommand( _
     "SELECT emp_id FROM employee", conn _
   )
 Dim reader As ULDataReader = cmd.ExecuteReader()

 // C# ULCommand cmd = new ULCommand(
     "SELECT emp_id FROM employee", conn
   );
 ULDataReader reader = cmd.ExecuteReader();
```

UL 拡張：ADO.NET 標準 では、結果セットで前方専用の動作だけが必要ですが、ULDataReader は双方向です。ULDataReader の Move メソッドによって、結果セットを移動するときに最大限の柔軟性が得られます。

ULDataReader は、読み込み専用の結果セットです。より柔軟なオブジェクトで結果を操作する必要がある場合は、ULDataAdapter クラス を使用します。ULDataReader は必要に応じてローを取得しますが、ULDataAdapter クラス の場合、結果セットのすべてのローを取得しないと、オブジェクトに対してアクションを実行できません。結果セットのサイズが大きい場合、この違いのために ULDataReader の方が応答時間が速くなります。

UL 拡張：ULDataReader のすべてのカラムは、GetString メソッド を使用して取得できます。

実装：IDataReader、IDataRecord、IDisposable

ULDataReader メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
Depth プロパティ	現在のローのネストの深さを返します。最も外側のテーブルの深さは 0 です。
FieldCount プロパティ	このカーソル内のカラム数を返します。
IsBOF プロパティ	**UL 拡張：**現在のローの位置が最初のローの前であるかどうかを確認します。
IsClosed プロパティ	カーソルが現在開いているかどうかを確認します。
IsEOF プロパティ	**UL 拡張：**現在のローの位置が最後のローの後であるかどうかを確認します。
Item プロパティ	指定したカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。
Item プロパティ	指定した名前のカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。
RecordsAffected プロパティ	SQL 文の実行によって変更、挿入、または削除されたローの数を返します。SELECT 文または CommandType.TableDirect テーブルの場合、この値は -1 です。
RowCount プロパティ	**UL 拡張：**カーソル内のロー数を返します。
Schema プロパティ	**UL 拡張：**このカーソルのスキーマを保持します。

パブリック・インスタンス・メソッド

メンバ	説明
Close メソッド	カーソルを閉じます。
Dispose メソッド	ULDataReader によって使用されるアンマネージ・リソースを解放するほか、必要に応じてマネージ・リソースを解放します。
GetBoolean メソッド	指定したカラムの値を Boolean として返します。
GetByte メソッド	指定したカラムの値を符号なし 8 ビット値 (Byte) として返します。
GetBytes メソッド	指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。
GetBytes メソッド	**UL 拡張：**指定したカラムの値を Byte 配列として返します。ULDbType 列挙 型、ULDbType 列挙 型、ULDbType 列挙 型のカラムの場合にのみ有効です。
GetChar メソッド	このメソッドは Ultra Light.NET ではサポートされていません。
GetChars メソッド	指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。
GetData メソッド	このメソッドは Ultra Light.NET ではサポートされていません。
GetDataTypeName メソッド	指定したカラムのプロバイダのデータ型の名前を返します。
GetDateTime メソッド	指定したカラムの値を、ミリ秒の精度の DateTime として返します。
GetDecimal メソッド	指定したカラムの値を Decimal として返します。
GetDouble メソッド	指定したカラムの値を Double として返します。
GetFieldType メソッド	指定したカラムに最適な Type を返します。

メンバ	説明
GetFloat メソッド	指定したカラムの値を Single として返します。
GetGuid メソッド	指定したカラムの値を UUID (Guid) として返します。
GetInt16 メソッド	指定したカラムの値を Int16 として返します。
GetInt32 メソッド	指定したカラムの値を Int32 として返します。
GetInt64 メソッド	指定したカラムの値を Int64 として返します。
GetName メソッド	指定したカラムの名前を返します。
GetOrdinal メソッド	指定したカラムのカラム ID を返します。
GetSchemaTable メソッド	ULDataReader のカラムのメタデータが記述された DataTable を返します。
GetString メソッド	指定したカラムの値を String として返します。
GetTimeSpan メソッド	指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。
GetUInt16 メソッド	指定したカラムの値を UInt16 として返します。
GetUInt32 メソッド	指定したカラムの値を UInt32 として返します。
GetUInt64 メソッド	指定したカラムの値を UInt64 として返します。
GetValue メソッド	指定したカラムの値をネイティブ・フォーマットで返します。
GetValues メソッド	現在のローのすべてのカラム値を返します。
IsDBNull メソッド	指定されたカラムの値が NULL かどうかをチェックします。
MoveAfterLast メソッド	**UL 拡張：**カーソルの最後のローの後に、カーソルを配置します。
MoveBeforeFirst メソッド	**UL 拡張：**カーソルの最初のローの前に、カーソルを配置します。
MoveFirst メソッド	**UL 拡張：**カーソルの最初のローに、カーソルを配置します。

メンバ	説明
MoveLast メソッド	**UL 拡張：**カーソルの最後のローに、カーソルを配置します。
MoveNext メソッド	**UL 拡張：**カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。
MovePrevious メソッド	**UL 拡張：**カーソルを前のローに配置するか、最初のローの前に配置します。
MoveRelative メソッド	**UL 拡張：**現在のローを基準としてカーソルを配置します。
NextResult メソッド	バッチ SQL 文の結果を読み込むときに ULDataReader を次の結果に進めます。
Read メソッド	カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	ULDataReader によって使用されるアンマネージ・リソースを解放します。

Depth プロパティ

現在のローのネストの深さを返します。最も外側のテーブルの深さは 0 です。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property Depth As Integer _
      Implements IDataReader.Depth

// C#
    public int Depth {get;}
```

プロパティ値

Ultra Light.NET のすべての結果セットの深さはゼロです。

例外

- ULException クラス - ULDataReader が開かれていません。

実装

IDataReader.Depth

FieldCount プロパティ

このカーソル内のカラム数を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property FieldCount As Integer _
      Implements IDataRecord.FieldCount

// C#
    public int FieldCount {get;}
```

戻り値

整数としてのカーソル内のカラム数。カーソルが閉じている場合は 0 を返します。

説明

このメソッドは、ColumnCount プロパティと同じです。

実装

IDataRecord.FieldCount

IsBOF プロパティ

UL 拡張：現在のローの位置が最初のローの前であるかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsBOF As Boolean

// C#
    public bool IsBOF {get;}
```

プロパティ値

現在のローの位置が最初のローの前であれば true、そうでなければ false

例外

- ULException クラス - SQL エラーが発生しました。

IsClosed プロパティ

カーソルが現在開いているかどうかを確認します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property IsClosed As Boolean _
      Implements IDataReader.IsClosed

// C#
    public bool IsClosed {get;}
```

プロパティ値 カーソルが現在開いている場合は true、閉じている場合は false

実装 IDataReader.IsClosed

IsEOF プロパティ

UL 拡張：現在のローの位置が最後のローの後であるかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsEOF As Boolean

// C#
    public bool IsEOF {get;}
```

プロパティ値 現在のローの位置が最後のローの後であれば true、そうでなければ false

例外

- ULException クラス - SQL エラーが発生しました。

Item プロパティ

指定したカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property Item As Object _
      Implements IDataRecord.Item
```

// C#

public object **Item** {get;}

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

そのカラムに最適な .NET 型としてのカラム値、またはカラムが NULL の場合は DBNull

説明

このメソッドは、機能的には GetValue メソッドと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.Item

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetFieldType メソッド
- Item プロパティ

Item プロパティ

指定した名前のカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。

プロトタイプ

' Visual Basic

NotOverridable Public Readonly Property **Item** As Object _
Implements IDataRecord.Item

// C#

public object **Item** {get;}

パラメータ

- **name** カラムの名前

戻り値

そのカラムに最適な .NET 型としてのカラム値、またはカラムが NULL の場合は DBNull

説明

結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。エイリアスを使用していない場合は、非計算カラムの名前には、そのカラムの元となるテーブルの名前がプレフィクスとして付けられます。たとえば、MyTable.ID は、クエリ "SELECT ID FROM MyTable" の結果セットに含まれる唯一のカラムの名前です。

カラムに何回もアクセスするときは、名前ではなく、カラム ID でアクセスすると効率が良くなります。

このメソッドは次と同じです。

```
dataReader.GetValue( dataReader.GetOrdinal( name ) )
```

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.Item

参照

- ULDataReader クラス
- ULDataReader メンバ
- Item プロパティ
- GetOrdinal メソッド
- GetValue メソッド
- GetFieldType メソッド

RecordsAffected プロパティ

SQL 文の実行によって変更、挿入、または削除されたローの数を返します。SELECT 文または CommandType.TableDirect テーブルの場合、この値は -1 です。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property RecordsAffected As Integer _
      Implements IDataReader.RecordsAffected

// C#
    public int RecordsAffected {get;}
```

プロパティ値

SQL 文の実行によって変更、挿入、または削除されたローの数

実装　IDataReader.RecordsAffected

RowCount プロパティ

UL 拡張：カーソル内のロー数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property RowCount As Integer

// C#
    public int RowCount {get;}
```

プロパティ値　カーソル内のロー数

説明　RowCount は、古いローを削除して領域を節約するタイミングを決定するときに使用します。StopSynchronizationDelete メソッドを使用すると、統合データベースからは削除しないで Ultra Light データベースから古いローを削除できます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDataReader クラス
- ULDataReader メンバ
- StartSynchronizationDelete メソッド
- StopSynchronizationDelete メソッド

Schema プロパティ

UL 拡張：このカーソルのスキーマを保持します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Schema As ULCursorSchema

// C#
    public ULCursorSchema Schema {get;}
```

プロパティ値　結果セットの場合、結果セットのスキーマを表す ULResultSetSchema クラス オブジェクト。テーブルの場合、テーブルのスキーマを表す ULTableSchema クラス オブジェクト。

説明

このプロパティは、GetSchemaTable メソッドからの結果に示されない Ultra Light.NET の詳細情報を含め、カーソルの完全なスキーマを示します。

Close メソッド

カーソルを閉じます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Close() _
      Implements IDataReader.Close

// C#
    public void Close();
```

説明

すでに閉じられているカーソルを閉じるのはエラーではありません。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataReader.Close

Dispose メソッド

ULDataReader によって使用されるアンマネージ・リソースを解放するほか、必要に応じてマネージ・リソースを解放します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Dispose() _
      Implements IDisposable.Dispose

// C#
    public void Dispose();
```

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDisposable.Dispose

Finalize メソッド

ULDataReader によって使用されるアンマネージ・リソースを解放します。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

GetBoolean メソッド

指定したカラムの値を Boolean として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetBoolean( _
      ByVal columnID As Integer _
    ) As Boolean _
      Implements IDataRecord.GetBoolean

// C#
    public bool GetBoolean(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Boolean としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetBoolean

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetByte メソッド

指定したカラムの値を符号なし 8 ビット値 (Byte) として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetByte( _
      ByVal columnID As Integer _
    ) As Byte _
      Implements IDataRecord.GetByte

// C#
    public byte GetByte(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Byte としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetByte

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetBytes メソッド

指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function GetBytes( _
      ByVal columnID As Integer, _
      ByVal srcOffset As Long, _
```

```
    ByVal dst As Byte(), _
    ByVal dstOffset As Integer, _
    ByVal count As Integer _
  ) As Long _
    Implements IDataRecord.GetBytes

// C#
  public long GetBytes(
    int columnID,
    long srcOffset,
    byte[] dst,
    int dstOffset,
    int count
  );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **srcOffset** カラム値の開始位置。ゼロが値の冒頭です。
- **dst** コピー先の配列
- **dstOffset** コピー先の配列の開始位置
- **count** コピーされるバイト数

戻り値

実際にコピーされたバイト数

説明

null 参照 (Visual Basic の Nothing) である *dst* バッファを渡すと、GetBytes はフィールドの長さをバイト数で返します。

値の *srcOffset* から *srcOffset*+*count*-1 までの位置のバイトが、コピー先の配列の *dstOffset* から *dstOffset*+*count*-1 までの位置に、それぞれコピーされます。*count* のバイト数がコピーされる前に、値の末尾が検出された場合は、コピー先の配列の残りは変更されないままになります。

次のいずれかに該当する場合は、コード ULSQLCode 列挙 とともに ULException クラス がスローされ、コピー先は修正されません。

- *srcOffset* が負である。
- *dstOffset* が負である。

- *count* が負である。
- *dstOffset*+*count* が *dst*.Length よりも大きい。

その他のエラーの場合は、それに応じたエラー・コードとともに ULException クラス がスローされます。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetBytes

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド
- GetBytes メソッド

GetBytes メソッド

UL 拡張 : 指定したカラムの値を Byte 配列として返します。ULDbType 列挙 型、ULDbType 列挙 型、ULDbType 列挙 型のカラムの場合にのみ有効です。

プロトタイプ

```
' Visual Basic
    Overloads Public Function GetBytes( _
      ByVal columnID As Integer _
    ) As Byte()

// C#
    public byte[] GetBytes(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Byte 配列としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド
- GetBytes メソッド

GetChar メソッド

このメソッドは Ultra Light.NET ではサポートされていません。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetChar( _
      ByVal columnID As Integer _
    ) As Char _
      Implements IDataRecord.GetChar

// C#
    public char GetChar(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

このメソッドは Ultra Light.NET ではサポートされていません。

例外

- ULException クラス - このメソッドは Ultra Light.NET ではサポートされていません。

実装

IDataRecord.GetChar

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド
- GetString メソッド

GetChars メソッド

指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetChars( _
      ByVal columnID As Integer, _
      ByVal srcOffset As Long, _
      ByVal dst As Char(), _
      ByVal dstOffset As Integer, _
      ByVal count As Integer _
    ) As Long _
      Implements IDataRecord.GetChars

// C#
    public long GetChars(
      int columnID,
      long srcOffset,
      char[] dst,
      int dstOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **srcOffset** カラム値の開始位置。ゼロが値の冒頭です。
- **dst** コピー先の配列
- **dstOffset** コピー先の配列の開始位置
- **count** コピーされる文字数

戻り値

実際にコピーされた文字数

説明

null 参照 (Visual Basic の Nothing) である *dst* バッファを渡すと、GetChars はフィールドの長さを文字数として返します。

値の *srcOffset* から *srcOffset*+*count*-1 までの位置の文字が、コピー先の配列の *dstOffset* から *dstOffset*+*count*-1 までの位置に、それぞれコピーされます。*count* の文字数がコピーされる前に、値の末尾が検出された場合は、コピー先の配列の残りは変更されないままになります。

次のいずれかに該当する場合は、コード ULSQLCode 列挙 とともに ULException クラス がスローされ、コピー先は修正されません。

- *srcOffset* が負である。
- *dstOffset* が負である。
- *count* が負である。
- *dstOffset*+*count* が *dst*.Length よりも大きい。

その他のエラーの場合は、それに応じたエラー・コードとともに ULException クラス がスローされます。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetChars

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetData メソッド

このメソッドは Ultra Light.NET ではサポートされていません。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetData( _
      ByVal i As Integer _
    ) As IDataReader _
      Implements IDataRecord.GetData

// C#
    public IDataReader GetData(
      int i
    );
```

パラメータ

- **i** 整数値

戻り値

このメソッドは Ultra Light.NET ではサポートされていません。

例外

- ULException クラス - このメソッドは Ultra Light.NET ではサポートされていません。

実装

IDataRecord.GetData

GetDataTypeName メソッド

指定したカラムのプロバイダのデータ型の名前を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetDataTypeName( _
      ByVal columnID As Integer _
    ) As String _
      Implements IDataRecord.GetDataTypeName

// C#
    public string  GetDataTypeName(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

カラムの ULDbType 列挙 に対応する文字列

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetDataTypeName

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetColumnULDbType メソッド

GetDateTime メソッド

指定したカラムの値を、ミリ秒の精度の DateTime として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetDateTime( _
      ByVal columnID As Integer _
    ) As Date _
      Implements IDataRecord.GetDate

// C#
    public DateTime GetDateTime(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

DateTime としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetDateTime

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetDecimal メソッド

指定したカラムの値を Decimal として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetDecimal( _
      ByVal columnID As Integer _
    ) As Decimal _
      Implements IDataRecord.GetDecimal
```

```
// C#
    public decimal GetDecimal(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Decimal としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetDecimal

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetDouble メソッド

指定したカラムの値を Double として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetDouble( _
      ByVal columnID As Integer _
    ) As Double _
      Implements IDataRecord.GetDouble

// C#
    public double GetDouble(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Double としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetDouble

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetFieldType メソッド

指定したカラムに最適な Type を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetFieldType( _
      ByVal columnID As Integer _
    ) As Type _
      Implements IDataRecord.GetFieldType

// C#
    public Type GetFieldType(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

カラムの Type 値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetFieldType

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetDataTypeName メソッド
- GetColumnULDbType メソッド

GetFloat メソッド

指定したカラムの値を Single として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetFloat( _
      ByVal columnID As Integer _
    ) As Single _
      Implements IDataRecord.GetFloat

// C#
    public float  GetFloat(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Single としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetFloat

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetGuid メソッド

指定したカラムの値を UUID (Guid) として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetGuid( _
      ByVal columnID As Integer _
    ) As Guid _
      Implements IDataRecord.GetGuid
```

```
// C#
    public Guid GetGuid(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Guid としてのカラム値

説明

このメソッドが有効なのは、ULDbType 列挙 型のカラム、または長さが 16 の ULDbType 列挙 型のカラムの場合だけです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetGuid

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド
- GetColumnULDbType メソッド
- GetColumnSize メソッド

GetInt16 メソッド

指定したカラムの値を Int16 として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetInt16( _
      ByVal columnID As Integer _
    ) As Short _
      Implements IDataRecord.GetShort

// C#
    public short GetInt16(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Int16 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetInt16

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetInt32 メソッド

指定したカラムの値を Int32 として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetInt32( _
      ByVal columnID As Integer _
    ) As Integer _
      Implements IDataRecord.GetInteger

// C#
    public int GetInt32(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Int32 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetInt32

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetInt64 メソッド

指定したカラムの値を Int64 として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetInt64( _
      ByVal columnID As Integer _
    ) As Long _
      Implements IDataRecord.GetLong

// C#
    public long GetInt64(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

Int64 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetInt64

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetName メソッド

指定したカラムの名前を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetName( _
      ByVal columnID As Integer _
    ) As String _
      Implements IDataRecord.GetName

// C#
    public string  GetName(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

カラムの名前、またはカラムに名前がない場合は null 参照 (Visual Basic の Nothing)。SQL クエリでカラムのエイリアスが使用されている場合は、そのエイリアスが返されます。

説明

結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。エイリアスを使用していない場合は、非計算カラムの名前には、そのカラムの元となるテーブルの名前がプレフィクスとして付けられます。たとえば、MyTable.ID は、クエリ "SELECT ID FROM MyTable" の結果セットに含まれる唯一のカラムの名前です。

このメソッドは、GetColumnName メソッドと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetName

参照

- ULDataReader クラス
- ULDataReader メンバ
- FieldCount プロパティ
- GetSchemaTable メソッド

GetOrdinal メソッド

指定したカラムのカラム ID を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetOrdinal( _
      ByVal columnName As String _
    ) As Integer _
      Implements IDataRecord.GetOrdinal

// C#
    public int GetOrdinal(
      string  columnName
    );
```

パラメータ

- **columnName** カラムの名前

戻り値

指定したカラムのカラム ID

説明

カラム ID の範囲は、0 ～ FieldCount プロパティ -1 です。

結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。エイリアスを使用していない場合は、非計算カラムの名前には、そのカラムの元となるテーブルの名前がプレフィクスとして付けられます。たとえば、MyTable.ID は、クエリ "SELECT ID FROM MyTable" の結果セットに含まれる唯一のカラムの名前です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

このメソッドは、GetColumnID メソッドと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetOrdinal

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetSchemaTable メソッド

GetSchemaTable メソッド

ULDataReader のカラムのメタデータが記述された DataTable を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetSchemaTable() As DataTable _
      Implements IDataReader.GetSchemaTable

// C#
    public DataTable GetSchemaTable();
```

戻り値

ULDataReader の各カラムのスキーマが記述された DataTable

説明

GetSchemaTable メソッドは、各カラムに関するメタデータを次の順で返します。

DataTable カラム	説明
ColumnName	カラムの名前、またはカラムに名前がない場合は null 参照 (Visual Basic の Nothing)。SQL クエリでカラムのエイリアスが使用されている場合は、そのエイリアスが返されます。結果セットでは、すべてのカラムに名前があるとは限らないほか、すべてのカラム名がユニークであるとは限りません。
ColumnOrdinal	カラムの ID。値の範囲は、[0,FieldCount プロパティ -1] です。
ColumnSize	サイズ指定されたカラムの場合、カラムの値の最大長。その他のカラムの場合、これは、そのデータ型のバイト単位のサイズです。
NumericPrecision	数値カラム (ProviderType ULDbType 列挙 または ULDbType 列挙) の精度、またはカラムが数値型ではない場合は DBNull。
NumericScale	数値カラム (ProviderType ULDbType 列挙 または ULDbType 列挙) の位取り、またはカラムが数値型ではない場合は DBNull。

DataTable カラム	説明
IsUnique	カラムが取得元のテーブル (BaseTableName) でユニークな非計算カラムである場合、true です。
IsKey	カラムが、結果セットのユニーク・キーからともに取得された結果セットの一連のカラムのいずれかである場合、true です。IsKey が true に設定されたカラムのセットは、結果セット内のローをユニークに識別する最小限のセットである必要はありません。
BaseCatalogName	カラムが含まれているデータベース内のカタログの名前。Ultra Light.NET の場合、この値は常に DBNull です。
BaseColumnN ame	データベースのテーブル BaseTableName にあるカラムの元の名前。カラムが計算される場合、およびこの情報を特定できない場合は DBNull です。
BaseSchemaName	カラムが含まれているデータベース内のスキーマの名前。Ultra Light.NET の場合、この値は常に DBNull です。
BaseTableName	カラムが含まれているデータベースのテーブルの名前。カラムが計算される場合、およびこの情報を特定できない場合は DBNull です。
DataType	この型のカラムに最適な .NET データ型。
AllowDBNull	カラムが null 入力可能である場合は true、null 入力が不可能である場合またはこの情報を特定できない場合は false。
ProviderType	カラムの ULDbType 列挙。
IsIdentity	カラムが identity カラムである場合は true、identity カラムでない場合は false。Ultra Light.NET の場合、この値は常に false です。
IsAutoIncrement	カラムがオートインクリメント・カラムまたはグローバル・オートインクリメント・カラムである場合は true、それ以外のカラムである場合 (または、この情報を特定できない場合) は false。

DataTable カラム	説明
IsRowVersion	書き込みできない永続的なロー識別子がカラムに含まれており、ローを識別する以外は意味を持たない値がカラムにある場合は true。Ultra Light.NET の場合、この値は常に false です。
IsLong	カラムが ULDbType 列挙 カラムまたは ULDbType 列挙 カラムである場合は true、それ以外のカラムの場合は false。
IsReadOnly	カラムが読み込み専用である場合は true、カラムが修正可能であるか、カラムのアクセス権を特定できない場合は false。

実装

IDataReader.GetSchemaTable

参照

- ULDataReader クラス
- ULDataReader メンバ
- Schema プロパティ

GetString メソッド

指定したカラムの値を String として返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetString( _
      ByVal columnID As Integer _
    ) As String _
      Implements IDataRecord.GetString

// C#
    public string  GetString(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値　String としてのカラム値

例外
- ULException クラス - SQL エラーが発生しました。

実装　IDataRecord.GetString

参照
- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetTimeSpan メソッド

指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTimeSpan( _
      ByVal columnID As Integer _
    ) As TimeSpan

// C#
    public TimeSpan GetTimeSpan(
      int columnID
    );
```

パラメータ
- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値　TimeSpan としてのカラム値

例外
- ULException クラス - SQL エラーが発生しました。

参照
- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetUInt16 メソッド

指定したカラムの値を UInt16 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUInt16( _
      ByVal columnID As Integer _
    ) As UInt16

// C#
    public ushort GetUInt16(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

UInt16 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetUInt32 メソッド

指定したカラムの値を UInt32 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUInt32( _
      ByVal columnID As Integer _
    ) As UInt32

// C#
    public uint GetUInt32(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

UInt32 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetUInt64 メソッド

指定したカラムの値を UInt64 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUInt64( _
      ByVal columnID As Integer _
    ) As UInt64

// C#
    public ulong GetUInt64(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

UInt64 としてのカラム値

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetValue メソッド

指定したカラムの値をネイティブ・フォーマットで返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetValue( _
      ByVal columnID As Integer _
    ) As Object _
      Implements IDataRecord.GetValue

// C#
    public object GetValue(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

そのカラムに最適な .NET 型としてのカラム値、またはカラムが NULL の場合は DBNull

説明

このメソッドは、機能的には Item プロパティと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.GetValue

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

GetValues メソッド

現在のローのすべてのカラム値を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetValues( _
      ByVal values As Object() _
    ) As Integer _
      Implements IDataRecord.GetValues

// C#
    public int GetValues(
      object[] values
    );
```

パラメータ

- **values** ロー全体を保持する Object 配列

戻り値

取り出されるカラム値の数。配列の長さがカラムの数 (FieldCount プロパティ) よりも大きい場合は、FieldCount の項目のみが取り出され、配列の残りは変更されないままになります。

説明

ほとんどのアプリケーションについて、GetValues メソッドは、各カラムを個々に取り出すのではなく、すべてのカラムを取り出す効率的な方法を提供します。

結果のローに含まれるカラムの数より少ないカラムが含まれる Object 配列を渡すことができます。Object 配列が保持するデータ量のみが配列にコピーされます。また、結果のローに含まれるカラムの数より長い Object 配列を渡すこともできます。

このメソッドは、NULL データベース・カラムに対して DBNull を返します。その他のカラムの場合は、カラムの値をネイティブ・フォーマットで返します。

例外

- ULException クラス - SQL エラーが発生しました。
- System.ArgumentNullException*values* 配列が NULL であるか、長さがゼロです。

実装

IDataRecord.GetValues

参照

- ULDataReader クラス
- ULDataReader メンバ
- FieldCount プロパティ
- GetFieldType メソッド
- GetValue メソッド

IsDBNull メソッド

指定されたカラムの値が NULL かどうかをチェックします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function IsDBNull( _
      ByVal columnID As Integer _
    ) As Boolean _
      Implements IDataRecord.IsDBNull

// C#
    public bool IsDBNull(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

戻り値

値が NULL の場合は true、NULL でない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataRecord.IsDBNull

参照

- ULDataReader クラス
- ULDataReader メンバ
- GetOrdinal メソッド
- GetFieldType メソッド

MoveAfterLast メソッド

UL 拡張：カーソルの最後のローの後に、カーソルを配置します。

プロトタイプ

```
' Visual Basic
    Public Sub MoveAfterLast()

// C#
    public void MoveAfterLast();
```

例外

- ULException クラス - SQL エラーが発生しました。

MoveBeforeFirst メソッド

UL 拡張：カーソルの最初のローの前に、カーソルを配置します。

プロトタイプ

```
' Visual Basic
    Public Sub MoveBeforeFirst()

// C#
    public void MoveBeforeFirst();
```

例外

- ULException クラス - SQL エラーが発生しました。

MoveFirst メソッド

UL 拡張：カーソルの最初のローに、カーソルを配置します。

プロトタイプ

```
' Visual Basic
    Public Function MoveFirst() As Boolean

// C#
    public bool MoveFirst();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、ローがない場合、メソッドは失敗します。

例外

- ULException クラス - SQL エラーが発生しました。

MoveLast メソッド

UL 拡張：カーソルの最後のローに、カーソルを配置します。

プロトタイプ

```
' Visual Basic
    Public Function MoveLast() As Boolean

// C#
    public bool MoveLast();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、ローがない場合、メソッドは失敗します。

例外

- ULException クラス - SQL エラーが発生しました。

MoveNext メソッド

UL 拡張：カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。

プロトタイプ

```
' Visual Basic
    Public Function MoveNext() As Boolean

// C#
    public bool MoveNext();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、それ以上ローがない場合、メソッドは失敗します。

説明

このメソッドは、Read メソッドと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

MovePrevious メソッド

UL 拡張：カーソルを前のローに配置するか、最初のローの前に配置します。

プロトタイプ

```
' Visual Basic
    Public Function MovePrevious() As Boolean

// C#
    public bool MovePrevious();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、それ以上ローがない場合、メソッドは失敗します。

例外

- ULException クラス - SQL エラーが発生しました。

MoveRelative メソッド

UL 拡張：現在のローを基準としてカーソルを配置します。

プロトタイプ

```
' Visual Basic
    Public Function MoveRelative( _
      ByVal offset As Integer _
    ) As Boolean

// C#
    public bool MoveRelative(
      int offset
    );
```

パラメータ

- **offset** 移動するローの数。負の値を指定すると、後方に移動します。

戻り値

成功した場合は true、失敗した場合は false。たとえば、最初または最後のローを超えて配置した場合、メソッドは失敗します。

説明

指定した移動先にローがない場合には、false が返されます。その場合のカーソル位置は、*offset* が正であるときには最後のローの後 (IsEOF プロパティ) になり、*offset* が負であるときには最初のローの前 (IsBOF プロパティ) になります。

例外

- ULException クラス - SQL エラーが発生しました。

NextResult メソッド

バッチ SQL 文の結果を読み込むときに ULDataReader を次の結果に進めます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function NextResult() As Boolean _
      Implements IDataReader.NextResult

// C#
    public bool NextResult();
```

戻り値

さらに結果セットがある場合は true、そうでない場合は false。Ultra Light.NET では、常に false が返されます。

説明

UL 拡張：Ultra Light.NET ではバッチ SQL 文はサポートされていないため、ULDataReader は最初 (唯一) の結果セットに常に配置されます。NextResult は、呼び出しても機能しません。

例外

- ULException クラス - ULDataReader が開かれていません。

実装

IDataReader.NextResult

Read メソッド

カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function Read() As Boolean _
      Implements IDataReader.Read

// C#
    public bool Read();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、それ以上ローがない場合、メソッドは失敗します。

説明

このメソッドは、MoveNext メソッドと同じです。

例外

- ULException クラス - SQL エラーが発生しました。

実装

IDataReader.Read

ULDbType 列挙

Ultra Light.NET データベースのデータ型を列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULDbType

// C#
    public enum ULDbType
```

説明

下の表には、各 ULDbType との互換性がある .NET 型がリストされています。整数型の場合、テーブルのカラムは、常により小さい整数型を使用して設定することができるほか、実際の値がその型の範囲内にある限り、より大きい型を使用して設定することも可能です。

ULDbType	互換性のある .NET 型	C# の組み込み型	Visual Basic の組み込み型
Binary、**VarBinary**	System.Byte[]、またはサイズが 16 の場合は System.Guid	byte[]	Byte()
Bit	System.Boolean	bool	Boolean
Char、**VarChar**	System.String	String	String
Date	System.DateTime	DateTime 組み込み型はなし	Date
Double	System.Double	double	Double
LongBinary	System.Byte[]	byte[]	Byte()
LongVarchar	System.String	String	String
Decimal、**Numeric**	System.String	decimal	Decimal
Float、**Real**	System.Single	float	Single

ULDbType	互換性のある .NET 型	C# の組み込み型	Visual Basic の組み込み型
BigInt	System.Int64	long	Long
Integer	System.Int32	int	Integer
SmallInt	System.Int16	short	Short
Time	System.TimeSpan	TimeSpan 組み込み型はなし	TimeSpan 組み込み型はなし
DateTime、**TimeStamp**	System.DateTime	DateTime 組み込み型はなし	Date
TinyInt	System.Byte	byte	Byte
UnsignedBigInt	System.UInt64	ulong	UInt64 組み込み型はなし
UnsignedInt	System.UInt32	uint	UInt32 組み込み型はなし
UnsignedSmallInt	System.UInt16	ushort	UInt16 組み込み型はなし
UniqueIdentifier	System.Guid	Guid 組み込み型はなし	Guid 組み込み型はなし

長さが 16 のバイナリ・カラムには、UniqueIdentifier 型との完全な互換性があります。

メンバ

メンバ	説明
BigInt	符号付き 64 ビット整数値
Binary	指定した最大長のバイナリ・データ。列挙値 **Binary** と **VarBinary** は互いのエイリアスです。
Bit	1 ビット・フラグ

メンバ	説明
Char	指定した長さの文字データ。Ultra Light.NET では、この型は常に Unicode 文字をサポートします。**Char** 型と **VarChar** 型は、完全に互換性があります。
Date	日付情報
DateTime	タイムスタンプ情報 (日付、時刻)。列挙値 **DateTime** と **TimeStamp** は互いのエイリアスです。
Decimal	精度と桁数が指定された正確な数値データ。列挙値 **Decimal** と **Numeric** は互いのエイリアスです。
Double	倍精度浮動小数点数 (8 バイト)
Float	単精度浮動小数点数 (4 バイト)。列挙値 **Float** と **Real** は互いのエイリアスです。
Integer	符号なし 32 ビット整数値
LongBinary	可変長のバイナリ・データ
LongVarchar	可変長の文字データ。Ultra Light.NET では、この型は常に Unicode 文字をサポートします。
Numeric	精度と桁数が指定された正確な数値データ。列挙値 **Decimal** と **Numeric** は互いのエイリアスです。
Real	単精度浮動小数点数 (4 バイト)。列挙値 **Float** と **Real** は互いのエイリアスです。
SmallInt	符号付き 16 ビット整数値
Time	時刻情報
TimeStamp	タイムスタンプ情報 (日付、時刻)。列挙値 **DateTime** と **TimeStamp** は互いのエイリアスです。
TinyInt	符号なし 8 ビット整数値
UniqueIdentifier	ユニバーサル・ユニーク識別子 (UUID/GUID)
UnsignedBigInt	符号なし 64 ビット整数値

メンバ	説明
UnsignedInt	符号なし 32 ビット整数値
UnsignedSmallInt	符号なし 16 ビット整数値
VarBinary	指定した最大長のバイナリ・データ。列挙値 **Binary** と **VarBinary** は互いのエイリアスです。
VarChar	指定した最大長の文字データ。Ultra Light.NET では、この型は常に Unicode 文字をサポートします。**Char** 型と **VarChar** 型は、完全に互換性があります。

参照

- GetFieldType メソッド
- GetDataTypeName メソッド
- GetColumnULDbType メソッド

ULException クラス

Ultra Light.NET データベースによって返される SQL エラーを示します。

プロトタイプ

```
' Visual Basic
Public Class ULException
  Inherits ApplicationException

// C#
public class ULException :
  ApplicationException
```

説明

.NET Compact Framework では、このクラスは直列可能ではありません。

ULException メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULException コンストラクタ	指定されたエラー・コードを持つ ULException を作成します。

パブリック・インスタンス・プロパティ

メンバ	説明
HelpLink (継承 Exception)	例外に関連付けられているヘルプ ファイルへのリンクを取得または設定します。
InnerException (継承 Exception)	現在の例外を発生させた System.Exception インスタンスを取得します。
Message (継承 Exception)	現在の例外を説明するメッセージを取得します。
NativeError プロパティ	データベースによって返される SQL コードを返します。

メンバ	説明
Source プロパティ	エラーを生成したプロバイダの名前を返します。
StackTrace (継承 Exception)	現在の例外がスローされたときにコール スタックにあったフレームの文字列形式を取得します。
TargetSite (継承 Exception)	現在の例外をスローするメソッドを取得します。

パブリック・インスタンス・メソッド

メンバ	説明
GetBaseException (継承 Exception)	派生クラスでオーバーライドされた場合、それ以後に発生する 1 つ以上の例外の主要な原因である System.Exception を返します。
GetObjectData メソッド	この ULException の直列化に必要なデータを SerializationInfo に移植します。
ToString (継承 Exception)	現在の例外の文字列形式を作成して返します。

プロテクト・インスタンス・プロパティ

メンバ	説明
HResult (継承 Exception)	特定の例外に割り当てられているコード化数値である HRESULT を取得または設定します。

ULException コンストラクタ

指定されたエラー・コードを持つ ULException を作成します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal code As ULSQLCode, _
      ByVal s1 As String, _
```

```
    ByVal s2 As String, _
    ByVal s3 As String _
  )

// C#
  public ULException(
    ULSQLCode code,
    string  s1,
    string  s2,
    string  s3
  );
```

パラメータ

- **code** 例外のコード
- **s1** フォーマットされたメッセージの 1 番目の文字列
- **s2** フォーマットされたメッセージの 2 番目の文字列
- **s3** フォーマットされたメッセージの 3 番目の文字列

説明

指定した ULSQLCode 列挙 に対応するメッセージ文字列は、**iAnywhere.Data.UltraLite.resources** アセンブリから取得されます。リソースは、System.Globalization.CultureInfo.CurrentUICulture、System.Globalization.CultureInfo.CurrentCulture、国 "EN" の順序で国別に検索されます。

NativeError プロパティ

データベースによって返される SQL コードを返します。

プロトタイプ

```
' Visual Basic
  Public Readonly Property NativeError As ULSQLCode

// C#
  public ULSQLCode NativeError {get;}
```

プロパティ値

データベースによって返される ULSQLCode 列挙 値

Source プロパティ

エラーを生成したプロバイダの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Source As String

// C#
    public string  Source {get;}
```

プロパティ値 プロバイダが Ultra Light.NET であることを示す文字列値

GetObjectData メソッド

この ULException の直列化に必要なデータを SerializationInfo に移植します。

プロトタイプ

```
' Visual Basic
    Overrides Public Sub GetObjectData( _
      ByVal info As System.Runtime.Serialization.SerializationInfo, _
      ByVal context As System.Runtime.Serialization.StreamingContext _
    ) _
      Implements ISerializable.GetObjectData

// C#
    public override void GetObjectData(
      System.Runtime.Serialization.SerializationInfo info,
      System.Runtime.Serialization.StreamingContext context
    );
```

パラメータ

- **info** データを移植する SerializationInfo
- **context** この直列化の宛先

説明 .NET Compact Framework では、このメソッドはサポートされていません。

実装 ISerializable.GetObjectData

ULIndexSchema クラス

UL 拡張： Ultra Light テーブルのインデックスのスキーマを示します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **ULIndexSchema**

// C#
public sealed class **ULIndexSchema**

説明

このクラスを直接インスタンス化することはできません。インデックスのスキーマは、ULTableSchema クラスの PrimaryKey プロパティ メソッド、GetIndex メソッド、GetOptimalIndex メソッドを使用して作成されます。

ULIndexSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ	インデックス内のカラム数を返します。
IsForeignKey プロパティ	インデックスが外部キーであるかどうかをチェックします。
IsForeignKeyCheckOnCommit プロパティ	外部キーの参照整合性のチェックが、コミット時に行われるか、挿入時および更新時に行われるかを確認します。
IsForeignKeyNullable プロパティ	外部キーが null 入力可能であるかどうかをチェックします。
IsOpen プロパティ	インデックス・スキーマが開いているか、閉じているかを調べます。
IsPrimaryKey プロパティ	インデックスがプライマリ・キーであるかどうかをチェックします。

メンバ	説明
IsUniqueIndex プロパティ	インデックスがユニークであるかどうかをチェックします。
IsUniqueKey プロパティ	インデックスがユニーク・キーであるかどうかをチェックします。
Name プロパティ	インデックスの名前を返します。
ReferencedIndexName プロパティ	インデックスが外部キーである場合、参照されるプライマリ・インデックスの名前です。
ReferencedTableName プロパティ	インデックスが外部キーである場合、参照されるプライマリ・テーブルの名前です。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnName メソッド	このインデックス内の *colOrdinalInIndex* 番目のカラム名を返します。
IsColumnDescending メソッド	指定したカラムが、インデックスによって降順で使用されるかどうかをチェックします。

ColumnCount プロパティ

インデックス内のカラム数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ColumnCount As Short

// C#
    public short ColumnCount {get;}
```

プロパティ値

インデックス内のカラム数

説明

インデックス内のカラム序数の範囲は、1 ～ ColumnCount です。

カラムの序数とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスからのカラム序数は、特定のテーブルの同じ物理的カラムを参照する場合であっても、テーブルまたは別のインデックスのカラム ID とは異なります。

IsForeignKey プロパティ

インデックスが外部キーであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsForeignKey As Boolean

// C#
    public bool IsForeignKey {get;}
```

プロパティ値

インデックスが外部キーである場合は true、外部キーでない場合は false

説明

外部キー内のカラムは、別のテーブルの null 以外のユニーク・インデックスを参照することがあります。

例外

- ULException クラス - SQL エラーが発生しました。

IsForeignKeyCheckOnCommit プロパティ

外部キーの参照整合性のチェックが、コミット時に行われるか、挿入時および更新時に行われるかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsForeignKeyCheckOnCommit As Boolean

// C#
    public bool IsForeignKeyCheckOnCommit {get;}
```

プロパティ値

参照整合性がコミット時にチェックされる場合は true、挿入時および更新時にチェックされる場合は false

例外

- ULException クラス - SQL エラーが発生しました (インデックスが外部キーではない場合を含む)。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- IsForeignKey プロパティ

IsForeignKeyNullable プロパティ

外部キーが null 入力可能であるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsForeignKeyNullable As Boolean

// C#
    public bool IsForeignKeyNullable {get;}
```

プロパティ値

外部キーが null 入力可能であれば true、null 入力不可であれば false

例外

- ULException クラス - SQL エラーが発生しました (インデックスが外部キーではない場合を含む)。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- IsForeignKey プロパティ

IsOpen プロパティ

インデックス・スキーマが開いているか、閉じているかを調べます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

インデックス・スキーマが開いている場合は true、閉じている場合は false

IsPrimaryKey プロパティ

インデックスがプライマリ・キーであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsPrimaryKey As Boolean

// C#
    public bool IsPrimaryKey {get;}
```

プロパティ値　インデックスがプライマリ・キーである場合は true、プライマリ・キーでない場合は false

説明　プライマリ・キー内のカラムでは null は許可されません。

例外

- ULException クラス - SQL エラーが発生しました。

IsUniqueIndex プロパティ

インデックスがユニークであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsUniqueIndex As Boolean

// C#
    public bool IsUniqueIndex {get;}
```

プロパティ値　インデックスがユニークである場合は true、ユニークでない場合は false

説明　ユニークなインデックス内のカラムは null であることがあります。

例外

- ULException クラス - SQL エラーが発生しました。

IsUniqueKey プロパティ

インデックスがユニーク・キーであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsUniqueKey As Boolean

// C#
    public bool IsUniqueKey {get;}
```

プロパティ値 インデックスがユニーク・キーである場合は true、ユニーク・キーでない場合は false

説明 ユニーク・キー内のカラムでは null は許可されません。

例外

- ULException クラス - SQL エラーが発生しました。

Name プロパティ

インデックスの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

プロパティ値 インデックスの名前を指定する文字列

例外

- ULException クラス - SQL エラーが発生しました。

ReferencedIndexName プロパティ

インデックスが外部キーである場合、参照されるプライマリ・インデックスの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReferencedIndexName As String

// C#
    public string  ReferencedIndexName {get;}
```

プロパティ値 参照されるプライマリ・インデックスの名前を指定する文字列

例外

- ULException クラス - SQL エラーが発生しました (インデックスが外部キーではない場合を含む)。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- IsForeignKey プロパティ

ReferencedTableName プロパティ

インデックスが外部キーである場合、参照されるプライマリ・テーブルの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReferencedTableName As String

// C#
    public string  ReferencedTableName {get;}
```

プロパティ値

参照されるプライマリ・テーブルの名前を指定する文字列

例外

- ULException クラス - SQL エラーが発生しました (インデックスが外部キーではない場合を含む)。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- IsForeignKey プロパティ

GetColumnName メソッド

このインデックス内の *colOrdinalInIndex* 番目のカラム名を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnName( _
      ByVal colOrdinalInIndex As Short _
    ) As String
```

```
// C#
    public string  GetColumnName(
      short colOrdinalInIndex
    );
```

パラメータ

- **colOrdinalInIndex** インデックス内の対象のカラムの序数。値は、[1,ColumnCount プロパティ] の範囲内である必要があります。

戻り値

カラムの名前

説明

カラムの序数とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスからのカラム序数は、特定のテーブルの同じ物理的カラムを参照する場合であっても、テーブルまたは別のインデックスのカラム ID とは異なります。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- ColumnCount プロパティ

IsColumnDescending メソッド

指定したカラムが、インデックスによって降順で使用されるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnDescending( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnDescending(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムが降順で使用される場合は true、昇順で使用される場合は false

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULIndexSchema クラス
- ULIndexSchema メンバ
- GetColumnName メソッド
- ColumnCount プロパティ

ULInfoMessageEventArgs クラス

InfoMessage イベントのデータを提供します。

プロトタイプ

```
' Visual Basic
NotInheritable Public Class ULInfoMessageEventArgs
  Inherits EventArgs

// C#
public sealed class ULInfoMessageEventArgs :
  EventArgs
```

ULInfoMessageEventArgs メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
Message プロパティ	データベースによって返される情報メッセージまたは警告メッセージの文字列です。
NativeError プロパティ	データベースによって返される情報メッセージまたは警告に対応する SQL コードです。
Source プロパティ	メッセージを返す ADO.NET データ・プロバイダの名前です。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド	InfoMessage イベントの文字列表現を返します。

Message プロパティ

データベースによって返される情報メッセージまたは警告メッセージの文字列です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Message As String

// C#
    public string  Message {get;}
```

プロパティ値　情報メッセージまたは警告メッセージが含まれる文字列

NativeError プロパティ

データベースによって返される情報メッセージまたは警告に対応する SQL コードです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property NativeError As ULSQLCode

// C#
    public ULSQLCode NativeError {get;}
```

プロパティ値　情報メッセージまたは警告の ULSQLCode 列挙 値

Source プロパティ

メッセージを返す ADO.NET データ・プロバイダの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Source As String

// C#
    public string  Source {get;}
```

プロパティ値　文字列 "UltraLite.NET Data Provider"

ToString メソッド

InfoMessage イベントの文字列表現を返します。

プロトタイプ

```
' Visual Basic
    Overrides Public Function ToString() As String

// C#
    public override string  ToString();
```

プロパティ値

文字列 "UltraLite.NET Data Provider"

戻り値

情報メッセージまたは警告メッセージの文字列

ULInfoMessageEventHandler 委譲

InfoMessage イベントを処理するメソッドを示します。

プロトタイプ

```
' Visual Basic
Public Delegate Sub ULInfoMessageEventHandler( _
  ByVal obj As Object, _
  ByVal args As ULInfoMessageEventArgs _
)

// C#
public delegate void ULInfoMessageEventHandler(
  object obj,
  ULInfoMessageEventArgs args
);
```

パラメータ

- **obj** イベントを送信する接続
- **args** イベント・データが含まれる ULInfoMessageEventArgs クラス オブジェクト

ULParameter クラス

ULCommand クラス のパラメータを示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULParameter
     Inherits MarshalByRefObject

// C#
    public sealed class ULParameter :
     MarshalByRefobject
```

説明

ULParameter オブジェクトは、その多数のコンストラクタのいずれかを使用したり、CreateParameter メソッドを使用したりして直接作成できます。定数 0 および 0.0 の特別な処理と、オーバロードされたメソッドの解決方法のため、ULParameter コンストラクタを使用するときは、定数値を型オブジェクトに明示的にキャストすることを強くおすすめします。次に例を示します。

```
' Visual Basic Dim p As ULParameter = New ULParameter(
"", CType( 0, Object ) )

 // C# ULParameter p = new ULParameter( "", (object)0 );
```

パラメータ (CreateParameter メソッド によって作成されたものを含む) は、使用される Parameters プロパティ コレクションに追加する必要があります。すべてのパラメータは、位置パラメータとして扱われ、コマンドによって追加された順序で使用されます。

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

実装：IDbDataParameter、IDataParameter

ULParameter メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULParameter コンストラクタ	値として null (Visual Basic の Nothing) を使用して、ULParameter オブジェクトを初期化します。
ULParameter コンストラクタ	ULParameter オブジェクトを、指定したパラメータ名と値で初期化します。
ULParameter コンストラクタ	ULParameter オブジェクトを、指定したパラメータ名とデータ型で初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。
ULParameter コンストラクタ	ULParameter オブジェクトを、指定したパラメータ名とデータ型で初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。
ULParameter コンストラクタ	ULParameter オブジェクトを、指定したパラメータ名、データ型、長さで初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。
ULParameter コンストラクタ	指定したパラメータ名、データ型、長さ、方向、null 入力属性、数値精度、数値の位取り、ソース・カラム、ソースのバージョン、値で、ULParameter オブジェクトを初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。

パブリック・インスタンス・プロパティ

メンバ	説明
DbType プロパティ	パラメータの DbType を指定します。
Direction プロパティ	パラメータが入力専用、出力専用、双方向性、またはストアド・プロシージャ戻り値パラメータのいずれであるかを示す値です。

メンバ	説明
IsNullable プロパティ	パラメータが null 値を受け入れるかどうかを指定します。
Offset プロパティ	Value プロパティのオフセットを指定します。
ParameterName プロパティ	パラメータの名前を指定します。
Precision プロパティ	Value プロパティを示すために使用される最大桁数を指定します。
Scale プロパティ	Value プロパティ が解析される小数点の桁の数を指定します。
Size プロパティ	カラム内のデータの最大サイズを指定します。
SourceColumn プロパティ	DataSet にマッピングされ、値をロードしたり返したりするときに使用するソース・カラムの名前を指定します。
SourceVersion プロパティ	DataRowVersion をロードするときに使用する Value プロパティ です。
ULDbType プロパティ	パラメータの ULDbType 列挙 を指定します。
Value プロパティ	パラメータの値を指定します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド	このインスタンスの文字列表現を返します。

ULParameter コンストラクタ

値として null (Visual Basic の Nothing) を使用して、ULParameter オブジェクトを初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()
```

```
// C#
    public ULParameter();
```

例

次のコードでは、値が 3 である ULParameter パラメータが作成され、cmd という ULCommand クラス に追加されます。

```
' Visual Basic Dim p As ULParameter = New ULParameter
 p.Value = 3
 cmd.Parameters.Add( p )

 // C# ULParameter p = new ULParameter();
 p.Value = 3;
 cmd.Parameters.Add( p );
```

参照

- ULParameter クラス
- ULParameter メンバ
- Value プロパティ
- ULParameter コンストラクタ

ULParameter コンストラクタ

ULParameter オブジェクトを、指定したパラメータ名と値で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal parameterName As String, _
      ByVal value As Object _
    )

// C#
    public ULParameter(
      string  parameterName,
      object value
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。

- **value** パラメータの値である Object

説明

定数 0 および 0.0 の特別な処理と、オーバロードされたメソッドの解決方法のため、このコンストラクタを使用するときは、定数値を型オブジェクトに明示的にキャストすることを強くおすすめします。

例

次のコードでは、値が 0 である ULParameter パラメータが作成され、cmd という ULCommand クラス が追加されます。

```
' Visual Basic cmd.Parameters.Add( New ULParameter( "",
CType( 0, Object ) ) )

 // C# cmd.Parameters.Add( new ULParameter( "",
(object)0 ) );
```

参照

- ULParameter クラス
- ULParameter メンバ
- ULParameter コンストラクタ

ULParameter コンストラクタ

ULParameter オブジェクトを、指定したパラメータ名とデータ型で初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal parameterName As String, _
      ByVal dbType As ULDbType _
    )

// C#
    public ULParameter(
      string  parameterName,
      ULDbType dbType
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **dbType** ULDbType 列挙 値の 1 つ

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

参照

- ULParameter クラス
- ULParameter メンバ
- ULParameter コンストラクタ
- ULParameter コンストラクタ

ULParameter コンストラクタ

ULParameter オブジェクトを、指定したパラメータ名とデータ型で初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal parameterName As String, _
      ByVal dbType As ULDbType, _
      ByVal size As Integer _
    )

// C#
    public ULParameter(
      string  parameterName,
      ULDbType dbType,
      int size
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。

- **dbType** ULDbType 列挙 値の 1 つ
- **size** パラメータの長さ

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

参照

- ULParameter クラス
- ULParameter メンバ
- ULParameter コンストラクタ
- ULParameter コンストラクタ

ULParameter コンストラクタ

ULParameter オブジェクトを、指定したパラメータ名、データ型、長さで初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal parameterName As String, _
      ByVal dbType As ULDbType, _
      ByVal size As Integer, _
      ByVal sourceColumn As String _
    )

// C#
    public ULParameter(
      string  parameterName,
      ULDbType dbType,
      int size,
      string  sourceColumn
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。

- **dbType** ULDbType 列挙 値の 1 つ
- **size** パラメータの長さ
- **sourceColumn** マッピングするソース・カラムの名前

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

参照

- ULParameter クラス
- ULParameter メンバ
- ULParameter コンストラクタ
- ULParameter コンストラクタ

ULParameter コンストラクタ

指定したパラメータ名、データ型、長さ、方向、null 入力属性、数値精度、数値の位取り、ソース・カラム、ソースのバージョン、値で、ULParameter オブジェクトを初期化します。このコンストラクタの使用はおすすめしません。これは、他のデータ・プロバイダとの互換性のために用意されています。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal parameterName As String, _
      ByVal dbType As ULDbType, _
      ByVal size As Integer, _
      ByVal direction As ParameterDirection, _
      ByVal isNullable As Boolean, _
      ByVal precision As Byte, _
      ByVal scale As Byte, _
      ByVal sourceColumn As String, _
      ByVal sourceVersion As DataRowVersion, _
      ByVal value As Object _
    )

// C#
    public ULParameter(
      string  parameterName,
      ULDbType dbType,
      int size,
```

```
  ParameterDirection direction,
  bool isNullable,
  byte precision,
  byte scale,
  string  sourceColumn,
  DataRowVersion sourceVersion,
  object value
);
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **dbType** ULDbType 列挙 値の 1 つ
- **size** パラメータの長さ
- **direction** ParameterDirection 値の 1 つ
- **isNullable** フィールドの値を null にできる場合は true、できない場合は false
- **precision** Value が解析される小数点の左右の桁の合計数
- **scale** Value が解析される小数点の桁の合計数
- **sourceColumn** マッピングするソース・カラムの名前
- **sourceVersion** DataRowVersion 値の 1 つ
- **value** パラメータの値である Object

例外

- ULException クラス - Ultra Light.NET では、ParameterDirection.Input 方向しかサポートされていません。

参照

- ULParameter クラス
- ULParameter メンバ
- ULParameter コンストラクタ
- ULParameter コンストラクタ

DbType プロパティ

パラメータの DbType を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property DbType As DbType _
      Implements IDataParameter.DbType

// C#
    public DbType DbType {get;set;}
```

プロパティ値

DbType 値の 1 つ

説明

ULDbType プロパティ と DbType プロパティはリンクされます。このため、DbType を設定すると、ULDbType プロパティ がサポートされている ULDbType 列挙 に変更されます。

例外

- ArgumentException - 指定した値から ULDbType 列挙 へのマッピングは存在しないため、指定した値はサポートされません。

実装

IDataParameter.DbType

Direction プロパティ

パラメータが入力専用、出力専用、双方向性、またはストアド・プロシージャ戻り値パラメータのいずれであるかを示す値です。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property Direction As ParameterDirection _
      Implements IDataParameter.Direction

// C#
    public ParameterDirection Direction {get;set;}
```

プロパティ値

ParameterDirection 値の 1 つ

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

例外

- ULException クラス - Ultra Light.NET では、ParameterDirection.Input 方向しかサポートされていません。

実装 IDataParameter.Direction

IsNullable プロパティ

パラメータが null 値を受け入れるかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property IsNullable As Boolean _
      Implements IDataParameter.IsNullable

// C#
    public bool IsNullable {get;set;}
```

プロパティ値 null 値を受け入れる場合は true、受け入れない場合は false。デフォルトは false です。null 値は DBNull クラスを使用して処理されます。

説明 Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

実装 IDataParameter.IsNullable

Offset プロパティ

Value プロパティのオフセットを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Offset As Integer

// C#
    public int Offset {get;set;}
```

プロパティ値 値のオフセット。デフォルトは 0 です。

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティだけです。

ParameterName プロパティ

パラメータの名前を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property ParameterName As String _
      Implements IDataParameter.ParameterName

// C#
    public string  ParameterName {get;set;}
```

プロパティ値

パラメータの名前を示す文字列、または名前のないパラメータの場合は空の文字列 ('')。null 参照 (Visual Basic の Nothing) を指定すると、空の文字列が使用されます。

説明

Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。すべてのパラメータは、位置パラメータとして扱われ、コマンドによって追加された順序で使用されます。

実装

IDataParameter.ParameterName

Precision プロパティ

Value プロパティ を示すために使用される最大桁数を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property Precision As Byte _
      Implements IDbDataParameter.Precision

// C#
    public byte Precision {get;set;}
```

プロパティ値

Value プロパティ を示すために使用される最大桁数。デフォルト値は 0 です。これは、データ・プロバイダが Value プロパティ の精度を設定することを示します。

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティ だけです。

例外

- ArgumentException - 値が 38 を超えています。

実装

IDbDataParameter.Precision

Scale プロパティ

Value プロパティ が解析される小数点の桁の数を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property Scale As Byte _
      Implements IDbDataParameter.Scale

// C#
    public byte Scale {get;set;}
```

プロパティ値

Value プロパティ が解析される小数点の桁の数。デフォルトは 0 です。

説明

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティ だけです。

実装

IDbDataParameter.Scale

Size プロパティ

カラム内のデータの最大サイズを指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property Size As Integer _
      Implements IDbDataParameter.Size

// C#
    public int Size {get;set;}
```

プロパティ値　カラム内のデータの最大サイズ。デフォルト値はパラメータ値から推測されます。Size プロパティは、バイナリおよび文字列型に対して使用されます。

説明　Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティ だけです。

実装　IDbDataParameter.Size

SourceColumn プロパティ

DataSet にマッピングされ、値をロードしたり返したりするときに使用するソース・カラムの名前を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property SourceColumn As String _
      Implements IDataParameter.SourceColumn

// C#
    public string  SourceColumn {get;set;}
```

プロパティ値　DataSet にマッピングされ、値をロードしたり返したりするときに使用するソース・カラムの名前を指定する文字列

説明　Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティ だけです。

実装　IDataParameter.SourceColumn

SourceVersion プロパティ

DataRowVersion をロードするときに使用する Value プロパティ です。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property SourceVersion As DataRowVersion _
      Implements IDataParameter.SourceVersion
```

```
// C#
    public DataRowVersion SourceVersion {get;set;}
```

実装 IDataParameter.SourceVersion

ULDbType プロパティ

パラメータの ULDbType 列挙 を指定します。

プロトタイプ

```
' Visual Basic
    Public Property ULDbType As ULDbType

// C#
    public ULDbType ULDbType {get;set;}
```

プロパティ値 ULDbType 列挙 値の 1 つ

説明

ULDbType と DbType プロパティ はリンクされます。このため、ULDbType を設定すると、DbType プロパティ がサポートされている DbType に変更されます。

Ultra Light.NET では、パラメータは IN パラメータとしてのみ使用できるほか、マッピング情報はすべて無視されます。重要なのは、Value プロパティ だけです。

Value プロパティ

パラメータの値を指定します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Property Value As Object _
      Implements IDataParameter.Value

// C#
    public object Value {get;set;}
```

プロパティ値 パラメータの値を指定する Object

説明　値は、型変換やマッピングは行われずに、データ・プロバイダにそのまま送信されます。コマンドが実行されると、コマンドは必要な型に値を変換しますが、値を変換できない場合は ULSQLCode 列挙 とともに ULException クラス を通知します。

実装　IDataParameter.Value

ToString メソッド

このインスタンスの文字列表現を返します。

プロトタイプ

```
' Visual Basic
    Overrides Public Function ToString() As String

// C#
    public override string  ToString();
```

戻り値　パラメータの名前

ULParameterCollection クラス

ULCommand クラス のすべてのパラメータを示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULParameterCollection

// C#
    public sealed class ULParameterCollection
```

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ の疑問符のプレースホルダと同じ順序で指定されます。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

ULParameterCollection にはコンストラクタがありません。ULParameterCollection は、Parameters プロパティ から取得します。

実装：IDataParameterCollection

ULParameterCollection メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
Count プロパティ	コレクション内の ULParameter クラス オブジェクトの数を返します。
Item プロパティ	指定したインデックスの ULParameter クラス を返します。C# では、このプロパティは ULParameterCollection クラスのインデクサです。
Item プロパティ	指定した名前の ULParameter クラス を返します。C# では、このプロパティは ULParameterCollection クラスのインデクサです。

パブリック・インスタンス・メソッド

メンバ	説明
Add メソッド	ULParameter クラス をコレクションに追加します。
Add メソッド	ULParameter クラス をコレクションに追加します。
Add メソッド	指定したパラメータ名と値を使用して作成された新しい ULParameter クラス をコレクションに追加します。
Add メソッド	指定したパラメータ名とデータ型を使用して作成された新しい ULParameter クラス をコレクションに追加します。
Add メソッド	指定したパラメータ名、データ型、長さを使用して作成された新しい ULParameter クラス をコレクションに追加します。
Add メソッド	指定したパラメータ名、データ型、長さ、ソース・カラム名を使用して作成された新しい ULParameter クラス をコレクションに追加します。
Clear メソッド	すべてのパラメータをコレクションから削除します。
Contains メソッド	ULParameter クラス がコレクション内に存在するかどうかをチェックします。
Contains メソッド	指定した名前の ULParameter クラス がコレクション内に存在するかどうかをチェックします。
CopyTo メソッド	ULParameter クラス オブジェクトを ULParameterCollection から指定の配列にコピーします。
GetEnumerator メソッド	コレクションの列挙子を返します。
IndexOf メソッド	コレクション内の ULParameter クラス のロケーションを返します。

メンバ	説明
IndexOf メソッド	指定した名前を持つ ULParameter クラス のコレクション内のロケーションを返します。
Insert メソッド	コレクション内の指定されたインデックス位置に ULParameter クラス を挿入します。
Remove メソッド	ULParameter クラス をコレクションから削除します。
RemoveAt メソッド	コレクションの指定したインデックス位置のパラメータを削除します。
RemoveAt メソッド	指定した名前のパラメータをコレクションから削除します。

Count プロパティ

コレクション内の ULParameter クラス オブジェクトの数を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property Count As Integer _
      Implements ICollection.Count

// C#
    public int Count {get;}
```

プロパティ値 コレクション内の ULParameter クラス オブジェクトの数

実装 ICollection.Count

Item プロパティ

指定したインデックスの ULParameter クラス を返します。C# では、このプロパティは ULParameterCollection クラスのインデクサです。

プロトタイプ

```
' Visual Basic
    Public Property Item As ULParameter
```

// C#
public ULParameter **Item** {get;set;}

パラメータ

- **index** 取り出すパラメータの 0 から始まるインデックス。値は、[0,Count プロパティ -1] の範囲内である必要があります。コレクションの最初のパラメータのインデックス値はゼロです。

戻り値

指定したインデックスの ULParameter クラス

例外

- System.IndexOutOfRangeException - インデックスが無効です。

Item プロパティ

指定した名前の ULParameter クラス を返します。C# では、このプロパティは ULParameterCollection クラスのインデクサです。

プロトタイプ

' Visual Basic
Public Property **Item** As ULParameter

// C#
public ULParameter **Item** {get;set;}

パラメータ

- **parameterName** 取り出すパラメータの名前

戻り値

指定した名前の ULParameter クラス

例外

- System.ArgumentNullException - パラメータ名に null (Visual Basic の Nothing) を使用してパラメータを設定することはできません。
- System.IndexOutOfRangeException - 指定した名前のパラメータがありません。

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Item プロパティ
- GetOrdinal メソッド
- GetValue メソッド
- GetFieldType メソッド

Add メソッド

ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function Add( _
      ByVal value As Object _
    ) As Integer _
      Implements IList.Add

// C#
    public int Add(
      object value
    );
```

パラメータ

- **value** コレクションに追加される ULParameter クラス オブジェクト

戻り値

新しい ULParameter クラス オブジェクトのインデックス

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

例外

- ArgumentExceptionULParameter クラス オブジェクトは、一度しかコレクションに追加できません。
- System.ArgumentNullException - 値に null (Visual Basic の Nothing) を指定することはできません。
- System.InvalidCastException - 指定される値は ULParameter クラス でなければなりません。

実装

IList.Add

参照

- ULParameterCollection クラス

- ULParameterCollection メンバ
- Add メソッド
- Add メソッド

Add メソッド

ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Add( _
      ByVal value As ULParameter _
    ) As ULParameter

// C#
    public ULParameter Add(
      ULParameter value
    );
```

パラメータ

- **value** コレクションに追加される ULParameter クラス オブジェクト

戻り値

新しい ULParameter クラス オブジェクト

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

例外

- ArgumentExceptionULParameter クラス オブジェクトは、一度しかコレクションに追加できません。
- System.ArgumentNullException - 値に null (Visual Basic の Nothing) を指定することはできません。

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ

- Add メソッド

Add メソッド

指定したパラメータ名と値を使用して作成された新しい ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Add( _
      ByVal parameterName As String, _
      ByVal value As Object _
    ) As ULParameter

// C#
    public ULParameter Add(
      string  parameterName,
      object value
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **value** パラメータの値である Object

戻り値

新しい ULParameter クラス オブジェクト

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

定数 0 および 0.0 の特別な処理と、オーバロードされたメソッドの解決方法のため、このメソッドを使用するときは、定数値を型オブジェクトに明示的にキャストすることを強くおすすめします。

例

次のコードでは、値が 0 である ULParameter パラメータが cmd という ULCommand クラス に追加されます。

```
' Visual Basic cmd.Parameters.Add( "", CType( 0, Object
) )

 // C# cmd.Parameters.Add( "", (object)0 );
```

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Add メソッド

Add メソッド

指定したパラメータ名とデータ型を使用して作成された新しい ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Add( _
      ByVal parameterName As String, _
      ByVal ulDbType As ULDbType _
    ) As ULParameter

// C#
    public ULParameter Add(
      string  parameterName,
      ULDbType ulDbType
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **ulDbType** ULDbType 列挙 値の 1 つ

戻り値

新しい ULParameter クラス オブジェクト

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Add メソッド
- Add メソッド

Add メソッド

指定したパラメータ名、データ型、長さを使用して作成された新しい ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Add( _
      ByVal parameterName As String, _
      ByVal ulDbType As ULDbType, _
      ByVal size As Integer _
    ) As ULParameter

// C#
    public ULParameter Add(
      string  parameterName,
      ULDbType ulDbType,
      int size
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **ulDbType** ULDbType 列挙 値の 1 つ

- **size** パラメータの長さ

戻り値

新しい ULParameter クラス オブジェクト

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Add メソッド
- Add メソッド

Add メソッド

指定したパラメータ名、データ型、長さ、ソース・カラム名を使用して作成された新しい ULParameter クラス をコレクションに追加します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function Add( _
      ByVal parameterName As String, _
      ByVal ulDbType As ULDbType, _
      ByVal size As Integer, _
      ByVal sourceColumn As String _
    ) As ULParameter

// C#
    public ULParameter Add(
      string  parameterName,
      ULDbType ulDbType,
      int size,
      string  sourceColumn
    );
```

パラメータ

- **parameterName** パラメータの名前。名前のないパラメータの場合、この値に空の文字列 ("") または null 参照 (Visual Basic の Nothing) を使用します。Ultra Light.NET では、ULCommand クラス はパラメータの名前を使用しません。
- **ulDbType** ULDbType 列挙値の 1 つ
- **size** パラメータの長さ
- **sourceColumn** マッピングするソース・カラムの名前

戻り値

新しい ULParameter クラス オブジェクト

説明

コレクション内のすべてのパラメータは、位置パラメータとして扱われ、CommandText プロパティ 内の対応する疑問符のプレースホルダと同じ順序でコレクションに追加される必要があります。たとえば、コレクション内の最初のパラメータは、SQL 文の最初の疑問符に対応し、コレクション内の 2 番目のパラメータは、SQL 文の 2 番目の疑問符に対応します。CommandText プロパティ 内の疑問符の数は、少なくともコレクション内のパラメータの数と同じでなければなりません。パラメータが不足している場合は、null が代用されます。

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Add メソッド
- Add メソッド

Clear メソッド

すべてのパラメータをコレクションから削除します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Clear() _
      Implements IList.Clear

// C#
    public void Clear();
```

実装

IList.Clear

Contains メソッド

ULParameter クラス がコレクション内に存在するかどうかをチェックします。

プロトタイプ

```
' Visual Basic
Overloads NotOverridable Public Function Contains( _
  ByVal value As Object _
) As Boolean _
  Implements IList.Contains

// C#
public bool Contains(
  object value
);
```

パラメータ

- **value** チェック対象の ULParameter クラス オブジェクト

戻り値

コレクションに ULParameter クラス がある場合は true、ない場合は false

実装

IList.Contains

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Contains メソッド

Contains メソッド

指定した名前の ULParameter クラス がコレクション内に存在するかどうかをチェックします。

プロトタイプ

```
' Visual Basic
Overloads NotOverridable Public Function Contains( _
  ByVal value As String _
) As Boolean _
  Implements IDataParameterCollection.Contains
```

```
// C#
    public bool Contains(
      string  value
    );
```

パラメータ

- **value** 検索対象のパラメータの名前

戻り値

コレクションに、その ULParameter クラス がある場合は true、ない場合は false

実装

IDataParameterCollection.Contains

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- Contains メソッド

CopyTo メソッド

ULParameter クラス オブジェクトを ULParameterCollection から指定の配列にコピーします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub CopyTo( _
      ByVal array As System.Array, _
      ByVal index As Integer _
    ) _
      Implements ICollection.CopyTo

// C#
    public void CopyTo(
      System.Array array,
      int index
    );
```

パラメータ

- **array** ULParameter クラス オブジェクトのコピー先の配列
- **index** 配列の開始インデックス

実装

ICollection.CopyTo

GetEnumerator メソッド

コレクションの列挙子を返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Function GetEnumerator() As
   System.Collections.IEnumerator _
     Implements IEnumerable.GetEnumerator

// C#
    public System.Collections.IEnumerator GetEnumerator();
```

戻り値　コレクション内のパラメータを列挙する ArrayList 列挙子

実装　IEnumerable.GetEnumerator

IndexOf メソッド

コレクション内の ULParameter クラス のロケーションを返します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function IndexOf( _
      ByVal value As Object _
    ) As Integer _
      Implements IList.IndexOf

// C#
    public int IndexOf(
      object value
    );
```

パラメータ

- **value** 検索する ULParameter クラス オブジェクト

戻り値　コレクション内の ULParameter クラス の 0 から始まるインデックス、またはパラメータが見つからない場合は -1

例外

- System.InvalidCastException - 指定される値は ULParameter クラス でなければなりません。

実装　IList.IndexOf

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- IndexOf メソッド

IndexOf メソッド

指定した名前を持つ ULParameter クラス のコレクション内のロケーションを返します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Function IndexOf( _
      ByVal parameterName As String _
    ) As Integer _
      Implements IDataParameterCollection.IndexOf

// C#
    public int IndexOf(
      string  parameterName
    );
```

パラメータ

- **parameterName** 検索するパラメータの名前

戻り値

コレクション内の ULParameter クラス の 0 から始まるインデックス、またはパラメータが見つからない場合は -1

実装

IDataParameterCollection.IndexOf

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- IndexOf メソッド

Insert メソッド

コレクション内の指定されたインデックス位置に ULParameter クラス を挿入します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Insert( _
      ByVal index As Integer, _
```

```
    ByVal value As Object _
) _
    Implements IList.Insert

// C#
public void Insert(
    int index,
    object value
);
```

パラメータ

- **index** コレクションでパラメータが挿入されるロケーションの 0 から始まるインデックス
- **value** 挿入する ULParameter クラス オブジェクト

例外

- System.ArgumentNullException - null 参照 (Visual Basic の Nothing) を使用してパラメータを設定することはできません。
- System.IndexOutOfRangeException - インデックスが無効です。
- System.InvalidCastException - 指定される値は ULParameter クラスでなければなりません。

実装

IList.Insert

Remove メソッド

ULParameter クラス をコレクションから削除します。

プロトタイプ

```
' Visual Basic
NotOverridable Public Sub Remove( _
    ByVal value As Object _
) _
    Implements IList.Remove

// C#
public void Remove(
    object value
);
```

パラメータ

- **value** 挿入する ULParameter クラス オブジェクト

例外

- ArgumentException - コレクション内に、指定したパラメータがありません。
- System.ArgumentNullException - null 参照 (Visual Basic の Nothing) を使用してパラメータを設定することはできません。
- System.InvalidCastException - 指定される値は ULParameter クラスでなければなりません。

実装 IList.Remove

RemoveAt メソッド

コレクションの指定したインデックス位置のパラメータを削除します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Sub RemoveAt( _
      ByVal index As Integer _
    ) _
      Implements IList.RemoveAt

// C#
    public void RemoveAt(
      int index
    );
```

パラメータ

- **index** 削除するパラメータの 0 から始まるインデックス。値は、[0,Count プロパティ -1] の範囲内である必要があります。コレクションの最初のパラメータのインデックス値はゼロです。

例外

- System.IndexOutOfRangeException - インデックスが無効です。

実装 IList.RemoveAt

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- RemoveAt メソッド

RemoveAt メソッド

指定した名前のパラメータをコレクションから削除します。

プロトタイプ

```
' Visual Basic
    Overloads NotOverridable Public Sub RemoveAt( _
      ByVal parameterName As String _
    ) _
      Implements IDataParameterCollection.RemoveAt

// C#
    public void RemoveAt(
      string  parameterName
    );
```

パラメータ

- **parameterName** 取り出すパラメータの名前

例外

- System.IndexOutOfRangeException - 指定した名前のパラメータがありません。

実装

IDataParameterCollection.RemoveAt

参照

- ULParameterCollection クラス
- ULParameterCollection メンバ
- RemoveAt メソッド

ULPublicationSchema クラス

UL 拡張： Ultra Light パブリケーションのインデックスのスキーマを示します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **ULPublicationSchema**

// C#
public sealed class **ULPublicationSchema**

説明

このクラスを直接インスタンス化することはできません。パブリケーションのスキーマは、ULDatabaseSchema クラスの GetPublicationSchema メソッドを使用して作成されます。

パブリケーション・マスクが必要な Ultra Light のメソッドには、実際にはチェック対象のパブリケーションのセットが必要です。セットは、個々のパブリケーションのパブリケーション・マスクの論理和で形成されます。次に例を示します。

```
' Visual Basic Dim mask As Integer = pub1.Mask Or
pub2.Mask

 // C# int mask = pub1.Mask | pub2.Mask;
```

特別な 2 つのマスク値も、このクラスによって提供されます。SYNC_ALL_DB フィールドは、データベース全体に対応します。SYNC_ALL_PUBS フィールドは、すべてのパブリケーションに対応します。

注意：パブリケーション・マスクは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされているマスクをスキーマのアップグレード後にリフレッシュします。

参照

- ULPublicationSchema メンバ
- Mask プロパティ
- GetLastDownloadTime メソッド
- CountUploadRows メソッド
- Schema プロパティ

ULPublicationSchema メンバ

パブリック静的フィールド (共有)

メンバ	説明
SYNC_ALL_DB フィールド	データベース全体に対応するパブリケーション・マスクです。
SYNC_ALL_PUBS フィールド	すべてのパブリケーションに対応するパブリケーション・マスクです。

パブリック・インスタンス・プロパティ

メンバ	説明
IsOpen プロパティ	パブリケーション・スキーマが開いているか、閉じているかを調べます。
Mask プロパティ	パブリケーションのパブリケーション・マスクを返します。
Name プロパティ	このパブリケーションの名前を返します。

SYNC_ALL_DB フィールド

データベース全体に対応するパブリケーション・マスクです。

プロトタイプ

```
' Visual Basic
    Public Shared SYNC_ALL_DB As Integer

// C#
    public const int SYNC_ALL_DB;
```

SYNC_ALL_PUBS フィールド

すべてのパブリケーションに対応するパブリケーション・マスクです。

プロトタイプ

```
' Visual Basic
    Public Shared SYNC_ALL_PUBS As Integer

// C#
    public const int SYNC_ALL_PUBS;
```

IsOpen プロパティ

パブリケーション・スキーマが開いているか、閉じているかを調べます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

パブリケーション・スキーマが開いている場合は true、閉じている場合は false

Mask プロパティ

パブリケーションのパブリケーション・マスクを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Mask As Integer

// C#
    public int Mask {get;}
```

プロパティ値

パブリケーションのパブリケーション・マスク

説明

パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされているマスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーによる例外が発生した場合。スキーマのアップグレードでこのパブリケーションが削除されるか、名前が変更された場合は、PUBLICATION_NOT_FOUND が発行されます。

Name プロパティ

このパブリケーションの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

プロパティ値

パブリケーションの名前を指定する文字列

例外

- ULException クラス - SQL エラーが発生しました。スキーマのアップグレードでパブリケーションが削除されるか、名前が変更された場合は、PUBLICATION_NOT_FOUND が発行されます。

ULResultSetSchema クラス

UL 拡張： Ultra Light の結果セットのスキーマを示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULResultSetSchema
      Inherits ULCursorSchema

// C#
    public sealed class ULResultSetSchema :
      ULCursorSchema
```

説明

このクラスを直接インスタンス化することはできません。ULResultSetSchema クラス オブジェクトは、その Schema プロパティとして結果セットにアタッチされます。

結果セットのスキーマが有効なのは、データ・リーダが開かれている間だけです。

参照

- ULResultSetSchema メンバ
- ULCommand クラス
- ULDataReader クラス

ULResultSetSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ (継承 ULCursorSchema)	このカーソル内のカラム数を返します。
IsOpen プロパティ (継承 ULCursorSchema)	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	カーソルの名前を返します。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnID メソッド (継承 ULCursorSchema)	指定したカラムのカラム ID を返します。
GetColumnName メソッド (継承 ULCursorSchema)	指定したカラム ID で識別されたカラムの名前を返します。
GetColumnPrecision メソッド (継承 ULCursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの精度を返します。
GetColumnScale メソッド (継承 ULCursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの位取りを返します。
GetColumnSize メソッド (継承 ULCursorSchema)	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラム ID によって識別されたカラムのサイズを返します。
GetColumnULDbType メソッド (継承 ULCursorSchema)	指定したカラム ID によって識別されたカラムの Ultra Light.NET データ型を返します。
GetSchemaTable メソッド (継承 ULCursorSchema)	ULDataReader クラス のカラムのスキーマが記述された DataTable を返します。

Name プロパティ

カーソルの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

プロパティ値

ULResultSetSchema を生成した SQL 文

ULRowUpdatedEventArgs クラス

RowUpdated イベントのデータを提供します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULRowUpdatedEventArgs
      Inherits RowUpdatedEventArgs

// C#
    public sealed class ULRowUpdatedEventArgs :
      RowUpdatedEventArgs
```

ULRowUpdatedEventArgs メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULRowUpdatedEventArgs コンストラクタ	ULRowUpdatedEventArgs クラスの新しいインスタンスを初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
Command プロパティ	DbDataAdapter.Update が呼び出されたときに実行された ULCommand クラス を返します。
Errors (継承 RowUpdatedEventArgs)	RowUpdatedEventArgs.Command の実行時に .NET Framework データ プロバイダが生成したエラーを取得します。
RecordsAffected プロパティ	SQL 文の実行によって変更、挿入、または削除されたローの数を返します。SELECT 文の場合、この値は -1 です。
Row (継承 RowUpdatedEventArgs)	DbDataAdapter.Update を通じて送信された DataRow を取得します。

メンバ	説明
StatementType (継承 RowUpdatedEventArgs)	実行された SQL ステートメントの種類を取得します。
Status (継承 RowUpdatedEventArgs)	RowUpdatedEventArgs.Command プロパティの UpdateStatus を取得します。
TableMapping (継承 RowUpdatedEventArgs)	DbDataAdapter.Update を通じて送信された DataTableMapping を取得します。

ULRowUpdatedEventArgs コンストラクタ

ULRowUpdatedEventArgs クラスの新しいインスタンスを初期化します。

プロトタイプ

```
' Visual Basic
    Public Sub New( _
      ByVal row As DataRow, _
      ByVal command As IDbCommand, _
      ByVal statementType As StatementType, _
      ByVal tableMapping As DataTableMapping _
    )

// C#
    public ULRowUpdatedEventArgs(
      DataRow row,
      IDbCommand command,
      StatementType statementType,
      DataTableMapping tableMapping
    );
```

パラメータ

- **row** DbDataAdapter.Update を介して送信された DataRow
- **command** DbDataAdapter.Update が呼び出されたときに実行された IDbCommand
- **statementType** 実行されたクエリのタイプを指定する StatementType 値の 1 つ

- **tableMapping** DbDataAdapter.Update を介して送信された DataTableMapping

Command プロパティ

DbDataAdapter.Update が呼び出されたときに実行された ULCommand クラス を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Command As ULCommand

// C#
    public ULCommand Command {get;}
```

プロパティ値 更新で実行された ULCommand クラス オブジェクト

説明 これは、RowUpdatedEventArgs.Command が厳密に型指定されたものです。

RecordsAffected プロパティ

SQL 文の実行によって変更、挿入、または削除されたローの数を返します。SELECT 文の場合、この値は -1 です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property RecordsAffected As Integer

// C#
    public int RecordsAffected {get;}
```

プロパティ値 変更、挿入、または削除されたローの数。文が失敗したときにローが影響されなかった場合は 0。SELECT 文の場合は -1

ULRowUpdatedEventHandler 委譲

RowUpdated イベントを処理するメソッドを示します。

プロトタイプ

```
' Visual Basic
Public Delegate Sub ULRowUpdatedEventHandler( _
  ByVal sender As Object, _
  ByVal e As ULRowUpdatedEventArgs _
)

// C#
public delegate void ULRowUpdatedEventHandler(
  object sender,
  ULRowUpdatedEventArgs e
);
```

パラメータ

- **sender** イベントを送信する接続
- **e** イベント・データが含まれる ULRowUpdatedEventArgs クラス オブジェクト

ULRowUpdatingEventArgs クラス

RowUpdating イベントのデータを提供します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULRowUpdatingEventArgs
     Inherits RowUpdatingEventArgs

// C#
    public sealed class ULRowUpdatingEventArgs :
     RowUpdatingEventArgs
```

ULRowUpdatingEventArgs メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ULRowUpdatingEventArgs コンストラクタ	ULRowUpdatingEventArgs クラスの新しいインスタンスを初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
Command プロパティ	DbDataAdapter.Update の実行時に実行する ULCommand クラス を指定します。
Errors (継承 RowUpdatingEventArgs)	RowUpdatedEventArgs.Command の実行時に .NET Framework データ プロバイダが生成したエラーを取得します。
Row (継承 RowUpdatingEventArgs)	DbDataAdapter.Update を通じて送信する DataRow を取得します。
StatementType (継承 RowUpdatingEventArgs)	実行する SQL ステートメントの種類を取得します。
Status (継承 RowUpdatingEventArgs)	RowUpdatedEventArgs.Command プロパティの UpdateStatus を取得します。

メンバ	説明
TableMapping (継承 RowUpdatingEventArgs)	DbDataAdapter.Update を通じて送信する DataTableMapping を取得します。

ULRowUpdatingEventArgs コンストラクタ

ULRowUpdatingEventArgs クラスの新しいインスタンスを初期化します。

プロトタイプ

```
' Visual Basic
    Public Sub New( _
      ByVal row As DataRow, _
      ByVal command As IDbCommand, _
      ByVal statementType As StatementType, _
      ByVal tableMapping As DataTableMapping _
    )

// C#
    public ULRowUpdatingEventArgs(
      DataRow row,
      IDbCommand command,
      StatementType statementType,
      DataTableMapping tableMapping
    );
```

パラメータ

- **row** 更新する DataRow
- **command** 更新時に実行する IDbCommand
- **statementType** 実行されたクエリのタイプを指定する StatementType 値の 1 つ
- **tableMapping** DbDataAdapter.Update を介して送信された DataTableMapping

Command プロパティ

DbDataAdapter.Update の実行時に実行する ULCommand クラス を指定します。

プロトタイプ

```
' Visual Basic
    Public Property Command As ULCommand

// C#
    public ULCommand Command {get;set;}
```

プロパティ値

更新時に実行する ULCommand クラス オブジェクト

説明

これは、RowUpdatingEventArgs.Command が厳密に型指定されたものです。

ULRowUpdatingEventHandler 委譲

RowUpdating イベントを処理するメソッドを示します。

プロトタイプ

```
' Visual Basic
Public Delegate Sub ULRowUpdatingEventHandler( _
  ByVal sender As Object, _
  ByVal e As ULRowUpdatingEventArgs _
)

// C#
public delegate void ULRowUpdatingEventHandler(
  object sender,
  ULRowUpdatingEventArgs e
);
```

パラメータ

- **sender** イベントを送信する接続
- **e** イベント・データが含まれる ULRowUpdatingEventArgs クラス オブジェクト

ULRuntimeType 列挙

UL 拡張：Ultra Light.NET ランタイムのタイプを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULRuntimeType
      Inherits Short

// C#
    public enum ULRuntimeType :
      short
```

メンバ

メンバ	説明
STANDALONE_UL	スタンドアロンの Ultra Light.NET ランタイムを選択します。 スタンドアロンのランタイムは、データベースに直接アクセスします。データベースへのアクセスは高速になりますが、データベースを共有することはできません。
UL_ENGINE_CLIENT	Ultra Light エンジンのランタイムを選択します。 Ultra Light.NET エンジン・クライアントは、Ultra Light エンジンと通信してデータベースにアクセスします。つまり、別々のアプリケーションでデータベースを共有できます。

参照

- RuntimeType プロパティ

ULSchemaUpgradeData クラス

UL 拡張：スキーマのアップグレードのモニタリング・データを返します。

プロトタイプ

' Visual Basic
Public Class **ULSchemaUpgradeData**

// C#
public class **ULSchemaUpgradeData**

参照

- ULSchemaUpgradeData メンバ
- ULSchemaUpgradeListener インタフェース

ULSchemaUpgradeData メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
FinalProgressCount プロパティ	最終的な進行状況のカウントの推定値を返します。
ProgressCounter プロパティ	スキーマのアップグレードの進行状況のカウンタを返します。この数は、現在までの進行状況を表す近似値です。
State プロパティ	現在のスキーマのアップグレードのステータスを返します。
UpgradeOperations プロパティ	実行されたスキーマのアップグレード操作の数を返します。

FinalProgressCount プロパティ

最終的な進行状況のカウントの推定値を返します。

プロトタイプ

' Visual Basic
Public Readonly Property **FinalProgressCount** As Long

```
// C#
    public long FinalProgressCount {get;}
```

プロパティ値　推定される最終的な進行状況のカウント

参照

- ULSchemaUpgradeData クラス
- ULSchemaUpgradeData メンバ
- ULSchemaUpgradeState 列挙
- ProgressCounter プロパティ

ProgressCounter プロパティ

スキーマのアップグレードの進行状況のカウンタを返します。この数は、現在までの進行状況を表す近似値です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ProgressCounter As Long

// C#
    public long ProgressCounter {get;}
```

プロパティ値　スキーマのアップグレードの進行状況のカウンタ (0 ～ FinalProgressCount プロパティ の数)

参照

- ULSchemaUpgradeData クラス
- ULSchemaUpgradeData メンバ
- ULSchemaUpgradeState 列挙

State プロパティ

現在のスキーマのアップグレードのステータスを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property State As ULSchemaUpgradeState

// C#
    public ULSchemaUpgradeState State {get;}
```

プロパティ値　ULSchemaUpgradeState 列挙 値の 1 つ

UpgradeOperations プロパティ

実行されたスキーマのアップグレード操作の数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UpgradeOperations As Long

// C#
    public long UpgradeOperations {get;}
```

プロパティ値

現在までに実行されたスキーマ操作の数

説明

この数は、スキーマのアップグレード中に行われた作業の量の近似値です。この数は 0 から開始し、アップグレードの進行に伴って増加します。進行状況のカウンタよりも頻繁に更新されるため、ほかのスキーマ・アップグレードと比較するための相対的な計測値として使用できます。

参照

- ULSchemaUpgradeData クラス
- ULSchemaUpgradeData メンバ
- ULSchemaUpgradeState 列挙

ULSchemaUpgradeListener インタフェース

UL 拡張：スキーマのアップグレードのプログレス・イベントを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ULSchemaUpgradeListener

// C#
    public interface ULSchemaUpgradeListener
```

参照

- ULSchemaUpgradeListener メンバ
- ApplyFile メソッド

ULSchemaUpgradeListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
SchemaUpgrading メソッド	ユーザに進行状況を通知するために、スキーマのアップグレード中に起動されます。ステータスが ULSchemaUpgradeState 列挙 であるときは、このメソッドは true を返してスキーマのアップグレードをキャンセルするか、false を返して続行する必要があります。その他のステータスの場合、このメソッドの戻り値は無視されます。

SchemaUpgrading メソッド

ユーザに進行状況を通知するために、スキーマのアップグレード中に起動されます。ステータスが ULSchemaUpgradeState 列挙 であるときは、このメソッドは true を返してスキーマのアップグレードをキャンセルするか、false を返して続行する必要があります。その他のステータスの場合、このメソッドの戻り値は無視されます。

プロトタイプ

```
' Visual Basic
    Public Function SchemaUpgrading( _
      ByVal data As ULSchemaUpgradeData _
    ) As Boolean

// C#
    public bool SchemaUpgrading(
      ULSchemaUpgradeData data
    );
```

パラメータ

- **data** 最新のスキーマのアップグレードのプログレス・データを保持している ULSchemaUpgradeData クラス オブジェクト

戻り値

このメソッドは、false を返して続行するか、true を返してスキーマのアップグレードをキャンセルする必要があります (キャンセルするには、ステータスは ULSchemaUpgradeState 列挙である必要があります)。

説明

SchemaUpgrading の呼び出し中に、Ultra Light.NET API のメソッドを呼び出さないでください。

ULSchemaUpgradeState 列挙

UL 拡張：スキーマのアップグレード時に発生する可能性のあるすべてのステータスを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULSchemaUpgradeState
      Inherits Short

// C#
    public enum ULSchemaUpgradeState :
      short
```

説明

STATE_STARTING は、アップグレードをキャンセルできる唯一のステータスです。アップグレードがキャンセルされると、ステータス **STATE_ERROR** を持つ 2 番目のイベントを受け取ります。

メンバ

メンバ	説明
STATE_ABORT	スキーマのアップグレードはアボートされ、古いデータベースが保持されています。回復可能なエラーまたはユーザによるアボートの結果、このステータスになることがあります。
STATE_DONE	スキーマのアップグレードが正常に完了しました。
STATE_ERROR	重大なエラーが発生したため、データベースは使用できません。
STATE_LAST	内部的に使用されるステータスであり、SchemaUpgrading メソッド に送信されません。
STATE_STARTING	スキーマのアップグレードが開始されています。これは、アップグレードをキャンセルできる唯一のステータスです。アップグレードがキャンセルされると、ステータス **STATE_ABORT** を持つ 2 番目のイベントを受け取ります。
STATE_UPGRADING	スキーマのアップグレードが進行中です。

参照

- ULSchemaUpgradeData クラス

ULServerSyncListener インタフェース

UL 拡張：サーバの同期メッセージを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ULServerSyncListener

// C#
    public interface ULServerSyncListener
```

ULServerSyncListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
ServerSyncInvoked メソッド	サーバ起動同期用の Mobile Link Listener がアプリケーションを呼び出して同期を実行するときに起動されます。

ServerSyncInvoked メソッド

サーバ起動同期用の Mobile Link Listener がアプリケーションを呼び出して同期を実行するときに起動されます。

プロトタイプ

```
' Visual Basic
    Public Sub ServerSyncInvoked( _
      ByVal messageName As String _
    )

// C#
    public void ServerSyncInvoked(
      string  messageName
    );
```

パラメータ

- **messageName** アプリケーションに送信されるメッセージの名前

説明

このメソッドは、別のスレッドによって呼び出されます。マルチスレッドの問題を回避するには、このメソッドがユーザ・インタフェースにイベントを通知する必要があります。マルチスレッドを使用する場合は、スレッドごとに別々の接続を使用し、lock キーワードを使用して、アプリケーションの共有オブジェクトにアクセスすることをおすすめします。

例

次のコード・フラグメントは、サーバ同期要求の受信方法および UI スレッドでの同期の実行方法を示してします。

```
' Visual Basic Imports iAnywhere.Data.UltraLite

 Public Class MainWindow
   Inherits System.Windows.Forms.Form
   Implements ULServerSyncListener
   Private conn As ULConnection


  Public Sub New(ByVal args() As String)

     MyBase.New()

     'This call is required by the Windows Form
Designer.
     InitializeComponent()

     'Add any initialization after the
InitializeComponent() call

ULConnection.DatabaseManager.SetServerSyncListener( _
         "myCompany.mymsg", "myCompany.myapp", Me _
       )
     'Create Connection
     ...
   End Sub


  Protected Overrides Sub OnClosing( _
         ByVal e As
System.ComponentModel.CancelEventArgs _
       )

ULConnection.DatabaseManager.SetServerSyncListener( _
```

```
        Nothing, Nothing, Nothing _
      )
    MyBase.OnClosing(e)
  End Sub


  Public Sub ServerSyncInvoked(ByVal messageName As
String) _
      Implements
ULServerSyncListener.ServerSyncInvoked
    Me.Invoke(New EventHandler(AddressOf
Me.ServerSyncAction))
  End Sub


  Public Sub ServerSyncAction( _
      ByVal sender As Object, ByVal e As EventArgs _
    )
    ' Do Server sync
    conn.Synchronize()
  End Sub
 End Class


// C# using iAnywhere.Data.UltraLite;
 public class Form1 : System.Windows.Forms.Form,
ULServerSyncListener
 {
   private System.Windows.Forms.MainMenu mainMenu1;
   private ULConnection conn;


  public Form1()
   {
     //
     // Required for Windows Form Designer support
     //
     InitializeComponent();

     //
     // TODO: Add any constructor code after
     // InitializeComponent call
     //

ULConnection.DatabaseManager.SetServerSyncListener(
        "myCompnay.mymsg", "myCompany.myapp", this
```

```
        );
      // Create connection
      ...
    }


  protected override void Dispose( bool disposing )
    {
      base.Dispose( disposing );
    }


  protected override void OnClosing(
       System.ComponentModel.CancelEventArgs e
      )
    {

ULConnection.DatabaseManager.SetServerSyncListener(
         null, null, null
        );
      base.OnClosing(e);
    }


  public void ServerSyncInvoked( string messageName )
    {
      this.Invoke( new EventHandler( ServerSyncHandler )
);
    }


  internal void ServerSyncHandler(object sender,
EventArgs e)
    {
      conn.Synchronize();
    }
 }
```

ULSQLCode 列挙

UL 拡張：Ultra Light.NET によってレポートされる可能性のある SQL コードを列挙します。

プロトタイプ

' Visual Basic

Public Enum **ULSQLCode**

// C#

public enum **ULSQLCode**

メンバ

メンバ	説明
SQLE_AGGREGATES_NOT_ALLOWED	『ASA エラー・メッセージ』>「集合関数の不正な使用です。」を参照してください。
SQLE_ALIAS_NOT_UNIQUE	『ASA エラー・メッセージ』>「エイリアス '%1' がユニークではありません。」を参照してください。
SQLE_ALIAS_NOT_YET_DEFINED	『ASA エラー・メッセージ』>「エイリアス '%1' の定義は、最初の参照前に記述する必要があります。」を参照してください。
SQLE_AMBIGUOUS_INDEX_NAME	『ASA エラー・メッセージ』>「インデックス名 '%1' があいまいです。」を参照してください。
SQLE_ARGUMENT_CANNOT_BE_NULL	『ASA エラー・メッセージ』>「プロシージャ '%2' の引数 %1 に NULL は指定できません。」を参照してください。
SQLE_BAD_ENCRYPTION_KEY	『ASA エラー・メッセージ』>「暗号化キーが不正であるか、または見つかりません。」を参照してください。
SQLE_BAD_PARAM_INDEX	『ASA エラー・メッセージ』>「入力パラメータ・インデックスが範囲外です。」を参照してください。
SQLE_CANNOT_ACCESS_FILE	『ASA エラー・メッセージ』>「ファイル '%1' にアクセスできません -- %2」を参照してください。

メンバ	説明
SQLE_CANNOT_ACCESS_SCHEMA_FILE	『ASA エラー・メッセージ』>「スキーマ・ファイル '%1' にアクセスできません。」を参照してください。
SQLE_CANNOT_CHANGE_USER_NAME	『ASA エラー・メッセージ』>「前回のアップロードのステータスが不明の場合、同期の user_name は変更できません。」を参照してください。
SQLE_CANNOT_EXECUTE_STMT	『ASA エラー・メッセージ』>「文を実行することができませんでした。」を参照してください。
SQLE_CANNOT_MODIFY	『ASA エラー・メッセージ』>「テーブル '%2' のカラム '%1' を変更できません。」を参照してください。
SQLE_CANNOT_REGISTER_LISTENER	『ASA エラー・メッセージ』>「指定されたリスナを登録できませんでした。」を参照してください。
SQLE_COLUMN_AMBIGUOUS	『ASA エラー・メッセージ』>「カラム '%1' が複数のテーブルで見つかりました。相関名が必要です。」を参照してください。
SQLE_COLUMN_CANNOT_BE_NULL	『ASA エラー・メッセージ』>「テーブル '%2' のカラム '%1' は NULL 値を持つことはできません。」を参照してください。
SQLE_COLUMN_IN_INDEX	『ASA エラー・メッセージ』>「インデックスのカラムを変更することはできません。」を参照してください。
SQLE_COLUMN_NOT_FOUND	『ASA エラー・メッセージ』>「カラム '%1' が見つかりません。」を参照してください。
SQLE_COLUMN_NOT_FOUND_IN_TABLE	『ASA エラー・メッセージ』>「テーブル '%2' にカラム '%1' がありません。」を参照してください。
SQLE_COMMUNICATIONS_ERROR	『ASA エラー・メッセージ』>「通信エラーが発生しました。」を参照してください。
SQLE_CONNECTION_ALREADY_EXISTS	『ASA エラー・メッセージ』>「この接続はすでに存在します。」を参照してください。
SQLE_CONNECTION_NOT_FOUND	『ASA エラー・メッセージ』>「接続が見つかりません。」を参照してください。

メンバ	説明
SQLE_CONNECTION_RESTORED	『ASA エラー・メッセージ』> 「Ultra Light の接続がリストアされました。」を参照してください。
SQLE_CONVERSION_ERROR	『ASA エラー・メッセージ』> 「値 %1 をデータ型 %2 に変換できません。」を参照してください。
SQLE_CORRELATION_NAME_NOT_FOUND	『ASA エラー・メッセージ』> 「相関名 '%1' が見つかりません。」を参照してください。
SQLE_COULD_NOT_FIND_FUNCTION	『ASA エラー・メッセージ』> 「ダイナミック・ライブラリ '%2' に '%1' が見つかりませんでした。」を参照してください。
SQLE_COULD_NOT_LOAD_LIBRARY	『ASA エラー・メッセージ』> 「ダイナミック・ライブラリ '%1' をロードできませんでした。」を参照してください。
SQLE_CURSOR_ALREADY_OPEN	『ASA エラー・メッセージ』> 「カーソルはすでに開いています。」を参照してください。
SQLE_CURSOR_NOT_OPEN	『ASA エラー・メッセージ』> 「カーソルが開きません。」を参照してください。
SQLE_CURSOR_RESTORED	『ASA エラー・メッセージ』> 「Ultra Light のカーソル (あるいは結果セットまたはテーブル) がリストアされました。」を参照してください。
SQLE_CURSOROP_NOT_ALLOWED	『ASA エラー・メッセージ』> 「不正なカーソル処理をしようとしました。」を参照してください。
SQLE_DATABASE_CREATED	『ASA エラー・メッセージ』> 「データベースが作成されました。」を参照してください。
SQLE_DATABASE_ERROR	『ASA エラー・メッセージ』> 「データベースの内部エラー %1 -- トランザクションはロールバックされました。」を参照してください。
SQLE_DATABASE_NAME_REQUIRED	『ASA エラー・メッセージ』> 「サーバを起動するには、データベース名が必要です。」を参照してください。
SQLE_DATABASE_NEW	『ASA エラー・メッセージ』> 「データベースがスキーマなしで作成されました。」を参照してください。

メンバ	説明
SQLE_DATABASE_NOT_CREATED	『ASA エラー・メッセージ』> 「データベースの作成に失敗しました : %1」を参照してください。
SQLE_DATABASE_UPGRADE_NOT_POSSIBLE	『ASA エラー・メッセージ』> 「データベースのアップグレードができません。」を参照してください。
SQLE_DATATYPE_NOT_ALLOWED	『ASA エラー・メッセージ』> 「式にサポートされていないデータ型があります。」を参照してください。
SQLE_DBSPACE_FULL	『ASA エラー・メッセージ』> 「データベース領域が最大ファイル・サイズに達しています。」を参照してください。
SQLE_DEVICE_IO_FAILED	『ASA エラー・メッセージ』> 「'%1' のファイル I/O に失敗しました。」を参照してください。
SQLE_DIV_ZERO_ERROR	『ASA エラー・メッセージ』> 「ゼロで除算しようとしました。」を参照してください。
SQLE_DOWNLOAD_CONFLICT	『ASA エラー・メッセージ』> 「既存のローと競合しているため、ダウンロードに失敗しました。」を参照してください。
SQLE_DROP_DATABASE_FAILED	『ASA エラー・メッセージ』> 「データベース '%1' の削除に失敗しました。」を参照してください。
SQLE_DUPLICATE_FOREIGN_KEY	『ASA エラー・メッセージ』> 「テーブル '%2' の外部キー '%1' は、既存の外部キーと重複しています。」を参照してください。
SQLE_DYNAMIC_MEMORY_EXHAUSTED	『ASA エラー・メッセージ』> 「動的メモリが足りません。」を参照してください。
SQLE_ENGINE_ALREADY_RUNNING	『ASA エラー・メッセージ』> 「データベース・サーバはすでに起動しています。」を参照してください。
SQLE_ENGINE_NOT_MULTIUSER	『ASA エラー・メッセージ』> 「データベース・サーバはマルチユーザ・モードで実行していません。」を参照してください。

メンバ	説明
SQLE_ERROR	『ASA エラー・メッセージ』> 「実行時 SQL エラーです -- %1」を参照してください。
SQLE_ERROR_CALLING_FUNCTION	『ASA エラー・メッセージ』> 「外部関数の呼び出しのためのリソースを割り付けられませんでした。」を参照してください。
SQLE_EXPRESSION_ERROR	『ASA エラー・メッセージ』> 「'%1' 付近に無効な式があります。」を参照してください。
SQLE_FOREIGN_KEY_NAME_NOT_FOUND	『ASA エラー・メッセージ』> 「外部キー '%1' は見つかりません。」を参照してください。
SQLE_IDENTIFIER_TOO_LONG	『ASA エラー・メッセージ』> 「識別子 '%1' が長すぎます。」を参照してください。
SQLE_INCORRECT_VOLUME_ID	『ASA エラー・メッセージ』> 「'%1' のボリューム ID が不正です。」を参照してください。
SQLE_INDEX_NAME_NOT_UNIQUE	『ASA エラー・メッセージ』> 「インデックス名 '%1' はユニークではありません。」を参照してください。
SQLE_INDEX_NOT_FOUND	『ASA エラー・メッセージ』> 「インデックス '%1' が見つかりません。」を参照してください。
SQLE_INDEX_NOT_UNIQUE	『ASA エラー・メッセージ』> 「テーブル '%2' のインデックス '%1' はユニークでなければなりません。」を参照してください。
SQLE_INTERRUPTED	『ASA エラー・メッセージ』> 「文の実行がユーザによって中断させられました。」を参照してください。
SQLE_INVALID_DATABASE	『ASA エラー・メッセージ』> 「指定されたデータベースは無効です。」を参照してください。
SQLE_INVALID_FOREIGN_KEY	『ASA エラー・メッセージ』> 「テーブル '%2' の外部キー '%1' に対応するプライマリ・キーの値がありません。」を参照してください。

メンバ	説明
SQLE_INVALID_FOREIGN_KEY_DEF	『ASA エラー・メッセージ』>「外部キーのカラム '%1' にプライマリー・キーと異なる定義があります。」を参照してください。
SQLE_INVALID_GROUP_SELECT	『ASA エラー・メッセージ』>「'%1' に対する関数またはカラムの参照も GROUP BY 句に記述する必要があります。」を参照してください。
SQLE_INVALID_OPTION_SETTING	『ASA エラー・メッセージ』>「不正なオプション '%1' を設定しています。」を参照してください。
SQLE_INVALID_ORDER	『ASA エラー・メッセージ』>「ORDER BY 句の指定が不正です。」を参照してください。
SQLE_INVALID_PARAMETER	『ASA エラー・メッセージ』>「不正なパラメータです。」を参照してください。
SQLE_INVALID_PARSE_PARAMETER	『ASA エラー・メッセージ』>「解析エラーです : %1」を参照してください。
SQLE_INVALID_PASSWORD	『ASA エラー・メッセージ』>「ユーザ ID またはパスワードが無効です。」を参照してください。
SQLE_INVALID_SQL_IDENTIFIER	『ASA エラー・メッセージ』>「SQL の識別子が無効です。」を参照してください。
SQLE_LOCKED	『ASA エラー・メッセージ』>「'%2' のローは、ユーザ '%1' によってロックされています。」を参照してください。
SQLE_MEMORY_ERROR	『ASA エラー・メッセージ』>「メモリエラー -- トランザクションはロールバックされました。」を参照してください。
SQLE_METHOD_CANNOT_BE_CALLED	『ASA エラー・メッセージ』>「メソッド '%1' は現時点では呼び出せません。」を参照してください。
SQLE_NAME_NOT_UNIQUE	『ASA エラー・メッセージ』>「アイテム '%1' はすでに存在しています。」を参照してください。
SQLE_NO_COLUMN_NAME	『ASA エラー・メッセージ』>「抽出されたテーブル '%1' にはカラム %2 に対する名前がありません。」を参照してください。

メンバ	説明
SQLE_NO_CURRENT_ROW	『ASA エラー・メッセージ』>「カーソルの現在のローがありません。」を参照してください。
SQLE_NO_INDICATOR	『ASA エラー・メッセージ』>「NULL に対して、インジケータ変数が用意されていません。」を参照してください。
SQLE_NO_MATCHING_SELECT_ITEM	『ASA エラー・メッセージ』>「派生テーブル '%1' の Select リストに '%2' と一致する式がありません。」を参照してください。
SQLE_NO_PRIMARY_KEY	『ASA エラー・メッセージ』>「テーブル '%1' にはプライマリ・キーが定義されていません。」を参照してください。
SQLE_NOERROR	SQLE_NOERROR(0) ? このコードは、エラーまたは警告がなかったことを示します。
SQLE_NOT_IMPLEMENTED	『ASA エラー・メッセージ』>「'%1' の機能は実装されていません。」を参照してください。
SQLE_NOT_SUPPORTED_IN_ULTRALITE	『ASA エラー・メッセージ』>「Ultra Light では使用できない機能です。」を参照してください。
SQLE_NOTFOUND	『ASA エラー・メッセージ』>「ローが見つかりません。」を参照してください。
SQLE_OVERFLOW_ERROR	『ASA エラー・メッセージ』>「値 %1 は、対象先にとって大きすぎます。」を参照してください。
SQLE_PAGE_SIZE_INVALID	『ASA エラー・メッセージ』>「無効なデータベース・ページ・サイズです。」を参照してください。
SQLE_PERMISSION_DENIED	『ASA エラー・メッセージ』>「パーミッションがありません : %1」を参照してください。
SQLE_PRIMARY_KEY_NOT_UNIQUE	『ASA エラー・メッセージ』>「テーブル '%1' のプライマリ・キーがユニークではありません。」を参照してください。
SQLE_PRIMARY_KEY_TWICE	『ASA エラー・メッセージ』>「テーブルに 2 つのプライマリ・キーを定義することはできません。」を参照してください。

メンバ	説明
SQLE_PRIMARY_KEY_VALUE_REF	『ASA エラー・メッセージ』>「テーブル '%1' 内のローのプライマリ・キーがテーブル '%3' 内の外部キー '%2' によって参照されています。」を参照してください。
SQLE_PUBLICATION_NOT_FOUND	『ASA エラー・メッセージ』>「パブリケーション '%1' が見つかりません。」を参照してください。
SQLE_RESOURCE_GOVERNOR_EXCEEDED	『ASA エラー・メッセージ』>「%1 のリソース・ガバナーが制限を超えています。」を参照してください。
SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY	『ASA エラー・メッセージ』>「参照整合性を保つためにテーブル %1 からローが削除されました。」を参照してください。
SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE	『ASA エラー・メッセージ』>「ローを新しいスキーマ・フォーマットに変換できませんでした。」を参照してください。
SQLE_SCHEMA_UPGRADE_NOT_ALLOWED	『ASA エラー・メッセージ』>「スキーマのアップグレードは現在認められません。」を参照してください。
SQLE_SERVER_SYNCHRONIZATION_ERROR	『ASA エラー・メッセージ』>「サーバ %1 でエラーが発生したため、同期に失敗しました。」を参照してください。
SQLE_START_STOP_DATABASE_DENIED	『ASA エラー・メッセージ』>「データベースの起動／停止の要求は拒否されました。」を参照してください。
SQLE_STRING_RIGHT_TRUNCATION	『ASA エラー・メッセージ』>「文字列データの右側がトランケートされます。」を参照してください。
SQLE_SUBQUERY_SELECT_LIST	『ASA エラー・メッセージ』>「select リストの中にカラムが 2 つ以上指定されています。」を参照してください。
SQLE_SYNC_INFO_INVALID	『ASA エラー・メッセージ』>「同期の情報が不完全か無効です。'%1' を確認してください。」を参照してください。

メンバ	説明
SQLE_SYNC_STATUS_UNKNOWN	『ASA エラー・メッセージ』>「最後の同期アップロードのステータスは不明です。」を参照してください。
SQLE_SYNCHRONIZATION_NOT_FOUND	『ASA エラー・メッセージ』>「'%1' という名前の同期サブスクリプションが見つかりません。」を参照してください。
SQLE_SYNTAX_ERROR	『ASA エラー・メッセージ』>「'%1' %2 の近くに構文エラーがあります。」を参照してください。
SQLE_TABLE_HAS_PUBLICATIONS	『ASA エラー・メッセージ』>「テーブル '%1' はパブリケーションを持っています。」を参照してください。
SQLE_TABLE_IN_USE	『ASA エラー・メッセージ』>「テーブルは使用されています。」を参照してください。
SQLE_TABLE_NOT_FOUND	『ASA エラー・メッセージ』>「テーブル '%1' は見つかりませんでした。」を参照してください。
SQLE_TOO_MANY_CONNECTIONS	『ASA エラー・メッセージ』>「データベース・サーバに接続できる限界数を超えています。」を参照してください。
SQLE_TOO_MANY_TEMP_TABLES	『ASA エラー・メッセージ』>「接続しているテンポラリ・テーブルが多すぎます。」を参照してください。
SQLE_ULTRALITE_DATABASE_NOT_FOUND	『ASA エラー・メッセージ』>「データベース '%1' が見つかりませんでした。」を参照してください。
SQLE_ULTRALITE_OBJ_CLOSED	『ASA エラー・メッセージ』>「閉じられた '%1' に対する操作は無効です。」を参照してください。
SQLE_ULTRALITE_RUNTIME_LIBRARY_MISMATCH	『ASA エラー・メッセージ』>「Ultra Light ランタイム・ライブラリは、データベース・ファイルと互換性がありません。」を参照してください。
SQLE_UNABLE_TO_CONNECT	『ASA エラー・メッセージ』>「データベースが起動できません -- %1」を参照してください。

メンバ	説明
SQLE_UNABLE_TO_START_DATABASE	『ASA エラー・メッセージ』>「指定されたデータベースを起動できません : %1」を参照してください。
SQLE_UNABLE_TO_START_DATABASE_VER_NEWER	『ASA エラー・メッセージ』>「指定されたデータベースを起動できません : データベース %1 を起動するにはサーバをアップグレードする必要があります。」を参照してください。
SQLE_UNCOMMITTED_TRANSACTIONS	『ASA エラー・メッセージ』>「コミットされていないトランザクションの同期またはアップグレードはできません。」を参照してください。
SQLE_UNKNOWN_FUNC	『ASA エラー・メッセージ』>「関数 '%1' はありません。」を参照してください。
SQLE_UNKNOWN_USERID	『ASA エラー・メッセージ』>「'%1' というユーザ ID はありません。」を参照してください。
SQLE_UNRECOGNIZED_OPTION	『ASA エラー・メッセージ』>「オプション '%1' が認識されません。」を参照してください。
SQLE_UNSUPPORTED_CHARACTER_SET_ERROR	『ASA エラー・メッセージ』>「データベース・サーバでは文字セット '%1' を変換できません。」を参照してください。
SQLE_UPLOAD_FAILED_AT_SERVER	『ASA エラー・メッセージ』>「同期サーバがアップロードのコミットに失敗しました。」を参照してください。
SQLE_WRONG_NUM_OF_INSERT_COLS	『ASA エラー・メッセージ』>「INSERT コマンドへの値が正しくありません。」を参照してください。
SQLE_WRONG_PARAMETER_COUNT	『ASA エラー・メッセージ』>「関数 '%1' のパラメータ数が誤りです。」を参照してください。

ULStreamErrorCode 列挙

UL 拡張：同期中にストリームによって報告される可能性のあるエラー・コードを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULStreamErrorCode
     Inherits Short

// C#
    public enum ULStreamErrorCode :
     short
```

メンバ

メンバ	説明
ACTSYNC_NO_PORT	ACTSYNC_NO_PORT(75)
ACTSYNC_NOT_INSTALLED	ACTSYNC_NOT_INSTALLED(76)
CREATE_RANDOM_OBJECT	CREATE_RANDOM_OBJECT(17)
DEQUEUING_CONNECTION	DEQUEUING_CONNECTION(19)
END_READ	END_READ(11)
END_WRITE	END_WRITE(10)
GENERATE_RANDOM	GENERATE_RANDOM(14)
HTTP_BAD_STATUS_CODE	HTTP_BAD_STATUS_CODE(86)
HTTP_BUFFER_SIZE_OUT_OF_RANGE	HTTP_BUFFER_SIZE_OUT_OF_RANGE(79)
HTTP_CHUNK_LEN_BAD_CHARACTER	HTTP_CHUNK_LEN_BAD_CHARACTER(85)
HTTP_CHUNK_LEN_ENCODED_MISSING	HTTP_CHUNK_LEN_ENCODED_MISSING(84)

メンバ	説明
HTTP_CLIENT_ID_NOT_SET	HTTP_CLIENT_ID_NOT_SET(78)
HTTP_CONTENT_TYPE_NOT_SPECIFIED	HTTP_CONTENT_TYPE_NOT_SPECIFIED(77)
HTTP_CRLF_ENCODED_MISSING	HTTP_CRLF_ENCODED_MISSING(81)
HTTP_CRLF_MISSING	HTTP_CRLF_MISSING(82)
HTTP_EXPECTED_POST	HTTP_EXPECTED_POST(89)
HTTP_EXTRA_DATA_END_READ	HTTP_EXTRA_DATA_END_READ(80)
HTTP_NO_CONTD_CONNECTION	HTTP_NO_CONTD_CONNECTION(83)
HTTP_UNABLE_TO_PARSE_COOKIE	HTTP_UNABLE_TO_PARSE_COOKIE(88)
HTTP_UNKNOWN_TRANSFER_ENCODING	HTTP_UNKNOWN_TRANSFER_ENCODING(87)
HTTP_VERSION	HTTP_VERSION(54)
INIT_RANDOM	INIT_RANDOM(15)
LOAD_NETWORK_LIBRARY	LOAD_NETWORK_LIBRARY(74)
MEMORY_ALLOCATION	MEMORY_ALLOCATION(6)
NONE	NONE(0)
NOT_IMPLEMENTED	NOT_IMPLEMENTED(12)
PARAMETER	PARAMETER(1)
PARAMETER_NOT_BOOLEAN	PARAMETER_NOT_BOOLEAN(4)

メンバ	説明
PARAMETER_NOT_HEX	PARAMETER_NOT_HEX(5)
PARAMETER_NOT_UINT32	PARAMETER_NOT_UINT32(2)
PARAMETER_NOT_UINT32_RANGE	PARAMETER_NOT_UINT32_RANGE(3)
PARSE	PARSE(7)
READ	READ(8)
SECURE_ADD_CERTIFICATE	SECURE_ADD_CERTIFICATE(39)
SECURE_ADD_TRUSTED_CERTIFICATE	SECURE_ADD_TRUSTED_CERTIFICATE(48)
SECURE_CERTIFICATE_CHAIN_FUNC	SECURE_CERTIFICATE_CHAIN_FUNC(28)
SECURE_CERTIFICATE_CHAIN_LENGTH	SECURE_CERTIFICATE_CHAIN_LENGTH(22)
SECURE_CERTIFICATE_CHAIN_REF	SECURE_CERTIFICATE_CHAIN_REF(29)
SECURE_CERTIFICATE_COMMON_NAME	SECURE_CERTIFICATE_COMMON_NAME(52)
SECURE_CERTIFICATE_COMPANY_NAME	SECURE_CERTIFICATE_COMPANY_NAME(21)
SECURE_CERTIFICATE_COMPANY_UNIT	SECURE_CERTIFICATE_COMPANY_UNIT(51)
SECURE_CERTIFICATE_COUNT	SECURE_CERTIFICATE_COUNT(42)
SECURE_CERTIFICATE_EXPIRED	SECURE_CERTIFICATE_EXPIRED(50)
SECURE_CERTIFICATE_EXPIRY_DATE	SECURE_CERTIFICATE_EXPIRY_DATE(37)

メンバ	説明
SECURE_CERTIFICATE_FILE_NOT_FOUND	SECURE_CERTIFICATE_FILE_NOT_FOUND(33)
SECURE_CERTIFICATE_NOT_TRUSTED	SECURE_CERTIFICATE_NOT_TRUSTED(24)
SECURE_CERTIFICATE_REF	SECURE_CERTIFICATE_REF(23)
SECURE_CERTIFICATE_ROOT	SECURE_CERTIFICATE_ROOT(20)
SECURE_CREATE_CERTIFICATE	SECURE_CREATE_CERTIFICATE(43)
SECURE_CREATE_PRIVATE_KEY_OBJECT	SECURE_CREATE_PRIVATE_KEY_OBJECT(49)
SECURE_DUPLICATE_CONTEXT	SECURE_DUPLICATE_CONTEXT(25)
SECURE_ENABLE_NON_BLOCKING	SECURE_ENABLE_NON_BLOCKING(30)
SECURE_EXPORT_CERTIFICATE	SECURE_EXPORT_CERTIFICATE(38)
SECURE_HANDSHAKE	SECURE_HANDSHAKE(53)
SECURE_IMPORT_CERTIFICATE	SECURE_IMPORT_CERTIFICATE(44)
SECURE_READ_CERTIFICATE	SECURE_READ_CERTIFICATE(34)
SECURE_READ_PRIVATE_KEY	SECURE_READ_PRIVATE_KEY(35)
SECURE_SET_CHAIN_NUMBER	SECURE_SET_CHAIN_NUMBER(32)
SECURE_SET_CIPHER_SUITES	SECURE_SET_CIPHER_SUITES(31)
SECURE_SET_IO	SECURE_SET_IO(26)

メンバ	説明
SECURE_SET_IO_SEMANTICS	SECURE_SET_IO_SEMANTICS(27)
SECURE_SET_PRIVATE_KEY	SECURE_SET_PRIVATE_KEY(36)
SECURE_SET_PROTOCOL_SIDE	SECURE_SET_PROTOCOL_SIDE(47)
SECURE_SET_RANDOM_FUNC	SECURE_SET_RANDOM_FUNC(46)
SECURE_SET_RANDOM_REF	SECURE_SET_RANDOM_REF(45)
SECURE_SET_READ_FUNC	SECURE_SET_READ_FUNC(55)
SECURE_SET_WRITE_FUNC	SECURE_SET_WRITE_FUNC(56)
SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND	SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND(40)
SECURE_TRUSTED_CERTIFICATE_READ	SECURE_TRUSTED_CERTIFICATE_READ(41)
SEED_RANDOM	SEED_RANDOM(16)
SHUTTING_DOWN	SHUTTING_DOWN(18)
SOCKET_BIND	SOCKET_BIND(62)
SOCKET_CLEANUP	SOCKET_CLEANUP(63)
SOCKET_CLOSE	SOCKET_CLOSE(64)
SOCKET_CONNECT	SOCKET_CONNECT(65)
SOCKET_CREATE_TCPIP	SOCKET_CREATE_TCPIP(60)
SOCKET_CREATE_UDP	SOCKET_CREATE_UDP(61)

メンバ	説明
SOCKET_GET_HOST_BY_ADDR	SOCKET_GET_HOST_BY_ADDR(58)
SOCKET_GET_NAME	SOCKET_GET_NAME(66)
SOCKET_GET_OPTION	SOCKET_GET_OPTION(67)
SOCKET_HOST_NAME_NOT_FOUND	SOCKET_HOST_NAME_NOT_FOUND(57)
SOCKET_LISTEN	SOCKET_LISTEN(69)
SOCKET_LOCALHOST_NAME_NOT_FOUND	SOCKET_LOCALHOST_NAME_NOT_FOUND(59)
SOCKET_PORT_OUT_OF_RANGE	SOCKET_PORT_OUT_OF_RANGE(73)
SOCKET_SELECT	SOCKET_SELECT(71)
SOCKET_SET_OPTION	SOCKET_SET_OPTION(68)
SOCKET_SHUTDOWN	SOCKET_SHUTDOWN(70)
SOCKET_STARTUP	SOCKET_STARTUP(72)
WOULD_BLOCK	WOULD_BLOCK(13)
WRITE	WRITE(9)

参照

- StreamErrorCode プロパティ

ULStreamErrorContext 列挙

UL 拡張：ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULStreamErrorContext
     Inherits Short

// C#
    public enum ULStreamErrorContext :
     short
```

メンバ

メンバ	説明
CLOSE	CLOSE(6)
CREATE	CREATE(3)
DESTROY	DESTROY(4)
END_READ	END_READ(11)
END_WRITE	END_WRITE(10)
GETVALUE	GETVALUE(14)
OPEN	OPEN(5)
READ	READ(7)
REGISTER	REGISTER(1)
SOFTSHUTDOWN	SOFTSHUTDOWN(13)
UNKNOWN	UNKNOWN(0)
UNREGISTER	UNREGISTER(2)
WRITE	WRITE(8)
WRITE_FLUSH	WRITE_FLUSH(9)
YIELD	YIELD(12)

参照

- StreamErrorContext プロパティ

ULStreamErrorID 列挙

UL 拡張：同期中にエラーを報告する可能性のあるネットワーク・レイヤを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULStreamErrorID
     Inherits Short

// C#
    public enum ULStreamErrorID :
     short
```

メンバ

メンバ	説明
ACTIVESYNC	ACTIVESYNC(22)
CERTICOM	CERTICOM(11)
CERTICOM_SSL	CERTICOM_SSL(13)
CERTICOM_TLS	CERTICOM_TLS(14)
DH_CAST	DH_CAST(9)
EMAIL	EMAIL(20)
FAKE	FAKE(2)
FILE	FILE(21)
HTTP	HTTP(7)
HTTPS	HTTPS(8)
JAVA_CERTICOM	JAVA_CERTICOM(12)
JAVA_RSA	JAVA_RSA(24)
NETTECH	NETTECH(5)
PALM_CONDUIT	PALM_CONDUIT(3)
PALM_SS	PALM_SS(4)

メンバ	説明
REPLAY	REPLAY(17)
RIMBB	RIMBB(6)
RSA_TLS	RSA_TLS(23)
SECURE	SECURE(10)
SERIAL	SERIAL(1)
STRM	STRM(18)
TCPIP	TCPIP(0)
UDP	UDP(19)
WIRELESS	WIRELESS(16)
WIRESTRM	WIRESTRM(15)

参照

- StreamErrorID プロパティ

ULStreamType 列挙

UL 拡張：同期に使用する Mobile Link 同期ストリームのタイプを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULStreamType
     Inherits Short

// C#
    public enum ULStreamType :
     short
```

説明

特定のストリームのタイプの設定方法については、『Mobile Link クライアント』> 「Ultra Light 同期クライアントのネットワーク・プロトコルのオプション」を参照してください。

メンバ

メンバ	説明
ACTIVE_SYNC	ActiveSync 同期 (Windows CE のみ) です。ActiveSync 用 Mobile Link プロバイダから同期を通知されたときは、Ultra Light アプリケーションは ActiveSync 同期のみを使用する必要があります。ULActiveSyncListener インタフェース を実装し、SetActiveSyncListener メソッド を呼び出すことによって、アプリケーションはこのような通知を受信できます。
HTTP	HTTP を介して同期します。HTTP ストリームは基本となるトランスポートとして TCP/IP を使用します。Ultra Light アプリケーションは Web ブラウザとして機能し、Mobile Link 同期サーバは Web サーバとして機能します。Ultra Light アプリケーションは、サーバへのデータ送信のために POST 要求を送り、サーバからのデータの読み込みのために GET 要求を送ります。

メンバ	説明
HTTPS	HTTPS を介して同期します (RSA トランスポート・レイヤ・セキュリティを適用した HTTP)。トランスポート・レイヤ・セキュリティに関するコンポーネントには別途ライセンスが必要です。インストールする場合はご注文ください。このコンポーネントのご注文については、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。
TCPIP	TCP/IP を介して同期します。
UNKNOWN	未知の同期ストリームであるか、同期ストリームがありません。ユーザがストリーム・タイプを設定していません。

参照

- Stream プロパティ

ULSyncParms クラス

UL 拡張： Ultra Light データベースの同期方法を定義する同期パラメータを表します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULSyncParms

// C#
    public sealed class ULSyncParms
```

説明

このクラスを直接インスタンス化することはできません。各接続には、SyncParms プロパティとしてアタッチされた、固有の ULSyncParms インスタンスがあります。

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、ULSQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

その他の ULSQLCode 列挙 エラーの原因には、Stream プロパティ 値または Version プロパティ 値を指定していないことが含まれます。

参照

- ULSyncParms メンバ
- ULConnection クラス
- SyncParms プロパティ
- Synchronize メソッド

ULSyncParms メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
AuthenticationParms プロパティ	カスタム・ユーザ認証スクリプト (Mobile Link authenticate_parameters 接続イベント) のパラメータを指定します。

メンバ	説明
CheckpointStore プロパティ	クライアントが、追加のストア・チェックポイントを実行して、同期中にデータベース・ストアのサイズの増加を制御するかどうかを指定します。
DisableConcurrency プロパティ	同期の実行中に Ultra Light への同時アクセスを無効にするか、有効にするかを指定します。
DownloadOnly プロパティ	同期時のアップロードを無効にするか、有効にするかを指定します。
KeepPartialDownload プロパティ	同期時の部分的なダウンロードを無効にするか、有効にするかを指定します。
NewPassword プロパティ	UserName で指定されたユーザの新しい Mobile Link パスワードを指定します。
Password プロパティ	UserName で指定されたユーザの Mobile Link パスワードです。
PingOnly プロパティ	実際に同期を行う代わりに、クライアントが Mobile Link 同期サーバに ping のみを行うかどうかを指定します。
PublicationMask プロパティ	同期させるパブリケーションを指定します。
ResumePartialDownload プロパティ	前の部分的なダウンロードを再開するか、破棄するかを指定します。
SendColumnNames プロパティ	同期中に、クライアントが Mobile Link 同期サーバにカラム名を送信するかどうかを指定します。
SendDownloadAck プロパティ	同期中に、クライアントが Mobile Link 同期サーバにダウンロード確認を送信するかどうかを指定します。
Stream プロパティ	同期に使用する Mobile Link 同期ストリームを指定します。
StreamParms プロパティ	同期ストリームの設定パラメータを指定します。
UploadOnly プロパティ	同期時のダウンロードを無効にするか、有効にするかを指定します。

メンバ	説明
CheckpointStore プロパティ	クライアントが、追加のストア・チェックポイントを実行して、同期中にデータベース・ストアのサイズの増加を制御するかどうかを指定します。
DisableConcurrency プロパティ	同期の実行中に Ultra Light への同時アクセスを無効にするか、有効にするかを指定します。
DownloadOnly プロパティ	同期時のアップロードを無効にするか、有効にするかを指定します。
KeepPartialDownload プロパティ	同期時の部分的なダウンロードを無効にするか、有効にするかを指定します。
NewPassword プロパティ	UserName で指定されたユーザの新しい Mobile Link パスワードを指定します。
Password プロパティ	UserName で指定されたユーザの Mobile Link パスワードです。
PingOnly プロパティ	実際に同期を行う代わりに、クライアントが Mobile Link 同期サーバに ping のみを行うかどうかを指定します。
PublicationMask プロパティ	同期させるパブリケーションを指定します。
ResumePartialDownload プロパティ	前の部分的なダウンロードを再開するか、破棄するかを指定します。
SendColumnNames プロパティ	同期中に、クライアントが Mobile Link 同期サーバにカラム名を送信するかどうかを指定します。
SendDownloadAck プロパティ	同期中に、クライアントが Mobile Link 同期サーバにダウンロード確認を送信するかどうかを指定します。
Stream プロパティ	同期に使用する Mobile Link 同期ストリームを指定します。
StreamParms プロパティ	同期ストリームの設定パラメータを指定します。
UploadOnly プロパティ	同期時のダウンロードを無効にするか、有効にするかを指定します。

メンバ	説明
UserName プロパティ	Mobile Link 同期サーバが Mobile Link クライアントをユニークに識別するユーザ名です。
Version プロパティ	使用する同期スクリプトを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
CopyFrom メソッド	指定した ULSyncParms クラス オブジェクトのプロパティを、この ULSyncParms クラス オブジェクトにコピーします。

AuthenticationParms プロパティ

カスタム・ユーザ認証スクリプト (Mobile Link authenticate_parameters 接続イベント) のパラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Property AuthenticationParms As String()

// C#
    public string [] AuthenticationParms {get;set;}
```

プロパティ値

それぞれに認証パラメータが格納された、文字列の配列 (配列のエントリが null であると、同期エラーになります)。デフォルト値は null 参照 (Visual Basic の Nothing) で、認証パラメータは指定されません。

説明

最初の 255 文字のみが使用されます。また、各文字列は 128 文字以下である必要があります (長すぎる文字列は、Mobile Link 同期サーバに送信されるときにトランケートされます)。

CheckpointStore プロパティ

クライアントが、追加のストア・チェックポイントを実行して、同期中にデータベース・ストアのサイズの増加を制御するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property CheckpointStore As Boolean

// C#
    public bool CheckpointStore {get;set;}
```

プロパティ値

クライアントが追加のストア・チェックポイントを実行する必要があると指定する場合は true。デフォルトは false で、必要なチェックポイントのみが実行されます。

説明

チェックポイント操作は、アプリケーションの I/O 操作を増やすため、同期が低速になります。このオプションは、多くの更新を伴う大量のダウンロードに最適です。低速なフラッシュ・メモリを使用するデバイスでは、追加のチェックポイントの実行によって、望ましくないパフォーマンスの低下が生じることがあります。

DisableConcurrency プロパティ

同期の実行中に Ultra Light への同時アクセスを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property DisableConcurrency As Boolean

// C#
    public bool DisableConcurrency {get;set;}
```

プロパティ値

同期の実行中に Ultra Light への同時アクセスを無効にする場合は true、有効にする場合は false。デフォルトは false です。

説明

デフォルトでは、1 つのスレッドが同期しているときに、別のスレッドが Ultra Light の操作を行うことがあります。同時同期が無効になっていると、別のスレッドは同期が完了するまで Ultra Light の呼び出しをブロックされます。

DownloadOnly プロパティ

同期時のアップロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property DownloadOnly As Boolean

// C#
    public bool DownloadOnly {get;set;}
```

プロパティ値

同期時のアップロードを無効にする場合は true、有効にする場合は false。デフォルトは false です。

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、ULSQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- UploadOnly プロパティ

KeepPartialDownload プロパティ

同期時の部分的なダウンロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property KeepPartialDownload As Boolean

// C#
    public bool KeepPartialDownload {get;set;}
```

プロパティ値

同期時の部分的なダウンロードを有効にする場合は true、無効にする場合は false。デフォルトは false です。

説明

Ultra Light.NET では、通信エラーや、ユーザによる ULSyncProgressListener からのアボートが原因で失敗したダウンロードを再起動できます。Ultra Light.NET は、ダウンロードを受信しなが

ら処理します。ダウンロードが中断した場合は、部分的なダウンロード・トランザクションがデータベース内に残るため、次の同期中に再開できます。

Ultra Light.NET で部分的なダウンロードを保存する必要があることを示すには、connection.SyncParms.KeepPartialDownload=true と指定します。指定しないと、エラーが発生した場合にダウンロードがロールバックされます。

部分的なダウンロードが保持された場合、connection.Synchronize() の終了時に、出力フィールド connection.SyncResult.PartialDownloadRetained プロパティ が true に設定されます。

PartialDownloadRetained が設定されている場合は、ダウンロードを再開できます。再開するには、true に設定された connection.SyncParms.ResumePartialDownload プロパティ を使用して connection.Synchronize() を呼び出します。別の通信エラーの発生に備えて、KeepPartialDownload も true に設定しておくことをおすすめします。ダウンロードが省略された場合は、アップロードは行われません。

再開したダウンロードで受信するダウンロードは、最初にダウンロードを開始したときと同じものです。最新のデータが必要な場合は、再開された特別なダウンロードが完了した直後に、もう一度ダウンロードを行うことができます。

ダウンロードを再開する場合、ULSyncParms フィールドの多くは関係ありません。たとえば、PublicationMask フィールドが使用されません。受信するパブリケーションは、最初のダウンロード時に要求したものです。設定する必要があるフィールドは、ResumePartialDownload プロパティ と UserName プロパティ だけです。KeepPartialDownload フィールドと DisableConcurrency プロパティ フィールドを必要に応じて設定すると、正常に機能します。

部分的なダウンロードが存在するが、このダウンロードが必要ではなくなった場合は、RollbackPartialDownload メソッド を呼び出して、失敗したダウンロード・トランザクションをロールバックできます。また、同期をもう一度実行したときに ResumePartialDownload を指定しなかった場合は、部分的なダウンロードがロールバックされてから、次の同期が開始されます。

詳細については、『Mobile Link 管理ガイド』> 「失敗したダウンロードの再開」を参照してください。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- PartialDownloadRetained プロパティ
- ResumePartialDownload プロパティ
- RollbackPartialDownload メソッド

NewPassword プロパティ

UserName で指定されたユーザの新しい Mobile Link パスワードを指定します。

プロトタイプ

```
' Visual Basic
    Public Property NewPassword As String

// C#
    public string  NewPassword {get;set;}
```

プロパティ値

新しい Mobile Link パスワードを指定する文字列。デフォルト値は null 参照 (Visual Basic の Nothing) で、パスワードは変更されません。

説明

新しいパスワードが有効になるのは、次の同期の後です。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- UserName プロパティ

Password プロパティ

UserName で指定されたユーザの Mobile Link パスワードです。

プロトタイプ

```
' Visual Basic
    Public Property Password As String

// C#
    public string  Password {get;set;}
```

プロパティ値

Mobile Link パスワードを指定する文字列。デフォルト値は null 参照 (Visual Basic の Nothing) で、パスワードは指定されません。

説明

Mobile Link ユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link 同期サーバに対して識別し、認証するために使用されます。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- NewPassword プロパティ
- UserName プロパティ

PingOnly プロパティ

実際に同期を行う代わりに、クライアントが Mobile Link 同期サーバに ping のみを行うかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property PingOnly As Boolean

// C#
    public bool PingOnly {get;set;}
```

プロパティ値

クライアントが Mobile Link 同期サーバに ping のみを行うように指定する場合は true、クライアントが実際に同期を行うように指定する場合は false。デフォルトは false です。

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、ULSQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

PublicationMask プロパティ

同期させるパブリケーションを指定します。

プロトタイプ

```
' Visual Basic
    Public Property PublicationMask As Integer

// C#
    public int PublicationMask {get;set;}
```

プロパティ値

パブリケーション・マスク のビット処理の組み合わせとして、特別値 SYNC_ALL_PUBS フィールド、または特別値 SYNC_ALL_DB フィールド。デフォルトは SYNC_ALL_DB フィールド です。パブリケーション・マスクの詳細については、ULPublicationSchema クラス を参照してください。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- Mask プロパティ

ResumePartialDownload プロパティ

前の部分的なダウンロードを再開するか、破棄するかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property ResumePartialDownload As Boolean

// C#
    public bool ResumePartialDownload {get;set;}
```

プロパティ値

前の部分的なダウンロードを再開する場合は true、破棄する場合は false。デフォルトは false です。

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、ULSQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

部分的なダウンロードの詳細については、KeepPartialDownload プロパティ を参照してください。

参照

- ULSyncParms クラス
- ULSyncParms メンバ

- PartialDownloadRetained プロパティ

SendColumnNames プロパティ

同期中に、クライアントが Mobile Link 同期サーバにカラム名を送信するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property SendColumnNames As Boolean

// C#
    public bool SendColumnNames {get;set;}
```

プロパティ値

クライアントがカラム名を Mobile Link 同期サーバに送信する必要があると指定する場合は true、カラム名を送信しないように指定する場合は false。デフォルトは false です。

説明

このパラメータは、通常は同期スクリプトを自動的に生成するために Mobile Link 同期サーバで -za オプションまたは -ze オプションとともに使用されます。

SendDownloadAck プロパティ

同期中に、クライアントが Mobile Link 同期サーバにダウンロード確認を送信するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property SendDownloadAck As Boolean

// C#
    public bool SendDownloadAck {get;set;}
```

プロパティ値

クライアントがダウンロード確認を Mobile Link 同期サーバに送信する必要があると指定する場合は true、ダウンロード確認を送信しないように指定する場合は false。デフォルトは false です。

説明

クライアントがダウンロード確認を送信する場合、Mobile Link 同期サーバのワーカ・スレッドは、クライアントがダウンロードを適用するまで待機します。クライアントがダウンロード確認を送信しない場合、Mobile Link 同期サーバは、次の同期のため、まもなく解放されます。

Stream プロパティ

同期に使用する Mobile Link 同期ストリームを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Stream As ULStreamType

// C#
    public ULStreamType Stream {get;set;}
```

プロパティ値

使用する同期ストリームのタイプを指定する ULStreamType 列挙 値の 1 つ。このパラメータにはデフォルト値がないので、明示的に設定してください。

説明

ほとんどの同期ストリームでは、Mobile Link 同期サーバのアドレスを識別したり、その他の動作を制御したりするパラメータが必要です。これらのパラメータは、StreamParms プロパティで指定します。

ストリーム・タイプが、プラットフォームに適さない無効な値に設定されていると、ストリーム・タイプは ULStreamType 列挙 に設定されます。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- ULStreamType 列挙
- StreamParms プロパティ

StreamParms プロパティ

同期ストリームの設定パラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Property StreamParms As String
```

```
// C#
    public string  StreamParms {get;set;}
```

プロパティ値

キーワードと値の組み合わせがセミコロンで区切られたリスト形式の、ストリームのパラメータを指定する文字列。デフォルト値は null 参照 (Visual Basic の Nothing) です。

説明

特定のストリームのタイプの設定方法については、『Mobile Link クライアント』>「Ultra Light 同期クライアントのネットワーク・プロトコルのオプション」を参照してください。

StreamParms は、同期ストリームに使用されるすべてのパラメータが含まれている文字列です。パラメータは、名前 = 値の組み合わせをセミコロンで区切ったリスト ("param1=value1;param2=value2") で指定します。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- Stream プロパティ
- ULStreamType 列挙

UploadOnly プロパティ

同期時のダウンロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property UploadOnly As Boolean

// C#
    public bool UploadOnly {get;set;}
```

プロパティ値

ダウンロードを無効にする場合は true、有効にする場合は false。デフォルトは false です。

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、ULSQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- DownloadOnly プロパティ

UserName プロパティ

Mobile Link 同期サーバが Mobile Link クライアントをユニークに識別するユーザ名です。

プロトタイプ

```
' Visual Basic
    Public Property UserName As String

// C#
    public string  UserName {get;set;}
```

プロパティ値

ユーザ名を指定する文字列。このパラメータにはデフォルト値がないので、明示的に設定してください。

説明

Mobile Link では、この値を使用して、ダウンロードする内容の決定、同期ステータスの記録、同期中の割り込みからの復帰を行います。このユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link 同期サーバに対して識別し、認証するために使用されます。

参照

- ULSyncParms クラス
- ULSyncParms メンバ
- Password プロパティ

Version プロパティ

使用する同期スクリプトを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Version As String

// C#
    public string  Version {get;set;}
```

プロパティ値

使用する同期スクリプトのバージョンを指定する文字列。このパラメータにはデフォルト値がないので、明示的に設定してください。

説明

統合データベースの同期スクリプトは、それぞれバージョン文字列で区別されます。たとえば、異なる文字列バージョンによって特定される 2 種類の download_cursor スクリプトがあります。Ultra Light アプリケーションは、バージョン文字列により、同期スクリプトのセットから選択できます。

CopyFrom メソッド

指定した ULSyncParms クラス オブジェクトのプロパティを、この ULSyncParms クラス オブジェクトにコピーします。

プロトタイプ

```
' Visual Basic
    Public Sub CopyFrom( _
      ByVal src As ULSyncParms _
    )

// C#
    public void CopyFrom(
      ULSyncParms src
    );
```

パラメータ

- **src** コピー元のオブジェクト

ULSyncProgressData クラス

UL 拡張：同期の進行状況のモニタリング・データを返します。

プロトタイプ

```
' Visual Basic
    Public Class ULSyncProgressData

// C#
    public class ULSyncProgressData
```

参照

- ULSyncProgressData メンバ
- ULSyncProgressListener インタフェース

ULSyncProgressData メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ErrorMessage プロパティ	同期中に発生したエラーを説明するエラー・メッセージを返します。
ReceivedBytes プロパティ	現在までに受信したバイト数を返します。この情報は、すべてのステータスで更新されます。
ReceivedDeletes プロパティ	現在までに受信した削除済みのローの数を返します。
ReceivedInserts プロパティ	現在までに受信した挿入済みのローの数を返します。
ReceivedUpdates プロパティ	現在までに受信した更新済みのローの数を返します。
SentBytes プロパティ	現在までに送信されたバイト数を返します。この情報は、すべてのステータスで更新されます。
SentDeletes プロパティ	現在までに送信された削除済みのローの数を返します。
SentInserts プロパティ	現在までに送信された挿入済みのローの数を返します。

メンバ	説明
SentUpdates プロパティ	現在までに送信された更新済みのローの数を返します。
SQLCode プロパティ	同期の SQL コードを返します。
State プロパティ	現在の同期のステータスを返します。
SyncParms プロパティ	接続の SyncParms オブジェクトへの参照を返します。
SyncResult プロパティ	接続の SyncResult オブジェクトへの参照を返します。このオブジェクトは、ULSyncProgressState 列挙 イベントおよび ULSyncProgressState 列挙 イベントでのみ更新されます。
TableCount プロパティ	現在までに送信または受信したテーブルの数です (TableTotal の TableCount)。
TableIndex プロパティ	現在同期中のテーブルのインデックスを返します (テーブルには 1 ～ DatabaseSchema.TableCount までの番号が付けられています)。
TableTotal プロパティ	同期中のテーブルの数を返します。

ErrorMessage プロパティ

同期中に発生したエラーを説明するエラー・メッセージを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ErrorMessage As String

// C#
    public string  ErrorMessage {get;}
```

プロパティ値

同期中に発生したエラーを説明する文字列

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙

ReceivedBytes プロパティ

現在までに受信したバイト数を返します。この情報は、すべてのステータスで更新されます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedBytes As Long

// C#
    public long ReceivedBytes {get;}
```

プロパティ値

現在までに受信したバイト数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

ReceivedDeletes プロパティ

現在までに受信した削除済みのローの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedDeletes As Integer

// C#
    public int ReceivedDeletes {get;}
```

プロパティ値

現在までに受信した削除済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

ReceivedInserts プロパティ

現在までに受信した挿入済みのローの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedInserts As Integer

// C#
    public int ReceivedInserts {get;}
```

プロパティ値　現在までに受信した挿入済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

ReceivedUpdates プロパティ

現在までに受信した更新済みのローの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedUpdates As Integer

// C#
    public int ReceivedUpdates {get;}
```

プロパティ値　現在までに受信した更新済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

SentBytes プロパティ

現在までに送信されたバイト数を返します。この情報は、すべてのステータスで更新されます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentBytes As Long
```

```
// C#
    public long SentBytes {get;}
```

プロパティ値　現在までに送信されたバイト数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

SentDeletes プロパティ

現在までに送信された削除済みのローの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentDeletes As Integer

// C#
    public int SentDeletes {get;}
```

プロパティ値　現在までに送信された削除済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

SentInserts プロパティ

現在までに送信された挿入済みのローの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentInserts As Integer

// C#
    public int SentInserts {get;}
```

プロパティ値　現在までに送信された挿入済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

SentUpdates プロパティ

現在までに送信された更新済みのローの数を返します。

プロトタイプ

' Visual Basic

Public Readonly Property **SentUpdates** As Integer

// C#

public int **SentUpdates** {get;}

プロパティ値

現在までに送信された更新済みのローの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

SQLCode プロパティ

同期の SQL コードを返します。

プロトタイプ

' Visual Basic

Public Readonly Property **SQLCode** As ULSQLCode

// C#

public ULSQLCode **SQLCode** {get;}

プロパティ値

同期エラーの ULSQLCode 列挙 値

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙

State プロパティ

現在の同期のステータスを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property State As ULSyncProgressState

// C#
    public ULSyncProgressState State {get;}
```

プロパティ値

現在の同期のステータスを指定する ULSyncProgressState 列挙 値の 1 つ

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙

SyncParms プロパティ

接続の SyncParms オブジェクトへの参照を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncParms As ULSyncParms

// C#
    public ULSyncParms SyncParms {get;}
```

プロパティ値

SyncParms プロパティ オブジェクトへの参照

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙
- SyncParms プロパティ

SyncResult プロパティ

接続の SyncResult オブジェクトへの参照を返します。このオブジェクトは、ULSyncProgressState 列挙 イベントおよび ULSyncProgressState 列挙 イベントでのみ更新されます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncResult As ULSyncResult

// C#
    public ULSyncResult SyncResult {get;}
```

プロパティ値

SyncResult プロパティ オブジェクトへの参照

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

TableCount プロパティ

現在までに送信または受信したテーブルの数です (TableTotal の TableCount)。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableCount As Integer

// C#
    public int TableCount {get;}
```

プロパティ値

送信または受信したテーブルの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

TableIndex プロパティ

現在同期中のテーブルのインデックスを返します (テーブルには 1 ～ DatabaseSchema.TableCount までの番号が付けられています)。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableIndex As Integer

// C#
    public int TableIndex {get;}
```

プロパティ値

現在同期中のテーブルのインデックス。テーブルには、1 ～ TableCount プロパティ までの番号が付けられています。

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

TableTotal プロパティ

同期中のテーブルの数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableTotal As Integer

// C#
    public int TableTotal {get;}
```

プロパティ値

同期中のテーブルの数

参照

- ULSyncProgressData クラス
- ULSyncProgressData メンバ
- ULSyncProgressState 列挙
- ULSyncProgressState 列挙

ULSyncProgressListener インタフェース

UL 拡張：同期プログレスイベントを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ULSyncProgressListener

// C#
    public interface ULSyncProgressListener
```

参照

- ULSyncProgressListener メンバ
- Synchronize メソッド

ULSyncProgressListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
SyncProgressed メソッド	ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

SyncProgressed メソッド

ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

プロトタイプ

```
' Visual Basic
    Public Function SyncProgressed( _
      ByVal data As ULSyncProgressData _
    ) As Boolean
```

```
// C#
    public bool SyncProgressed(
      ULSyncProgressData data
    );
```

パラメータ

- **data** 最新の同期のプログレス・データを保持している ULSyncProgressData クラス オブジェクト

戻り値

このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

説明

SyncProgressed の呼び出し中に、Ultra Light.NET API のメソッドを呼び出さないでください。

ULSyncProgressState 列挙

UL 拡張：同期中に発生する可能性のあるすべてのステータスを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum ULSyncProgressState
     Inherits Short

// C#
    public enum ULSyncProgressState :
     short
```

メンバ

メンバ	説明
STATE_CANCELLED	同期がキャンセルされました。
STATE_COMMITTING_DOWNLOAD	ダウンロードをコミットしています。受信された最終的なロー数が、このイベントに含まれます。ReceivedBytes プロパティ、ReceivedInserts プロパティ、ReceivedUpdates プロパティ、ReceivedDeletes プロパティ を参照してください。
STATE_CONNECTING	同期ストリームは構築されていますが、まだ開かれていません。
STATE_DISCONNECTING	同期ストリームが閉じようとしています。
STATE_DONE	同期が正常に完了しました。接続の SyncResult プロパティ オブジェクトが更新されています。
STATE_ERROR	同期は完了しましたが、エラーが発生しました。詳細については、SyncResult プロパティ、ErrorMessage プロパティ、SQLCode プロパティ を確認してください。

メンバ	説明
STATE_FINISHING_UPLOAD	アップロードが完了しています。送信された最終的なロー数が、このイベントに含まれます。SentBytes プロパティ、SentInserts プロパティ、SentUpdates プロパティ、SentDeletes プロパティ を参照してください。
STATE_LAST	同期の最後のステータスです。このステータスは、常にその他のステータスの後に入力されます。他の ULSyncProgressData フィールドには、有効なデータがないことがあります。
STATE_RECEIVING_DATA	現在のテーブルのデータを受信しています。ReceivedBytes プロパティ、ReceivedInserts プロパティ、ReceivedUpdates プロパティ、ReceivedDeletes プロパティ が更新されています。
STATE_RECEIVING_TABLE	テーブルが受信されています。進行状況をモニタするには、TableIndex プロパティ と TableCount プロパティ を使用します。
STATE_RECEIVING_UPLOAD_ACK	アップロードが完了したという確認が受信されています。
STATE_ROLLING_BACK_DOWNLOAD	ダウンロード中にエラーが発生したため、同期によってダウンロードがロールバックされています。後続の STATE_ERROR 進行状況レポートにエラーがレポートされます。
STATE_SENDING_DATA	現在のテーブルのデータが送信されています。SentBytes プロパティ、SentInserts プロパティ、SentUpdates プロパティ、SentDeletes プロパティ が更新されています。
STATE_SENDING_DOWNLOAD_ACK	ダウンロードが完了したという確認が送信されています。
STATE_SENDING_HEADER	同期ストリームが開かれ、ヘッダが送信されようとしています。
STATE_SENDING_TABLE	テーブルが送信されています。進行状況をモニタするには、TableIndex プロパティ と TableCount プロパティ を使用します。

メンバ	説明
STATE_STARTING	同期処理はまだ行われていません。

参照

- ULSyncProgressData クラス

ULSyncResult クラス

UL 拡張：前回の同期のステータスを表します。

プロトタイプ

```
' Visual Basic
    Public Class ULSyncResult

// C#
    public class ULSyncResult
```

説明

このクラスを直接インスタンス化することはできません。各接続には、SyncResult プロパティとしてアタッチされた、固有の ULSyncResult インスタンスがあります。ULSyncResult インスタンスが有効なのは、接続が開かれている間だけです。

参照

- ULSyncResult メンバ
- SyncResult プロパティ
- Synchronize メソッド

ULSyncResult メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
AuthStatus プロパティ	前回行われた同期の認証ステータス・コードを返します。
AuthValue プロパティ	カスタム・ユーザ認証同期スクリプトからの戻り値を返します。
IgnoredRows プロパティ	前回行われた同期で、アップロードされたローが無視されたかどうかを確認します。
PartialDownloadRetained プロパティ	前回行われた同期で、部分的なダウンロードが保持されたかどうかを確認します。
StreamErrorCode プロパティ	ストリーム自体によってレポートされるエラーを返します。

メンバ	説明
StreamErrorContext プロパティ	ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションを返します。
StreamErrorID プロパティ	エラーをレポートするネットワーク・レイヤの ID を返します。
StreamErrorSystem プロパティ	ストリーム・エラー・システム固有のコードを返します。
Timestamp プロパティ	前回の同期のタイムスタンプを返します。
UploadOK プロパティ	前回のアップロード同期が成功したかどうかをチェックします。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	ULSyncResult がガーベジ・コレクションによって回収される前に、アンマネージ・リソースを解放し、その他のクリーンアップ操作を行います。

AuthStatus プロパティ

前回行われた同期の認証ステータス・コードを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property AuthStatus As ULAuthStatusCode

// C#
    public ULAuthStatusCode AuthStatus {get;}
```

プロパティ値

前回行われた同期の認証ステータスを示す ULAuthStatusCode 列挙 値の 1 つ

AuthValue プロパティ

カスタム・ユーザ認証同期スクリプトからの戻り値を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property AuthValue As Long

// C#
    public long AuthValue {get;}
```

プロパティ値

カスタム・ユーザ認証同期スクリプトから返された long integer

IgnoredRows プロパティ

前回行われた同期で、アップロードされたローが無視されたかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IgnoredRows As Boolean

// C#
    public bool IgnoredRows {get;}
```

プロパティ値

前回の同期中にアップロードされたローが無視された場合は true、ローが無視されなかった場合は false

参照

- ULSyncResult クラス
- ULSyncResult メンバ
- DownloadOnly プロパティ

PartialDownloadRetained プロパティ

前回行われた同期で、部分的なダウンロードが保持されたかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PartialDownloadRetained As Boolean
```

// C#
public bool **PartialDownloadRetained** {get;}

プロパティ値　ダウンロードが中断され、部分的なダウンロードが保持された場合は true、ダウンロードが中断されなかった場合または部分的なダウンロードがロールバックされた場合は false

参照
- ULSyncResult クラス
- ULSyncResult メンバ
- KeepPartialDownload プロパティ

StreamErrorCode プロパティ

ストリーム自体によってレポートされるエラーを返します。

プロトタイプ

' Visual Basic
Public Readonly Property **StreamErrorCode** As ULStreamErrorCode

// C#
public ULStreamErrorCode **StreamErrorCode** {get;}

プロパティ値　ストリーム自体によってレポートされるエラーを示す ULStreamErrorCode 列挙 値の 1 つ。ただし、エラーが発生しなかった場合は ULStreamErrorCode 列挙。

StreamErrorContext プロパティ

ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションを返します。

プロトタイプ

' Visual Basic
Public Readonly Property **StreamErrorContext** As ULStreamErrorContext

// C#
public ULStreamErrorContext **StreamErrorContext** {get;}

プロパティ値　ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションを示す ULStreamErrorContext 列挙 値の 1 つ

StreamErrorID プロパティ

エラーをレポートするネットワーク・レイヤの ID を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorID As ULStreamErrorID

// C#
    public ULStreamErrorID StreamErrorID {get;}
```

プロパティ値 エラーをレポートするネットワーク・レイヤの ID を示す ULStreamErrorID 列挙 値の 1 つ

StreamErrorSystem プロパティ

ストリーム・エラー・システム固有のコードを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorSystem As Integer

// C#
    public int StreamErrorSystem {get;}
```

プロパティ値 ストリーム・エラー・システム固有のコードを示す整数

Timestamp プロパティ

前回の同期のタイムスタンプを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Timestamp As Date

// C#
    public DateTime Timestamp {get;}
```

プロパティ値 前回の同期のタイムスタンプを指定する DateTime

UploadOK プロパティ

前回のアップロード同期が成功したかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UploadOK As Boolean

// C#
    public bool UploadOK {get;}
```

プロパティ値

前回のアップロード同期が成功であった場合は true、不成功であった場合は false

Finalize メソッド

ULSyncResult がガーベジ・コレクションによって回収される前に、アンマネージ・リソースを解放し、その他のクリーンアップ操作を行います。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

ULTable クラス

UL 拡張： Ultra Light データベース内のテーブルを示します。

プロトタイプ

```
' Visual Basic
    Public Class ULTable
     Inherits ULDataReader

// C#
    public class ULTable :
     ULDataReader
```

説明

このクラスを直接インスタンス化することはできません。テーブルは、ULCommand クラスの ExecuteTable メソッドを使用して作成されます。

ULTable メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
Depth プロパティ (継承 ULDataReader)	現在のローのネストの深さを返します。最も外側のテーブルの深さは 0 です。
FieldCount プロパティ (継承 ULDataReader)	このカーソル内のカラム数を返します。
IsBOF プロパティ (継承 ULDataReader)	**UL 拡張：** 現在のローの位置が最初のローの前であるかどうかを確認します。
IsClosed プロパティ (継承 ULDataReader)	カーソルが現在開いているかどうかを確認します。
IsEOF プロパティ (継承 ULDataReader)	**UL 拡張：** 現在のローの位置が最後のローの後であるかどうかを確認します。
Item プロパティ (継承 ULDataReader)	指定したカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。

メンバ	説明
Item プロパティ (継承 ULDataReader)	指定した名前のカラムの値をネイティブ・フォーマットで返します。C# では、このプロパティは ULDataReader クラスのインデクサです。
RecordsAffected プロパティ (継承 ULDataReader)	SQL 文の実行によって変更、挿入、または削除されたローの数を返します。SELECT 文または CommandType.TableDirect テーブルの場合、この値は -1 です。
RowCount プロパティ (継承 ULDataReader)	**UL 拡張：**カーソル内のロー数を返します。
Schema プロパティ	テーブル・スキーマを保持します。このプロパティが有効なのは、接続が開かれている間だけです。

パブリック・インスタンス・メソッド

メンバ	説明
AppendBytes メソッド	指定した Byte 配列の指定したサブセットを、指定した ULDbType 列挙 カラムの新しい値に追加します。
AppendChars メソッド	指定した System.Char 配列の指定したサブセットを、指定した ULDbType 列挙 カラムの新しい値に追加します。
Close メソッド (継承 ULDataReader)	カーソルを閉じます。
Delete メソッド	現在の行を削除します。
DeleteAllRows メソッド	テーブルのすべてのローを削除します。
Dispose メソッド (継承 ULDataReader)	ULDataReader によって使用されるアンマネージ・リソースを解放するほか、必要に応じてマネージ・リソースを解放します。
FindBegin メソッド	テーブルで新規に検索を実行する準備を行います。

メンバ	説明
FindFirst メソッド	テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセット全体に完全に一致するローを検索します。
FindFirst メソッド	テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットの一部に完全に一致するローを検索します。
FindLast メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に完全に一致するローを検索します。
FindLast メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に完全に一致するローを検索します。
FindNext メソッド	現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。
FindNext メソッド	現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。
FindPrevious メソッド	現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。
FindPrevious メソッド	現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。
GetBoolean メソッド (継承 ULDataReader)	指定したカラムの値を Boolean として返します。
GetByte メソッド (継承 ULDataReader)	指定したカラムの値を符号なし 8 ビット値 (Byte) として返します。

メンバ	説明
GetBytes メソッド (継承 ULDataReader)	指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。
GetBytes メソッド (継承 ULDataReader)	**UL 拡張：**指定したカラムの値を Byte 配列として返します。ULDbType 列挙 型、ULDbType 列挙 型、ULDbType 列挙 型のカラムの場合にのみ有効です。
GetChar メソッド (継承 ULDataReader)	このメソッドは Ultra Light.NET ではサポートされていません。
GetChars メソッド (継承 ULDataReader)	指定したオフセットで始まる、指定した ULDbType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。
GetData メソッド (継承 ULDataReader)	このメソッドは Ultra Light.NET ではサポートされていません。
GetDataTypeName メソッド (継承 ULDataReader)	指定したカラムのプロバイダのデータ型の名前を返します。
GetDateTime メソッド (継承 ULDataReader)	指定したカラムの値を、ミリ秒の精度の DateTime として返します。
GetDecimal メソッド (継承 ULDataReader)	指定したカラムの値を Decimal として返します。
GetDouble メソッド (継承 ULDataReader)	指定したカラムの値を Double として返します。
GetFieldType メソッド (継承 ULDataReader)	指定したカラムに最適な Type を返します。
GetFloat メソッド (継承 ULDataReader)	指定したカラムの値を Single として返します。
GetGuid メソッド (継承 ULDataReader)	指定したカラムの値を UUID (Guid) として返します。

メンバ	説明
GetInt16 メソッド (継承 ULDataReader)	指定したカラムの値を Int16 として返します。
GetInt32 メソッド (継承 ULDataReader)	指定したカラムの値を Int32 として返します。
GetInt64 メソッド (継承 ULDataReader)	指定したカラムの値を Int64 として返します。
GetName メソッド (継承 ULDataReader)	指定したカラムの名前を返します。
GetOrdinal メソッド (継承 ULDataReader)	指定したカラムのカラム ID を返します。
GetSchemaTable メソッド (継承 ULDataReader)	ULDataReader のカラムのメタデータが記述された DataTable を返します。
GetString メソッド (継承 ULDataReader)	指定したカラムの値を String として返します。
GetTimeSpan メソッド (継承 ULDataReader)	指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。
GetUInt16 メソッド (継承 ULDataReader)	指定したカラムの値を UInt16 として返します。
GetUInt32 メソッド (継承 ULDataReader)	指定したカラムの値を UInt32 として返します。
GetUInt64 メソッド (継承 ULDataReader)	指定したカラムの値を UInt64 として返します。
GetValue メソッド (継承 ULDataReader)	指定したカラムの値をネイティブ・フォーマットで返します。
GetValues メソッド (継承 ULDataReader)	現在のローのすべてのカラム値を返します。

メンバ	説明
Insert メソッド	現在のカラム値 (set メソッドを使用して指定されます) で新しいローを挿入します。 挿入を行う前に InsertBegin メソッド を呼び出してください。
InsertBegin メソッド	現在のすべてのカラムをデフォルト値に設定して、テーブルに新しいローを挿入する準備を行います。
IsDBNull メソッド (継承 ULDataReader)	指定されたカラムの値が NULL かどうかをチェックします。
LookupBackward メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より小さい値を持つローを検索します。
LookupBackward メソッド	テーブルを最初から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より小さい値を持つローを検索します。
LookupBegin メソッド	テーブルで新規に検索を実行する準備を行います。検索する値は、テーブルを開いたインデックス内のカラムで適切な setType メソッドを呼び出して指定します。
LookupForward メソッド	テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より大きい値を持つローを検索します。
LookupForward メソッド	テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より大きい値を持つローを検索します。
MoveAfterLast メソッド (継承 ULDataReader)	**UL 拡張：**カーソルの最後のローの後に、カーソルを配置します。
MoveBeforeFirst メソッド (継承 ULDataReader)	**UL 拡張：**カーソルの最初のローの前に、カーソルを配置します。

メンバ	説明
MoveFirst メソッド (継承 ULDataReader)	**UL 拡張 :** カーソルの最初のローに、カーソルを配置します。
MoveLast メソッド (継承 ULDataReader)	**UL 拡張 :** カーソルの最後のローに、カーソルを配置します。
MoveNext メソッド (継承 ULDataReader)	**UL 拡張 :** カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。
MovePrevious メソッド (継承 ULDataReader)	**UL 拡張 :** カーソルを前のローに配置するか、最初のローの前に配置します。
MoveRelative メソッド (継承 ULDataReader)	**UL 拡張 :** 現在のローを基準としてカーソルを配置します。
NextResult メソッド (継承 ULDataReader)	バッチ SQL 文の結果を読み込むときに ULDataReader を次の結果に進めます。
Read メソッド (継承 ULDataReader)	カーソルを次のローに配置します。ただし、カーソルがすでに最後のローにある場合は最後のローの後に配置します。
SetBoolean メソッド	指定したカラムの値を、Boolean を使用して設定します。
SetByte メソッド	指定したカラムの値を、Byte (符号なし 8 ビット整数) を使用して設定します。
SetBytes メソッド	指定したカラムの値を、Byte 配列を使用して設定します。
SetDateTime メソッド	指定したカラムの値を、DateTime を使用して設定します。
SetDBNull メソッド	カラムを NULL に設定します。
SetDecimal メソッド	指定したカラムの値を、Decimal を使用して設定します。
SetDouble メソッド	指定したカラムの値を、Double を使用して設定します。

メンバ	説明
SetFloat メソッド	指定したカラムの値を、Single を使用して設定します。
SetGuid メソッド	指定したカラムの値を、Guid を使用して設定します。
SetInt16 メソッド	指定したカラムの値を、Int16 を使用して設定します。
SetInt32 メソッド	指定したカラムの値を、Int32 を使用して設定します。
SetInt64 メソッド	指定したカラムの値を、Int64 を使用して設定します。
SetString メソッド	指定したカラムの値を、String を使用して設定します。
SetTimeSpan メソッド	指定したカラムの値を、TimeSpan を使用して設定します。
SetToDefault メソッド	指定したカラムの値を、そのデフォルト値に設定します。
SetUInt16 メソッド	指定したカラムの値を、UInt16 を使用して設定します。
SetUInt32 メソッド	指定したカラムの値を、UInt32 を使用して設定します。
SetUInt64 メソッド	指定したカラムの値を、UInt64 を使用して設定します。
Truncate メソッド	テーブル内のすべてのローを削除し、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。
Update メソッド	現在のカラム値 (set メソッドを使用して指定されます) で新しいローを更新します。 更新を行う前に UpdateBegin メソッド を呼び出してください。
UpdateBegin メソッド	テーブルの現在のローを更新する準備を行います。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	ULTable がガーベジ・コレクションによって回収される前に、アンマネージ・リソースを解放し、その他のクリーンアップ操作を行います。

Schema プロパティ

テーブル・スキーマを保持します。このプロパティが有効なのは、接続が開かれている間だけです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Schema As ULTableSchema

// C#
    public ULTableSchema Schema {get;}
```

プロパティ値

テーブル・スキーマを示す ULTableSchema クラス オブジェクト

説明

このプロパティは、GetSchemaTable メソッド からの結果に示されない Ultra Light.NET の詳細情報を含め、テーブルの完全なスキーマを示します。

AppendBytes メソッド

指定した Byte 配列の指定したサブセットを、指定した ULDbType 列挙 カラムの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendBytes( _
      ByVal columnID As Integer, _
      ByVal val As Byte(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )
```

```
// C#
    public void AppendBytes(
      int columnID,
      byte[] val,
      int srcOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの現在の新しい値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置のバイトが、指定したカラムの値に追加されます。

挿入時には、InsertBegin メソッド は新しい値をカラムのデフォルト値に初期化します。ローのデータは、Insert メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

更新時の、カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード ULSQLCode 列挙 とともに ULException クラス がスローされ、処理対象は修正されません。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *val*.Length よりも大きい。

その他のエラーの場合は、それに応じたエラー・コードとともに ULException クラス がスローされます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

AppendChars メソッド

指定した System.Char 配列の指定したサブセットを、指定した ULDbType 列挙 カラムの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendChars( _
      ByVal columnID As Integer, _
      ByVal val As Char(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )

// C#
    public void AppendChars(
      int columnID,
      char[] val,
      int srcOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの現在の新しい値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置の文字が、指定したカラムの値に追加されます。挿入時には、InsertBegin メソッド は新しい値をカラムのデフォルト値に初期化します。ローのデータは、Insert メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

更新時の、カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード ULSQLCode 列挙 とともに ULException クラス がスローされ、コピー先は修正されません。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *value*.Length よりも大きい。

その他のエラーの場合は、それに応じたエラー・コードとともに ULException クラス がスローされます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

Delete メソッド

現在の行を削除します。

プロトタイプ

```
' Visual Basic
    Public Sub Delete()
```

```
// C#
    public void Delete();
```

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- StartSynchronizationDelete メソッド
- StopSynchronizationDelete メソッド

DeleteAllRows メソッド

テーブルのすべてのローを削除します。

プロトタイプ

```
' Visual Basic
    Public Sub DeleteAllRows()

// C#
    public void DeleteAllRows();
```

説明

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータ・セットをテーブルにダウンロードするほうが便利なことがあります。StopSynchronizationDelete メソッド を使用すると、統合データベースからは削除しないで Ultra Light データベースからローを削除できます。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- Truncate メソッド

Finalize メソッド

ULTable がガーベジ・コレクションによって回収される前に、アンマネージ・リソースを解放し、その他のクリーンアップ操作を行います。

プロトタイプ

' Visual Basic
Overrides Protected Sub **Finalize()**

// C#
protected override void **Finalize();**

FindBegin メソッド

テーブルで新規に検索を実行する準備を行います。

プロトタイプ

' Visual Basic
Public Sub **FindBegin()**

// C#
public void **FindBegin();**

説明

検索する値は、テーブルを開いたインデックス内のカラムで適切な setType メソッドを呼び出して指定します。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindFirst メソッド
- FindFirst メソッド
- FindLast メソッド
- FindLast メソッド

FindFirst メソッド

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセット全体に完全に一致するローを検索します。

プロトタイプ

' Visual Basic
Overloads Public Function **FindFirst()** As Boolean

// C#
public bool **FindFirst();**

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド
- FindFirst メソッド

FindFirst メソッド

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットの一部に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindFirst( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindFirst(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド
- FindFirst メソッド

FindLast メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindLast() As Boolean

// C#
    public bool FindLast();
```

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ

- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド
- FindLast メソッド

FindLast メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindLast( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindLast(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindBegin メソッド
- FindNext メソッド

- FindPrevious メソッド
- FindLast メソッド

FindNext メソッド

現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。

プロトタイプ

' Visual Basic
Overloads Public Function **FindNext()** As Boolean

// C#
public bool **FindNext();**

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは次のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の FindNext の動作は不確定です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindFirst メソッド
- FindNext メソッド

FindNext メソッド

現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindNext( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindNext(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは次のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の FindNext の動作は不確定です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindFirst メソッド
- FindNext メソッド

FindPrevious メソッド

現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindPrevious() As Boolean
```

```
// C#
    public bool FindPrevious();
```

戻り値　成功した場合は true、失敗した場合は false

説明　インデックスの値と完全に一致すると、カーソルは前のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の FindPrevious の動作は不確定です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindLast メソッド
- FindPrevious メソッド

FindPrevious メソッド

現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindPrevious( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindPrevious(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは前のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の FindPrevious の動作は不確定です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- FindLast メソッド
- FindPrevious メソッド

Insert メソッド

現在のカラム値 (set メソッドを使用して指定されます) で新しいローを挿入します。

挿入を行う前に InsertBegin メソッド を呼び出してください。

プロトタイプ

```
' Visual Basic
    Public Sub Insert()

// C#
    public void Insert();
```

例外

- ULException クラス - SQL エラーが発生しました。

InsertBegin メソッド

現在のすべてのカラムをデフォルト値に設定して、テーブルに新しいローを挿入する準備を行います。

プロトタイプ

```
' Visual Basic
    Public Sub InsertBegin()
```

// C#
public void **InsertBegin();**

説明

適切な SetType メソッドまたは AppendType メソッドを呼び出して、挿入するデフォルト以外の値を指定します。

Insert メソッド が実行されないと、ローが実際に挿入されることも、ロー内のデータが実際に変更されることもありません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- Insert メソッド

LookupBackward メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より小さい値を持つローを検索します。

プロトタイプ

' Visual Basic
Overloads Public Function **LookupBackward()** As Boolean

// C#
public bool **LookupBackward();**

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致する最初のローか、それより少ない値の最初のローで停止します。失敗した場合 (検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- LookupBegin メソッド
- LookupBackward メソッド

LookupBackward メソッド

テーブルを最初から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より小さい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupBackward( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool LookupBackward(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、ルックアップで使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致する最初のローか、それより少ない値の最初のローで停止します。失敗した場合 (検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス

- ULTable メンバ
- LookupBegin メソッド
- LookupBackward メソッド

LookupBegin メソッド

テーブルで新規に検索を実行する準備を行います。検索する値は、テーブルを開いたインデックス内のカラムで適切な setType メソッドを呼び出して指定します。

プロトタイプ

```
' Visual Basic
    Public Sub LookupBegin()

// C#
    public void LookupBegin();
```

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- LookupForward メソッド
- LookupForward メソッド
- LookupBackward メソッド
- LookupBackward メソッド

LookupForward メソッド

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より大きい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupForward() As Boolean

// C#
    public bool LookupForward();
```

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。失敗した場合 (検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- LookupBegin メソッド
- LookupForward メソッド

LookupForward メソッド

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より大きい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupForward( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool LookupForward(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、ルックアップで使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定してから、値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。失敗した場合 (検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- LookupBegin メソッド
- LookupForward メソッド

SetBoolean メソッド

指定したカラムの値を、Boolean を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBoolean( _
      ByVal columnID As Integer, _
      ByVal val As Boolean _
    )

// C#
    public void SetBoolean(
      int columnID,
      bool val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetByte メソッド

指定したカラムの値を、Byte (符号なし 8 ビット整数) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetByte( _
      ByVal columnID As Integer, _
      ByVal val As Byte _
    )

// C#
    public void SetByte(
      int columnID,
      byte val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetBytes メソッド

指定したカラムの値を、Byte 配列を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBytes( _
      ByVal columnID As Integer, _
      ByVal val As Byte() _
    )

// C#
    public void SetBytes(
      int columnID,
      byte[] val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ULDbType 列挙 型または ULDbType 列挙 型のカラム、あるいは val の長さが 16 の場合の ULDbType 列挙 型のカラムにのみ適しています。ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetDateTime メソッド

指定したカラムの値を、DateTime を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDateTime( _
      ByVal columnID As Integer, _
      ByVal val As Date _
    )

// C#
    public void SetDateTime(
      int columnID,
      DateTime val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

設定された値の精度はミリ秒です。ローのデータは、Insert メソッドまたは Update メソッドが実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ

- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetDBNull メソッド

カラムを NULL に設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDBNull( _
      ByVal columnID As Integer _
    )

// C#
    public void SetDBNull(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

説明

データは、Insert メソッド または Update メソッド を実行するまでは、実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド

- Schema プロパティ
- IsColumnNullable メソッド

SetDecimal メソッド

指定したカラムの値を、Decimal を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDecimal( _
      ByVal columnID As Integer, _
      ByVal val As Decimal _
    )

// C#
    public void SetDecimal(
      int columnID,
      decimal val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetDouble メソッド

指定したカラムの値を、Double を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDouble( _
      ByVal columnID As Integer, _
      ByVal val As Double _
    )

// C#
    public void SetDouble(
      int columnID,
      double val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetFloat メソッド

指定したカラムの値を、Single を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetFloat( _
      ByVal columnID As Integer, _
      ByVal val As Single _
    )

// C#
    public void SetFloat(
      int columnID,
      float  val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetGuid メソッド

指定したカラムの値を、Guid を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetGuid( _
      ByVal columnID As Integer, _
      ByVal val As Guid _
    )

// C#
    public void SetGuid(
      int columnID,
      Guid val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。ULDbType 列挙 型のカラム、または長さが 16 の ULDbType 列挙 型のカラムの場合にのみ有効です。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetNewUUID メソッド
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド
- GetColumnSize メソッド

SetInt16 メソッド

指定したカラムの値を、Int16 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetInt16( _
      ByVal columnID As Integer, _
      ByVal val As Short _
    )

// C#
    public void SetInt16(
      int columnID,
      short val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetInt32 メソッド

指定したカラムの値を、Int32 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetInt32( _
      ByVal columnID As Integer, _
      ByVal val As Integer _
    )

// C#
    public void SetInt32(
      int columnID,
      int val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetInt64 メソッド

指定したカラムの値を、Int64 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetInt64( _
      ByVal columnID As Integer, _
      ByVal val As Long _
    )

// C#
    public void SetInt64(
      int columnID,
      long val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetString メソッド

指定したカラムの値を、String を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetString( _
      ByVal columnID As Integer, _
      ByVal val As String _
    )

// C#
    public void SetString(
      int columnID,
      string  val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetTimeSpan メソッド

指定したカラムの値を、TimeSpan を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetTimeSpan( _
      ByVal columnID As Integer, _
      ByVal val As TimeSpan _
    )

// C#
    public void SetTimeSpan(
      int columnID,
      TimeSpan val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

設定された値は、ミリ秒の精度であり、0 ～ 24 時間の負でない値に正規化されます。ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetToDefault メソッド

指定したカラムの値を、そのデフォルト値に設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetToDefault( _
      ByVal columnID As Integer _
    )

// C#
    public void SetToDefault(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド
- GetColumnDefaultValue メソッド

SetUInt16 メソッド

指定したカラムの値を、UInt16 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUInt16( _
      ByVal columnID As Integer, _
      ByVal val As UInt16 _
    )

// C#
    public void SetUInt16(
      int columnID,
      ushort val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetUInt32 メソッド

指定したカラムの値を、UInt32 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUInt32( _
      ByVal columnID As Integer, _
      ByVal val As UInt32 _
    )

// C#
    public void SetUInt32(
      int columnID,
      uint val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

SetUInt64 メソッド

指定したカラムの値を、UInt64 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUInt64( _
      ByVal columnID As Integer, _
      ByVal val As UInt64 _
    )

// C#
    public void SetUInt64(
      int columnID,
      ulong val
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,FieldCount プロパティ - 1] の範囲内である必要があります。カーソルの最初のカラムの ID 値はゼロです。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- GetOrdinal メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetFieldType メソッド

Truncate メソッド

テーブル内のすべてのローを削除し、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。

プロトタイプ

```
' Visual Basic
    Public Sub Truncate()

// C#
    public void Truncate();
```

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTable クラス
- ULTable メンバ
- DeleteAllRows メソッド

Update メソッド

現在のカラム値 (set メソッドを使用して指定されます) で新しいローを更新します。

更新を行う前に UpdateBegin メソッド を呼び出してください。

プロトタイプ

```
' Visual Basic
    Public Sub Update()

// C#
    public void Update();
```

例外

- ULException クラス - SQL エラーが発生しました。

UpdateBegin メソッド

テーブルの現在のローを更新する準備を行います。

プロトタイプ

```
' Visual Basic
    Public Sub UpdateBegin()

// C#
    public void UpdateBegin();
```

説明

カラム値は、適切な setType メソッドまたは AppendType メソッドを呼び出すことによって修正します。カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

ローのデータは、Update メソッド が呼び出されるまで実際には変更されません。また、その変更は、コミットされないかぎり、永続化されません。

テーブルを開くのに使用されるインデックス内のカラムを修正すると、アクティブな検索処理に予期しない影響を及ぼします。テーブルのプライマリ・キー内のカラムは更新できません。

例外

- ULException クラス - SQL エラーが発生しました。

ULTableSchema クラス

UL 拡張： Ultra Light テーブルのスキーマを示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULTableSchema
      Inherits ULCursorSchema

// C#
    public sealed class ULTableSchema :
      ULCursorSchema
```

説明

このクラスを直接インスタンス化することはできません。ULTableSchema クラス オブジェクトは、その Schema プロパティ としてテーブルにアタッチされます。

ULTableSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ (継承 ULCursorSchema)	このカーソル内のカラム数を返します。
IndexCount プロパティ	テーブルのインデックス数を返します。
IsNeverSynchronized プロパティ	テーブルが、まったく同期されないようにマーク付けされているかどうかをチェックします。
IsOpen プロパティ (継承 ULCursorSchema)	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	テーブルの名前を返します。
PrimaryKey プロパティ	テーブルのプライマリ・キーのインデックス・スキーマを返します。

メンバ	説明
UploadUnchangedRows プロパティ	データベースが、変更されていないローをアップロードするかどうかをチェックします。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnDefaultValue メソッド	指定したカラムのデフォルト値を返します。
GetColumnID メソッド (継承 ULCursorSchema)	指定したカラムのカラム ID を返します。
GetColumnName メソッド (継承 ULCursorSchema)	指定したカラム ID で識別されたカラムの名前を返します。
GetColumnPartitionSize メソッド	指定したカラムに割り当てられているグローバル・オートインクリメントの分割サイズを返します。
GetColumnPrecision メソッド (継承 ULCursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの精度を返します。
GetColumnScale メソッド (継承 ULCursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラム ID によって識別されたカラムの位取りを返します。
GetColumnSize メソッド (継承 ULCursorSchema)	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラム ID によって識別されたカラムのサイズを返します。
GetColumnULDbType メソッド (継承 ULCursorSchema)	指定したカラム ID によって識別されたカラムの Ultra Light.NET データ型を返します。
GetIndex メソッド	指定したインデックスのインデックス・スキーマを返します。

メンバ	説明
GetIndexName メソッド	指定したインデックス ID で識別されたインデックスの名前を返します。
GetOptimalIndex メソッド	指定したカラムを使用してテーブルを検索するために最適なインデックスです。
GetSchemaTable メソッド (継承 ULCursorSchema)	ULDataReader クラス のカラムのスキーマが記述された DataTable を返します。
IsColumnAutoIncrement メソッド	指定したカラムのデフォルトがオートインクリメントに設定されているかどうかをチェックします。
IsColumnCurrentDate メソッド	指定したカラムのデフォルトが、現在の日付 (ULDbType 列挙) に設定されているかどうかをチェックします。
IsColumnCurrentTime メソッド	指定したカラムのデフォルトが、現在の時刻 (ULDbType 列挙) に設定されているかどうかをチェックします。
IsColumnCurrentTimestamp メソッド	指定したカラムのデフォルトが、現在のタイムスタンプ (ULDbType 列挙) に設定されているかどうかをチェックします。
IsColumnGlobalAutoIncrement メソッド	指定したカラムのデフォルトがグローバル・オートインクリメントに設定されているかどうかをチェックします。
IsColumnNewUUID メソッド	指定したカラムのデフォルトが新しい UUID (Guid) に設定されているかどうかをチェックします。
IsColumnNullable メソッド	指定したカラムが null 入力可能であるかどうかをチェックします。
IsInPublication メソッド	テーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

IndexCount プロパティ

テーブルのインデックス数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IndexCount As Integer

// C#
    public int IndexCount {get;}
```

プロパティ値

テーブルのインデックスの数、またはテーブル・スキーマが閉じている場合は 0

説明

インデックス ID の範囲は、1 ～ IndexCount です。

注意：インデックスの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

IsNeverSynchronized プロパティ

テーブルが、まったく同期されないようにマーク付けされているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsNeverSynchronized As Boolean

// C#
    public bool IsNeverSynchronized {get;}
```

プロパティ値

テーブルが、まったく同期されないようにマーク付けされている場合は true、そうでない場合は false

説明

まったく同期されないようにマーク付けされているテーブルは、パブリケーションに含まれているものであっても、まったく同期されていません。このようなテーブルは、「未同期」テーブルと呼ばれることもあります。

例外

- ULException クラス - SQL エラーが発生しました。

Name プロパティ

テーブルの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

プロパティ値　文字列としてのテーブル名

例外

- ULException クラス - SQL エラーが発生しました。

PrimaryKey プロパティ

テーブルのプライマリ・キーのインデックス・スキーマを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PrimaryKey As ULIndexSchema

// C#
    public ULIndexSchema PrimaryKey {get;}
```

プロパティ値　テーブルのプライマリ・キーを示す ULIndexSchema クラス オブジェクト

例外

- ULException クラス - SQL エラーが発生しました。

UploadUnchangedRows プロパティ

データベースが、変更されていないローをアップロードするかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UploadUnchangedRows As Boolean

// C#
    public bool UploadUnchangedRows {get;}
```

プロパティ値

テーブルが、同期時に常にすべてのローをアップロードするようにマーク付けされている場合は true、変更されたローのみをアップロードするようにマーク付けされている場合は false

説明

すべてのローをアップロードするようにマーク付けされているテーブルでは、その同期時に、変更されたローと未変更のローがアップロードされます。このようなテーブルは、「完全同期」テーブルと呼ばれることもあります。

例外

- ULException クラス - SQL エラーが発生しました。

GetColumnDefaultValue メソッド

指定したカラムのデフォルト値を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnDefaultValue( _
      ByVal columnID As Integer _
    ) As String

// C#
    public string  GetColumnDefaultValue(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

指定したカラムの文字列としてのデフォルト値、またはデフォルト値が null の場合は null 参照 (Visual Basic の Nothing)

例外

- ULException クラス - SQL エラーが発生しました。

GetColumnPartitionSize メソッド

指定したカラムに割り当てられているグローバル・オートインクリメントの分割サイズを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnPartitionSize( _
      ByVal columnID As Integer _
    ) As UInt64

// C#
    public ulong GetColumnPartitionSize(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムのグローバル・オートインクリメントの分割サイズ (UInt64)

説明

テーブルのすべてのグローバル・オートインクリメント・カラムは、同じグローバル・オートインクリメントの分割サイズを共有します。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTableSchema クラス
- ULTableSchema メンバ
- IsColumnGlobalAutoIncrement メソッド

GetIndex メソッド

指定したインデックスのインデックス・スキーマを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetIndex( _
      ByVal name As String _
    ) As ULIndexSchema

// C#
    public ULIndexSchema GetIndex(
      string  name
    );
```

パラメータ

- **name** インデックス名

戻り値 指定したインデックスを示す ULIndexSchema クラス オブジェクト

例外

- ULException クラス - SQL エラーが発生しました。

GetIndexName メソッド

指定したインデックス ID で識別されたインデックスの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetIndexName( _
      ByVal indexID As Integer _
    ) As String

// C#
    public string  GetIndexName(
      int indexID
    );
```

パラメータ

- **indexID** インデックスの ID。値は、[1,IndexCount プロパティ] の範囲内である必要があります。

戻り値 文字列としてのインデックス名

説明 インデックスの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTableSchema クラス
- ULTableSchema メンバ
- IndexCount プロパティ

GetOptimalIndex メソッド

指定したカラムを使用してテーブルを検索するために最適なインデックスです。

プロトタイプ

```
' Visual Basic
    Public Function GetOptimalIndex( _
      ByVal columnID As Integer _
    ) As ULIndexSchema

// C#
    public ULIndexSchema GetOptimalIndex(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

指定したカラムの最適なインデックスを示す ULIndexSchema クラスオブジェクト

説明

指定したカラムは、インデックス内の最初のカラムですが、インデックスには複数のカラムがある場合があります。

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnAutoIncrement メソッド

指定したカラムのデフォルトがオートインクリメントに設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnAutoIncrement( _
      ByVal columnID As Integer _
    ) As Boolean

// C#
    public bool IsColumnAutoIncrement(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムがオートインクリメントされる場合は true、オートインクリメントされない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnCurrentDate メソッド

指定したカラムのデフォルトが、現在の日付 (ULDbType 列挙) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentDate( _
      ByVal columnID As Integer _
    ) As Boolean

// C#
    public bool IsColumnCurrentDate(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムのデフォルトが現在の日付に設定されている場合は true、現在の日付に設定されていない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnCurrentTime メソッド

指定したカラムのデフォルトが、現在の時刻 (ULDbType 列挙) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentTime( _
      ByVal columnID As Integer _
    ) As Boolean
```

```
// C#
    public bool IsColumnCurrentTime(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムのデフォルトが現在の時刻に設定されている場合は true、現在の時刻に設定されていない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnCurrentTimestamp メソッド

指定したカラムのデフォルトが、現在のタイムスタンプ (ULDbType 列挙) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentTimestamp( _
      ByVal columnID As Integer _
    ) As Boolean

// C#
    public bool IsColumnCurrentTimestamp(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムのデフォルトが現在のタイムスタンプに設定されている場合は true、現在のタイムスタンプに設定されていない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnGlobalAutoIncrement メソッド

指定したカラムのデフォルトがグローバル・オートインクリメントに設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnGlobalAutoIncrement( _
      ByVal columnID As Integer _
    ) As Boolean

// C#
    public bool IsColumnGlobalAutoIncrement(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムがグローバル・オートインクリメントされる場合は true、グローバル・オートインクリメントされない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

参照

- ULTableSchema クラス
- ULTableSchema メンバ
- GetColumnPartitionSize メソッド
- DatabaseID プロパティ

IsColumnNewUUID メソッド

指定したカラムのデフォルトが新しい UUID (Guid) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnNewUUID( _
      ByVal columnID As Integer _
    ) As Boolean
```

```
// C#
    public bool IsColumnNewUUID(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムのデフォルトが新しい UUID に設定されている場合は true、新しい UUID に設定されていない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

IsColumnNullable メソッド

指定したカラムが null 入力可能であるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnNullable( _
      ByVal columnID As Integer _
    ) As Boolean

// C#
    public bool IsColumnNullable(
      int columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。値は、[0,ColumnCount プロパティ -1] の範囲内である必要があります。テーブルの最初のカラムの ID 値はゼロです。

戻り値

カラムが null 入力可能であれば true、null 入力が不可能であれば false

例外

- ULException クラス - SQL エラーが発生しました。

IsInPublication メソッド

テーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsInPublication( _
      ByVal pubName As String _
    ) As Boolean

// C#
    public bool IsInPublication(
      string  pubName
    );
```

パラメータ

- **pubName** パブリケーションの名前

戻り値

テーブルがパブリケーション内にある場合は true、パブリケーション内にない場合は false

例外

- ULException クラス - SQL エラーが発生しました。

ULTransaction クラス

SQL トランザクションを示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class ULTransaction

// C#
    public sealed class ULTransaction
```

説明

ULTransaction にはコンストラクタがありません。ULTransaction オブジェクトを取得するには、BeginTransaction メソッドを使用します。コマンドをトランザクションに関連付けるには、Transaction プロパティを使用します。

トランザクションがコミットまたはロールバックされると、接続はすべての操作の実行時にそれらを自動的にコミットする状態に戻ります。さらに操作をグループ化するには、新しいトランザクションを作成する必要があります。

実装：IDbTransaction、IDisposable

ULTransaction メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
Connection プロパティ	トランザクションに対応する接続を返します。
IsolationLevel プロパティ	トランザクションの独立性レベルを返します。

パブリック・インスタンス・メソッド

メンバ	説明
Commit メソッド	データベース・トランザクションをコミットします。
Dispose メソッド	ULTransaction によって使用されたリソースを解放し、コミットされていないコマンドをロールバックします。
Rollback メソッド	トランザクションの未処理の変更をデータベースにロールバックします。

Connection プロパティ

トランザクションに対応する接続を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Connection As ULConnection

// C#
    public ULConnection Connection {get;}
```

プロパティ値

トランザクションに関連付けられている ULConnection クラス オブジェクト。トランザクションが無効な場合は null 参照 (Visual Basic の Nothing)。

説明

これは、IDbTransaction.Connection が厳密に型指定されたものです。

参照

- ULTransaction クラス
- ULTransaction メンバ
- BeginTransaction メソッド

IsolationLevel プロパティ

トランザクションの独立性レベルを返します。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Readonly Property IsolationLevel As IsolationLevel
    _
      Implements IDbTransaction.IsolationLevel

// C#
    public IsolationLevel IsolationLevel {get;}
```

プロパティ値

IsolationLevel 値の 1 つ。Ultra Light.NET では、IsolationLevel.ReadUncommitted のみがサポートされています。

実装

IDbTransaction.IsolationLevel

参照

- ULTransaction クラス
- ULTransaction メンバ
- BeginTransaction メソッド

Commit メソッド

データベース・トランザクションをコミットします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Commit() _
      Implements IDbTransaction.Commit

// C#
    public void Commit();
```

説明

トランザクションがコミットまたはロールバックされると、接続はすべての操作の実行時にそれらを自動的にコミットする状態に戻ります。さらに操作をグループ化するには、新しいトランザクションを作成する必要があります。

実装

IDbTransaction.Commit

参照

- ULTransaction クラス
- ULTransaction メンバ
- Rollback メソッド

Dispose メソッド

ULTransaction によって使用されたリソースを解放し、コミットされていないコマンドをロールバックします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Dispose() _
      Implements IDisposable.Dispose

// C#
    public void Dispose();
```

実装

IDisposable.Dispose

Rollback メソッド

トランザクションの未処理の変更をデータベースにロールバックします。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Rollback() _
      Implements IDbTransaction.Rollback

// C#
    public void Rollback();
```

説明

トランザクションがコミットまたはロールバックされると、接続はすべての操作の実行時にそれらを自動的にコミットする状態に戻ります。さらに操作をグループ化するには、新しいトランザクションを作成する必要があります。

実装

IDbTransaction.Rollback

参照

- ULTransaction クラス
- ULTransaction メンバ
- Commit メソッド

第 5 章

iAnywhere.UltraLite ネームスペース

この章について

iAnywhere.UltraLite ネームスペースは、リリース 9.0.2 では使用されなくなりました。また、SQL Anywhere の次のメジャー・リリースでも利用できません。詳細については、『Ultra Light.NET ユーザーズ・ガイド』> 「iAnywhere.Data.UltraLite ネームスペース」を参照してください。

iAnywhere.UltraLite ネームスペースには、Ultra Light.NET クラス、インタフェース、列挙があり、これらを使用して、Ultra Light データベースを使用するアプリケーション用の C# または Visual Basic.NET コードを記述できます。

この Ultra Light.NET 用 API リファレンスは、サポートされる機能の完全な参照となります。各トピックには特定のクラス、インタフェース、または列挙に関する情報が記載されています。

アプリケーション・プログラムはまず DatabaseManager クラス のインスタンスを 1 つ生成してからでなければ、ほかの Ultra Light.NET クラスを使用できません。

iAnywhere.UltraLite アセンブリは **iAnywhere.UltraLite.resources** という名前のサテライト・リソース・アセンブリを使用します。メイン・アセンブリは、このリソース・アセンブリをカルチャで検索します。検索する順序は、初めに System.Globalization.CultureInfo.CurrentUICulture、次に System.Globalization.CultureInfo.CurrentCulture、最後にカルチャ "EN" の順です。

ActiveSyncListener インタフェース

ActiveSync イベントを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ActiveSyncListener

// C#
    public interface ActiveSyncListener
```

ActiveSyncListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
ActiveSyncInvoked メソッド	ActiveSync 用の Mobile Link プロバイダがアプリケーションを呼び出して同期を実行するときに起動されます。

ActiveSyncInvoked メソッド

ActiveSync 用の Mobile Link プロバイダがアプリケーションを呼び出して同期を実行するときに起動されます。

プロトタイプ

```
' Visual Basic
    Public Sub ActiveSyncInvoked( _
    ByVal launchedByProvider As Boolean _
    )

// C#
    public void ActiveSyncInvoked(
    bool launchedByProvider
    );
```

パラメータ

- **launchedByProvider** ActiveSync 同期を実行するために、Mobile Link プロバイダがアプリケーションを起動した場合は true です。同期の完了後、アプリケーションは自動的に停止

しなければなりません。ActiveSync 用 Mobile Link プロバイダが呼び出したときに、すでにアプリケーションが実行されていた場合は false です。

説明

このメソッドは、別のスレッドによって呼び出されます。マルチスレッドの問題を回避するには、このメソッドがユーザ・インタフェースにイベントを通知する必要があります。マルチスレッドを使用する場合は、スレッドごとに別々の接続を使用し、lock キーワードを使用して、アプリケーションの共有オブジェクトにアクセスすることをおすすめします。ActiveSyncInvoked への呼び出しによって発生した同期では、その接続の SyncInfo ストリームに StreamType.ACTIVE_SYNC を指定する必要があります。

例

次のコード・フラグメントは、ActiveSync 要求の受信方法および UI スレッドでの同期の実行方法を示してします。

```
[Visual Basic]
 Imports iAnywhere.UltraLite

 Public Class MainWindow
   Inherits System.Windows.Forms.Form
   Implements ActiveSyncListener
   Private dbMgr As DatabaseManager
   Private conn As Connection

   Public Sub New(ByVal args() As String)

     MyBase.New()

     'This call is required by the Windows Form
Designer.
     InitializeComponent()

     'Add any initialization after the
InitializeComponent() call
     dbMgr = New DatabaseManager
     dbMgr.SetActiveSyncListener( "myCompany.myapp", Me
)
     'Create Connection
     ...
   End Sub

   Protected Overrides Sub OnClosing(ByVal e As
System.ComponentModel.CancelEventArgs)
```

```
      If Not (dbMgr Is Nothing) Then
        dbMgr.SetActiveSyncListener(Nothing, Nothing)
        dbMgr = Nothing
      End If
    End Sub

    Public Sub ActiveSyncInvoked(ByVal
launchedByProvider As Boolean) Implements
iAnywhere.UltraLite.ActiveSyncListener.ActiveSyncInvoke
d
      Me.Invoke(New EventHandler(AddressOf
Me.ActiveSyncAction))
    End Sub

    Public Sub ActiveSyncAction(ByVal sender As Object,
ByVal e As EventArgs)
      ' Do active sync
      conn.SyncParms.Stream = StreamType.ACTIVE_SYNC
      conn.Synchronize()
    End Sub
 End Class

 [C#]
 using iAnywhere.UltraLite;
 public class Form1 : System.Windows.Forms.Form,
ActiveSyncListener
 {
   private System.Windows.Forms.MainMenu mainMenu1;
   private DatabaseManager dbMgr;
   private Connection conn;

   public Form1()
   {
     //
     // Required for Windows Form Designer support
     //
     InitializeComponent();

     //
     // TODO: Add any constructor code after
InitializeComponent call
     //
     dbMgr = new DatabaseManager();
     dbMgr.SetActiveSyncListener( "myCompany.myapp",
this );
     // Create connection
```

```
      ...
    }

    protected override void Dispose( bool disposing )
    {
      base.Dispose( disposing );
    }

    protected override void
OnClosing(System.ComponentModel.CancelEventArgs e)
    {
      if( dbMgr != null ) {
        dbMgr.SetActiveSyncListener( null, null );
        dbMgr = null;
      }
      base.OnClosing (e);
    }

    public void ActiveSyncInvoked(bool
launchedByProvider)
    {
     this.Invoke( new EventHandler( ActiveSyncHandler )
);
    }

    internal void ActiveSyncHandler(object sender,
EventArgs e)
    {
      conn.SyncParms.Stream = StreamType.ACTIVE_SYNC;
      conn.Synchronize();
    }
 }
```

AuthStatusCode 列挙

Mobile Link ユーザ認証の実行中にレポートされる可能性のあるステータス・コードを列挙します。

プロトタイプ

' Visual Basic

```
Public Enum AuthStatusCode
  Inherits Short
```

// C#

```
public enum AuthStatusCode :
  short
```

メンバ

メンバ	説明
EXPIRED	ユーザ ID またはパスワードの有効期限が切れているため、認証に失敗しました (EXPIRED = 3000)。
IN_USE	ユーザ ID がすでに使用されているため、認証に失敗しました (IN_USE = 5000)。
INVALID	ユーザ ID またはパスワードが正しくないため、認証に失敗しました (INVALID = 4000)。
UNKNOWN	認証ステータスが不明です。接続がまだ同期を実行していない可能性があります (UNKNOWN = 0)。
VALID	ユーザ ID とパスワードは、同期時には有効でした (VALID = 1000)。
VALID_BUT_EXPIRES_SOON	ユーザ ID とパスワードは、同期時には有効でしたが、まもなく有効期限が切れます (VALID_BUT_EXPIRES_SOON = 2000)。

参照

- AuthStatus プロパティ

Connection クラス

Ultra Light データベースへの接続を示します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class Connection

// C#
    public sealed class Connection
```

説明

このクラスを直接インスタンス化することはできません。接続をインスタンス化して開く処理には、OpenConnection メソッド が使用されます。ただし、データベースが存在しない場合は CreateDatabase メソッド が使用されます。

接続は、他の操作の実行前に開きます。また、その接続での操作がすべて終了したら、接続を閉じてからアプリケーションを終了します。接続で開いたテーブルをすべて閉じてから、その接続を閉じます。

SQLException がスローされたときは、この例外のエラー・コードは、LastSQLCode プロパティ のエラー・コード、または IsLastFetchOK プロパティ および IsLastCodeOK プロパティ の値よりも優先されます。

Connection メンバ

パブリック静的フィールド (共有)

メンバ	説明
INVALID_DATABASE_ID フィールド	DatabaseID プロパティ が設定されていないことを示すデータベース ID の定数です。

パブリック・インスタンス・フィールド

メンバ	説明
AutoCommit フィールド	insert 文、update 文、delete 文の後でコミットを行うかどうかを指定します。

パブリック・インスタンス・プロパティ

メンバ	説明
ConnectionOpenParms プロパティ	この接続を開くために使用されるパラメータです。
DatabaseID プロパティ	グローバル・オートインクリメント・カラムに使用されるデータベース ID の値です。
GlobalAutoIncrementUsage プロパティ	使用可能なグローバル・オートインクリメントの値の使用済み比率 (%) です。
IsDatabaseNew プロパティ	この接続が開かれたときに、データベースが新規であったかどうかをチェックします。
IsLastCodeOK プロパティ	最後の SQL コードを成功または失敗として分類します。
IsLastFetchOK プロパティ	最後に行われたローのフェッチについて、成功または失敗を確認するのに便利です。
IsOpen プロパティ	接続が開いているか、閉じているかを調べます。
LastIdentity プロパティ	直前に使用した identity の値です。
LastSQLCode プロパティ	最後のデータベース操作の成功または失敗に対応する SQL コードの値です。
Schema プロパティ	接続に関連付けられているデータベース スキーマです。
SyncParms プロパティ	この接続の同期設定です。
SyncResult プロパティ	この接続の最後の同期結果です。

パブリック・インスタンス・メソッド

メンバ	説明
ChangeEncryptionKey メソッド	データベースの暗号化キーを、指定した新しいキーに変更します。

メンバ	説明
Close メソッド	この接続を閉じます。
Commit メソッド	未処理の変更をデータベースにコミットします。
CountUploadRows メソッド	次回の同期でアップロードする必要のあるロー数を返します。
CountUploadRows メソッド	次回の同期でアップロードする必要のあるロー数を返します。
GetLastDownloadTime メソッド	指定されたパブリケーションの最後のダウンロードの時刻を返します。
GetNewUUID メソッド	新しい UUID を生成します (Guid)。
GetTable メソッド	データベース内の要求されたテーブルへの参照を作成して返します。
GrantConnectTo メソッド	指定されたパスワードを持つユーザ ID に、Ultra Light データベースへのアクセスを許可します。
PrepareStatement メソッド	IN パラメータ付きまたは IN パラメータなしで、SQL 文を事前にコンパイルして PreparedStatement オブジェクトに格納します。
ResetLastDownloadTime メソッド	最後のダウンロードの時刻をリセットします。
RevokeConnectFrom メソッド	指定したユーザ ID から、Ultra Light データベースへのアクセス権を取り消します。
Rollback メソッド	未処理の変更をデータベースにロールバックします。
RollbackPartialDownload メソッド	部分的なダウンロードから、未処理の変更をデータベースにロールバックします。
StartSynchronizationDelete メソッド	同期用に、この接続によって行われる以後のすべての削除にマークを付けます。
StopSynchronizationDelete メソッド	この関数が呼び出されると、削除操作が同期されなくなります。

メンバ	説明
Synchronize メソッド	現在の SyncParms プロパティ を使用してデータベースを同期します。
Synchronize メソッド	指定したリスナに送信されるプログレス・イベントとともに、現在の SyncParms プロパティ を使用してデータベースを同期します。

AutoCommit フィールド

insert 文、update 文、delete 文の後でコミットを行うかどうかを指定します。

プロトタイプ

```
' Visual Basic
    PublicAutoCommit As Boolean

// C#
    public bool AutoCommit;
```

説明

AutoCommit が false の場合、コミットまたはロールバックが実行されるのは、ユーザが Commit メソッドまたは Rollback メソッドを呼び出したときだけです。

デフォルトでは、各文の処理が成功した後でデータベースのコミットが実行されます。コミットに失敗する場合は、追加の SQL 文を実行してからコミットを再度実行するか、ロールバック文を実行できます。

参照

- Connection クラス
- Connection メンバ
- Commit メソッド
- Rollback メソッド

INVALID_DATABASE_ID フィールド

DatabaseID プロパティ が設定されていないことを示すデータベース ID の定数です。

プロトタイプ

```
' Visual Basic
    Public Shared INVALID_DATABASE_ID As Long

// C#
    public const long INVALID_DATABASE_ID;
```

ConnectionOpenParms プロパティ

この接続を開くために使用されるパラメータです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ConnectionOpenParms As ConnectionParms

// C#
    public ConnectionParms ConnectionOpenParms {get;}
```

プロパティ値

読み込み専用の ConnectionParms クラス インスタンスで、この接続を開くために使用されるパラメータ

説明

このプロパティを使用できるのは、接続が閉じられた後です。

参照

- Connection クラス
- Connection メンバ
- ConnectionParms クラス
- CreateDatabase メソッド
- OpenConnection メソッド

DatabaseID プロパティ

グローバル・オートインクリメント・カラムに使用されるデータベース ID の値です。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseID As Long

// C#
    public long DatabaseID {get;set;}
```

プロパティ値

データベース ID の値 は、`[0,UInt32.MaxValue]` の範囲内である必要があります。INVALID_DATABASE_ID フィールド の値は、データベース ID が設定されていないことを示すのに使用されます。

例外

- SQLException クラス - 新しい ID が無効の場合

GlobalAutoIncrementUsage プロパティ

使用可能なグローバル・オートインクリメントの値の使用済み比率 (%) です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property GlobalAutoIncrementUsage As Short

// C#
    public short GlobalAutoIncrementUsage {get;}
```

プロパティ値

使用可能なグローバル・オートインクリメントの値の使用済み比率 (%)。これは、0 ～ 100 の範囲内の整数です。

説明

比率が 100% に近づいたら、DatabaseID プロパティ を使用して、使用中のアプリケーションに新しいグローバル・データベース ID を設定してください。

例外

- SQLException クラス - SQL エラーが発生しました。

IsDatabaseNew プロパティ

この接続が開かれたときに、データベースが新規であったかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsDatabaseNew As Boolean

// C#
    public bool IsDatabaseNew {get;}
```

プロパティ値

この接続が開かれたときにデータベースが新規であった場合は true、この接続が開かれたときにデータベースがすでに存在していた場合は false

参照

- Connection クラス
- Connection メンバ
- CreateDatabase メソッド
- OpenConnection メソッド

IsLastCodeOK プロパティ

最後の SQL コードを成功または失敗として分類します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsLastCodeOK As Boolean

// C#
    public bool IsLastCodeOK {get;}
```

プロパティ値

最後の SQL コードが警告または成功を示している場合は true、エラーを示している場合は false。たとえば、SQLCode 列挙 はエラーではありません。

参照

- Connection クラス
- Connection メンバ
- LastSQLCode プロパティ
- SQLCode 列挙

IsLastFetchOK プロパティ

最後に行われたローのフェッチについて、成功または失敗を確認するのに便利です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsLastFetchOK As Boolean

// C#
    public bool IsLastFetchOK {get;}
```

プロパティ値　最後に行われたフェッチが成功した場合は true、失敗した場合は false

説明　SQLException がスローされたときは、この例外のエラー・コードは、LastSQLCode プロパティ のエラー・コード、または IsLastFetchOK プロパティ および IsLastCodeOK プロパティ の値よりも優先されます。

参照

- Connection クラス
- Connection メンバ
- LastSQLCode プロパティ

IsOpen プロパティ

接続が開いているか、閉じているかを調べます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値　接続が開いている場合は true、閉じている場合は false

LastIdentity プロパティ

直前に使用した identity の値です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property LastIdentity As UInt64

// C#
    public ulong LastIdentity {get;}
```

プロパティ値　直前に使用した identity の値。このプロパティは、次の SQL 文と同義です。

```
SELECT @@identity
```

LastIdentity は、グローバル・オートインクリメント・カラムで使うと特に便利です。

説明

このプロパティでは最後に割り当てられたデフォルト値がわかるだけなので、間違った結果を取らないために INSERT 文を実行した直後にこの値を取り出してください。

ときには、1 つの INSERT 文にグローバル・オートインクリメント型のカラムが複数含まれていることがあります。この場合、`LastIdentity` は生成されたデフォルト値のいずれかですが、そのうちのどの値であるかを判別する信頼できる方法はありません。このため、このような状況を回避するようなデータベースの設計と INSERT 文の記述を行ってください。

例外

- SQLException クラス - SQL エラーが発生しました。

LastSQLCode プロパティ

最後のデータベース操作の成功または失敗に対応する SQL コードの値です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property LastSQLCode As SQLCode

// C#
    public SQLCode LastSQLCode {get;}
```

説明

SQL コードは、Adaptive Server Anywhere の標準のコードであり、この接続の以後の Ultra Light データベース操作によってリセットされます。

SQLException がスローされたときは、この例外のエラー・コードは、LastSQLCode プロパティ によって返されたエラー・コード、または IsLastFetchOK プロパティ および IsLastCodeOK プロパティ からの結果よりも優先されます。

Schema プロパティ

接続に関連付けられているデータベース スキーマです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Schema As DatabaseSchema

// C#
    public DatabaseSchema Schema {get;}
```

プロパティ値 接続によって参照されているデータベースのスキーマへの参照

説明 このプロパティが有効なのは、接続が開かれている間だけです。

参照

- Connection クラス
- Connection メンバ
- DatabaseSchema クラス

SyncParms プロパティ

この接続の同期設定です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncParms As SyncParms

// C#
    public SyncParms SyncParms {get;}
```

プロパティ値 接続によって同期に使用されるパラメータへの参照。パラメータの修正は、この接続で行われる次の同期に反映されます。

参照

- Connection クラス
- Connection メンバ
- SyncParms プロパティ
- SyncResult プロパティ

SyncResult プロパティ

この接続の最後の同期結果です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncResult As SyncResult
```

```
// C#
    public SyncResult SyncResult {get;}
```

プロパティ値　この接続の最後の同期結果に対する参照

参照

- Connection クラス
- Connection メンバ
- SyncResult プロパティ
- SyncParms プロパティ

ChangeEncryptionKey メソッド

データベースの暗号化キーを、指定した新しいキーに変更します。

プロトタイプ

```
' Visual Basic
    Public Sub ChangeEncryptionKey( _
      ByVal newKey As String _
    )

// C#
    public void ChangeEncryptionKey(
      string  newKey
    );
```

パラメータ

- **newKey** データベースの新しい暗号化キー

説明　暗号化キーが失われた場合は、データベースを開くことができなくなります。

例外

- SQLException クラス - SQL エラーが発生しました。

Close メソッド

この接続を閉じます。

プロトタイプ

```
' Visual Basic
    Public Sub Close()

// C#
    public void Close();
```

説明

接続を閉じると、その接続をもう一度開くことはできません。接続をもう一度開くには、新しい Connection オブジェクトを作成して開く必要があります。

閉じた接続に関連付けられたオブジェクト (テーブル、スキーマなど) を使用するとエラーになります。

例外

- SQLException クラス - SQL エラーが発生しました。

Commit メソッド

未処理の変更をデータベースにコミットします。

プロトタイプ

```
' Visual Basic
    Public Sub Commit()

// C#
    public void Commit();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- Commit メソッド
- Rollback メソッド

CountUploadRows メソッド

次回の同期でアップロードする必要のあるロー数を返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function CountUploadRows( _
      ByVal mask As Integer, _
      ByVal threshold As UInt32 _
    ) As UInt32
```

```
// C#
    public uint CountUploadRows(
      int mask,
      uint threshold
    );
```

パラメータ

- **mask** ローのチェック対象となるパブリケーションのセット。詳細については、PublicationSchema クラスを参照してください。
- **threshold** カウントするローの最大数。`CountUploadRows` の所要時間を制限します。値 0 は制限が最大であることを示します。値 1 は、同期の必要なローがあるかどうかを判別する場合に使用します。

戻り値

指定したパブリケーションからアップロードする必要のあるロー数

例外

- SQLException クラス - SQL エラーが発生しました。

CountUploadRows メソッド

次回の同期でアップロードする必要のあるロー数を返します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function CountUploadRows( _
      ByVal mask As Integer, _
      ByVal threshold As Long _
    ) As Long

// C#
    public long CountUploadRows(
      int mask,
      long threshold
    );
```

パラメータ

- **mask** ローのチェック対象となるパブリケーションのセット。詳細については、PublicationSchema クラスを参照してください。

- **threshold** カウントするローの最大数。`CountUploadRows` の所要時間を制限します。値 0 は制限が最大であることを示します。値 1 は、同期の必要なローがあるかどうかを判別する場合に使用します。*threshold* は、`[0,0x0fffffffff]` の範囲内である必要があります。

戻り値

指定したパブリケーションからアップロードする必要のあるロー数

説明

このメソッドは、System.UInt32 型をネイティブでサポートしていない言語を対象として用意されています。この型がアプリケーションでサポートされている場合は、このメソッドの別のフォームを使用します。

例外

- SQLException クラス - SQL エラーが発生しました。

GetLastDownloadTime メソッド

指定されたパブリケーションの最後のダウンロードの時刻を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetLastDownloadTime( _
      ByVal mask As Integer _
    ) As Date

// C#
    public DateTime GetLastDownloadTime(
      int mask
    );
```

パラメータ

- **mask** チェックするパブリケーションのマスク。詳細については、PublicationSchema クラス を参照してください。

戻り値

最後のダウンロードのタイムスタンプ

説明

パラメータ *mask* は、単一のパブリケーションを参照するか、データベース全体の最後のダウンロード時刻の特殊定数 SYNC_ALL_DB フィールド である必要があります。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- ResetLastDownloadTime メソッド

GetNewUUID メソッド

新しい UUID を生成します (Guid)。

プロトタイプ

```
' Visual Basic
    Public Function GetNewUUID() As Guid

// C#
    public Guid GetNewUUID();
```

戻り値

Guid としての新しい UUID

説明

.NET Compact Framework に含まれていないため、ここで提供されています。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- SetUUID メソッド
- GetUUID メソッド

GetTable メソッド

データベース内の要求されたテーブルへの参照を作成して返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTable( _
      ByVal name As String _
    ) As Table

// C#
    public Table GetTable(
      string  name
    );
```

パラメータ

- **name** フェッチするテーブルの名前

戻り値

目的のテーブルを表す Table クラス オブジェクトのインスタンス

例外

- SQLException クラス - SQL エラーが発生しました。

GrantConnectTo メソッド

指定されたパスワードを持つユーザ ID に、Ultra Light データベースへのアクセスを許可します。

プロトタイプ

```
' Visual Basic
    Public Sub GrantConnectTo( _
      ByVal uid As String, _
      ByVal pwd As String _
    )

// C#
    public void GrantConnectTo(
      string  uid,
      string  pwd
    );
```

パラメータ

- **uid** データベースへのアクセス権を受け取るユーザ ID。ID の最大長は 16 文字です。
- **pwd** ID に関連付けられるパスワード。最大長は 16 文字です。

説明

既存のユーザ ID が指定されていれば、この関数を使用してそのユーザのパスワードを更新します。Ultra Light では、最大で 4 人のユーザがサポートされます。このメソッドが有効になるのは、接続が開かれたときにユーザ認証が有効になっていた場合だけです。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- OpenConnection メソッド

PrepareStatement メソッド

IN パラメータ付きまたは IN パラメータなしで、SQL 文を事前にコンパイルして PreparedStatement オブジェクトに格納します。

プロトタイプ

```
' Visual Basic
    Public Function PrepareStatement( _
      ByVal sql As String _
    ) As PreparedStatement

// C#
    public PreparedStatement PrepareStatement(
      string  sql
    );
```

パラメータ

- **sql** IN パラメータのプレースホルダ '?' が 1 つまたは複数含まれている可能性のある SQL 文

戻り値

事前にコンパイルされた文を含む新しい PreparedStatement オブジェクト

説明

準備文オブジェクトを使用すると、この文を効率的に何回も実行できます。

例外

- SQLException クラス - SQL エラーが発生しました。

ResetLastDownloadTime メソッド

最後のダウンロードの時刻をリセットします。

プロトタイプ

```
' Visual Basic
    Public Sub ResetLastDownloadTime( _
      ByVal mask As Integer _
    )

// C#
    public void ResetLastDownloadTime(
      int mask
    );
```

パラメータ

- **mask** リセットするパブリケーションのセット。詳細については、PublicationSchema クラス を参照してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- GetLastDownloadTime メソッド

RevokeConnectFrom メソッド

指定したユーザ ID から、Ultra Light データベースへのアクセス権を取り消します。

プロトタイプ

```
' Visual Basic
    Public Sub RevokeConnectFrom( _
      ByVal uid As String _
    )

// C#
    public void RevokeConnectFrom(
      string  uid
    );
```

パラメータ

- **uid** データベースへのアクセス権を取り消すユーザ ID。ユーザ ID の最大長は 16 文字です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- GrantConnectTo メソッド

Rollback メソッド

未処理の変更をデータベースにロールバックします。

プロトタイプ

```
' Visual Basic
    Public Sub Rollback()
```

```
// C#
    public void Rollback();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- Commit メソッド
- AutoCommit フィールド

RollbackPartialDownload メソッド

部分的なダウンロードから、未処理の変更をデータベースにロールバックします。

プロトタイプ

```
' Visual Basic
    Public Sub RollbackPartialDownload()
```

```
// C#
    public void RollbackPartialDownload();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- KeepPartialDownload プロパティ
- ResumePartialDownload プロパティ

StartSynchronizationDelete メソッド

同期用に、この接続によって行われる以後のすべての削除にマークを付けます。

プロトタイプ

```
' Visual Basic
    Public Sub StartSynchronizationDelete()
```

```
// C#
    public void StartSynchronizationDelete();
```

説明

この関数が呼び出されると、すべての削除操作が再度同期され、削除されたローが統合データベースからも削除されます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- StopSynchronizationDelete メソッド

StopSynchronizationDelete メソッド

この関数が呼び出されると、削除操作が同期されなくなります。

プロトタイプ

```
' Visual Basic
    Public Sub StopSynchronizationDelete()

// C#
    public void StopSynchronizationDelete();
```

説明

領域を節約するために、Ultra Light データベースから古い情報を削除して、統合データベースにはそれを残しておく場合に、このメソッドを使用すると便利です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- StartSynchronizationDelete メソッド

Synchronize メソッド

現在の SyncParms プロパティ を使用してデータベースを同期します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Synchronize()

// C#
    public void Synchronize();
```

説明

結果の詳細なステータスは、この接続の SyncResult プロパティ オブジェクトでレポートされます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ
- SyncParms プロパティ
- SyncResult プロパティ

Synchronize メソッド

指定したリスナに送信されるプログレス・イベントとともに、現在の SyncParms プロパティ を使用してデータベースを同期します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Synchronize( _
      ByVal listener As SyncProgressListener _
    )

// C#
    public void Synchronize(
      SyncProgressListener listener
    );
```

パラメータ

- **listener** 同期プログレス・イベントを受信するオブジェクト

説明

リスナに送信された最後のイベントのステータスは、SyncProgressState 列挙 です。

同期中のエラーは、SyncProgressState 列挙 イベントとして送信され、SQLException クラス としてスローされます。

結果の詳細なステータスは、この接続の SyncResult プロパティ オブジェクトで報告されます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Connection クラス
- Connection メンバ

- SyncParms プロパティ
- SyncResult プロパティ
- SyncProgressListener インタフェース

ConnectionParms クラス

Ultra Light データベースへの接続を開くためのパラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Class ConnectionParms
     Inherits DatabaseNameParms

// C#
    public class ConnectionParms :
     DatabaseNameParms
```

説明

データベースは、1 人の認証済みユーザ DBA で作成されます。このユーザの最初のパスワードは SQL です。デフォルトでは、ユーザ ID DBA とパスワード SQL を使用して、接続が開かれます。デフォルト・ユーザを無効にするには、RevokeConnectFrom メソッド を使用します。ユーザを追加したり、ユーザのパスワードを変更したりするには、GrantConnectTo メソッド を使用します。

複数の接続が作成される場合は、ConnectionName="*名前*"; を使用して、各接続にユニークな名前を付ける必要があります。

参照

- ConnectionParms メンバ
- AdditionalParms プロパティ

ConnectionParms メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
ConnectionParms コンストラクタ	ConnectionParms インスタンスを、そのデフォルト値で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AdditionalParms プロパティ (継承 DatabaseNameParms)	*名前*= *値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。
CacheSize プロパティ	キャッシュのサイズを指定します。
ConnectionName プロパティ	接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。
DatabaseOnCE プロパティ (継承 DatabaseNameParms)	Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。
DatabaseOnDesktop プロパティ (継承 DatabaseNameParms)	Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。
EncryptionKey プロパティ	データベースを暗号化するためのキーを指定します。
ParmsUsed プロパティ (継承 DatabaseNameParms)	このインスタンスを使用する最後の `DatabaseManager` メソッドによって実際に使用されたパラメータを返します。
Password プロパティ	認証済みユーザのパスワードを指定します。
UserID プロパティ	データベースで認証されるユーザを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド (継承 DatabaseNameParms)	このインスタンスの文字列表現を返します。

パブリック・インスタンス・イベント

メンバ	説明
UnusedEvent イベント (継承 DatabaseNameParms)	使われていません。

プロテクト・インスタンス・メソッド

メンバ	説明
Dispose メソッド (継承 DatabaseNameParms)	このオブジェクトによって使用されるリソースを解放します。

ConnectionParms コンストラクタ

ConnectionParms インスタンスを、そのデフォルト値で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public ConnectionParms();
```

例外

- SQLException クラス - SQL エラーが発生しました。

CacheSize プロパティ

キャッシュのサイズを指定します。

プロトタイプ

```
' Visual Basic
    Public Property CacheSize As String

// C#
    public string  CacheSize {get;set;}
```

説明

キャッシュ・サイズの値は、バイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス m または M を使用します。

たとえば、次の例ではキャッシュ・サイズが 128 KB に設定されます。

```
connParms.CacheSize = "128k";
```

デフォルトのキャッシュ・サイズは 16 ページです。デフォルトのページ・サイズは 4 KB なので、デフォルトのキャッシュ・サイズは 64 KB です。キャッシュの最小サイズは、プラットフォームによって異なります。

デフォルトのキャッシュ・サイズは小さめの値です。テスト結果により、パフォーマンスの向上が必要と判断した場合は、キャッシュ・サイズを大きくしてください。

キャッシュ・サイズをデータベース自体のサイズより大きくすると、パフォーマンスは向上しません。また、キャッシュ・サイズを大きくすると、使用できる他のアプリケーションの数に影響する場合があります。

キャッシュ・サイズが未指定であるか、正しく指定されていない場合は、デフォルトのサイズが使用されます。

ConnectionName プロパティ

接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。

プロトタイプ

```
' Visual Basic
    Public Property ConnectionName As String

// C#
    public string  ConnectionName {get;set;}
```

EncryptionKey プロパティ

データベースを暗号化するためのキーを指定します。

プロトタイプ

```
' Visual Basic
    Public Property EncryptionKey As String

// C#
    public string  EncryptionKey {get;set;}
```

説明

すべての openConnection 呼び出しは、データベースが作成されたときに指定されたキーと同じキーを使用する必要があります。キーを忘れた場合はデータベースにまったくアクセスできなくなります。

すべてのパスワードに共通することですが、最善の方法は、簡単には推測できないキー値を選択することです。キーの長さは任意ですが、短いと推測されやすいため、一般的には長い方が適しています。また、数字、文字、特殊文字を組み合わせると、キーは推測されにくくなります。

参照

- ConnectionParms クラス
- ConnectionParms メンバ
- ChangeEncryptionKey メソッド

Password プロパティ

認証済みユーザのパスワードを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Password As String

// C#
    public string  Password {get;set;}
```

説明

データベースが作成されると、ユーザ ID DBA のパスワードは SQL に設定されます。データベースで大文字と小文字が区別されない場合は、パスワードの大文字と小文字は区別されません。データベースで大文字と小文字が区別される場合は、パスワードの大文字と小文字も区別されます。デフォルト値は SQL です。

参照

- ConnectionParms クラス
- ConnectionParms メンバ
- UserID プロパティ

UserID プロパティ

データベースで認証されるユーザを指定します。

プロトタイプ

```
' Visual Basic
    Public Property UserID As String

// C#
    public string  UserID {get;set;}
```

説明

データベースは最初、DBA という名前の 1 人の認証済みユーザで作成されます。データベースで大文字と小文字が区別されない場合は、ユーザ ID の大文字と小文字は区別されません。データベースで大文字と小文字が区別される場合は、ユーザ ID の大文字と小文字も区別されます。デフォルト値は DBA です。

参照

- ConnectionParms クラス
- ConnectionParms メンバ
- Password プロパティ
- GrantConnectTo メソッド
- RevokeConnectFrom メソッド
- IsCaseSensitive プロパティ

CreateParms クラス

Ultra Light データベースを作成し、そのデータベースへの接続を開くためのパラメータを指定します。

プロトタイプ

' Visual Basic

```
NotInheritable Public Class CreateParms
  Inherits ConnectionParms
```

// C#

```
public sealed class CreateParms :
  ConnectionParms
```

説明

注意：一度にアクティブにできるデータベースは 1 つのみです。ほかの接続が開いているときに別のデータベースを作成しようとすると、エラーが発生します。

データベースを正常に作成するには、有効なスキーマ・ファイルを指定します。

参照

- CreateParms メンバ
- AdditionalParms プロパティ

CreateParms メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
CreateParms コンストラクタ	CreateParms インスタンスを、そのデフォルト値で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AdditionalParms プロパティ (継承 DatabaseNameParms)	*名前*= *値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。
CacheSize プロパティ (継承 ConnectionParms)	キャッシュのサイズを指定します。
ConnectionName プロパティ (継承 ConnectionParms)	接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。
DatabaseOnCE プロパティ (継承 DatabaseNameParms)	Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。
DatabaseOnDesktop プロパティ (継承 DatabaseNameParms)	Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。
EncryptionKey プロパティ (継承 ConnectionParms)	データベースを暗号化するためのキーを指定します。
ParmsUsed プロパティ (継承 DatabaseNameParms)	このインスタンスを使用する最後の `DatabaseManager` メソッドによって実際に使用されたパラメータを返します。
Password プロパティ (継承 ConnectionParms)	認証済みユーザのパスワードを指定します。
Schema プロパティ	Ultra Light データベースのスキーマを指定します。
UserID プロパティ (継承 ConnectionParms)	データベースで認証されるユーザを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド (継承 DatabaseNameParms)	このインスタンスの文字列表現を返します。

パブリック・インスタンス・イベント

メンバ	説明
UnusedEvent イベント (継承 DatabaseNameParms)	使われていません。

CreateParms コンストラクタ

CreateParms インスタンスを、そのデフォルト値で初期化します。

プロトタイプ

```
' Visual Basic
    Public Sub New()

// C#
    public CreateParms();
```

例外

- SQLException クラス - SQL エラーが発生しました。

Schema プロパティ

Ultra Light データベースのスキーマを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Schema As SchemaParms

// C#
    public SchemaParms Schema {get;set;}
```

参照

- CreateParms クラス
- CreateParms メンバ

- SchemaParms クラス

Cursor クラス

Ultra Light データベースのカーソルを示します。カーソルとは、テーブルまたは結果セットからの一連のローです。

プロトタイプ

```
' Visual Basic
    MustInherit Public Class Cursor

// C#
    public abstract class Cursor
```

説明

このクラスは、Table クラス および ResultSet クラス の抽象基本クラスです。

派生クラスには、ローの説明にアクセス可能な `Schema` プロパティがあります。

Cursor メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
IsBOF プロパティ	現在のローの位置が最初のローの前なら true を返します。
IsEOF プロパティ	現在のローの位置が最後のローの後なら true を返します。
IsOpen プロパティ	カーソルが現在開いているかどうかを確認します。
RowCount プロパティ	カーソル内のロー数を返します。

パブリック・インスタンス・メソッド

メンバ	説明
Close メソッド	カーソルを閉じます。

メンバ	説明
Fill メソッド	ADO .NET DataTable オブジェクトに、カーソル内のローを格納します。
GetBoolean メソッド	指定したカラムの値を Boolean として返します。
GetBytes メソッド	指定したカラムの値を Byte 配列として返します。SQLType 列挙 型、SQLType 列挙 型、SQLType 列挙 型のカラムの場合にのみ有効です。
GetBytes メソッド	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。
GetChars メソッド	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。
GetDouble メソッド	指定したカラムの値を Double として返します。
GetFloat メソッド	指定したカラムの値を Single として返します。
GetInt メソッド	指定したカラムの値を Int32 として返します。
GetLong メソッド	指定したカラムの値を Int64 として返します。
GetShort メソッド	指定したカラムの値を Int16 として返します。
GetString メソッド	指定したカラムの値を String として返します。
GetTime メソッド	指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。
GetTimestamp メソッド	指定したカラムの値を、ミリ秒の精度の DateTime として返します。
GetUInt メソッド	指定したカラムの値を UInt32 として返します。
GetULong メソッド	指定したカラムの値を UInt64 として返します。
GetUShort メソッド	指定したカラムの値を UInt16 として返します。

メンバ	説明
GetUUID メソッド	指定したカラムの値を UUID (Guid) として返します。
IsNull メソッド	指定されたカラムの値が NULL かどうかをチェックします。
MoveAfterLast メソッド	カーソルの最後のローの後に移動します。
MoveBeforeFirst メソッド	カーソルの最初のローの前に移動します。
MoveFirst メソッド	カーソルの最初のローに移動します。
MoveLast メソッド	カーソルの最後のローに移動します。
MoveNext メソッド	次のローに、または最後のローの後に移動します。
MovePrevious メソッド	前のローに、または最初のローの前に移動します。
MoveRelative メソッド	現在のローを基準として移動します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	Finalize メソッドです。

IsBOF プロパティ

現在のローの位置が最初のローの前なら true を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsBOF As Boolean

// C#
    public bool IsBOF {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。

IsEOF プロパティ

現在のローの位置が最後のローの後なら true を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsEOF As Boolean

// C#
    public bool IsEOF {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。

IsOpen プロパティ

カーソルが現在開いているかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

戻り値

カーソルが現在開いている場合は true、閉じている場合は false

RowCount プロパティ

カーソル内のロー数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property RowCount As Integer

// C#
    public int RowCount {get;}
```

説明

RowCount は、古いローを削除して領域を節約するタイミングを決定するときに使用します。StopSynchronizationDelete メソッド を使用すると、統合データベースからは削除しないで Ultra Light データベースから古いローを削除できます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- StartSynchronizationDelete メソッド
- StopSynchronizationDelete メソッド

Close メソッド

カーソルを閉じます。

プロトタイプ

```
' Visual Basic
    Public Sub Close()

// C#
    public void Close();
```

説明

すでに閉じられているカーソルを閉じるのはエラーではありません。

例外

- SQLException クラス - SQL エラーが発生しました。

Fill メソッド

ADO .NET DataTable オブジェクトに、カーソル内のローを格納します。

プロトタイプ

```
' Visual Basic
    Public Function Fill( _
      ByVal dt As DataTable _
    ) As Integer
```

```
// C#
    public int Fill(
      DataTable dt
    );
```

パラメータ

- **dt** 格納先の ADO .NET DataTable オブジェクト

戻り値

格納されたローの数

Finalize メソッド

Finalize メソッドです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

GetBoolean メソッド

指定したカラムの値を Boolean として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetBoolean( _
      ByVal columnID As Short _
    ) As Boolean

// C#
    public bool GetBoolean(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Boolean としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetBytes メソッド

指定したカラムの値を Byte 配列として返します。SQLType 列挙 型、SQLType 列挙 型、SQLType 列挙 型のカラムの場合にのみ有効です。

プロトタイプ

```
' Visual Basic
    Overloads Public Function GetBytes( _
      ByVal columnID As Short _
    ) As Byte()

// C#
    public byte[] GetBytes(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Byte 配列としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBoolean メソッド
- GetBytes メソッド
- GetChars メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetBytes メソッド

指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。

プロトタイプ

```
' Visual Basic
    Overloads Public Function GetBytes( _
      ByVal columnID As Short, _
      ByVal srcOffset As Integer, _
      ByVal dst As Byte(), _
      ByVal dstOffset As Integer, _
      ByVal count As Integer _
    ) As Integer
```

```
// C#
    public int GetBytes(
      short columnID,
      int srcOffset,
      byte[] dst,
      int dstOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。
- **srcOffset** カラム値の開始位置。ゼロが値の冒頭です。
- **dst** コピー先の配列
- **dstOffset** コピー先の配列の開始位置
- **count** コピーされるバイト数

戻り値

実際にコピーされたバイト数

説明

値の *srcOffset* から *srcOffset*+*count*-1 までの位置のバイトが、コピー先の配列の *dstOffset* から *dstOffset*+*count*-1 までの位置に、それぞれコピーされます。*count* のバイト数がコピーされる前に、値の末尾が検出された場合は、コピー先の配列の残りは変更されないままになります。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、コピー先は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *dst* が null である。
- *srcOffset* が負である。
- *dstOffset* が負である。
- *count* が負である。
- *dstOffset*+*count* が *dst*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス

- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBoolean メソッド
- GetBytes メソッド
- GetChars メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetChars メソッド

指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。

プロトタイプ

```
' Visual Basic
    Public Function GetChars( _
      ByVal columnID As Short, _
      ByVal srcOffset As Integer, _
      ByVal dst As Char(), _
      ByVal dstOffset As Integer, _
      ByVal count As Integer _
    ) As Integer

// C#
    public int GetChars(
      short columnID,
      int srcOffset,
      char[] dst,
```

```
    int dstOffset,
    int count
);
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。
- **srcOffset** カラム値の開始位置。ゼロが値の冒頭です。
- **dst** コピー先の配列
- **dstOffset** コピー先の配列の開始位置
- **count** コピーされる文字数

戻り値

実際にコピーされた文字数

説明

値の *srcOffset* から *srcOffset*+*count*-1 までの位置の文字が、コピー先の配列の *dstOffset* から *dstOffset*+*count*-1 までの位置に、それぞれコピーされます。*count* の文字数がコピーされる前に、値の末尾が検出された場合は、コピー先の配列の残りは変更されないままになります。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、コピー先は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *dst* が null である。
- *srcOffset* が負である。
- *dstOffset* が負である。
- *count* が負である。
- *dstOffset*+*count* が *dst*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBoolean メソッド

- GetBytes メソッド
- GetBytes メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetDouble メソッド

指定したカラムの値を Double として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetDouble( _
      ByVal columnID As Short _
    ) As Double

// C#
    public double GetDouble(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Double としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ

- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetFloat メソッド

指定したカラムの値を Single として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetFloat( _
      ByVal columnID As Short _
    ) As Single

// C#
    public float  GetFloat(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Single としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetInt メソッド

指定したカラムの値を Int32 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetInt( _
      ByVal columnID As Short _
    ) As Integer

// C#
    public int GetInt(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Int32 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetLong メソッド

指定したカラムの値を Int64 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetLong( _
      ByVal columnID As Short _
    ) As Long

// C#
    public long GetLong(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値 Int64 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetShort メソッド

指定したカラムの値を Int16 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetShort( _
      ByVal columnID As Short _
    ) As Short

// C#
    public short GetShort(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Int16 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetString メソッド

指定したカラムの値を String として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetString( _
      ByVal columnID As Short _
    ) As String
```

```
// C#
    public string  GetString(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

String としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetTime メソッド

指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTime( _
      ByVal columnID As Short _
    ) As TimeSpan

// C#
    public TimeSpan GetTime(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

TimeSpan としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetTimestamp メソッド

指定したカラムの値を、ミリ秒の精度の DateTime として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTimestamp( _
      ByVal columnID As Short _
    ) As Date

// C#
    public DateTime GetTimestamp(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

DateTime としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- DateTime
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド

- GetUUID メソッド
- GetColumnSQLType メソッド

GetUInt メソッド

指定したカラムの値を UInt32 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUInt( _
      ByVal columnID As Short _
    ) As UInt32

// C#
    public uint GetUInt(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

UInt32 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド

- GetTimestamp メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetULong メソッド

指定したカラムの値を UInt64 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetULong( _
      ByVal columnID As Short _
    ) As UInt64

// C#
    public ulong GetULong(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

UInt64 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド

- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetUShort メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetUShort メソッド

指定したカラムの値を UInt16 として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUShort( _
      ByVal columnID As Short _
    ) As UInt16

// C#
    public ushort GetUShort(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

UInt16 としてのカラム値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド

- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUUID メソッド
- GetColumnSQLType メソッド

GetUUID メソッド

指定したカラムの値を UUID (Guid) として返します。

プロトタイプ

```
' Visual Basic
    Public Function GetUUID( _
      ByVal columnID As Short _
    ) As Guid

// C#
    public Guid GetUUID(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

Guid としてのカラム値

説明

SQLType 列挙 型のカラム、または長さが 16 の SQLType 列挙 型のカラムの場合にのみ有効です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド

- IsNull メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetBoolean メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetColumnSQLType メソッド
- GetColumnSize メソッド

IsNull メソッド

指定されたカラムの値が NULL かどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsNull( _
      ByVal columnID As Short _
    ) As Boolean

// C#
    public bool IsNull(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。カーソルの最初のカラムの ID 値は 1 です。

戻り値

値が NULL の場合は true、NULL でない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Cursor クラス
- Cursor メンバ
- GetColumnID メソッド
- GetBoolean メソッド
- GetBytes メソッド
- GetBytes メソッド
- GetChars メソッド
- GetDouble メソッド
- GetFloat メソッド
- GetInt メソッド
- GetLong メソッド
- GetShort メソッド
- GetString メソッド
- GetTime メソッド
- GetTimestamp メソッド
- GetUInt メソッド
- GetULong メソッド
- GetUShort メソッド
- GetUUID メソッド

MoveAfterLast メソッド

カーソルの最後のローの後に移動します。

プロトタイプ

```
' Visual Basic
    Public Sub MoveAfterLast()

// C#
    public void MoveAfterLast();
```

例外

- SQLException クラス - SQL エラーが発生しました。

MoveBeforeFirst メソッド

カーソルの最初のローの前に移動します。

プロトタイプ

```
' Visual Basic
    Public Sub MoveBeforeFirst()
```

```
// C#
    public void MoveBeforeFirst();
```

例外

- SQLException クラス - SQL エラーが発生しました。

MoveFirst メソッド

カーソルの最初のローに移動します。

プロトタイプ

```
' Visual Basic
    Public Function MoveFirst() As Boolean

// C#
    public bool MoveFirst();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、ローがない場合、メソッドは失敗します。

例外

- SQLException クラス - SQL エラーが発生しました。

MoveLast メソッド

カーソルの最後のローに移動します。

プロトタイプ

```
' Visual Basic
    Public Function MoveLast() As Boolean

// C#
    public bool MoveLast();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、ローがない場合、メソッドは失敗します。

例外

- SQLException クラス - SQL エラーが発生しました。

MoveNext メソッド

次のローに、または最後のローの後に移動します。

プロトタイプ

```
' Visual Basic
    Public Function MoveNext() As Boolean

// C#
    public bool MoveNext();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、それ以上ローがない場合、メソッドは失敗します。

例外

- SQLException クラス - SQL エラーが発生しました。

MovePrevious メソッド

前のローに、または最初のローの前に移動します。

プロトタイプ

```
' Visual Basic
    Public Function MovePrevious() As Boolean

// C#
    public bool MovePrevious();
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、それ以上ローがない場合、メソッドは失敗します。

例外

- SQLException クラス - SQL エラーが発生しました。

MoveRelative メソッド

現在のローを基準として移動します。

プロトタイプ

```
' Visual Basic
    Public Function MoveRelative( _
      ByVal offset As Integer _
    ) As Boolean

// C#
    public bool MoveRelative(
      int offset
    );
```

パラメータ

- **offset** 移動するローの数。負の値を指定すると、後方に移動します。

戻り値

成功した場合は true、失敗した場合は false。たとえば、最初または最後のローを超えて移動する場合、メソッドは失敗します。

説明

指定した移動先にローがない場合には、false が返されます。その場合のカーソル位置は、*offset* が正であるときには最後のロー (`isEOF()`) になり、*offset* が負であるときには最初のロー (`isBOF()`) になります。

例外

- SQLException クラス - SQL エラーが発生しました。

CursorSchema クラス

Ultra Light のカーソルのスキーマを表します。

プロトタイプ

```
' Visual Basic
    MustInherit Public Class CursorSchema

// C#
    public abstract class CursorSchema
```

説明

このクラスは、TableSchema クラスおよび ResultSetSchema クラス の抽象基本クラ スです。

CursorSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ	このカーソル内のカラム数を返します。
IsOpen プロパティ	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	このカーソルの名前です。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnID メソッド	指定したカラムのカラム ID を返します。
GetColumnName メソッド	指定したカラム ID で識別されたカラムの名前を返します。
GetColumnPrecision メソッド	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの精度を返します。

メンバ	説明
GetColumnScale メソッド	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの位取りを返します。
GetColumnSize メソッド	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラムのサイズを返します。
GetColumnSQLType メソッド	指定したカラムの SQL データ型を取得します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	関連するネイティブ・オブジェクトをクリーンアップするデストラクタです。

ColumnCount プロパティ

このカーソル内のカラム数を返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ColumnCount As Short

// C#
    public short ColumnCount {get;}
```

説明

カラム ID の範囲は、1 ～ `ColumnCount` です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

IsOpen プロパティ

カーソルのスキーマが現在開いているかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

戻り値

カーソルのスキーマが現在開いている場合は true、閉じている場合は false

Name プロパティ

このカーソルの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

Finalize メソッド

関連するネイティブ・オブジェクトをクリーンアップするデストラクタです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

GetColumnID メソッド

指定したカラムのカラム ID を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnID( _
      ByVal name As String _
    ) As Short
```

```
// C#
    public short GetColumnID(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラム ID

説明

カラム ID の範囲は、1 ～ ColumnCount プロパティ です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- CursorSchema クラス
- CursorSchema メンバ
- ColumnCount プロパティ

GetColumnName メソッド

指定したカラム ID で識別されたカラムの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnName( _
      ByVal columnID As Short _
    ) As String

// C#
    public string  GetColumnName(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID。*columnID* は、`[1,CursorSchema.ColumnCount]` の範囲内である必要があります。

戻り値

カラムの名前

説明

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。カラムを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- CursorSchema クラス
- CursorSchema メンバ
- ColumnCount プロパティ

GetColumnPrecision メソッド

カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの精度を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnPrecision( _
      ByVal name As String _
    ) As Integer

// C#
    public int GetColumnPrecision(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

数値カラムの精度

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- CursorSchema クラス
- CursorSchema メンバ
- GetColumnSQLType メソッド

GetColumnScale メソッド

カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの位取りを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnScale( _
      ByVal name As String _
    ) As Integer

// C#
    public int GetColumnScale(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

数値カラムの位取り

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- CursorSchema クラス
- CursorSchema メンバ
- GetColumnSQLType メソッド

GetColumnSize メソッド

カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラムのサイズを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnSize( _
      ByVal name As String _
    ) As Integer

// C#
    public int GetColumnSize(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値　サイズ指定されたカラムのサイズ

例外
- SQLException クラス - SQL エラーが発生しました。

参照
- CursorSchema クラス
- CursorSchema メンバ
- GetColumnSQLType メソッド

GetColumnSQLType メソッド

指定したカラムの SQL データ型を取得します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnSQLType( _
      ByVal name As String _
    ) As SQLType

// C#
    public SQLType GetColumnSQLType(
      string  name
    );
```

パラメータ
- **name** カラムの名前

戻り値　SQLType 列挙整数

例外
- SQLException クラス - SQL エラーが発生しました。

参照
- CursorSchema クラス
- CursorSchema メンバ
- SQLType 列挙
- ColumnCount プロパティ

DatabaseManager クラス

Ultra Light データベースへの接続を管理します。複数回インスタンス化しないでください。

プロトタイプ

' Visual Basic

NotInheritable Public Class **DatabaseManager**

// C#

public sealed class **DatabaseManager**

説明

接続は、他の操作の実行前に開きます。また、その接続での操作がすべて終了したら、接続を閉じてからアプリケーションを終了します。接続で開いたテーブルをすべて閉じてから、その接続を閉じます。

このコンストラクタはスレッド対応ではありません。このクラスの他のメソッドは、すべてスレッド対応です。

DatabaseManager メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
DatabaseManager コンストラクタ	スタンドアロンのランタイムを使用して、iAnywhere.UltraLite アセンブリを初期化します。DatabaseManager のインスタンスは、アプリケーションごとに 1 つしか作成できません。
DatabaseManager コンストラクタ	ランタイムのタイプを指定して、iAnywhere.UltraLite アセンブリを初期化します。DatabaseManager のインスタンスは、アプリケーションごとに 1 つしか作成できません。

パブリック・インスタンス・プロパティ

メンバ	説明
RuntimeType プロパティ	Ultra Light.NET ランタイムのタイプを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
CreateDatabase メソッド	データベースを作成し、*parms* によって指定されたデータベースへの接続を開きます。
DropDatabase メソッド	指定したデータベースを削除します。 接続が開かれているデータベースを削除することはできません。
OpenConnection メソッド	*parms* によって指定されたデータベースへの接続を開きます。データベースが存在しない場合は、値 SQLCode 列挙 とともに SQLException クラス がスローされます。
SetActiveSyncListener メソッド	ActiveSync 用 Mobile Link プロバイダからの ActiveSync の呼び出しを処理するリスナ・オブジェクトを指定します。
SetServerSyncListener メソッド	指定したサーバ同期メッセージの処理に使用するリスナ・オブジェクトを指定します。

DatabaseManager コンストラクタ

スタンドアロンのランタイムを使用して、iAnywhere.UltraLite アセンブリを初期化します。DatabaseManager のインスタンスは、アプリケーションごとに 1 つしか作成できません。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public DatabaseManager();
```

例外

- SQLException クラス - 1 つのアプリケーションにつき、複数回呼び出された場合

参照

- DatabaseManager クラス
- DatabaseManager メンバ
- RuntimeType プロパティ

DatabaseManager コンストラクタ

ランタイムのタイプを指定して、iAnywhere.UltraLite アセンブリを初期化します。DatabaseManager のインスタンスは、アプリケーションごとに 1 つしか作成できません。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal ulType As RuntimeType _
    )

// C#
    public DatabaseManager(
      RuntimeType ulType
    );
```

パラメータ

- **ulType** 使用する Ultra Light.NET ランタイムのタイプを指定します。

例外

- SQLException クラス - 1 つのアプリケーションにつき、複数回呼び出された場合

参照

- DatabaseManager クラス
- DatabaseManager メンバ
- RuntimeType プロパティ

RuntimeType プロパティ

Ultra Light.NET ランタイムのタイプを指定します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property RuntimeType As RuntimeType
```

```
// C#
    public RuntimeType RuntimeType {get;}
```

プロパティ値

アンマネージ Ultra Light.NET ランタイムのタイプを示す列挙値
RuntimeType 列挙

CreateDatabase メソッド

データベースを作成し、*parms* によって指定されたデータベースへの接続を開きます。

プロトタイプ

```
' Visual Basic
    Public Function CreateDatabase( _
      ByVal parms As CreateParms _
    ) As Connection

// C#
    public Connection CreateDatabase(
      CreateParms parms
    );
```

パラメータ

- **parms** データベースを作成し、そのデータベースへの接続を開くためのパラメータ。詳細については、CreateParms クラスを参照してください。

戻り値

開かれた接続

説明

すでにデータベースが存在する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされます。

接続の IsDatabaseNew プロパティ は `true` に設定され、接続が開かれときにデータベースが作成されたことが示されます。

接続を閉じるのは、その接続でのすべての操作がすべて終了してから、アプリケーションを終了する前に行う必要があります。接続で開いたテーブルをすべて閉じてから、その接続を閉じます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseManager クラス

- DatabaseManager メンバ
- CreateParms クラス
- OpenConnection メソッド
- IsDatabaseNew プロパティ
- Close メソッド
- GrantConnectTo メソッド
- RevokeConnectFrom メソッド

DropDatabase メソッド

指定したデータベースを削除します。

接続が開かれているデータベースを削除することはできません。

プロトタイプ

```
' Visual Basic
    Public Sub DropDatabase( _
      ByVal parms As DatabaseNameParms _
    )

// C#
    public void DropDatabase(
      DatabaseNameParms parms
    );
```

パラメータ

- **parms** データベースを識別するためのパラメータ。詳細については、DatabaseNameParms クラス を参照してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseManager クラス
- DatabaseManager メンバ
- DatabaseNameParms クラス
- CreateDatabase メソッド
- OpenConnection メソッド

OpenConnection メソッド

parms によって指定されたデータベースへの接続を開きます。データベースが存在しない場合は、値 SQLCode 列挙 とともに SQLException クラス がスローされます。

プロトタイプ

```
' Visual Basic
    Public Function OpenConnection( _
      ByVal parms As ConnectionParms _
    ) As Connection

// C#
    public Connection OpenConnection(
      ConnectionParms parms
    );
```

パラメータ

- **parms** 接続を開くためのパラメータ。詳細については、ConnectionParms クラス を参照してください。

戻り値

開かれた接続

説明

IsDatabaseNew プロパティ は `false` に設定され、接続が開かれときにデータベースが作成されなかったことが示されます。

接続を閉じるのは、その接続でのすべての操作がすべて終了してから、アプリケーションを終了する前に行う必要があります。接続で開いたテーブルをすべて閉じてから、その接続を閉じます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseManager クラス
- DatabaseManager メンバ
- ConnectionParms クラス
- CreateDatabase メソッド
- IsDatabaseNew プロパティ
- Close メソッド
- GrantConnectTo メソッド
- RevokeConnectFrom メソッド

SetActiveSyncListener メソッド

ActiveSync 用 Mobile Link プロバイダからの ActiveSync の呼び出しを処理するリスナ・オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetActiveSyncListener( _
      ByVal appClassName As String, _
      ByVal listener As ActiveSyncListener _
    )

// C#
    public void SetActiveSyncListener(
      string  appClassName,
      ActiveSyncListener listener
    );
```

パラメータ

- **appClassName** アプリケーションのユニークなクラス名。これは、アプリケーションが ActiveSync で使用されるように登録されているときに使用されるクラス名です。
- **listener** リスナ・オブジェクト。前のリスナを削除するには、null を使用します。

説明

パラメータ *appClassName* は、アプリケーションの識別に使用されるユニークな識別子です。アプリケーションは、一度に 1 つの *appClassName* のみを使用できます。リスナが、特定の *appClassName* に登録されているときは、別の *appClassName* で SetServerSyncListener メソッド または SetActiveSyncListener メソッド を呼び出すと失敗します。

ActiveSync のリスナを削除するには、*listener* パラメータとして `null` を使用して SetActiveSyncListener メソッド を呼び出します。

すべてのリスナを削除するには、すべてのパラメータに `null` を使用して SetServerSyncListener メソッド を呼び出します。

アプリケーションは、すべてのリスナを削除してから終了する必要があります。

例外

- SQLException クラス - SQL エラーが発生しました。

例

SetActiveSyncListener の例については、ActiveSyncInvoked メソッドを参照してください。

SetServerSyncListener メソッド

指定したサーバ同期メッセージの処理に使用するリスナ・オブジェクトを指定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetServerSyncListener( _
      ByVal messageName As String, _
      ByVal appClassName As String, _
      ByVal listener As ServerSyncListener _
    )

// C#
    public void SetServerSyncListener(
      string  messageName,
      string  appClassName,
      ServerSyncListener listener
    );
```

パラメータ

- **messageName** メッセージの名前
- **appClassName** アプリケーションのユニークなクラス名。これは、アプリケーションの識別に使用されるユニークな識別子です。
- **listener** リスナ・オブジェクト。前のリスナを削除するには、null を使用します。

説明

パラメータ *appClassName* は、アプリケーションの識別に使用されるユニークな識別子です。アプリケーションは、一度に 1 つの *appClassName* のみを使用できます。リスナが、特定の *appClassName* に登録されているときは、別の *appClassName* で SetServerSyncListener メソッド または SetActiveSyncListener メソッドを呼び出すと失敗します。

特定のメッセージのリスナを削除するには、*listener* パラメータとして `null` を使用して SetServerSyncListener メソッドを呼び出します。

すべてのリスナを削除するには、すべてのパラメータに null を使用して SetServerSyncListener メソッド を呼び出します。

アプリケーションは、すべてのリスナを削除してから終了する必要があります。

例外

- SQLException クラス - SQL エラーが発生しました。

例

ServerSyncInvoked メソッドの例については、

SetActiveSyncListener メソッドのマニュアルを参照してください。

DatabaseNameParms クラス

Ultra Light データベースを識別します。

プロトタイプ

```
' Visual Basic
    Public Class DatabaseNameParms
     Inherits Component

// C#
    public class DatabaseNameParms :
     Component
```

説明

各インスタンスには、データベースへのプラットフォーム固有のパスと汎用のパスがあります。プラットフォーム固有のパスでは、1 つの値だけが使用され、汎用のパスよりも優先されます。次に例を示します。

```
[Visual Basic]

dbName.DatabaseOnCE="¥UltraLite¥mydb1.udb";

dbName.DatabaseOnDesktop="mydb2.udb";

[C#]

dbName.DatabaseOnCE="¥¥UltraLite¥¥mydb1.udb";

dbName.DatabaseOnDesktop="mydb2.udb";
```

パス ¥UltraLite¥mydb1.udb は Windows CE で使用され、mydb2.udb はその他のプラットフォームで使用されます。

インスタンスが Ultra Light のメソッドに渡されると、プロパティ ParmsUsed は Ultra Light のメソッドによって実際に使用されるパラメータを返します。

Ultra Light データベース・ファイルに推奨されるファイル拡張子は **.udb** です。Windows CE デバイスでは、デフォルトのデータベースは **¥UltraLiteDB¥ulstore.udb** です。その他の Windows プラットフォームでは、デフォルトのデータベースは **ulstore.udb** です。C# では、パス内の円記号をエスケープする必要があります。

DatabaseNameParms メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
DatabaseNameParms コンストラクタ	DatabaseNameParms インスタンスを、そのデフォルト値で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AdditionalParms プロパティ	*名前*= *値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。
DatabaseOnCE プロパティ	Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。
DatabaseOnDesktop プロパティ	Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。
ParmsUsed プロパティ	このインスタンスを使用する最後の `DatabaseManager` メソッドによって実際に使用されたパラメータを返します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド	このインスタンスの文字列表現を返します。

パブリック・インスタンス・イベント

メンバ	説明
UnusedEvent イベント	使われていません。

プロテクト・インスタンス・メソッド

メンバ	説明
Dispose メソッド	このオブジェクトによって使用されるリソースを解放します。

DatabaseNameParms コンストラクタ

DatabaseNameParms インスタンスを、そのデフォルト値で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public DatabaseNameParms();
```

例外

- SQLException クラス - SQL エラーが発生しました。

AdditionalParms プロパティ

名前= *値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。ここで指定されるのは、あまり一般的に使用されないパラメータです。

プロトタイプ

```
' Visual Basic
    Public Property AdditionalParms As String

// C#
    public string  AdditionalParms {get;set;}
```

説明

ページ・サイズや予約サイズのパラメータの値は、バイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス m または M を使用します。

追加のパラメータを次に示します。

キーワード	説明
dbn	接続する必要のある、ロードしたデータベースを識別します。 データベースが起動されると、データベース名が割り当てられます。データベース名は、dbn パラメータを使用して明示的に割り当てられるか、または拡張子とパスが削除されたベース・ファイル名を使用して Ultra Light によって割り当てられます。 接続が開かれると、Ultra Light は dbn が一致する実行中のデータベースをまず検索します。一致するデータベースが見つからない場合、Ultra Light は適切なデータベースのファイル名のパラメータ (DatabaseOnCE プロパティ または DatabaseOnDesktop プロパティ) を使用して、新しいデータベースを起動します。 このパラメータが必要なのは、同じベース・ファイル名を持つ 2 つの異なるデータベースにアプリケーション (または Ultra Light エンジン) がアクセスする場合です。 このパラメータが使用されるのは、OpenConnection メソッド または CreateDatabase メソッド を使用して接続を開いたときだけです。

キーワード	説明
obfuscate	1 に設定すると、難読化されたデータベースを作成することが指定されます。次に例を示します。 `createParms.`**`AdditionalParms = "obfuscate=1";`** データベースを読みにくくすることで、表示ツールでデータベース内のデータを直接表示しようとする単純な行為に対するセキュリティが提供されます。巧妙で容赦ないデータへのアクセスが試みられた場合は、このオプションでは防止できません。難読化は、パフォーマンスにほとんど影響しません。 このパラメータが使用されるのは、CreateDatabase メソッド 使用して新しいデータベースを作成したときだけです。
page_size	Ultra Light データベースはページ単位で格納されます。I/O 操作は一度に 1 ページずつ実行されます。Ultra Light データベースのデフォルト・ページ・サイズは 4 KB です。次の保管パラメータ文字列を使用すると 2 KB のページを指定できます。 `connParms.`**`AdditionalParms = "page_size=2k";`** ページ・サイズを 2 KB に設定すると、最大テーブル数が約 500 に減少します。 このパラメータが使用されるのは、CreateDatabase メソッド 使用して新しいデータベースを作成したときだけです。

キーワード	説明
reserve_size	Ultra Light 継続データの保管に使用する、ファイル・システム領域を予約します。 `reserve_size` パラメータを使用して、実際にデータを挿入することなく、Ultra Light データベースが必要とするファイル・システム領域を事前に割り付けることができます。ファイル・システムの領域を予約すると、パフォーマンスが多少向上し、メモリが不足するという障害を防ぐことができます。デフォルトでは、永続ストレージ・ファイルのサイズは、アプリケーションがデータベースを更新して、サイズを大きくする必要が生じた場合にだけ大きくなります。 `reserve_size` で予約されるファイル・システム領域には、ロー・データだけでなく、継続保管ファイルのメタデータも含まれることに注意してください。データベースのデータ量から、必要なファイル・システム領域を計算する場合は、メタデータのオーバヘッドとデータの圧縮を考慮してください。テスト・データを入れてデータベースを実行し、継続保管ファイルのサイズを確認することをおすすめします。 `reserve_size` パラメータは、起動時に継続保管ファイルを指定の予約サイズまで大きくすることによって領域を予約します。これは、そのファイルが以前に存在していたかどうかに関係なく行われます。このファイルはトランケートされません。次のパラメータ文字列では、起動時に継続保管ファイルのサイズが最低 2 MB 確保されます。 `createParms.AdditionalParms = "reserve_size=2m";` このパラメータが使用されるのは、OpenConnection メソッド または CreateDatabase メソッド を使用して接続を開いたときだけです。

DatabaseOnCE プロパティ

Windows CE 上の Ultra Light データベースのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseOnCE As String

// C#
    public string  DatabaseOnCE {get;set;}
```

説明

値が `null` の場合、デフォルトのデータベース **¥UltraLiteDB¥ulstore.udb** が使用されます。C# では、パス内の円記号をエスケープする必要があります。

DatabaseOnDesktop プロパティ

Windows デスクトップ・プラットフォーム上の Ultra Light データベースのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property DatabaseOnDesktop As String

// C#
    public string  DatabaseOnDesktop {get;set;}
```

説明

値が `null` の場合、デフォルトのデータベース **ulstore.udb** が使用されます。C# では、パス内の円記号をエスケープする必要があります。

ParmsUsed プロパティ

このインスタンスを使用する最後の `DatabaseManager` メソッドによって実際に使用されたパラメータを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ParmsUsed As String

// C#
    public string  ParmsUsed {get;}
```

参照

- DatabaseNameParms クラス
- DatabaseNameParms メンバ
- CreateDatabase メソッド
- DropDatabase メソッド
- OpenConnection メソッド

Dispose メソッド

このオブジェクトによって使用されるリソースを解放します。

プロトタイプ

```
' Visual Basic
    Overloads Overrides Protected Sub Dispose( _
      ByVal disposing As Boolean _
    )

// C#
    protected override void Dispose(
      bool disposing
    );
```

パラメータ

- **disposing** マネージ・リソースとアンマネージ・リソースの両方を解放する場合は true、アンマネージ・リソースだけを解放する場合は false

ToString メソッド

このインスタンスの文字列表現を返します。

プロトタイプ

```
' Visual Basic
    Overrides Public Function ToString() As String

// C#
    public override string  ToString();
```

戻り値

このインスタンスの文字列表現

UnusedEvent イベント

使われていません。

プロトタイプ

```
' Visual Basic
    Public Event UnusedEvent As DatabaseNameParms.UnusedEventHandler

// C#
    public event DatabaseNameParms.UnusedEventHandler UnusedEvent;
```

説明

この public Event は、Visual Basic .NET プロジェクトでこのクラスを組み込む場合の Visual Studio .NET のバグを修正するために追加されています。実用的な用途はありません。

DatabaseNameParms.UnusedEventHandler 委譲

使われていません。

プロトタイプ

```
' Visual Basic
    Delegate Sub DatabaseNameParms.UnusedEventHandler( _
      ByVal sender As Object, _
      ByVal args As System.EventArgs _
    )

// C#
    delegate void DatabaseNameParms.UnusedEventHandler(
      object sender,
      System.EventArgs args
    );
```

パラメータ

- **sender** 送信側のオブジェクト
- **args** イベント引数

説明

この public Delegate は、Visual Basic .NET プロジェクトでこのクラスを組み込む場合の Visual Studio .NET のバグを修正するために追加されています。実用的な用途はありません。

DatabaseSchema クラス

Ultra Light データベースのスキーマを表します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class DatabaseSchema

// C#
    public sealed class DatabaseSchema
```

説明

このクラスを直接インスタンス化することはできません。DatabaseSchema クラス オブジェクトは、Schema プロパティ として接続にアタッチされ、接続が開いている間のみ有効です。

DatabaseSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
CollationName プロパティ	データベースの照合順の名前です。
IsCaseSensitive プロパティ	データベースで大文字と小文字が区別されるかどうかをチェックします。
IsOpen プロパティ	データベース・スキーマが開いているかどうかを示します。
PublicationCount プロパティ	データベース内のパブリケーションの数です。
Signature プロパティ	このデータベースのシグニチャです。
TableCount プロパティ	データベース内のテーブルの数です。

パブリック・インスタンス・メソッド

メンバ	説明
ApplyFile メソッド	データベース・スキーマ・ファイルをデータベースに適用するとともに、リスナに進行状況をモニタさせます。
ApplyFile メソッド	データベース・スキーマ・ファイルをデータベースに適用します。
GetDatabaseProperty メソッド	指定したデータベースのプロパティの値を返します。
GetPublicationName メソッド	指定したパブリケーション ID で識別されたパブリケーションの名前を返します。パブリケーション ID は、パブリケーション・マスクではありません。
GetPublicationSchema メソッド	指定したパブリケーションに対応するパブリケーション・スキーマを返します。
GetTableCountInPublications メソッド	指定したパブリケーション・マスクに含まれるテーブルの数を返します。
GetTableName メソッド	指定したテーブル ID で識別されたテーブルの名前を返します。

CollationName プロパティ

データベースの照合順の名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property CollationName As String

// C#
    public string  CollationName {get;}
```

説明

データベースの照合順は、テーブルのインデックスおよび結果セットのソート方法に影響します。

IsCaseSensitive プロパティ

データベースで大文字と小文字が区別されるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsCaseSensitive As Boolean

// C#
    public bool IsCaseSensitive {get;}
```

プロパティ値

データベースで大文字と小文字が区別される場合は true、区別されない場合は false

説明

データベースで大文字と小文字が区別されるかどうかは、テーブルのインデックスおよび結果セットのソート方法に影響します。また、大文字と小文字の区別は、UserID プロパティ と Password プロパティ の検証方法に影響します。

IsOpen プロパティ

データベース・スキーマが開いているかどうかを示します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

このデータベース・スキーマが現在開いていれば true

PublicationCount プロパティ

データベース内のパブリケーションの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PublicationCount As Integer
```

// C#
public int **PublicationCount** {get;}

説明

パブリケーション ID の範囲は、1 ～ `PublicationCount` です。パブリケーション ID は、パブリケーション・マスクではありません。データベースが開いていない場合、PublicationCount の値は 0 です。

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- GetPublicationName メソッド

Signature プロパティ

このデータベースのシグニチャです。

プロトタイプ

' Visual Basic
Public Readonly Property **Signature** As String

// C#
public string **Signature** {get;}

例外

- SQLException クラス - SQL エラーが発生しました。

TableCount プロパティ

データベース内のテーブルの数です。

プロトタイプ

' Visual Basic
Public Readonly Property **TableCount** As Integer

// C#
public int **TableCount** {get;}

説明

テーブル ID の範囲は、1 ～ `TableCount` です。接続が開いていない場合、TableCount の値は 0 です。

注意：テーブルの ID とカウントは、スキーマのアップグレード中に変更されることがあります。テーブルを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

ApplyFile メソッド

データベース・スキーマ・ファイルをデータベースに適用するとともに、リスナに進行状況をモニタさせます。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub ApplyFile( _
      ByVal parms As SchemaParms, _
      ByVal listener As SchemaUpgradeListener _
    )

// C#
    public void ApplyFile(
      SchemaParms parms,
      SchemaUpgradeListener listener
    );
```

パラメータ

- **parms** データベースに適用されるスキーマを指定するためのパラメータ。詳細については、SchemaParms クラス を参照してください。
- **listener** スキーマのアップグレードのプログレス・イベントを受信するリスナ

説明

TableSchema クラス、IndexSchema クラス、PublicationSchema クラス のすべてのインスタンスは無効になるため、置き換える必要があります。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- ApplyFile メソッド

- SchemaParms クラス

ApplyFile メソッド

データベース・スキーマ・ファイルをデータベースに適用します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub ApplyFile( _
      ByVal parms As SchemaParms _
    )

// C#
    public void ApplyFile(
      SchemaParms parms
    );
```

パラメータ

- **parms** データベースに適用されるスキーマを指定するためのパラメータ。詳細については、SchemaParms クラス を参照してください。

説明

TableSchema クラス、IndexSchema クラス、PublicationSchema クラスのすべてのインスタンスは無効になるため、置き換える必要があります。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- ApplyFile メソッド
- SchemaParms クラス

GetDatabaseProperty メソッド

指定したデータベースのプロパティの値を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetDatabaseProperty( _
      ByVal name As String _
    ) As String
```

```
// C#
    public string  GetDatabaseProperty(
      string  name
    );
```

パラメータ

- **name** データベースのプロパティの名前

戻り値

プロパティの文字列としての値

説明

識別されるプロパティは次のとおりです。

- "DATE_FORMAT" データベースによる文字列変換に使用される日付フォーマットです。

  このフォーマットは、DateTime によって使用されるフォーマットと同じであるとは限りません。

- "DATE_ORDER" データベースによる文字列変換に使用される日付順です。

- "NEAREST_CENTURY" データベースによる文字列変換に使用される最も近い世紀です。

- "PRECISION" データベースによる文字列変換に使用される浮動小数点の精度です。

- "SCALE" データベースによる文字列変換中に、計算結果が最大 PRECISION にトランケートされるときの小数点以下の最大桁数です。

- "TIME_FORMAT" データベースによる文字列変換に使用される時刻フォーマットです。

  このフォーマットは、TimeSpan によって使用されるフォーマットと同じであるとは限りません。

- "TIMESTAMP_FORMAT" データベースによる文字列変換に使用されるタイムスタンプのフォーマットです。

  このフォーマットは、DateTime によって使用されるフォーマットと同じであるとは限りません。

- `"TIMESTAMP_INCREMENT"` 2 つのユニークなタイムスタンプ間のナノ秒 (1,000,000 分の 1 秒) 単位の最小差です。

例外

- SQLException クラス - SQL エラーが発生しました。

GetPublicationName メソッド

指定したパブリケーション ID で識別されたパブリケーションの名前を返します。パブリケーション ID は、パブリケーション・マスクではありません。

プロトタイプ

```
' Visual Basic
    Public Function GetPublicationName( _
      ByVal publD As Integer _
    ) As String

// C#
    public string  GetPublicationName(
      int publD
    );
```

パラメータ

- **pubID** パブリケーションの ID。*publD* は、`[1,DatabaseSchema.PublicationCount]` の範囲内である必要があります。

戻り値

パブリケーション名

説明

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- PublicationCount プロパティ

GetPublicationSchema メソッド

指定したパブリケーションに対応するパブリケーション・スキーマを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetPublicationSchema( _
      ByVal name As String _
    ) As PublicationSchema

// C#
    public PublicationSchema GetPublicationSchema(
      string  name
    );
```

パラメータ

- **name** パブリケーションの名前

戻り値

指定したパブリケーションのパブリケーション・スキーマ

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- GetPublicationName メソッド
- PublicationSchema クラス

GetTableCountInPublications メソッド

指定したパブリケーション・マスクに含まれるテーブルの数を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTableCountInPublications( _
      ByVal mask As Integer _
    ) As Integer

// C#
    public int GetTableCountInPublications(
      int mask
    );
```

パラメータ

- **mask** チェック対象のパブリケーションのセット

戻り値

パブリケーションのセットに含まれるテーブル数

説明

カウントには、名前が _nosync で終わるテーブルは含まれません。

注意：パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされている ID、マスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

GetTableName メソッド

指定したテーブル ID で識別されたテーブルの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetTableName( _
      ByVal tableID As Integer _
    ) As String

// C#
    public string  GetTableName(
      int tableID
    );
```

パラメータ

- **tableID** テーブルの ID。*tableID* は、`[1,TableCount]` の範囲内である必要があります。

戻り値

テーブル名

説明

テーブル ID は、スキーマのアップグレード中に変更されることがあります。テーブルを正しく識別するには、名前でアクセスするか、キャッシュされている ID をスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- DatabaseSchema クラス
- DatabaseSchema メンバ
- TableCount プロパティ

IndexSchema クラス

Ultra Light テーブルのインデックスのスキーマを表します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **IndexSchema**

// C#
public sealed class **IndexSchema**

説明

このクラスを直接インスタンス化することはできません。インデックスのスキーマは、TableSchema クラスの PrimaryKey プロパティ メソッド、GetIndex メソッド、GetOptimalIndex メソッド を使用して作成されます。

IndexSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ	このインデックス内のカラム数です。
IsForeignKey プロパティ	インデックスが外部キーであるかどうかをチェックします。
IsForeignKeyCheckOnCommit プロパティ	この外部キーの参照整合性のチェックが、コミット時に行われるか、挿入時および更新時に行われるかを確認します。
IsForeignKeyNullable プロパティ	この外部キーが null 入力可能であるかどうかをチェックします。
IsOpen プロパティ	インデックス・スキーマが開いているか、閉じているかを調べます。
IsPrimaryKey プロパティ	インデックスがプライマリ・キーであるかどうかをチェックします。

メンバ	説明
IsUniqueIndex プロパティ	インデックスがユニークであるかどうかをチェックします。
IsUniqueKey プロパティ	インデックスがユニーク・キーであるかどうかをチェックします。
Name プロパティ	このインデックスの名前です。
ReferencedIndexName プロパティ	このインデックスが外部キーである場合、参照されるプライマリ・インデックスの名前です。
ReferencedTableName プロパティ	インデックスが外部キーである場合、参照されるプライマリ・テーブルの名前です。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnName メソッド	このインデックス内の *collDInIndex* 番目のカラム名を返します。
IsColumnDescending メソッド	指定したカラムが、このインデックスによって降順で使用されるかどうかをチェックします。

ColumnCount プロパティ

このインデックス内のカラム数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ColumnCount As Short

// C#
    public short ColumnCount {get;}
```

説明

インデックス内のカラム ID の範囲は、1 ～ `ColumnCount` です。

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスからのカラム ID は、テーブルまたは別のインデックスのカラム ID とは異なります。

IsForeignKey プロパティ

インデックスが外部キーであるかどうかをチェックします。

プロトタイプ

' Visual Basic
Public Readonly Property **IsForeignKey** As Boolean

// C#
public bool **IsForeignKey** {get;}

プロパティ値

インデックスが外部キーである場合は true、外部キーでない場合は false

説明

外部キー内のカラムは、別のテーブルの null 以外のユニーク・インデックスを参照することがあります。

例外

- SQLException クラス - SQL エラーが発生しました。

IsForeignKeyCheckOnCommit プロパティ

この外部キーの参照整合性のチェックが、コミット時に行われるか、挿入時および更新時に行われるかを確認します。

プロトタイプ

' Visual Basic
Public Readonly Property **IsForeignKeyCheckOnCommit** As Boolean

// C#
public bool **IsForeignKeyCheckOnCommit** {get;}

プロパティ値

参照整合性がコミット時にチェックされる場合は true、挿入時および更新時にチェックされる場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- IndexSchema クラス
- IndexSchema メンバ
- IsForeignKey プロパティ

IsForeignKeyNullable プロパティ

この外部キーが null 入力可能であるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsForeignKeyNullable As Boolean

// C#
    public bool IsForeignKeyNullable {get;}
```

プロパティ値

この外部キーが null 入力可能であれば true、null 入力不可であれば false

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- IndexSchema クラス
- IndexSchema メンバ
- IsForeignKey プロパティ

IsOpen プロパティ

インデックス・スキーマが開いているか、閉じているかを調べます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

インデックス・スキーマが開いている場合は true、閉じている場合は false

IsPrimaryKey プロパティ

インデックスがプライマリ・キーであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsPrimaryKey As Boolean
```

```
// C#
    public bool IsPrimaryKey {get;}
```

プロパティ値　インデックスがプライマリ・キーである場合は true、プライマリ・キーでない場合は false

説明　プライマリ・キー内のカラムでは null は許可されません。

例外
- SQLException クラス - SQL エラーが発生しました。

IsUniqueIndex プロパティ

インデックスがユニークであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsUniqueIndex As Boolean

// C#
    public bool IsUniqueIndex {get;}
```

プロパティ値　インデックスがユニークである場合は true、ユニークでない場合は false

説明　ユニークなインデックス内のカラムは null であることがあります。

例外
- SQLException クラス - SQL エラーが発生しました。

IsUniqueKey プロパティ

インデックスがユニーク・キーであるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsUniqueKey As Boolean

// C#
    public bool IsUniqueKey {get;}
```

プロパティ値　インデックスがユニーク・キーである場合は true、ユニーク・キーでない場合は false

説明 ユニーク・キー内のカラムでは null は許可されません。

例外
- SQLException クラス - SQL エラーが発生しました。

Name プロパティ

このインデックスの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

例外
- SQLException クラス - SQL エラーが発生しました。

ReferencedIndexName プロパティ

このインデックスが外部キーである場合、参照されるプライマリ・インデックスの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReferencedIndexName As String

// C#
    public string  ReferencedIndexName {get;}
```

例外
- SQLException クラス - SQL エラーが発生しました。

ReferencedTableName プロパティ

インデックスが外部キーである場合、参照されるプライマリ・テーブルの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReferencedTableName As String
```

```
// C#
    public string  ReferencedTableName {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。

GetColumnName メソッド

このインデックス内の *collDInIndex* 番目のカラム名を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnName( _
      ByVal collDInIndex As Short _
    ) As String

// C#
    public string  GetColumnName(
      short collDInIndex
    );
```

パラメータ

- **colIDInIndex** このカラムのインデックス内の ID。*collDInIndex* は、`[1,IndexSchema.ColumnCount]` の範囲内である必要があります。

戻り値

カラムの名前

説明

カラムの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスからのカラム ID は、テーブルまたは別のインデックスのカラム ID とは異なります。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- IndexSchema クラス
- IndexSchema メンバ
- ColumnCount プロパティ

IsColumnDescending メソッド

指定したカラムが、このインデックスによって降順で使用されるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnDescending( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnDescending(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムが降順で使用される場合は true、昇順で使用される場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- IndexSchema クラス
- IndexSchema メンバ
- ColumnCount プロパティ

PreparedStatement クラス

IN パラメータ付きまたは IN パラメータなしで事前にコンパイルされた SQL 文を表します。

プロトタイプ

' Visual Basic

```
Public Class PreparedStatement
```

// C#

```
public class PreparedStatement
```

説明

このクラスを直接インスタンス化することはできません。準備文は、Connection クラス の PrepareStatement メソッド を使用して作成されます。このオブジェクトを使用すると、この文を効率的に何回も実行できます。

準備文が閉じられると、それに関連する ResultSet オブジェクトと ResultSetSchema オブジェクトもすべて閉じられます。PreparedStatement オブジェクトは、ガーベジ・コレクションの実行時に閉じられます。ただし、すべての ResultSet オブジェクトと ResultSetSchema オブジェクトは準備文を参照するため、開いているすべての ResultSet オブジェクトと ResultSetSchema オブジェクトのガーベジ・コレクションが実行できる状態になるまで、準備文のガーベジ・コレクションは行われません。

リソース管理の理由により、準備文を使用し終わったら、その準備文を明示的に閉じることをおすすめします。

PreparedStatement メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
HasResultSet プロパティ	準備文が結果セットを生成するかどうかを示します。
IsOpen プロパティ	準備文が現在開いているかどうかを確認します。

メンバ	説明
Plan プロパティ	クエリを実行するのに Ultra Light が使用するアクセス・プランを返します。このプロパティは、主に開発中の使用を目的とします。
ResultSetSchema プロパティ	このクエリ文の結果セットを記述するスキーマを保持します。

パブリック・インスタンス・メソッド

メンバ	説明
AppendBytesParameter メソッド	指定した Byte 配列 (C# では **byte[]**、Visual Basic では **Byte()**) の指定したサブセットを、指定した SQLType 列挙 パラメータの新しい値に追加します。
AppendCharsParameter メソッド	指定した System.Char 配列 (C# では **char[]**、Visual Basic では **Char()**) の指定したサブセットを、指定した SQLType 列挙 パラメータの新しい値に追加します。
Close メソッド	準備文を閉じます。
ExecuteQuery メソッド	SQL SELECT 文を実行し、結果セットを返します。
ExecuteStatement メソッド	SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。
SetBoolParameter メソッド	指定したパラメータの値を、Boolean (C# では **bool**) を使用して設定します。
SetBytesParameter メソッド	指定したパラメータの値を、Byte 配列 (C# では **byte[]**、Visual Basic では **Byte()**) を使用して設定します。
SetDateTimeParameter メソッド	指定したパラメータの値を、DateTime (Visual Basic では **Date**) を使用して設定します。
SetDoubleParameter メソッド	指定したパラメータの値を、Double (C# では **double**) を使用して設定します。

メンバ	説明
SetFloatParameter メソッド	指定したパラメータの値を、Single (C# では **float**) を使用して設定します。
SetIntParameter メソッド	指定したパラメータの値を、Int32 (C# では **long**、Visual Basic では **Long**) を使用して設定します。
SetLongParameter メソッド	指定したパラメータの値を、Int64 を使用して設定します。
SetNullParameter メソッド	パラメータを NULL に設定します。
SetShortParameter メソッド	指定したパラメータの値を、Int16 (C# では **short**、Visual Basic では **Short**) を使用して設定します。
SetStringParameter メソッド	指定したパラメータの値を、String を使用して設定します。
SetTimeSpanParameter メソッド	指定したパラメータの値を、TimeSpan を使用して設定します。
SetUIntParameter メソッド	指定したパラメータの値を、UInt32 (C# では **uint**) を使用して設定します。
SetULongParameter メソッド	指定したパラメータの値を、UInt64 (C# では **ulong**) を使用して設定します。
SetUShortParameter メソッド	指定したパラメータの値を、UInt16 (C# では **ushort**) を使用して設定します。
SetUUIDParameter メソッド	指定したパラメータの値を、UUID (Guid) を使用して設定します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	Finalize メソッドです。

HasResultSet プロパティ

準備文が結果セットを生成するかどうかを示します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property HasResultSet As Boolean

// C#
    public bool HasResultSet {get;}
```

説明

この文が実行されたときに結果セットが生成される場合は true、生成されない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

IsOpen プロパティ

準備文が現在開いているかどうかを確認します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

戻り値

この準備文が現在開いている場合は true、閉じている場合は false

Plan プロパティ

クエリを実行するのに Ultra Light が使用するアクセス・プランを返します。このプロパティは、主に開発中の使用を目的とします。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Plan As String

// C#
    public string  Plan {get;}
```

戻り値

クエリ実行プランのテキストベースの記述

例外

- SQLException クラス - SQL エラーが発生しました。

ResultSetSchema プロパティ

このクエリ文の結果セットを記述するスキーマを保持します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ResultSetSchema As ResultSetSchema

// C#
    public ResultSetSchema ResultSetSchema {get;}
```

戻り値

この文の結果セットのスキーマ

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- HasResultSet プロパティ
- ExecuteQuery メソッド

AppendBytesParameter メソッド

指定した Byte 配列 (C# では **byte[]**、Visual Basic では **Byte()**) の指定したサブセットを、指定した SQLType 列挙 パラメータの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendBytesParameter( _
      ByVal parameterID As Short, _
      ByVal val As Byte(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )

// C#
    public void AppendBytesParameter(
      short parameterID,
      byte[] val,
```

```
    int srcOffset,
    int count
);
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの現在の新しい値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置のバイトが、指定したパラメータの値に追加されます。

パラメータを指定すると、実行後のパラメータへの最初の追加では、現在の値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、処理対象は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *val*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド

- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

AppendCharsParameter メソッド

指定した System.Char 配列 (C# では **char[]**、Visual Basic では **Char()**) の指定したサブセットを、指定した SQLType 列挙 パラメータの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendCharsParameter( _
      ByVal parameterID As Short, _
      ByVal val As Char(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )

// C#
    public void AppendCharsParameter(
      short parameterID,
      char[] val,
      int srcOffset,
      int count
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの現在の値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置の文字が、指定したパラメータの値に追加されます。

パラメータを指定すると、実行後のパラメータへの最初の追加では、現在の値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、処理対象は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *value*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

Close メソッド

準備文を閉じます。

プロトタイプ

```
' Visual Basic
    Public Sub Close()

// C#
    public void Close();
```

説明

準備文が閉じられると、それに関連する ResultSet オブジェクトと ResultSetSchema オブジェクトもすべて閉じられます。

例外

- SQLException クラス - SQL エラーが発生しました。

ExecuteQuery メソッド

SQL SELECT 文を実行し、結果セットを返します。

プロトタイプ

```
' Visual Basic
    Public Function ExecuteQuery() As ResultSet

// C#
    public ResultSet ExecuteQuery();
```

戻り値

文からのローの結果セット

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- HasResultSet プロパティ
- ExecuteStatement メソッド

ExecuteStatement メソッド

SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。

プロトタイプ

```
' Visual Basic
    Public Function ExecuteStatement() As Long

// C#
    public long ExecuteStatement();
```

戻り値

文の影響を受けるローの数

説明

AutoCommit フィールド が true の場合は、文が 1 つまたは複数のローに影響するときだけ、文がコミットされます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- HasResultSet プロパティ
- ExecuteQuery メソッド

Finalize メソッド

Finalize メソッドです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

SetBoolParameter メソッド

指定したパラメータの値を、Boolean (C# では **bool**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBoolParameter( _
      ByVal parameterID As Short, _
      ByVal val As Boolean _
    )
```

```
// C#
    public void SetBoolParameter(
      short parameterID,
      bool val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetBytesParameter メソッド

指定したパラメータの値を、Byte 配列 (C# では **byte[]**、Visual Basic では **Byte()**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBytesParameter( _
      ByVal parameterID As Short, _
      ByVal val As Byte() _
    )

// C#
    public void SetBytesParameter(
      short parameterID,
      byte[] val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

説明

SQLType 列挙 型、SQLType 列挙 型のパラメータにのみ適しています。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetDateTimeParameter メソッド

指定したパラメータの値を、DateTime (Visual Basic では **Date**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDateTimeParameter( _
      ByVal parameterID As Short, _
      ByVal val As Date _
    )

// C#
    public void SetDateTimeParameter(
      short parameterID,
      DateTime val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

説明

設定された値の精度はミリ秒です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetUIntParameter メソッド

- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetDoubleParameter メソッド

指定したパラメータの値を、Double (C# では **double**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDoubleParameter( _
      ByVal parameterID As Short, _
      ByVal val As Double _
    )

// C#
    public void SetDoubleParameter(
      short parameterID,
      double val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド

- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetFloatParameter メソッド

指定したパラメータの値を、Single (C# では **float**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetFloatParameter( _
      ByVal parameterID As Short, _
      ByVal val As Single _
    )

// C#
    public void SetFloatParameter(
      short parameterID,
      float  val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetIntParameter メソッド

- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetIntParameter メソッド

指定したパラメータの値を、Int32 (C# では **long**、Visual Basic では **Long**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetIntParameter( _
      ByVal parameterID As Short, _
      ByVal val As Integer _
    )

// C#
    public void SetIntParameter(
      short parameterID,
      int val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド

- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetLongParameter メソッド

指定したパラメータの値を、Int64 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetLongParameter( _
      ByVal parameterID As Short, _
      ByVal val As Long _
    )

// C#
    public void SetLongParameter(
      short parameterID,
      long val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド

- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetNullParameter メソッド

パラメータを NULL に設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetNullParameter( _
      ByVal parameterID As Short _
    )

// C#
    public void SetNullParameter(
      short parameterID
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド

- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetShortParameter メソッド

指定したパラメータの値を、Int16 (C# では **short**、Visual Basic では **Short**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetShortParameter( _
      ByVal parameterID As Short, _
      ByVal val As Short _
    )

// C#
    public void SetShortParameter(
      short parameterID,
      short val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド

- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetStringParameter メソッド

指定したパラメータの値を、String を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetStringParameter( _
      ByVal parameterID As Short, _
      ByVal val As String _
    )

// C#
    public void SetStringParameter(
      short parameterID,
      string  val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド

- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetTimeSpanParameter メソッド

指定したパラメータの値を、TimeSpan を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetTimeSpanParameter( _
      ByVal parameterID As Short, _
      ByVal val As TimeSpan _
    )

// C#
    public void SetTimeSpanParameter(
      short parameterID,
      TimeSpan val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

説明

設定された値は、ミリ秒の精度であり、0 ～ 24 時間の負でない値に正規化されます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetUIntParameter メソッド

指定したパラメータの値を、UInt32 (C# では **uint**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUIntParameter( _
      ByVal parameterID As Short, _
      ByVal val As UInt32 _
    )

// C#
    public void SetUIntParameter(
      short parameterID,
      uint val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetULongParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetULongParameter メソッド

指定したパラメータの値を、UInt64 (C# では **ulong**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetULongParameter( _
      ByVal parameterID As Short, _
      ByVal val As UInt64 _
    )
```

```
// C#
public void SetULongParameter(
  short parameterID,
  ulong val
);
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetUShortParameter メソッド
- SetUUIDParameter メソッド

SetUShortParameter メソッド

指定したパラメータの値を、UInt16 (C# では **ushort**) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUShortParameter( _
      ByVal parameterID As Short, _
      ByVal val As UInt16 _
    )

// C#
    public void SetUShortParameter(
      short parameterID,
      ushort val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetStringParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド
- SetUUIDParameter メソッド

SetUUIDParameter メソッド

指定したパラメータの値を、UUID (Guid) を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUUIDParameter( _
      ByVal parameterID As Short, _
      ByVal val As Guid _
    )

// C#
    public void SetUUIDParameter(
      short parameterID,
      Guid val
    );
```

パラメータ

- **parameterID** パラメータの ID 番号。文の最初のパラメータの ID 値は 1 です。
- **val** パラメータの新しい値

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- PreparedStatement クラス
- PreparedStatement メンバ
- GetNewUUID メソッド
- SetNullParameter メソッド
- AppendBytesParameter メソッド
- AppendCharsParameter メソッド
- SetBytesParameter メソッド
- SetBoolParameter メソッド
- SetDoubleParameter メソッド
- SetFloatParameter メソッド
- SetIntParameter メソッド
- SetLongParameter メソッド
- SetShortParameter メソッド
- SetTimeSpanParameter メソッド
- SetDateTimeParameter メソッド
- SetUIntParameter メソッド
- SetULongParameter メソッド

- SetUShortParameter メソッド

PublicationSchema クラス

Ultra Light パブリケーションのスキーマを表します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **PublicationSchema**

// C#
public sealed class **PublicationSchema**

説明

このクラスを直接インスタンス化することはできません。パブリケーションのスキーマは、DatabaseSchema クラスの GetPublicationSchema メソッド を使用して作成されます。

パブリケーション・マスクが必要な Ultra Light のメソッドには、実際にはチェック対象のパブリケーションのセットが必要です。セットは、個々のパブリケーションのパブリケーション・マスクの論理和で形成されます。次に例を示します。

```
pub1.Mask | pub2.Mask
```

特別な 2 つのマスク値も、このクラスによって提供されます。SYNC_ALL_DB フィールド は、データベース全体に対応します。SYNC_ALL_PUBS フィールド は、すべてのパブリケーションに対応します。

注意：パブリケーション・マスクは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされているマスクをスキーマのアップグレード後にリフレッシュします。

参照

- PublicationSchema メンバ
- Mask プロパティ
- GetLastDownloadTime メソッド
- CountUploadRows メソッド
- Schema プロパティ

PublicationSchema メンバ

パブリック静的フィールド (共有)

メンバ	説明
SYNC_ALL_DB フィールド	データベース全体に対応するパブリケーション・マスクです。
SYNC_ALL_PUBS フィールド	すべてのパブリケーションに対応するパブリケーション・マスクです。

パブリック・インスタンス・プロパティ

メンバ	説明
IsOpen プロパティ	パブリケーション・スキーマが開いているか、閉じているかを調べます。
Mask プロパティ	このパブリケーションのパブリケーション・マスクです。
Name プロパティ	このパブリケーションの名前です。

SYNC_ALL_DB フィールド

データベース全体に対応するパブリケーション・マスクです。

プロトタイプ

```
' Visual Basic
    Public Shared SYNC_ALL_DB As Integer

// C#
    public const int SYNC_ALL_DB;
```

SYNC_ALL_PUBS フィールド

すべてのパブリケーションに対応するパブリケーション・マスクです。

プロトタイプ

```
' Visual Basic
    Public Shared SYNC_ALL_PUBS As Integer
```

```
// C#
    public const int SYNC_ALL_PUBS;
```

IsOpen プロパティ

パブリケーション・スキーマが開いているか、閉じているかを調べます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsOpen As Boolean

// C#
    public bool IsOpen {get;}
```

プロパティ値

パブリケーション・スキーマが開いている場合は true、閉じている場合は false

Mask プロパティ

このパブリケーションのパブリケーション・マスクです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Mask As Integer

// C#
    public int Mask {get;}
```

説明

パブリケーションの ID、マスク、カウントは、スキーマのアップグレード中に変更されることがあります。パブリケーションを正しく識別するには、名前でアクセスするか、キャッシュされているマスク、カウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーによる例外が発生した場合。スキーマのアップグレードでこのパブリケーションが削除されるか、名前が変更された場合は、PUBLICATION_NOT_FOUND が発行されます。

Name プロパティ

このパブリケーションの名前です。

プロトタイプ

' Visual Basic

```
Public Readonly Property Name As String
```

// C#

```
public string  Name {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。スキーマのアップグレードでこのパブリケーションが削除されるか、名前が変更された場合は、PUBLICATION_NOT_FOUND が発行されます。

ResultSet クラス

Ultra Light データベースの結果セットを示します。実行時に ExecuteQuery メソッド を使用して作成されます。

プロトタイプ

```
' Visual Basic
    Public Class ResultSet
      Inherits Cursor

// C#
    public class ResultSet :
      Cursor
```

説明

このクラスを直接インスタンス化することはできません。結果セットは、PreparedStatement クラス の ExecuteQuery メソッドを使用して作成されます。

結果セットが有効なのは、準備文が開かれている間だけです。

ResultSet メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
IsBOF プロパティ (継承 Cursor)	現在のローの位置が最初のローの前なら true を返します。
IsEOF プロパティ (継承 Cursor)	現在のローの位置が最後のローの後なら true を返します。
IsOpen プロパティ (継承 Cursor)	カーソルが現在開いているかどうかを確認します。
RowCount プロパティ (継承 Cursor)	カーソル内のロー数を返します。
Schema プロパティ	この結果セットのスキーマを保持します。このプロパティが有効なのは、その準備文が開かれている間だけです。

パブリック・インスタンス・メソッド

メンバ	説明
Close メソッド	結果セットを閉じます。
Fill メソッド (継承 Cursor)	ADO .NET DataTable オブジェクトに、カーソル内のローを格納します。
GetBoolean メソッド (継承 Cursor)	指定したカラムの値を Boolean として返します。
GetBytes メソッド (継承 Cursor)	指定したカラムの値を Byte 配列として返します。SQLType 列挙 型、SQLType 列挙 型、SQLType 列挙 型のカラムの場合にのみ有効です。
GetBytes メソッド (継承 Cursor)	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。
GetChars メソッド (継承 Cursor)	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。
GetDouble メソッド (継承 Cursor)	指定したカラムの値を Double として返します。
GetFloat メソッド (継承 Cursor)	指定したカラムの値を Single として返します。
GetInt メソッド (継承 Cursor)	指定したカラムの値を Int32 として返します。
GetLong メソッド (継承 Cursor)	指定したカラムの値を Int64 として返します。
GetShort メソッド (継承 Cursor)	指定したカラムの値を Int16 として返します。
GetString メソッド (継承 Cursor)	指定したカラムの値を String として返します。

メンバ	説明
GetTime メソッド (継承 Cursor)	指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。
GetTimestamp メソッド (継承 Cursor)	指定したカラムの値を、ミリ秒の精度の DateTime として返します。
GetUInt メソッド (継承 Cursor)	指定したカラムの値を UInt32 として返します。
GetULong メソッド (継承 Cursor)	指定したカラムの値を UInt64 として返します。
GetUShort メソッド (継承 Cursor)	指定したカラムの値を UInt16 として返します。
GetUUID メソッド (継承 Cursor)	指定したカラムの値を UUID (Guid) として返します。
IsNull メソッド (継承 Cursor)	指定されたカラムの値が NULL かどうかをチェックします。
MoveAfterLast メソッド (継承 Cursor)	カーソルの最後のローの後に移動します。
MoveBeforeFirst メソッド (継承 Cursor)	カーソルの最初のローの前に移動します。
MoveFirst メソッド (継承 Cursor)	カーソルの最初のローに移動します。
MoveLast メソッド (継承 Cursor)	カーソルの最後のローに移動します。
MoveNext メソッド (継承 Cursor)	次のローに、または最後のローの後に移動します。
MovePrevious メソッド (継承 Cursor)	前のローに、または最初のローの前に移動します。
MoveRelative メソッド (継承 Cursor)	現在のローを基準として移動します。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	Finalize メソッドです。

Schema プロパティ

この結果セットのスキーマを保持します。このプロパティが有効なのは、その準備文が開かれている間だけです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Schema As ResultSetSchema

// C#
    public ResultSetSchema Schema {get;}
```

参照

- ResultSet クラス
- ResultSet メンバ
- ResultSetSchema クラス

Close メソッド

結果セットを閉じます。

プロトタイプ

```
' Visual Basic
    NotOverridable Public Sub Close()

// C#
    public void Close();
```

説明

すでに閉じられている結果セットを閉じるのはエラーではありません。

例外

- SQLException クラス - SQL エラーが発生しました。

Finalize メソッド

Finalize メソッドです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

ResultSetSchema クラス

Ultra Light の結果セットのスキーマを表します。

プロトタイプ

```
' Visual Basic
NotInheritable Public Class ResultSetSchema
  Inherits CursorSchema

// C#
public sealed class ResultSetSchema :
  CursorSchema
```

説明

このクラスを直接インスタンス化することはできません。ResultSetSchema クラス オブジェクトは、その Schema プロパティ として結果セットにアタッチされます。準備文がクエリを表している場合 (準備文の HasResultSet プロパティ が true のとき)、結果セットのスキーマも、ResultSetSchema プロパティとして準備文にアタッチされます。

結果セットのスキーマが有効なのは、準備文が開かれている間だけです。

参照

- ResultSetSchema メンバ
- PreparedStatement クラス
- ResultSet クラス

ResultSetSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ (継承 CursorSchema)	このカーソル内のカラム数を返します。
IsOpen プロパティ (継承 CursorSchema)	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	このカーソルの名前です。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnID メソッド (継承 CursorSchema)	指定したカラムのカラム ID を返します。
GetColumnName メソッド (継承 CursorSchema)	指定したカラム ID で識別されたカラムの名前を返します。
GetColumnPrecision メソッド (継承 CursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの精度を返します。
GetColumnScale メソッド (継承 CursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの位取りを返します。
GetColumnSize メソッド (継承 CursorSchema)	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラムのサイズを返します。
GetColumnSQLType メソッド (継承 CursorSchema)	指定したカラムの SQL データ型を取得します。

Name プロパティ

このカーソルの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

戻り値

この ResultSetSchema を生成した SQL 文

RuntimeType 列挙

Ultra Light.NET ランタイムのタイプを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum RuntimeType
     Inherits Short

// C#
    public enum RuntimeType :
     short
```

メンバ

メンバ	説明
STANDALONE_UL	スタンドアロンの Ultra Light.NET ランタイムを選択します。 スタンドアロンのランタイムは、データベースに直接アクセスします。データベースへのアクセスは高速になりますが、データベースを共有することはできません。
UL_ENGINE_CLIENT	Ultra Light エンジンを使用することを選択します。Ultra Light.NET エンジン・クライアントは、Ultra Light エンジンと通信してデータベースにアクセスします。つまり、別々のアプリケーションでデータベースを共有できます。

参照

- DatabaseManager コンストラクタ

SchemaParms クラス

Ultra Light データベースのスキーマを指定します。

プロトタイプ

```
' Visual Basic
    NotInheritable Public Class SchemaParms
     Inherits Component

// C#
    public sealed class SchemaParms :
     Component
```

説明

各インスタンスには、スキーマ・ファイルへのプラットフォーム固有のパスと汎用のパスがあります。プラットフォーム固有のキーワードでは、1 つのファイル値だけを使用し、汎用キーワードより優先されます。たとえば、`schemaParms.SchemaOnCE="¥¥s1.usm";` および `schemaParms.SchemaOnDesktop="s2.usm";` では、パス `¥s1.usm` は Windows CE で使用され、`s2.usm` はその他のプラットフォームで使用されます。

インスタンスが Ultra Light のメソッドに渡されると、プロパティ `ParmsUsed` は Ultra Light のメソッドによって実際に使用されるパラメータを返します。

Ultra Light スキーマ・ファイルに推奨されるファイル拡張子は **.usm** です。パス内の円記号は、エスケープする必要があります。

SchemaParms メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
SchemaParms コンストラクタ	SchemaParms インスタンスを、そのデフォルト値で初期化します。
SchemaParms コンストラクタ	SchemaParms インスタンスを、指定した値で初期化します。

パブリック・インスタンス・プロパティ

メンバ	説明
AdditionalParms プロパティ	*名前*= *値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。これは、今後の使用のために予約されています。
ParmsUsed プロパティ	このインスタンスを使用する最後の DatabaseSchema メソッドによって実際に使用されたパラメータを返します。
SchemaOnCE プロパティ	Windows CE 上の Ultra Light スキーマのパスとファイル名を指定します。
SchemaOnDesktop プロパティ	Windows デスクトップ・プラットフォーム 上の Ultra Light スキーマのパスとファイル名を指定します。

パブリック・インスタンス・メソッド

メンバ	説明
ToString メソッド	このインスタンスの文字列表現を返します。

SchemaParms コンストラクタ

SchemaParms インスタンスを、そのデフォルト値で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New()

// C#
    public SchemaParms();
```

例外

- SQLException クラス - SQL エラーが発生しました。

SchemaParms コンストラクタ

SchemaParms インスタンスを、指定した値で初期化します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal schemaOnCE As String, _
      ByVal schemaOnDesktop As String, _
      ByVal additionalParms As String _
    )

// C#
    public SchemaParms(
      string  schemaOnCE,
      string  schemaOnDesktop,
      string  additionalParms
    );
```

パラメータ

- **schemaOnCE** SchemaOnCE プロパティ の値
- **schemaOnDesktop** SchemaOnDesktop プロパティ の値
- **additionalParms** AdditionalParms プロパティ の値

例外

- SQLException クラス - SQL エラーが発生しました。

AdditionalParms プロパティ

*名前= 値*の組み合わせをセミコロンで区切ったリストで、追加のパラメータを指定します。これは、今後の使用のために予約されています。

プロトタイプ

```
' Visual Basic
    Public Property AdditionalParms As String

// C#
    public string  AdditionalParms {get;set;}
```

ParmsUsed プロパティ

このインスタンスを使用する最後の DatabaseSchema メソッドによって実際に使用されたパラメータを返します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ParmsUsed As String

// C#
    public string  ParmsUsed {get;}
```

参照

- SchemaParms クラス
- SchemaParms メンバ
- ApplyFile メソッド

SchemaOnCE プロパティ

Windows CE 上の Ultra Light スキーマのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property SchemaOnCE As String

// C#
    public string  SchemaOnCE {get;set;}
```

SchemaOnDesktop プロパティ

Windows デスクトップ・プラットフォーム 上の Ultra Light スキーマのパスとファイル名を指定します。

プロトタイプ

```
' Visual Basic
    Public Property SchemaOnDesktop As String

// C#
    public string  SchemaOnDesktop {get;set;}
```

ToString メソッド

このインスタンスの文字列表現を返します。

プロトタイプ

```
' Visual Basic
    Overrides Public Function ToString() As String

// C#
    public override string  ToString();
```

戻り値

このインスタンスの文字列表現

SchemaUpgradeData クラス

スキーマのアップグレードのモニタリング・データを返します。

プロトタイプ

' Visual Basic
Public Class **SchemaUpgradeData**

// C#
public class **SchemaUpgradeData**

参照

- SchemaUpgradeData メンバ
- SchemaUpgradeListener インタフェース

SchemaUpgradeData メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
FinalProgressCount プロパティ	最終的な進行状況のカウントの推定値です。
ProgressCounter プロパティ	スキーマのアップグレードの進行状況のカウンタです。この数は、現在までの進行状況を表す近似値です。
State プロパティ	現在のスキーマのアップグレードのステータスです。
UpgradeOperations プロパティ	実行されたスキーマのアップグレード操作の数です。

FinalProgressCount プロパティ

最終的な進行状況のカウントの推定値です。

プロトタイプ

' Visual Basic
Public Readonly Property **FinalProgressCount** As Long

```
// C#
    public long FinalProgressCount {get;}
```

戻り値

推定される最終的な進行状況のカウント

参照

- SchemaUpgradeData クラス
- SchemaUpgradeData メンバ
- SchemaUpgradeState 列挙
- ProgressCounter プロパティ

ProgressCounter プロパティ

スキーマのアップグレードの進行状況のカウンタです。この数は、現在までの進行状況を表す近似値です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ProgressCounter As Long

// C#
    public long ProgressCounter {get;}
```

戻り値

スキーマのアップグレードの進行状況のカウンタ (0 ～ FinalProgressCount プロパティ の数)

参照

- SchemaUpgradeData クラス
- SchemaUpgradeData メンバ
- SchemaUpgradeState 列挙

State プロパティ

現在のスキーマのアップグレードのステータスです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property State As SchemaUpgradeState

// C#
    public SchemaUpgradeState State {get;}
```

参照

- SchemaUpgradeData クラス

- SchemaUpgradeData メンバ
- SchemaUpgradeState 列挙

UpgradeOperations プロパティ

実行されたスキーマのアップグレード操作の数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UpgradeOperations As Long

// C#
    public long UpgradeOperations {get;}
```

説明

この数は、スキーマのアップグレード中に行われた作業の量の近似値です。この数は 0 から開始し、アップグレードの進行に伴って増加します。進行状況のカウンタよりも頻繁に更新されるため、ほかのスキーマ・アップグレードと比較するための相対的な計測値として使用できます。

参照

- SchemaUpgradeData クラス
- SchemaUpgradeData メンバ
- SchemaUpgradeState 列挙

SchemaUpgradeListener インタフェース

スキーマのアップグレードのプログレス・イベントを受信するためのリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface SchemaUpgradeListener

// C#
    public interface SchemaUpgradeListener
```

参照

- SchemaUpgradeListener メンバ
- ApplyFile メソッド

SchemaUpgradeListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
SchemaUpgrading メソッド	ユーザに進行状況を通知するために、スキーマのアップグレード中に起動されます。ステータスが SchemaUpgradeState 列挙 であるときは、このメソッドは true を返してスキーマのアップグレードをキャンセルするか、false を返して続行する必要があります。その他のステータスの場合、このメソッドの戻り値は無視されます。

SchemaUpgrading メソッド

ユーザに進行状況を通知するために、スキーマのアップグレード中に起動されます。ステータスが SchemaUpgradeState 列挙 であるときは、このメソッドは true を返してスキーマのアップグレードをキャンセルするか、false を返して続行する必要があります。その他のステータスの場合、このメソッドの戻り値は無視されます。

プロトタイプ

```
' Visual Basic
    Public Function SchemaUpgrading( _
      ByVal data As SchemaUpgradeData _
    ) As Boolean

// C#
    public bool SchemaUpgrading(
      SchemaUpgradeData data
    );
```

パラメータ

- **data** 最新のスキーマのアップグレードのプログレス・データを保持しているオブジェクト

戻り値

このメソッドは、false を返して続行するか、true を返してスキーマのアップグレードをキャンセルする必要があります (キャンセルするには、ステータスは SchemaUpgradeState 列挙 である必要があります)。

説明

`SchemaUpgrading` の呼び出し中に、Ultra Light.NET API のメソッドを呼び出さないでください。

SchemaUpgradeState 列挙

スキーマのアップグレード時に発生する可能性のあるすべてのステータスを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum SchemaUpgradeState
      Inherits Short

// C#
    public enum SchemaUpgradeState :
      short
```

説明

STATE_STARTING は、アップグレードをキャンセルできる唯一のステータスです。アップグレードがキャンセルされると、ステータス **STATE_ERROR** を持つ 2 番目のイベントを受け取ります。

メンバ

メンバ	説明
STATE_ABORT	スキーマのアップグレードはアボートされ、古いデータベースが保持されています。回復可能なエラーまたはユーザによるアボートの結果、このステータスになることがあります。
STATE_DONE	スキーマのアップグレードが正常に完了しました。
STATE_ERROR	重大なエラーが発生したため、データベースを使用できません。
STATE_LAST	内部的に使用されるステータスであり、SchemaUpgrading メソッド に送信されません。
STATE_STARTING	スキーマのアップグレードが開始されています。これは、アップグレードをキャンセルできる唯一のステータスです。アップグレードがキャンセルされると、ステータス **STATE_ABORT** を持つ 2 番目のイベントを受け取ります。
STATE_UPGRADING	スキーマのアップグレードが進行中です。

参照

- State プロパティ
- ProgressCounter プロパティ
- FinalProgressCount プロパティ
- UpgradeOperations プロパティ

ServerSyncListener インタフェース

サーバの同期メッセージを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface ServerSyncListener

// C#
    public interface ServerSyncListener
```

ServerSyncListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
ServerSyncInvoked メソッド	サーバ起動同期用の Mobile Link Listener がアプリケーションを呼び出して同期を実行するときに起動されます。

ServerSyncInvoked メソッド

サーバ起動同期用の Mobile Link Listener がアプリケーションを呼び出して同期を実行するときに起動されます。

プロトタイプ

```
' Visual Basic
    Public Sub ServerSyncInvoked( _
      ByVal messageName As String _
    )

// C#
    public void ServerSyncInvoked(
      string  messageName
    );
```

パラメータ

- **messageName** アプリケーションに送信されるメッセージの名前

説明

このメソッドは、別のスレッドによって呼び出されます。マルチスレッドの問題を回避するには、このメソッドがユーザ・インタフェースにイベントを通知する必要があります。マルチスレッドを使用する場合は、スレッドごとに別々の接続を使用し、lock キーワードを使用して、アプリケーションの共有オブジェクトにアクセスすることをおすすめします。

例

次のコード・フラグメントは、サーバ同期要求の受信方法および UI スレッドでの同期の実行方法を示してします。

```
[Visual Basic]
 Imports iAnywhere.UltraLite

 Public Class MainWindow
   Inherits System.Windows.Forms.Form
   Implements ServerSyncListener
   Private dbMgr As DatabaseManager
   Private conn As Connection

   Public Sub New(ByVal args() As String)

     MyBase.New()

     'This call is required by the Windows Form
Designer.
     InitializeComponent()

     'Add any initialization after the
InitializeComponent() call
     dbMgr = New DatabaseManager
     dbMgr.SetServerSyncListener( "myCompany.mymsg",
"myCompany.myapp", Me )
     'Create Connection
     ...
   End Sub

   Protected Overrides Sub OnClosing(ByVal e As
System.ComponentModel.CancelEventArgs)
     If Not (dbMgr Is Nothing) Then
       dbMgr.SetServerSyncListener(Nothing, Nothing,
Nothing)
       dbMgr = Nothing
     End If
   End Sub
```

```
   Public Sub ServerSyncInvoked(ByVal messageName As
String) Implements
iAnywhere.UltraLite.ServerSyncListener.ServerSyncInvoke
d
     Me.Invoke(New EventHandler(AddressOf
Me.ServerSyncAction))
   End Sub

   Public Sub ServerSyncAction(ByVal sender As Object,
ByVal e As EventArgs)
     ' Do Server sync
     conn.Synchronize()
   End Sub
 End Class

 [C#]
 using iAnywhere.UltraLite;
 public class Form1 : System.Windows.Forms.Form,
ServerSyncListener
 {
   private System.Windows.Forms.MainMenu mainMenu1;
   private DatabaseManager dbMgr;
   private Connection conn;

   public Form1()
   {
     //
     // Required for Windows Form Designer support
     //
     InitializeComponent();

     //
     // TODO: Add any constructor code after
InitializeComponent call
     //
     dbMgr = new DatabaseManager();
     dbMgr.SetServerSyncListener( "myCompnay.mymsg",
"myCompany.myapp", this );
     // Create connection
     ...
   }

   protected override void Dispose( bool disposing )
   {
     base.Dispose( disposing );
   }
```

```
   protected override void
OnClosing(System.ComponentModel.CancelEventArgs e)
   {
     if( dbMgr != null ) {
       dbMgr.SetServerSyncListener( null, null, null );
       dbMgr = null;
     }
     base.OnClosing (e);
   }

   public void ServerSyncInvoked( string messageName )
   {
    this.Invoke( new EventHandler( ServerSyncHandler )
);
   }

   internal void ServerSyncHandler(object sender,
EventArgs e)
   {
     conn.Synchronize();
   }
 }
```

SQLCode 列挙

Ultra Light.NET によってレポートされる可能性のある SQL コードを列挙します。

プロトタイプ

' Visual Basic
Public Enum **SQLCode**

// C#
public enum **SQLCode**

メンバ

メンバ	説明
SQLE_AGGREGATES_NOT_ALLOWED	SQLE_AGGREGATES_NOT_ALLOWED(-150)
SQLE_ALIAS_NOT_UNIQUE	SQLE_ALIAS_NOT_UNIQUE(-830)
SQLE_ALIAS_NOT_YET_DEFINED	SQLE_ALIAS_NOT_YET_DEFINED(-831)
SQLE_AMBIGUOUS_INDEX_NAME	SQLE_AMBIGUOUS_INDEX_NAME(-678)
SQLE_ARGUMENT_CANNOT_BE_NULL	SQLE_ARGUMENT_CANNOT_BE_NULL(-90)
SQLE_BAD_ENCRYPTION_KEY	SQLE_BAD_ENCRYPTION_KEY(-840)
SQLE_BAD_PARAM_INDEX	SQLE_BAD_PARAM_INDEX(-689)
SQLE_CANNOT_ACCESS_FILE	SQLE_CANNOT_ACCESS_FILE(-602)
SQLE_CANNOT_ACCESS_SCHEMA_FILE	SQLE_CANNOT_ACCESS_SCHEMA_FILE(-951)
SQLE_CANNOT_CHANGE_USER_NAME	SQLE_CANNOT_CHANGE_USER_NAME(-867)

メンバ	説明
SQLE_CANNOT_EXECUTE_STMT	SQLE_CANNOT_EXECUTE_STMT(111)
SQLE_CANNOT_MODIFY	SQLE_CANNOT_MODIFY(-191)
SQLE_CANNOT_REGISTER_LISTENER	SQLE_CANNOT_REGISTER_LISTENER(-992)
SQLE_COLUMN_AMBIGUOUS	SQLE_COLUMN_AMBIGUOUS(-144)
SQLE_COLUMN_CANNOT_BE_NULL	SQLE_COLUMN_CANNOT_BE_NULL(-195)
SQLE_COLUMN_IN_INDEX	SQLE_COLUMN_IN_INDEX(-127)
SQLE_COLUMN_NOT_FOUND	SQLE_COLUMN_NOT_FOUND(-143)
SQLE_COLUMN_NOT_FOUND_IN_TABLE	SQLE_COLUMN_NOT_FOUND_IN_TABLE(-834)
SQLE_COMMUNICATIONS_ERROR	SQLE_COMMUNICATIONS_ERROR(-85)
SQLE_CONNECTION_ALREADY_EXISTS	SQLE_CONNECTION_ALREADY_EXISTS(-955)
SQLE_CONNECTION_NOT_FOUND	SQLE_CONNECTION_NOT_FOUND(-108)
SQLE_CONNECTION_RESTORED	SQLE_CONNECTION_RESTORED(133)
SQLE_CONVERSION_ERROR	SQLE_CONVERSION_ERROR(-157)
SQLE_CORRELATION_NAME_NOT_FOUND	SQLE_CORRELATION_NAME_NOT_FOUND(-142)
SQLE_COULD_NOT_FIND_FUNCTION	SQLE_COULD_NOT_FIND_FUNCTION(-621)

メンバ	説明
SQLE_COULD_NOT_LOAD_LIBRARY	SQLE_COULD_NOT_LOAD_LIBRARY(-620)
SQLE_CURSOR_ALREADY_OPEN	SQLE_CURSOR_ALREADY_OPEN(-172)
SQLE_CURSOR_NOT_OPEN	SQLE_CURSOR_NOT_OPEN(-180)
SQLE_CURSOR_RESTORED	SQLE_CURSOR_RESTORED(134)
SQLE_CURSOROP_NOT_ALLOWED	SQLE_CURSOROP_NOT_ALLOWED(-187)
SQLE_DATABASE_CREATED	SQLE_DATABASE_CREATED(136)
SQLE_DATABASE_ERROR	SQLE_DATABASE_ERROR(-301)
SQLE_DATABASE_NAME_REQUIRED	SQLE_DATABASE_NAME_REQUIRED(-87)
SQLE_DATABASE_NEW	SQLE_DATABASE_NEW(123)
SQLE_DATABASE_NOT_CREATED	SQLE_DATABASE_NOT_CREATED(-645)
SQLE_DATABASE_UPGRADE_NOT_POSSIBLE	SQLE_DATABASE_UPGRADE_NOT_POSSIBLE(-673)
SQLE_DATATYPE_NOT_ALLOWED	SQLE_DATATYPE_NOT_ALLOWED(-624)
SQLE_DBSPACE_FULL	SQLE_DBSPACE_FULL(-604)
SQLE_DEVICE_IO_FAILED	SQLE_DEVICE_IO_FAILED(-974)
SQLE_DIV_ZERO_ERROR	SQLE_DIV_ZERO_ERROR(-628)
SQLE_DOWNLOAD_CONFLICT	SQLE_DOWNLOAD_CONFLICT(-839)

メンバ	説明
SQLE_DROP_DATABASE_FAILED	SQLE_DROP_DATABASE_FAILED(-651)
SQLE_DUPLICATE_FOREIGN_KEY	
SQLE_DYNAMIC_MEMORY_EXHAUSTED	SQLE_DYNAMIC_MEMORY_EXHAUSTED(-78)
SQLE_ENGINE_ALREADY_RUNNING	SQLE_ENGINE_ALREADY_RUNNING(-96)
SQLE_ENGINE_NOT_MULTIUSER	SQLE_ENGINE_NOT_MULTIUSER(-89)
SQLE_ERROR	SQLE_ERROR(-300)
SQLE_ERROR_CALLING_FUNCTION	SQLE_ERROR_CALLING_FUNCTION(-622)
SQLE_EXPRESSION_ERROR	SQLE_EXPRESSION_ERROR(-156)
SQLE_FOREIGN_KEY_NAME_NOT_FOUND	SQLE_FOREIGN_KEY_NAME_NOT_FOUND(-145)
SQLE_IDENTIFIER_TOO_LONG	SQLE_IDENTIFIER_TOO_LONG(-250)
SQLE_INCORRECT_VOLUME_ID	SQLE_INCORRECT_VOLUME_ID(-975)
SQLE_INDEX_NAME_NOT_UNIQUE	SQLE_INDEX_NAME_NOT_UNIQUE(-111)
SQLE_INDEX_NOT_FOUND	SQLE_INDEX_NOT_FOUND(-183)
SQLE_INDEX_NOT_UNIQUE	SQLE_INDEX_NOT_UNIQUE(-196)
SQLE_INTERRUPTED	SQLE_INTERRUPTED(-299)
SQLE_INVALID_DATABASE	SQLE_INVALID_DATABASE(-84)

メンバ	説明
SQLE_INVALID_FOREIGN_KEY	SQLE_INVALID_FOREIGN_KEY(-194)
SQLE_INVALID_FOREIGN_KEY_DEF	SQLE_INVALID_FOREIGN_KEY_DEF(-113)
SQLE_INVALID_GROUP_SELECT	SQLE_INVALID_GROUP_SELECT(-149)
SQLE_INVALID_LOGON	SQLE_INVALID_LOGON(-103)
SQLE_INVALID_OPTION_SETTING	SQLE_INVALID_OPTION_SETTING(-201)
SQLE_INVALID_ORDER	SQLE_INVALID_ORDER(-152)
SQLE_INVALID_PARAMETER	SQLE_INVALID_PARAMETER(-735)
SQLE_INVALID_PARSE_PARAMETER	SQLE_INVALID_PARSE_PARAMETER(-95)
SQLE_INVALID_SQL_IDENTIFIER	SQLE_INVALID_SQL_IDENTIFIER(-760)
SQLE_LOCKED	SQLE_LOCKED(-210)
SQLE_MEMORY_ERROR	SQLE_MEMORY_ERROR(-309)
SQLE_METHOD_CANNOT_BE_CALLED	SQLE_METHOD_CANNOT_BE_CALLED(-669)
SQLE_NAME_NOT_UNIQUE	SQLE_NAME_NOT_UNIQUE(-110)
SQLE_NO_COLUMN_NAME	SQLE_NO_COLUMN_NAME(-163)
SQLE_NO_CURRENT_ROW	SQLE_NO_CURRENT_ROW(-197)
SQLE_NO_INDICATOR	SQLE_NO_INDICATOR(-181)

メンバ	説明
SQLE_NO_MATCHING_SELECT_ITEM	SQLE_NO_MATCHING_SELECT_ITEM(-812)
SQLE_NO_PRIMARY_KEY	SQLE_NO_PRIMARY_KEY(-118)
SQLE_NOERROR	SQLE_NOERROR(0)
SQLE_NOT_IMPLEMENTED	SQLE_NOT_IMPLEMENTED(-134)
SQLE_NOT_SUPPORTED_IN_ULTRALITE	SQLE_NOT_SUPPORTED_IN_ULTRALITE(-749)
SQLE_NOTFOUND	SQLE_NOTFOUND(100)
SQLE_OVERFLOW_ERROR	SQLE_OVERFLOW_ERROR(-158)
SQLE_PAGE_SIZE_INVALID	SQLE_PAGE_SIZE_INVALID(-644)
SQLE_PERMISSION_DENIED	SQLE_PERMISSION_DENIED(-121)
SQLE_PRIMARY_KEY_NOT_UNIQUE	SQLE_PRIMARY_KEY_NOT_UNIQUE(-193)
SQLE_PRIMARY_KEY_TWICE	SQLE_PRIMARY_KEY_TWICE(-126)
SQLE_PRIMARY_KEY_VALUE_REF	SQLE_PRIMARY_KEY_VALUE_REF(-198)
SQLE_PUBLICATION_NOT_FOUND	SQLE_PUBLICATION_NOT_FOUND(-280)
SQLE_RESOURCE_GOVERNOR_EXCEEDED	SQLE_RESOURCE_GOVERNOR_EXCEEDED(-685)
SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY	SQLE_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY(137)

メンバ	説明
SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE	SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE(130)
SQLE_SCHEMA_UPGRADE_NOT_ALLOWED	SQLE_SCHEMA_UPGRADE_NOT_ALLOWED(-953)
SQLE_SERVER_SYNCHRONIZATION_ERROR	SQLE_SERVER_SYNCHRONIZATION_ERROR(-857)
SQLE_START_STOP_DATABASE_DENIED	SQLE_START_STOP_DATABASE_DENIED(-75)
SQLE_STRING_RIGHT_TRUNCATION	SQLE_STRING_RIGHT_TRUNCATION(-638)
SQLE_SUBQUERY_SELECT_LIST	SQLE_SUBQUERY_SELECT_LIST(-151)
SQLE_SYNC_INFO_INVALID	SQLE_SYNC_INFO_INVALID(-956)
SQLE_SYNC_STATUS_UNKNOWN	SQLE_SYNC_STATUS_UNKNOWN(-952)
SQLE_SYNCHRONIZATION_NOT_FOUND	SQLE_SYNCHRONIZATION_NOT_FOUND(-767)
SQLE_SYNTAX_ERROR	SQLE_SYNTAX_ERROR(-131)
SQLE_TABLE_HAS_PUBLICATIONS	SQLE_TABLE_HAS_PUBLICATIONS(-281)
SQLE_TABLE_IN_USE	SQLE_TABLE_IN_USE(-214)
SQLE_TABLE_NOT_FOUND	SQLE_TABLE_NOT_FOUND(-141)
SQLE_TOO_MANY_CONNECTIONS	SQLE_TOO_MANY_CONNECTIONS(-102)
SQLE_TOO_MANY_TEMP_TABLES	SQLE_TOO_MANY_TEMP_TABLES(-817)
SQLE_ULTRALITE_DATABASE_NOT_FOUND	SQLE_ULTRALITE_DATABASE_NOT_FOUND(-954)

メンバ	説明
SQLE_ULTRALITE_OBJ_CLOSED	SQLE_ULTRALITE_OBJ_CLOSED(-908)
SQLE_ULTRALITE_RUNTIME_LIBRARY_MISMATCH	SQLE_ULTRALITE_RUNTIME_LIBRARY_MISMATCH(-977)
SQLE_UNABLE_TO_CONNECT	SQLE_UNABLE_TO_CONNECT(-105)
SQLE_UNABLE_TO_START_DATABASE	SQLE_UNABLE_TO_START_DATABASE(-82)
SQLE_UNABLE_TO_START_DATABASE_VER_NEWER	SQLE_UNABLE_TO_START_DATABASE_VER_NEWER(-934)
SQLE_UNCOMMITTED_TRANSACTIONS	SQLE_UNCOMMITTED_TRANSACTIONS(-755)
SQLE_UNKNOWN_FUNC	SQLE_UNKNOWN_FUNC(-148)
SQLE_UNKNOWN_USERID	SQLE_UNKNOWN_USERID(-140)
SQLE_UNRECOGNIZED_OPTION	SQLE_UNRECOGNIZED_OPTION(-1002)
SQLE_UNSUPPORTED_CHARACTER_SET_ERROR	SQLE_UNSUPPORTED_CHARACTER_SET_ERROR(-869)
SQLE_UPLOAD_FAILED_AT_SERVER	SQLE_UPLOAD_FAILED_AT_SERVER(-794)
SQLE_WRONG_NUM_OF_INSERT_COLS	SQLE_WRONG_NUM_OF_INSERT_COLS(-207)
SQLE_WRONG_PARAMETER_COUNT	SQLE_WRONG_PARAMETER_COUNT(-154)

SQLException クラス

Ultra Light データベースによって返される SQL エラーを示します。

プロトタイプ

```
' Visual Basic
    Public Class SQLException
      Inherits ApplicationException

// C#
    public class SQLException :
      ApplicationException
```

説明

.NET Compact Framework では、このクラスは直列可能ではありません。

SQLException メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
SQLException コンストラクタ	指定されたエラー・コードを持つ SQLException を作成します。

パブリック・インスタンス・プロパティ

メンバ	説明
ErrorCode プロパティ	データベースによって返される SQL コードです。
HelpLink (継承 Exception)	例外に関連付けられているヘルプ ファイルへのリンクを取得または設定します。
InnerException (継承 Exception)	現在の例外を発生させた System.Exception インスタンスを取得します。
Message (継承 Exception)	現在の例外を説明するメッセージを取得します。

メンバ	説明
Source (継承 Exception)	エラーの原因となったアプリケーションまたはオブジェクトの名前を取得または設定します。
StackTrace (継承 Exception)	現在の例外がスローされたときにコール スタックにあったフレームの文字列形式を取得します。
TargetSite (継承 Exception)	現在の例外をスローするメソッドを取得します。

パブリック・インスタンス・メソッド

メンバ	説明
GetBaseException (継承 Exception)	派生クラスでオーバーライドされた場合、それ以後に発生する 1 つ以上の例外の主要な原因である System.Exception を返します。
GetObjectData メソッド	この SQLException の直列化に必要なデータを SerializationInfo に移植します。
ToString (継承 Exception)	現在の例外の文字列形式を作成して返します。

プロテクト・インスタンス・プロパティ

メンバ	説明
HResult (継承 Exception)	特定の例外に割り当てられているコード化数値である HRESULT を取得または設定します。

SQLException コンストラクタ

指定されたエラー・コードを持つ SQLException を作成します。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub New( _
      ByVal code As SQLCode, _
      ByVal s1 As String, _
```

```
        ByVal s2 As String, _
        ByVal s3 As String _
    )

// C#
    public SQLException(
        SQLCode code,
        string  s1,
        string  s2,
        string  s3
    );
```

パラメータ

- **code** この例外のコード
- **s1** フォーマットされたメッセージの 1 番目の文字列
- **s2** フォーマットされたメッセージの 2 番目の文字列
- **s3** フォーマットされたメッセージの 3 番目の文字列

説明

指定した SQLCode 列挙 に対応するメッセージ文字列は、**iAnywhere.UltraLite.resources** アセンブリから取得されます。リソースは、System.Globalization.CultureInfo.CurrentUICulture、System.Globalization.CultureInfo.CurrentCulture、国 "EN" の順序で国別に検索されます。

ErrorCode プロパティ

データベースによって返される SQL コードです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ErrorCode As SQLCode

// C#
    public SQLCode ErrorCode {get;}
```

GetObjectData メソッド

この SQLException の直列化に必要なデータを SerializationInfo に移植します。

プロトタイプ

```
' Visual Basic
    Overrides Public Sub GetObjectData( _
      ByVal info As System.Runtime.Serialization.SerializationInfo, _
      ByVal context As System.Runtime.Serialization.StreamingContext _
    ) _
      Implements ISerializable.GetObjectData

// C#
    public override void GetObjectData(
      System.Runtime.Serialization.SerializationInfo info,
      System.Runtime.Serialization.StreamingContext context
    );
```

パラメータ

- **info** データを移植する SerializationInfo
- **context** この直列化の宛先

説明

.NET Compact Framework では、このメソッドはサポートされていません。

実装

ISerializable.GetObjectData

SQLType 列挙

テーブル・カラムの型として使用されている Ultra Light SQL データベースの型を列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum SQLType
     Inherits Short

// C#
    public enum SQLType :
     short
```

説明

下の表には、各 SQL 型との互換性がある .NET 型がリストされています。整数型の場合、テーブルのカラムは、常により小さい整数型を使用して設定することができるほか、実際の値が SQL 型の範囲内にある限り、より大きい型を使用して設定することが可能です。次に例を示します。

```
[Visual Basic]
 Dim t as iAnywhere.UltraLite.Table
 Dim s as Short
 ...
 ' Column 1 is of type S_LONG (Equivalent to a .NET
'System.Int32')
 t.SetShort( 1, s )  ' will succeed for all values in s
 t.SetLong( 1, 10 )  ' (long)10 is ok
 t.SetLong( 1, Int32.MaxValue + 1 )  ' throws
SQLException

 ' Column 2 is of type U_SHORT
 Dim i as Integer = t.GetInt( 2 )  ' will succeed for
all column values
 t.SetInt( 2, 0x0ffff )  ' largest valid 'int' column
can be set with
 t.SetInt( 2, -1 )  ' SQLException

 [C#]
 Table t;
 short s;
 ...
 // Column 1 is of type S_LONG (Equivalent to a .NET
'System.Int32')
 t.SetShort( 1, s );  // will succeed for all values in
```

```
s
 t.SetLong( 1, 10 );  // (long)10 is ok
 t.SetLong( 1, Int32.MaxValue + 1 );  // throws
SQLException

 // Column 2 is of type U_SHORT
 int i = t.GetInt( 2 );  // will succeed for all column
values
 t.SetInt( 2, 0x0ffff );  // largest valid 'int' column
can be set with
 t.SetInt( 2, -1 );  // SQLException
```

SQL 型	互換性のある .NET 型	C# 組み込みタイプ	Visual Basic 組み込みタイプ
BINARY	System.Byte[]、または System.Guid (サイズが 16 の場合)	byte[]	Byte()
BIT	System.Boolean	bool	Boolean
CHAR	System.String	String	String
DATE	System.DateTime	DateTime 組み込みタイプはなし	Date
DOUBLE	System.Double	double	Double
LONGBINARY	System.Byte[]	byte[]	Byte()
LONGVARCHAR	System.String	String	String
NUMERIC	System.String	String	String
REAL	System.Single	float	Single
S_BIG	System.Int64	long	Long

SQL 型	互換性のある .NET 型	C# 組み込みタイプ	Visual Basic 組み込みタイプ
S_LONG	System.Int32	int	Integer
S_SHORT	System.Int16	short	Short
TIME	System.TimeSpan	TimeSpan 組み込みタイプはなし	TimeSpan 組み込みタイプはなし
TIMESTAMP	System.DateTime	DateTime 組み込みタイプはなし	Date
TINY	System.Byte	byte	Byte
U_BIG	System.UInt64	ulong	UInt64 組み込みタイプはなし
U_LONG	System.UInt32	uint	UInt32 組み込みタイプはなし
U_SHORT	System.UInt16	ushort	UInt16 組み込みタイプはなし
UUID	System.Guid	Guid 組み込みタイプはなし	Guid 組み込みタイプはなし

長さが 16 の BINARY カラムは、UUID カラムと呼ばれることがあります。

メンバ

メンバ	説明
BAD_INDEX	正しくないカラム識別子が GetColumnSQLType メソッド に渡された場合に返されるカラムの型です。
BINARY	指定した長さのバイナリ・データ
BIT	1 ビット・フラグ
CHAR	指定した長さの文字データ
DATE	日付情報
DOUBLE	倍精度浮動小数点数 (8 バイト)

メンバ	説明
LONGBINARY	可変長のバイナリ・データ
LONGVARCHAR	可変長の文字データ
NUMERIC	精度と桁数が指定された正確な数値データ
REAL	単精度浮動小数点数 (4 バイト)
S_BIG	符号付き 64 ビット整数値
S_LONG	符号付き 32 ビット整数値
S_SHORT	符号付き 16 ビット整数値
TIME	時刻情報
TIMESTAMP	タイムスタンプ情報 (日付、時刻)
TINY	符号なし 8 ビット整数値
U_BIG	符号なし 64 ビット整数値
U_LONG	符号なし 32 ビット整数値
U_SHORT	符号なし 16 ビット整数値
UUID	ユニバーサル・ユニーク識別子

StreamErrorCode 列挙

同期中にストリームによって報告される可能性のあるエラー・コードを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum StreamErrorCode
      Inherits Short

// C#
    public enum StreamErrorCode :
      short
```

メンバ

メンバ	説明
ACTSYNC_NO_PORT	ACTSYNC_NO_PORT(75)
ACTSYNC_NOT_INSTALLED	ACTSYNC_NOT_INSTALLED(76)
CREATE_RANDOM_OBJECT	CREATE_RANDOM_OBJECT(17)
DEQUEUING_CONNECTION	DEQUEUING_CONNECTION(19)
END_READ	END_READ(11)
END_WRITE	END_WRITE(10)
GENERATE_RANDOM	GENERATE_RANDOM(14)
HTTP_BAD_STATUS_CODE	HTTP_BAD_STATUS_CODE(86)
HTTP_BUFFER_SIZE_OUT_OF_RANGE	HTTP_BUFFER_SIZE_OUT_OF_RANGE(79)
HTTP_CHUNK_LEN_BAD_CHARACTER	HTTP_CHUNK_LEN_BAD_CHARACTER(85)
HTTP_CHUNK_LEN_ENCODED_MISSING	HTTP_CHUNK_LEN_ENCODED_MISSING(84)

メンバ	説明
HTTP_CLIENT_ID_NOT_SET	HTTP_CLIENT_ID_NOT_SET(78)
HTTP_CONTENT_TYPE_NOT_SPECIFIED	HTTP_CONTENT_TYPE_NOT_SPECIFIED(77)
HTTP_CRLF_ENCODED_MISSING	HTTP_CRLF_ENCODED_MISSING(81)
HTTP_CRLF_MISSING	HTTP_CRLF_MISSING(82)
HTTP_EXPECTED_POST	HTTP_EXPECTED_POST(89)
HTTP_EXTRA_DATA_END_READ	HTTP_EXTRA_DATA_END_READ(80)
HTTP_NO_CONTD_CONNECTION	HTTP_NO_CONTD_CONNECTION(83)
HTTP_UNABLE_TO_PARSE_COOKIE	HTTP_UNABLE_TO_PARSE_COOKIE(88)
HTTP_UNKNOWN_TRANSFER_ENCODING	HTTP_UNKNOWN_TRANSFER_ENCODING(87)
HTTP_VERSION	HTTP_VERSION(54)
INIT_RANDOM	INIT_RANDOM(15)
LOAD_NETWORK_LIBRARY	LOAD_NETWORK_LIBRARY(74)
MEMORY_ALLOCATION	MEMORY_ALLOCATION(6)
NONE	NONE(0)
NOT_IMPLEMENTED	NOT_IMPLEMENTED(12)
PARAMETER	PARAMETER(1)
PARAMETER_NOT_BOOLEAN	PARAMETER_NOT_BOOLEAN(4)

メンバ	説明
PARAMETER_NOT_HEX	PARAMETER_NOT_HEX(5)
PARAMETER_NOT_UINT32	PARAMETER_NOT_UINT32(2)
PARAMETER_NOT_UINT32_RANGE	PARAMETER_NOT_UINT32_RANGE(3)
PARSE	PARSE(7)
READ	READ(8)
SECURE_ADD_CERTIFICATE	SECURE_ADD_CERTIFICATE(39)
SECURE_ADD_TRUSTED_CERTIFICATE	SECURE_ADD_TRUSTED_CERTIFICATE(48)
SECURE_CERTIFICATE_CHAIN_FUNC	SECURE_CERTIFICATE_CHAIN_FUNC(28)
SECURE_CERTIFICATE_CHAIN_LENGTH	SECURE_CERTIFICATE_CHAIN_LENGTH(22)
SECURE_CERTIFICATE_CHAIN_REF	SECURE_CERTIFICATE_CHAIN_REF(29)
SECURE_CERTIFICATE_COMMON_NAME	SECURE_CERTIFICATE_COMMON_NAME(52)
SECURE_CERTIFICATE_COMPANY_NAME	SECURE_CERTIFICATE_COMPANY_NAME(21)
SECURE_CERTIFICATE_COMPANY_UNIT	SECURE_CERTIFICATE_COMPANY_UNIT(51)
SECURE_CERTIFICATE_COUNT	SECURE_CERTIFICATE_COUNT(42)
SECURE_CERTIFICATE_EXPIRED	SECURE_CERTIFICATE_EXPIRED(50)
SECURE_CERTIFICATE_EXPIRY_DATE	SECURE_CERTIFICATE_EXPIRY_DATE(37)

メンバ	説明
SECURE_CERTIFICATE_FILE_NOT_FOUND	SECURE_CERTIFICATE_FILE_NOT_FOUND(33)
SECURE_CERTIFICATE_NOT_TRUSTED	SECURE_CERTIFICATE_NOT_TRUSTED(24)
SECURE_CERTIFICATE_REF	SECURE_CERTIFICATE_REF(23)
SECURE_CERTIFICATE_ROOT	SECURE_CERTIFICATE_ROOT(20)
SECURE_CREATE_CERTIFICATE	SECURE_CREATE_CERTIFICATE(43)
SECURE_CREATE_PRIVATE_KEY_OBJECT	SECURE_CREATE_PRIVATE_KEY_OBJECT(49)
SECURE_DUPLICATE_CONTEXT	SECURE_DUPLICATE_CONTEXT(25)
SECURE_ENABLE_NON_BLOCKING	SECURE_ENABLE_NON_BLOCKING(30)
SECURE_EXPORT_CERTIFICATE	SECURE_EXPORT_CERTIFICATE(38)
SECURE_HANDSHAKE	SECURE_HANDSHAKE(53)
SECURE_IMPORT_CERTIFICATE	SECURE_IMPORT_CERTIFICATE(44)
SECURE_READ_CERTIFICATE	SECURE_READ_CERTIFICATE(34)
SECURE_READ_PRIVATE_KEY	SECURE_READ_PRIVATE_KEY(35)
SECURE_SET_CHAIN_NUMBER	SECURE_SET_CHAIN_NUMBER(32)
SECURE_SET_CIPHER_SUITES	SECURE_SET_CIPHER_SUITES(31)
SECURE_SET_IO	SECURE_SET_IO(26)

メンバ	説明
SECURE_SET_IO_SEMANTICS	SECURE_SET_IO_SEMANTICS(27)
SECURE_SET_PRIVATE_KEY	SECURE_SET_PRIVATE_KEY(36)
SECURE_SET_PROTOCOL_SIDE	SECURE_SET_PROTOCOL_SIDE(47)
SECURE_SET_RANDOM_FUNC	SECURE_SET_RANDOM_FUNC(46)
SECURE_SET_RANDOM_REF	SECURE_SET_RANDOM_REF(45)
SECURE_SET_READ_FUNC	SECURE_SET_READ_FUNC(55)
SECURE_SET_WRITE_FUNC	SECURE_SET_WRITE_FUNC(56)
SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND	SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND(40)
SECURE_TRUSTED_CERTIFICATE_READ	SECURE_TRUSTED_CERTIFICATE_READ(41)
SEED_RANDOM	SEED_RANDOM(16)
SHUTTING_DOWN	SHUTTING_DOWN(18)
SOCKET_BIND	SOCKET_BIND(62)
SOCKET_CLEANUP	SOCKET_CLEANUP(63)
SOCKET_CLOSE	SOCKET_CLOSE(64)
SOCKET_CONNECT	SOCKET_CONNECT(65)
SOCKET_CREATE_TCPIP	SOCKET_CREATE_TCPIP(60)
SOCKET_CREATE_UDP	SOCKET_CREATE_UDP(61)

メンバ	説明
SOCKET_GET_HOST_BY_ADDR	SOCKET_GET_HOST_BY_ADDR(58)
SOCKET_GET_NAME	SOCKET_GET_NAME(66)
SOCKET_GET_OPTION	SOCKET_GET_OPTION(67)
SOCKET_HOST_NAME_NOT_FOUND	SOCKET_HOST_NAME_NOT_FOUND(57)
SOCKET_LISTEN	SOCKET_LISTEN(69)
SOCKET_LOCALHOST_NAME_NOT_FOUND	SOCKET_LOCALHOST_NAME_NOT_FOUND(59)
SOCKET_PORT_OUT_OF_RANGE	SOCKET_PORT_OUT_OF_RANGE(73)
SOCKET_SELECT	SOCKET_SELECT(71)
SOCKET_SET_OPTION	SOCKET_SET_OPTION(68)
SOCKET_SHUTDOWN	SOCKET_SHUTDOWN(70)
SOCKET_STARTUP	SOCKET_STARTUP(72)
WOULD_BLOCK	WOULD_BLOCK(13)
WRITE	WRITE(9)

参照

- StreamErrorCode プロパティ

StreamErrorContext 列挙

ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum StreamErrorContext
      Inherits Short

// C#
    public enum StreamErrorContext :
      short
```

メンバ

メンバ	説明
CLOSE	CLOSE(6)
CREATE	CREATE(3)
DESTROY	DESTROY(4)
END_READ	END_READ(11)
END_WRITE	END_WRITE(10)
GETVALUE	GETVALUE(14)
OPEN	OPEN(5)
READ	READ(7)
REGISTER	REGISTER(1)
SOFTSHUTDOWN	SOFTSHUTDOWN(13)
UNKNOWN	UNKNOWN(0)
UNREGISTER	UNREGISTER(2)
WRITE	WRITE(8)
WRITE_FLUSH	WRITE_FLUSH(9)
YIELD	YIELD(12)

参照

- StreamErrorContext プロパティ

StreamErrorID 列挙

同期中にエラーを報告する可能性のあるネットワーク・レイヤを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum StreamErrorID
      Inherits Short

// C#
    public enum StreamErrorID :
      short
```

メンバ

メンバ	説明
ACTIVESYNC	ACTIVESYNC(22)
CERTICOM	CERTICOM(11)
CERTICOM_SSL	CERTICOM_SSL(13)
CERTICOM_TLS	CERTICOM_TLS(14)
DH_CAST	DH_CAST(9)
EMAIL	EMAIL(20)
FAKE	FAKE(2)
FILE	FILE(21)
HTTP	HTTP(7)
HTTPS	HTTPS(8)
JAVA_CERTICOM	JAVA_CERTICOM(12)
JAVA_RSA	JAVA_RSA(24)
NETTECH	NETTECH(5)
PALM_CONDUIT	PALM_CONDUIT(3)
PALM_SS	PALM_SS(4)

メンバ	説明
REPLAY	REPLAY(17)
RIMBB	RIMBB(6)
RSA_TLS	RSA_TLS(23)
SECURE	SECURE(10)
SERIAL	SERIAL(1)
STRM	STRM(18)
TCPIP	TCPIP(0)
UDP	UDP(19)
WIRELESS	WIRELESS(16)
WIRESTRM	WIRESTRM(15)

参照

- StreamErrorID プロパティ

StreamType 列挙

同期に使用する Mobile Link 同期ストリームのタイプを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum StreamType
      Inherits Short

// C#
    public enum StreamType :
      short
```

説明

特定のストリームのタイプの設定については、オンライン・マニュアル『**Ultra Light データベース・ユーザーズ・ガイド**』の「**同期ストリーム・パラメータ・リファレンス**」の項を参照してください。

メンバ

メンバ	説明
ACTIVE_SYNC	ActiveSync 同期 (Windows CE のみ) です。 ActiveSync 用 Mobile Link プロバイダから同期を通知されたときは、Ultra Light アプリケーションは ActiveSync 同期のみを使用する必要があります。ActiveSyncListener インタフェース を実装し、SetActiveSyncListener メソッド を呼び出すことによって、アプリケーションはこのような通知を受信できます。
HTTP	HTTP を介して同期します。 HTTP ストリームは基本となるトランスポートとして TCP/IP を使用します。Ultra Light アプリケーションは Web ブラウザとして機能し、Mobile Link 同期サーバは Web サーバとして機能します。Ultra Light アプリケーションは、サーバへのデータ送信のために POST 要求を送り、サーバからのデータの読み込みのために GET 要求を送ります。

メンバ	説明
HTTPS	HTTPS を介して同期します (RSA トランスポート・レイヤ・セキュリティを適用した HTTP) トランスポート・レイヤ・セキュリティに関するコンポーネントには別途ライセンスが必要です。インストールする場合はご注文ください。このコンポーネントのご注文については、ご購入いただいた販売代理店または弊社営業担当までご連絡ください。
TCPIP	TCP/IP を介して同期します。
UNKNOWN	未知の同期ストリームであるか、同期ストリームがありません。ユーザがストリーム・タイプを設定していません。

参照

- Stream プロパティ

SyncParms クラス

Ultra Light データベースの同期方法を定義する同期パラメータを表します。

プロトタイプ

' Visual Basic
NotInheritable Public Class **SyncParms**

// C#
public sealed class **SyncParms**

説明

このクラスを直接インスタンス化することはできません。各接続には、SyncParms プロパティとしてアタッチされた固有の SyncParms インスタンスがあります。

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、SQLCode 列挙 SQLException が Synchronize メソッドによってスローされます。

その他の SQLCode 列挙 エラーの原因には、Stream プロパティ 値または Version プロパティ 値を指定していないことが含まれます。

参照

- SyncParms メンバ
- Connection クラス
- SyncParms プロパティ
- Synchronize メソッド

SyncParms メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
AuthenticationParms プロパティ	カスタム・ユーザ認証スクリプト (Mobile Link authenticate_parameters 接続イベント) のパラメータを指定します。

メンバ	説明
CheckpointStore プロパティ	クライアントが、追加のストア・チェックポイントを実行して、同期中にデータベース・ストアのサイズの増加を制御するかどうかを指定します。
DisableConcurrency プロパティ	同期の実行中に Ultra Light への同時アクセスを無効にするか、有効にするかを指定します。
DownloadOnly プロパティ	同期時のアップロードを無効にするか、有効にするかを指定します。
KeepPartialDownload プロパティ	同期時の部分的なダウンロードを無効にするか、有効にするかを指定します。
NewPassword プロパティ	UserName で指定されたユーザの新しい Mobile Link パスワードを指定します。
Password プロパティ	UserName で指定されたユーザの Mobile Link パスワードです。
PingOnly プロパティ	実際に同期を行う代わりに、クライアントが Mobile Link 同期サーバに ping のみを行うかどうかを指定します。
PublicationMask プロパティ	同期させるパブリケーションを指定します。
ResumePartialDownload プロパティ	前の部分的なダウンロードを再開するか、破棄するかを指定します。
SendColumnNames プロパティ	同期中に、クライアントが Mobile Link 同期サーバにカラム名を送信するかどうかを指定します。
SendDownloadAck プロパティ	同期中に、クライアントが Mobile Link 同期サーバにダウンロード確認を送信するかどうかを指定します。
Stream プロパティ	同期に使用する Mobile Link 同期ストリームを指定します。
StreamParms プロパティ	同期ストリームの設定パラメータを指定します。

メンバ	説明
UploadOnly プロパティ	同期時のダウンロードを無効にするか、有効にするかを指定します。
UserName プロパティ	Mobile Link 同期サーバが Mobile Link クライアントをユニークに識別するユーザ名です。
Version プロパティ	使用する同期スクリプトを指定します。

パブリック・インスタンス・メソッド

メンバ	説明
CopyFrom メソッド	指定した SyncParms クラス オブジェクトのプロパティを、この SyncParms クラス オブジェクトにコピーします。

AuthenticationParms プロパティ

カスタム・ユーザ認証スクリプト (Mobile Link authenticate_parameters 接続イベント) のパラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Property AuthenticationParms As String()

// C#
    public string [] AuthenticationParms {get;set;}
```

説明

それぞれに認証パラメータが格納された、文字列の配列として指定します (配列のエントリが null であると、同期エラーになります)。

最初の 255 文字のみが使用されます。また、各文字列は 128 文字以下である必要があります (長すぎる文字列は、Mobile Link に送信されるときにトランケートされます)。

デフォルトは、認証パラメータなしです。

CheckpointStore プロパティ

クライアントが、追加のストア・チェックポイントを実行して、同期中にデータベース・ストアのサイズの増加を制御するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property CheckpointStore As Boolean

// C#
    public bool CheckpointStore {get;set;}
```

説明

チェックポイント操作は、アプリケーションの I/O 操作を増やすため、同期が低速になります。このオプションは、多くの更新を伴う大量のダウンロードに最適です。低速なフラッシュ・メモリを使用するデバイスでは、パフォーマンスの低下が望ましくないことがあります。

デフォルトは false です (必要なチェックポイントのみが実行されます)。

DisableConcurrency プロパティ

同期の実行中に Ultra Light への同時アクセスを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property DisableConcurrency As Boolean

// C#
    public bool DisableConcurrency {get;set;}
```

説明

デフォルトでは、1 つのスレッドが同期しているときに、別のスレッドが Ultra Light の操作を行うことがあります。同時同期が無効になっていると、別のスレッドは同期が完了するまで Ultra Light の呼び出しをブロックされます。

DownloadOnly プロパティ

同期時のアップロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property DownloadOnly As Boolean

// C#
    public bool DownloadOnly {get;set;}
```

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、SQLCode 列挙 SQLException が Synchronize メソッドによってスローされます。

デフォルトは false です。

参照

- SyncParms クラス
- SyncParms メンバ
- UploadOnly プロパティ

KeepPartialDownload プロパティ

同期時の部分的なダウンロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property KeepPartialDownload As Boolean

// C#
    public bool KeepPartialDownload {get;set;}
```

説明

Ultra Light.NET では、通信エラーや、ユーザによる SyncProgressListener からのアボートが原因で失敗したダウンロードを再起動できます。Ultra Light.NET は、ダウンロードを受信しながら処理します。ダウンロードが中断した場合は、部分的なダウンロード・トランザクションがデータベース内に残るため、次の同期中に再開できます。

Ultra Light.NET で部分的なダウンロードを保存する必要があることを示すには、connection.SyncParms.KeepPartialDownload=true; と指定します。指定しないと、エラーが発生した場合にダウンロードがロールバックされます。

部分的なダウンロードが保持された場合、connection.Synchronize() の終了時に、出力フィールド connection.SyncResult.SyncResult.PartialDownloadRetained が true に設定されます。

PartialDownloadRetained が設定されている場合は、ダウンロードを再開できます。再開するには、connection.SyncParms.SyncParms.ResumePartialDownload を true に設定して connection.Synchronize() を呼び出します。多くの場合、別の通信エラーの発生に備えて、KeepPartialDownload も true に設定する必要があります。ダウンロードが省略された場合は、アップロードは行われません。

再開したダウンロードで受信するダウンロードは、最初にダウンロードを開始したときと同じものです。最新のデータが必要な場合は、再開された特別なダウンロードが完了した直後に、もう一度ダウンロードを行うことができます。

ダウンロードを再開する場合、SyncParms フィールドの多くは関係ありません。たとえば、PublicationMask フィールドは使用されません。受信するパブリケーションは、最初のダウンロード時に要求したものです。設定する必要があるフィールドは、ResumePartialDownload プロパティ と UserName プロパティ だけです。KeepPartialDownload フィールドと DisableConcurrency プロパティ フィールドを必要に応じて設定すると、正常に機能します。

部分的なダウンロードが存在するが、このダウンロードが必要ではなくなった場合は、RollbackPartialDownload メソッド を呼び出して、失敗したダウンロード・トランザクションをロールバックできます。また、同期をもう一度実行したときに ResumePartialDownload を指定しなかった場合は、部分的なダウンロードがロールバックされてから、次の同期が開始されます。

詳細については、『**Mobile Link 同期ユーザーズ・ガイド**』の「**再起動可能なダウンロード**」を参照してください。

デフォルトは false です。

参照

- SyncParms クラス
- SyncParms メンバ
- PartialDownloadRetained プロパティ
- ResumePartialDownload プロパティ
- RollbackPartialDownload メソッド

NewPassword プロパティ

UserName で指定されたユーザの新しい Mobile Link パスワードを指定します。

プロトタイプ

```
' Visual Basic
    Public Property NewPassword As String

// C#
    public string  NewPassword {get;set;}
```

説明

新しいパスワードが有効になるのは、次の同期の後です。

デフォルトは null です (パスワードは置き換えられません)。

参照

- SyncParms クラス
- SyncParms メンバ
- UserName プロパティ

Password プロパティ

UserName で指定されたユーザの Mobile Link パスワードです。

プロトタイプ

```
' Visual Basic
    Public Property Password As String

// C#
    public string  Password {get;set;}
```

説明

Mobile Link ユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link 同期サーバに対して識別し、認証するために使用されます。

デフォルトは null です (パスワードはありません)。

参照

- SyncParms クラス
- SyncParms メンバ
- NewPassword プロパティ
- UserName プロパティ

PingOnly プロパティ

実際に同期を行う代わりに、クライアントが Mobile Link 同期サーバに ping のみを行うかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property PingOnly As Boolean

// C#
    public bool PingOnly {get;set;}
```

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、SQLCode 列挙 SQLException が Synchronize メソッドによってスローされます。

デフォルトは false です。

PublicationMask プロパティ

同期させるパブリケーションを指定します。

プロトタイプ

```
' Visual Basic
    Public Property PublicationMask As Integer

// C#
    public int PublicationMask {get;set;}
```

説明

デフォルトは SYNC_ALL_DB フィールド です。

参照

- SyncParms クラス
- SyncParms メンバ
- PublicationSchema クラス

ResumePartialDownload プロパティ

前の部分的なダウンロードを再開するか、破棄するかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property ResumePartialDownload As Boolean

// C#
    public bool ResumePartialDownload {get;set;}
```

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、SQLCode 列挙 SQLException が Synchronize メソッド によってスローされます。

デフォルトは false です。

参照

- SyncParms クラス
- SyncParms メンバ
- KeepPartialDownload プロパティ
- PartialDownloadRetained プロパティ

SendColumnNames プロパティ

同期中に、クライアントが Mobile Link 同期サーバにカラム名を送信するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property SendColumnNames As Boolean

// C#
    public bool SendColumnNames {get;set;}
```

説明

このパラメータは、通常は同期スクリプトを自動的に生成するために Mobile Link 同期サーバで -za オプションまたは -ze オプションとともに使用されます。

デフォルトは false です (カラム名は送信されません)。

SendDownloadAck プロパティ

同期中に、クライアントが Mobile Link 同期サーバにダウンロード確認を送信するかどうかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property SendDownloadAck As Boolean

// C#
    public bool SendDownloadAck {get;set;}
```

説明

クライアントがダウンロード確認を送信する場合、Mobile Link 同期サーバのワーカ・スレッドは、クライアントがダウンロードを適用するまで待機します。クライアントがダウンロード確認を送信しない場合、Mobile Link 同期サーバは、次の同期のため、まもなく解放されます。

デフォルトは false です (ダウンロード確認は送信されません)。

Stream プロパティ

同期に使用する Mobile Link 同期ストリームを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Stream As StreamType

// C#
    public StreamType Stream {get;set;}
```

説明

ほとんどの同期ストリームでは、Mobile Link 同期サーバのアドレスを識別したり、その他の動作を制御したりするパラメータが必要です。これらのパラメータは、StreamParms プロパティで指定します。

ストリーム・タイプが、プラットフォームに適さない無効な値に設定されていると、ストリーム・タイプは StreamType 列挙 に設定されます。

このパラメータにはデフォルト値がないので、明示的に設定してください。

参照

- SyncParms クラス
- SyncParms メンバ
- StreamType 列挙
- StreamParms プロパティ

StreamParms プロパティ

同期ストリームの設定パラメータを指定します。

プロトタイプ

```
' Visual Basic
    Public Property StreamParms As String

// C#
    public string  StreamParms {get;set;}
```

説明

特定のストリームのタイプの設定については、オンライン・マニュアル『**Ultra Light データベース・ユーザーズ・ガイド**』の「**同期ストリーム・パラメータ・リファレンス**」の項を参照してください。

StreamParms は、同期ストリームに使用されるすべてのパラメータが含まれている文字列です。パラメータは、名前 = 値の組み合わせをセミコロンで区切ったリスト (`"param1=value1;param2=value2"`) で指定します。

デフォルトは null です。

参照

- SyncParms クラス
- SyncParms メンバ
- Stream プロパティ
- StreamType 列挙

UploadOnly プロパティ

同期時のダウンロードを無効にするか、有効にするかを指定します。

プロトタイプ

```
' Visual Basic
    Public Property UploadOnly As Boolean

// C#
    public bool UploadOnly {get;set;}
```

説明

同期コマンド (DownloadOnly プロパティ、PingOnly プロパティ、ResumePartialDownload プロパティ、UploadOnly プロパティ) は、一度に 1 つしか指定できません。これらの複数のパラメータが true に設定されていると、SQLCode 列挙 SQLException が Synchronize メソッドによってスローされます。

デフォルトは false です。

参照

- SyncParms クラス
- SyncParms メンバ
- DownloadOnly プロパティ

UserName プロパティ

Mobile Link 同期サーバが Mobile Link クライアントをユニークに識別するユーザ名です。

プロトタイプ

```
' Visual Basic
    Public Property UserName As String

// C#
    public string  UserName {get;set;}
```

説明

Mobile Link では、この値を使用して、ダウンロードする内容の決定、同期ステータスの記録、同期中の割り込みからの復帰を行います。このユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link 同期サーバに対して識別し、認証するために使用されます。

このパラメータにはデフォルト値がないので、明示的に設定してください。

参照

- SyncParms クラス
- SyncParms メンバ
- Password プロパティ

Version プロパティ

使用する同期スクリプトを指定します。

プロトタイプ

```
' Visual Basic
    Public Property Version As String

// C#
    public string  Version {get;set;}
```

説明

統合データベースの同期スクリプトは、それぞれバージョン文字列でマーク付けされます。たとえば、異なる文字列バージョンによって特定される 2 種類の download_cursor スクリプトがあります。Ultra Light アプリケーションは、バージョン文字列により、同期スクリプトのセットから選択できます。

このパラメータにはデフォルト値がないので、明示的に設定してください。

CopyFrom メソッド

指定した SyncParms クラス オブジェクトのプロパティを、この SyncParms クラス オブジェクトにコピーします。

プロトタイプ

```
' Visual Basic
    Public Sub CopyFrom( _
      ByVal src As SyncParms _
    )

// C#
    public void CopyFrom(
      SyncParms src
    );
```

パラメータ

- **src** コピー元のオブジェクト

SyncProgressData クラス

同期の進行状況のモニタリング・データを返します。

プロトタイプ

```
' Visual Basic
    Public Class SyncProgressData

// C#
    public class SyncProgressData
```

参照

- SyncProgressData メンバ
- SyncProgressListener インタフェース

SyncProgressData メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ErrorMessage プロパティ	同期中に発生したエラーを説明するエラー・メッセージです。
ReceivedBytes プロパティ	現在までに受信したバイト数です。この情報は、すべてのステータスで更新されます。
ReceivedDeletes プロパティ	現在までに受信した削除済みのローの数です。
ReceivedInserts プロパティ	現在までに受信した挿入済みのローの数です。
ReceivedUpdates プロパティ	現在までに受信した更新済みのローの数です。
SentBytes プロパティ	現在までに送信されたバイト数です。この情報は、すべてのステータスで更新されます。
SentDeletes プロパティ	現在までに送信された削除済みのローの数です。
SentInserts プロパティ	現在までに送信された挿入済みのローの数です。

メンバ	説明
SentUpdates プロパティ	現在までに送信された更新済みのローの数です。
SQLCode プロパティ	同期の SQL コードです。
State プロパティ	現在の同期のステータスです。
SyncParms プロパティ	接続の SyncParms オブジェクトへの参照です。
SyncResult プロパティ	接続の SyncResult オブジェクトへの参照です。このオブジェクトは、STATE_DONE イベントおよび STATE_ERROR イベントでのみ更新されます。
TableCount プロパティ	現在までに送信または受信したテーブルの数です (TableTotal の TableCount)。
TableIndex プロパティ	現在同期中のテーブルのインデックスです (テーブルには 1 ～ DatabaseSchema.TableCount までの番号が付けられます)。
TableTotal プロパティ	同期中のテーブルの数です。

ErrorMessage プロパティ

同期中に発生したエラーを説明するエラー・メッセージです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ErrorMessage As String

// C#
    public string  ErrorMessage {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙

ReceivedBytes プロパティ

現在までに受信したバイト数です。この情報は、すべてのステータスで更新されます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedBytes As Long

// C#
    public long ReceivedBytes {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

ReceivedDeletes プロパティ

現在までに受信した削除済みのローの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedDeletes As Integer

// C#
    public int ReceivedDeletes {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

ReceivedInserts プロパティ

現在までに受信した挿入済みのローの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property ReceivedInserts As Integer
```

// C#

public int **ReceivedInserts** {get;}

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

ReceivedUpdates プロパティ

現在までに受信した更新済みのローの数です。

プロトタイプ

' Visual Basic

Public Readonly Property **ReceivedUpdates** As Integer

// C#

public int **ReceivedUpdates** {get;}

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

SentBytes プロパティ

現在までに送信されたバイト数です。この情報は、すべてのステータスで更新されます。

プロトタイプ

' Visual Basic

Public Readonly Property **SentBytes** As Long

// C#

public long **SentBytes** {get;}

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

SentDeletes プロパティ

現在までに送信された削除済みのローの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentDeletes As Integer

// C#
    public int SentDeletes {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

SentInserts プロパティ

現在までに送信された挿入済みのローの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentInserts As Integer

// C#
    public int SentInserts {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

SentUpdates プロパティ

現在までに送信された更新済みのローの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SentUpdates As Integer

// C#
    public int SentUpdates {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

SQLCode プロパティ

同期の SQL コードです。

プロトタイプ

' Visual Basic

```
Public Readonly Property SQLCode As SQLCode
```

// C#

```
public SQLCode SQLCode {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙

State プロパティ

現在の同期のステータスです。

プロトタイプ

' Visual Basic

```
Public Readonly Property State As SyncProgressState
```

// C#

```
public SyncProgressState State {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙

SyncParms プロパティ

接続の SyncParms オブジェクトへの参照です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncParms As SyncParms

// C#
    public SyncParms SyncParms {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙
- SyncParms プロパティ

SyncResult プロパティ

接続の SyncResult オブジェクトへの参照です。このオブジェクトは、STATE_DONE イベントおよび STATE_ERROR イベントでのみ更新されます。

プロトタイプ

```
' Visual Basic
    Public Readonly Property SyncResult As SyncResult

// C#
    public SyncResult SyncResult {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙
- SyncResult プロパティ

TableCount プロパティ

現在までに送信または受信したテーブルの数です (TableTotal の TableCount)。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableCount As Integer
```

```
// C#
    public int TableCount {get;}
```

戻り値　送信または受信したテーブルの数

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

TableIndex プロパティ

現在同期中のテーブルのインデックスです (テーブルには 1 ～ DatabaseSchema.TableCount までの番号が付けられます)。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableIndex As Integer
```

```
// C#
    public int TableIndex {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙
- SyncProgressState 列挙

TableTotal プロパティ

同期中のテーブルの数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property TableTotal As Integer
```

```
// C#
    public int TableTotal {get;}
```

参照

- SyncProgressData クラス
- SyncProgressData メンバ
- SyncProgressState 列挙

- SyncProgressState 列挙

SyncProgressDialog クラス

同期の進行状況のモニタリング・ダイアログを表示します。

プロトタイプ

```
' Visual Basic
Public Class SyncProgressDialog
  Inherits Form

// C#
public class SyncProgressDialog :
  Form
```

説明

このダイアログは、モーダルの単一用途ダイアログとして使用してください。アプリケーションが、ダイアログの ShowDialog メソッドを呼び出してダイアログを表示すると、同期が完了するか、ユーザが [キャンセル] ボタンをクリックするまでアプリケーションがブロックされます。

注意：「同期処理中に呼び出される」と記述されているメソッドは、接続、スキーマ、テーブルにアクセスする必要はありません。

参照

- SyncProgressDialog メンバ
- Synchronize メソッド

SyncProgressDialog メンバ

パブリック・インスタンス・コンストラクタ

メンバ	説明
SyncProgressDialog	指定した接続を使用して Ultra Light データベースを同期する SyncProgressDialog インスタンスを構成します。

パブリック・インスタンス・プロパティ

メンバ	説明
SQLCode プロパティ	最後の同期の SQL エラー・コードです。

パブリック・インスタンス・メソッド

メンバ	説明
ShowDialog メソッド	指定した親ウィンドウとともに、モーダル・ダイアログとしてフォームを表示し、同期を行います。このダイアログは、同期が終了したときに閉じます。
ShowDialog メソッド	親ウィンドウなしで、モーダル・ダイアログとしてフォームを表示し、同期を行います。このダイアログは、同期が終了したときに閉じます。
SyncProgressed メソッド	ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。 ほとんどのイベントのステータスについて (SyncProgressData.getState())、表示するメッセージを生成するために getMessage が呼び出され、送信および受信されたバイト数が更新されます。 STATE_SENDING_DATA イベントと STATE_RECEIVED_DATA イベントについては、適切なロー・カウントが更新されます。 STATE_DONE イベントまたは STATE_CANCELLED イベントを受信すると、少し経ってからダイアログが閉じます。 STATE_ERROR イベントを受信すると、エラー・メッセージを生成するために GetErrorMessage が呼び出されるほか、[キャンセル] ボタンが [破棄] ボタンに変化し、このボタンをユーザがクリックするまでダイアログは閉じません。 注意：安全な Ultra Light.NET API 呼び出しは、スキーマの確認の API 呼び出し (`DatabaseSchema`、`TableSchema`、`PublicationSchema`、`IndexSchema`) だけです。

プロテクト・インスタンス・メソッド

メンバ	説明
Dispose メソッド	使用されているリソースをクリーンアップします。
GetErrorMessage メソッド	STATE_ERROR イベントのステータスのメッセージを返します。メッセージは、このエラーの SQL コードを使用してフォーマットされます。 このメソッドは、同期中に呼び出されます。
GetMessage メソッド	イベントのステータスのメッセージを返します。メッセージは、`data.GetTableIndex()`、`data.GetTableCount()`、`data.GetTableTotal()` を使用してフォーマットされます。 このメソッドは、同期中に呼び出されます。
GetString メソッド	ローカライズされた (Ultra Light.NET でサポートされている言語の) 文字列を返します。
OnClosing メソッド	同期をキャンセルする閉じる動作を無効にし、同期でキャンセル要求が処理されるまで閉じません。
ResetValues メソッド	このダイアログに表示されている値を初期化します。

SyncProgressDialog

指定した接続を使用して Ultra Light データベースを同期する SyncProgressDialog インスタンスを構成します。

プロトタイプ

```
' Visual Basic
    Public Sub New( _
      ByVal dbMgr As DatabaseManager, _
      ByVal conn As Connection _
    )
```

```
// C#
public SyncProgressDialog(
  DatabaseManager dbMgr,
  Connection conn
);
```

パラメータ

- **dbMgr** アプリケーションの DatabaseManager
- **conn** 同期する接続

説明

このダイアログは、ShowDialog メソッドを使用してモーダル・ダイアログとして表示してください。

SQLCode プロパティ

最後の同期の SQL エラー・コードです。

プロトタイプ

```
' Visual Basic
Public Readonly Property SQLCode As SQLCode

// C#
public SQLCode SQLCode {get;}
```

プロパティ値

SQL エラー・コード

参照

- SyncProgressDialog クラス
- SyncProgressDialog メンバ
- SQLCode プロパティ

Dispose メソッド

使用されているリソースをクリーンアップします。

プロトタイプ

```
' Visual Basic
Overloads Overrides Protected Sub Dispose( _
  ByVal disposing As Boolean _
)
```

```
// C#
    protected override void Dispose(
      bool disposing
    );
```

パラメータ

- **disposing** マネージ・リソースとアンマネージ・リソースの両方を解放する場合は true、アンマネージ・リソースだけを解放する場合は false

GetErrorMessage メソッド

STATE_ERROR イベントのステータスのメッセージを返します。メッセージは、このエラーの SQL コードを使用してフォーマットされます。

このメソッドは、同期中に呼び出されます。

プロトタイプ

```
' Visual Basic
    Protected Function GetErrorMessage( _
      ByVal sqlCode As SQLCode, _
      ByVal errMsg As String, _
      ByVal uploadOK As Boolean _
    ) As String

// C#
    protected string  GetErrorMessage(
      SQLCode sqlCode,
      string  errMsg,
      bool uploadOK
    );
```

パラメータ

- **sqlCode** このエラーの SQL コード
- **errMsg** SQL コードに関連付けられているエラー・メッセージ
- **uploadOK** アップロードのステータス

戻り値

イベントのメッセージ

参照

- SyncProgressDialog クラス
- SyncProgressDialog メンバ

- SQLCode プロパティ
- SyncResult プロパティ

GetMessage メソッド

イベントのステータスのメッセージを返します。メッセージは、`data.GetTableIndex()`、`data.GetTableCount()`、`data.GetTableTotal()` を使用してフォーマットされます。

このメソッドは、同期中に呼び出されます。

プロトタイプ

```
' Visual Basic
    Protected Function GetMessage( _
      ByVal progress As SyncProgressData _
    ) As String

// C#
    protected string  GetMessage(
      SyncProgressData progress
    );
```

パラメータ

- **progress** 同期のプログレス・データ

戻り値

イベントのメッセージ

GetString メソッド

ローカライズされた (Ultra Light.NET でサポートされている言語の) 文字列を返します。

プロトタイプ

```
' Visual Basic
    Protected Function GetString( _
      ByVal name As String _
    ) As String

// C#
    protected string  GetString(
      string  name
    );
```

パラメータ

- **name** 文字列の名前

戻り値

ローカライズされた文字列

OnClosing メソッド

同期をキャンセルする閉じる動作を無効にし、同期でキャンセル要求が処理されるまで閉じません。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub OnClosing( _
      ByVal e As System.ComponentModel.CancelEventArgs _
    )

// C#
    protected override void OnClosing(
      System.ComponentModel.CancelEventArgs e
    );
```

パラメータ

- **e** イベント・データが含まれる `CancelEventArgs`

ResetValues メソッド

このダイアログに表示されている値を初期化します。

プロトタイプ

```
' Visual Basic
    Protected Sub ResetValues()

// C#
    protected void ResetValues();
```

ShowDialog メソッド

指定した親ウィンドウとともに、モーダル・ダイアログとしてフォームを表示し、同期を行います。このダイアログは、同期が終了したときに閉じます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ShowDialog( _
      ByVal parent As System.Windows.Forms.Form _
    ) As System.Windows.Forms.DialogResult

// C#
    public System.Windows.Forms.DialogResult ShowDialog(
      System.Windows.Forms.Form parent
    );
```

パラメータ

- **parent** モーダル・ダイアログを所有するトップレベル・ウィンドウを表す System.Windows.Forms.Form オブジェクト

戻り値

同期が正常に完了した場合は System.Windows.Forms.DialogResult.OK、同期がユーザによってキャンセルされた場合は System.Windows.Forms.DialogResult.Cancel、同期中にエラーが発生した場合は System.Windows.Forms.DialogResult.Abort (詳細なエラー・レポートについては、SQLCode プロパティ を参照してください)

説明

ShowDialog() よりも、このメソッドの使用をおすすめします。

このダイアログがアクティブになっている間、親は無効になります。

ShowDialog メソッド

親ウィンドウなしで、モーダル・ダイアログとしてフォームを表示し、同期を行います。このダイアログは、同期が終了したときに閉じます。

プロトタイプ

```
' Visual Basic
    Overloads Public Function ShowDialog() As
    System.Windows.Forms.DialogResult

// C#
    public System.Windows.Forms.DialogResult ShowDialog();
```

戻り値

同期が正常に完了した場合は System.Windows.Forms.DialogResult.OK、同期がユーザによってキャンセルされた場合は System.Windows.Forms.DialogResult.Cancel、同期中にエラーが発生した場合は System.Windows.Forms.DialogResult.Abort (詳細なエラー・レポートについては、SQLCode プロパティ を参照してください)

説明

このメソッドよりも、ShowDialog メソッド の使用をおすすめします。

SyncProgressed メソッド

ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

ほとんどのイベントのステータスについて (SyncProgressData.getState())、表示するメッセージを生成するために getMessage が呼び出され、送信および受信されたバイト数が更新されます。STATE_SENDING_DATA イベントと STATE_RECEIVED_DATA イベントについては、適切なロー・カウントが更新されます。

STATE_DONE イベントまたは STATE_CANCELLED イベントを受信すると、少し経ってからダイアログが閉じます。

STATE_ERROR イベントを受信すると、エラー・メッセージを生成するために GetErrorMessage が呼び出されるほか、[キャンセル] ボタンが [破棄] ボタンに変化し、このボタンをユーザがクリックするまでダイアログは閉じません。

注意：安全な Ultra Light.NET API 呼び出しは、スキーマの確認の API 呼び出し (`DatabaseSchema`、`TableSchema`、`PublicationSchema`、`IndexSchema`) だけです。

プロトタイプ

```
' Visual Basic
    Public Function SyncProgressed( _
      ByVal data As SyncProgressData _
    ) As Boolean _
      Implements SyncProgressListener.SyncProgressed
```

```
// C#
    public bool SyncProgressed(
      SyncProgressData data
    );
```

パラメータ

- **data** 最新の同期のプログレス・データを保持しているオブジェクト

戻り値

同期をキャンセルするには true、続行するには false

実装

SyncProgressListener インタフェース

SyncProgressListener インタフェース

同期プログレス・イベントを受信するリスナ・インタフェースです。

プロトタイプ

```
' Visual Basic
    Public Interface SyncProgressListener

// C#
    public interface SyncProgressListener
```

参照

- SyncProgressListener メンバ
- Synchronize メソッド

SyncProgressListener メンバ

パブリック・インスタンス・メソッド

メンバ	説明
SyncProgressed メソッド	ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

SyncProgressed メソッド

ユーザに進行状況を通知するために、同期処理中に起動されます。このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

プロトタイプ

```
' Visual Basic
    Public Function SyncProgressed( _
      ByVal data As SyncProgressData _
    ) As Boolean

// C#
    public bool SyncProgressed(
      SyncProgressData data
    );
```

パラメータ

- **data** 最新の同期のプログレス・データを保持しているオブジェクト

戻り値

このメソッドは true を返して同期をキャンセルするか、false を返して続行する必要があります。

説明

`SyncProgressed` の呼び出し中に、Ultra Light.NET API のメソッドを呼び出さないでください。

SyncProgressState 列挙

同期中に発生する可能性のあるすべてのステータスを列挙します。

プロトタイプ

```
' Visual Basic
    Public Enum SyncProgressState
      Inherits Short

// C#
    public enum SyncProgressState :
      short
```

メンバ

メンバ	説明
STATE_CANCELLED	同期がキャンセルされました。
STATE_COMMITTING_DOWNLOAD	ダウンロードをコミットしています。受信された最終的なロー数が、このイベントに含まれます。ReceivedBytes プロパティ、ReceivedInserts プロパティ、ReceivedUpdates プロパティ、ReceivedDeletes プロパティ を参照してください。
STATE_CONNECTING	同期ストリームは構築されていますが、まだ開かれていません。
STATE_DISCONNECTING	同期ストリームが閉じようとしています。
STATE_DONE	同期が正常に完了しました。接続の SyncResult プロパティ オブジェクトが更新されています。
STATE_ERROR	同期は完了しましたが、エラーが発生しました。詳細については、SyncResult プロパティ、ErrorMessage プロパティ、SQLCode プロパティ を確認してください。

メンバ	説明
STATE_FINISHING_UPLOAD	アップロードが完了しています。送信された最終的なロー数が、このイベントに含まれます。SentBytes プロパティ、SentInserts プロパティ、SentUpdates プロパティ、SentDeletes プロパティ を参照してください。
STATE_LAST	同期の最後のステータスです。このステータスは、常にその他のステータスの後に入力されます。他の SyncProgressData フィールドには、有効なデータがないことがあります。
STATE_RECEIVING_DATA	現在のテーブルのデータを受信しています。ReceivedBytes プロパティ、ReceivedInserts プロパティ、ReceivedUpdates プロパティ、ReceivedDeletes プロパティ が更新されています。
STATE_RECEIVING_TABLE	テーブルが受信されています。進行状況をモニタするには、TableIndex プロパティ と TableCount プロパティ を使用します。
STATE_RECEIVING_UPLOAD_ACK	アップロードが完了したという確認が受信されています。
STATE_ROLLING_BACK_DOWNLOAD	ダウンロード中にエラーが発生したため、同期によってダウンロードがロールバックされています。後続の STATE_ERROR 進行状況レポートにエラーがレポートされます。
STATE_SENDING_DATA	現在のテーブルのデータが送信されています。SentBytes プロパティ、SentInserts プロパティ、SentUpdates プロパティ、SentDeletes プロパティ が更新されています。
STATE_SENDING_DOWNLOAD_ACK	ダウンロードが完了したという確認が送信されています。
STATE_SENDING_HEADER	同期ストリームが開かれ、ヘッダが送信されようとしています。

メンバ	説明
STATE_SENDING_TABLE	テーブルが送信されています。進行状況をモニタするには、TableIndex プロパティ と TableCount プロパティ を使用します。
STATE_STARTING	同期処理はまだ行われていません。

参照

- SyncProgressData クラス

SyncResult クラス

最後の同期のステータスを表します。

プロトタイプ

```
' Visual Basic
    Public Class SyncResult

// C#
    public class SyncResult
```

説明

このクラスを直接インスタンス化することはできません。各接続には、SyncResult プロパティとしてアタッチされた、固有の SyncResult インスタンスがあります。SyncResult インスタンスが有効なのは、接続が開かれている間だけです。

参照

- SyncResult メンバ
- SyncResult プロパティ
- Synchronize メソッド

SyncResult メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
AuthStatus プロパティ	前回行われた同期試行の認証ステータス・コードです。有効な値については、AuthStatusCode 列挙 を参照してください。
AuthValue プロパティ	カスタム・ユーザ認証同期スクリプトからの戻り値です。
IgnoredRows プロパティ	前回の同期中にアップロードされたローが無視された場合は true、ローが無視されなかった場合は false です。
PartialDownloadRetained プロパティ	ダウンロードが中断され、部分ダウンロードが保持された場合は true です。ダウンロードが中断されなかったか、部分ダウンロードがロールバックされた場合は false です。

メンバ	説明
StreamErrorCode プロパティ	ストリーム自体によってレポートされるエラーを返します。既知のコードのリストについては、StreamErrorCode プロパティ を参照してください。
StreamErrorContext プロパティ	ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションです。既知のコンテキストのリストについては、StreamErrorContext プロパティ を参照してください。
StreamErrorID プロパティ	エラーをレポートするネットワーク・レイヤの ID です。既知の ID のリストについては、StreamErrorID プロパティ を参照してください。
StreamErrorSystem プロパティ	ストリーム・エラー・システム固有のコードです。
Timestamp プロパティ	最後の同期のタイムスタンプです。
UploadOK プロパティ	前回のアップロード同期が成功であった場合は true、不成功であった場合は false です。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	関連する JNI ステータスをクリーンアップするデストラクタです。

AuthStatus プロパティ

前回行われた同期試行の認証ステータス・コードです。有効な値については、AuthStatusCode 列挙 を参照してください。

プロトタイプ

```
' Visual Basic
    Public Readonly Property AuthStatus As AuthStatusCode
```

```
// C#
    public AuthStatusCode AuthStatus {get;}
```

AuthValue プロパティ

カスタム・ユーザ認証同期スクリプトからの戻り値です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property AuthValue As Long

// C#
    public long AuthValue {get;}
```

IgnoredRows プロパティ

前回の同期中にアップロードされたローが無視された場合は true、ローが無視されなかった場合は false です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IgnoredRows As Boolean

// C#
    public bool IgnoredRows {get;}
```

参照

- SyncResult クラス
- SyncResult メンバ
- DownloadOnly プロパティ

PartialDownloadRetained プロパティ

ダウンロードが中断され、部分ダウンロードが保持された場合は true です。ダウンロードが中断されなかったか、部分ダウンロードがロールバックされた場合は false です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PartialDownloadRetained As Boolean
```

```
// C#
    public bool PartialDownloadRetained {get;}
```

参照

- SyncResult クラス
- SyncResult メンバ
- KeepPartialDownload プロパティ

StreamErrorCode プロパティ

ストリーム自体によってレポートされるエラーを返します。既知のコードのリストについては、StreamErrorCode プロパティ を参照してください。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorCode As StreamErrorCode

// C#
    public StreamErrorCode StreamErrorCode {get;}
```

StreamErrorContext プロパティ

ストリーム・エラーが発生したときに実行される基本的なネットワーク・オペレーションです。既知のコンテキストのリストについては、StreamErrorContext プロパティ を参照してください。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorContext As StreamErrorContext

// C#
    public StreamErrorContext StreamErrorContext {get;}
```

StreamErrorID プロパティ

エラーをレポートするネットワーク・レイヤの ID です。既知の ID のリストについては、StreamErrorID プロパティ を参照してください。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorID As StreamErrorID
```

```
// C#
    public StreamErrorID StreamErrorID {get;}
```

StreamErrorSystem プロパティ

ストリーム・エラー・システム固有のコードです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property StreamErrorSystem As Integer

// C#
    public int StreamErrorSystem {get;}
```

Timestamp プロパティ

最後の同期のタイムスタンプです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Timestamp As Date

// C#
    public DateTime Timestamp {get;}
```

UploadOK プロパティ

前回のアップロード同期が成功であった場合は true、不成功であった場合は false です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UploadOK As Boolean

// C#
    public bool UploadOK {get;}
```

Finalize メソッド

関連する JNI ステータスをクリーンアップするデストラクタです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

Table クラス

Ultra Light データベース内のテーブルを示します。

プロトタイプ

```
' Visual Basic
    Public Class Table
      Inherits Cursor

// C#
    public class Table :
      Cursor
```

説明

このクラスを直接インスタンス化することはできません。テーブルは、Connection クラス の GetTable メソッド を使用して作成されます。

Table メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
IsBOF プロパティ (継承 Cursor)	現在のローの位置が最初のローの前なら true を返します。
IsEOF プロパティ (継承 Cursor)	現在のローの位置が最後のローの後なら true を返します。
IsOpen プロパティ (継承 Cursor)	カーソルが現在開いているかどうかを確認します。
RowCount プロパティ (継承 Cursor)	カーソル内のロー数を返します。
Schema プロパティ	このテーブルのスキーマを保持します。このプロパティが有効なのは、接続が開かれている間だけです。

パブリック・インスタンス・メソッド

メンバ	説明
AppendBytes メソッド	指定した Byte 配列の指定したサブセットを、指定した SQLType 列挙 カラムの新しい値に追加します。
AppendChars メソッド	指定した System.Char 配列の指定したサブセットを、指定した SQLType 列挙 カラムの新しい値に追加します。
Close メソッド (継承 Cursor)	カーソルを閉じます。
Delete メソッド	現在の行を削除します。
DeleteAllRows メソッド	テーブルのすべてのローを削除します。
Fill メソッド (継承 Cursor)	ADO .NET DataTable オブジェクトに、カーソル内のローを格納します。
FindBegin メソッド	このテーブルで新規に検索を実行する準備を行います。
FindFirst メソッド	テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセット全体に完全に一致するローを検索します。
FindFirst メソッド	テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットの一部に完全に一致するローを検索します。
FindLast メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に完全に一致するローを検索します。
FindLast メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に完全に一致するローを検索します。
FindNext メソッド	現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。

メンバ	説明
FindNext メソッド	現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。
FindPrevious メソッド	現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。
FindPrevious メソッド	現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。
GetBoolean メソッド (継承 Cursor)	指定したカラムの値を Boolean として返します。
GetBytes メソッド (継承 Cursor)	指定したカラムの値を Byte 配列として返します。SQLType 列挙 型、SQLType 列挙 型、SQLType 列挙 型のカラムの場合にのみ有効です。
GetBytes メソッド (継承 Cursor)	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の Byte 配列の指定したオフセットにコピーします。
GetChars メソッド (継承 Cursor)	指定したオフセットで始まる、指定した SQLType 列挙 カラムの値のサブセットを、コピー先の System.Char 配列の指定したオフセットにコピーします。
GetDouble メソッド (継承 Cursor)	指定したカラムの値を Double として返します。
GetFloat メソッド (継承 Cursor)	指定したカラムの値を Single として返します。
GetInt メソッド (継承 Cursor)	指定したカラムの値を Int32 として返します。

メンバ	説明
GetLong メソッド (継承 Cursor)	指定したカラムの値を Int64 として返します。
GetShort メソッド (継承 Cursor)	指定したカラムの値を Int16 として返します。
GetString メソッド (継承 Cursor)	指定したカラムの値を String として返します。
GetTime メソッド (継承 Cursor)	指定したカラムの値を、ミリ秒の精度の TimeSpan として返します。
GetTimestamp メソッド (継承 Cursor)	指定したカラムの値を、ミリ秒の精度の DateTime として返します。
GetUInt メソッド (継承 Cursor)	指定したカラムの値を UInt32 として返します。
GetULong メソッド (継承 Cursor)	指定したカラムの値を UInt64 として返します。
GetUShort メソッド (継承 Cursor)	指定したカラムの値を UInt16 として返します。
GetUUID メソッド (継承 Cursor)	指定したカラムの値を UUID (Guid) として返します。
Insert メソッド	現在のカラム値 (set メソッドを使用して指定されます) で新しいローを挿入します。 挿入を行う前に InsertBegin メソッド を呼び出してください。
InsertBegin メソッド	現在のすべてのカラムをデフォルト値に設定して、このテーブルに新しいローを挿入する準備を行います。
IsNull メソッド (継承 Cursor)	指定されたカラムの値が NULL かどうかをチェックします。
LookupBackward メソッド	テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より小さい値を持つローを検索します。

メンバ	説明
LookupBackward メソッド	テーブルを最初から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より小さい値を持つローを検索します。
LookupBegin メソッド	このテーブルで新規に検索を実行する準備を行います。検索する値は、このテーブルを開いたインデックス内のカラムで適切な set*Type* メソッドを呼び出して指定します。
LookupForward メソッド	テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より大きい値を持つローを検索します。
LookupForward メソッド	テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より大きい値を持つローを検索します。
MoveAfterLast メソッド (継承 Cursor)	カーソルの最後のローの後に移動します。
MoveBeforeFirst メソッド (継承 Cursor)	カーソルの最初のローの前に移動します。
MoveFirst メソッド (継承 Cursor)	カーソルの最初のローに移動します。
MoveLast メソッド (継承 Cursor)	カーソルの最後のローに移動します。
MoveNext メソッド (継承 Cursor)	次のローに、または最後のローの後に移動します。
MovePrevious メソッド (継承 Cursor)	前のローに、または最初のローの前に移動します。
MoveRelative メソッド (継承 Cursor)	現在のローを基準として移動します。
Open メソッド	プライマリ・キーを使用して、このテーブルをデータ・アクセス用に開きます。

メンバ	説明
Open メソッド	指定したインデックスを使用して、このテーブルをデータ・アクセス用に開きます。
SetBoolean メソッド	指定したカラムの値を、Boolean を使用して設定します。
SetBytes メソッド	指定したカラムの値を、Byte の配列を使用して設定します。
SetDouble メソッド	指定したカラムの値を、Double を使用して設定します。
SetFloat メソッド	指定したカラムの値を、Single を使用して設定します。
SetInt メソッド	指定したカラムの値を、Int32 を使用して設定します。
SetLong メソッド	指定したカラムの値を、Int64 を使用して設定します。
SetNull メソッド	カラムを NULL に設定します。
SetShort メソッド	指定したカラムの値を、Int16 を使用して設定します。
SetString メソッド	指定したカラムの値を、String を使用して設定します。
SetTime メソッド	指定したカラムの値を、TimeSpan を使用して設定します。
SetTimestamp メソッド	指定したカラムの値を、DateTime を使用して設定します。
SetToDefault メソッド	指定したカラムの値を、そのデフォルト値に設定します。
SetUInt メソッド	指定したカラムの値を、UInt32 を使用して設定します。
SetULong メソッド	指定したカラムの値を、UInt64 を使用して設定します。

メンバ	説明
SetUShort メソッド	指定したカラムの値を、UInt16 を使用して設定します。
SetUUID メソッド	指定したカラムの値を、Guid を使用して設定します。
Truncate メソッド	テーブル内のすべてのローを削除し、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。
Update メソッド	現在のカラム値 (set メソッドを使用して指定されます) で新しいローを更新します。 更新を行う前に UpdateBegin メソッド を呼び出してください。
UpdateBegin メソッド	このテーブルの現在のローを更新する準備を行います。

プロテクト・インスタンス・メソッド

メンバ	説明
Finalize メソッド	Finalize メソッドです。

Schema プロパティ

このテーブルのスキーマを保持します。このプロパティが有効なのは、接続が開かれている間だけです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Schema As TableSchema

// C#
    public TableSchema Schema {get;}
```

参照

- Table クラス
- Table メンバ
- TableSchema クラス

AppendBytes メソッド

指定した Byte 配列の指定したサブセットを、指定した SQLType 列挙カラムの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendBytes( _
      ByVal columnID As Short, _
      ByVal val As Byte(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )

// C#
    public void AppendBytes(
      short columnID,
      byte[] val,
      int srcOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの現在の新しい値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置のバイトが、指定したカラムの値に追加されます。

挿入時には、InsertBegin メソッド は新しい値をカラムのデフォルト値に初期化します。ローのデータは、Insert メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

更新時の、カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、処理対象は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *val*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

AppendChars メソッド

指定した System.Char 配列の指定したサブセットを、指定した SQLType 列挙 カラムの新しい値に追加します。

プロトタイプ

```
' Visual Basic
    Public Sub AppendChars( _
      ByVal columnID As Short, _
      ByVal val As Char(), _
      ByVal srcOffset As Integer, _
      ByVal count As Integer _
    )

// C#
    public void AppendChars(
      short columnID,
      char[] val,
      int srcOffset,
      int count
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの現在の新しい値に追加する値
- **srcOffset** ソース配列の開始位置
- **count** コピーされるバイト数

説明

srcOffset (0 から始まります) から、配列 *val* の *srcOffset*+*count*-1 までの位置の文字が、指定したカラムの値に追加されます。挿入時には、InsertBegin メソッド は新しい値をカラムのデフォルト値に初期化します。ローのデータは、Insert メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

更新時の、カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

次のいずれかに該当する場合は、コード SQLCode 列挙 とともに SQLException クラス がスローされ、コピー先は修正されません。その他のエラーの場合は、それに応じたエラー・コードとともに SQLException クラス がスローされます。

- *val* が null である。
- *srcOffset* が負である。
- *count* が負である。
- *srcOffset*+*count* が *value*.Length よりも大きい。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

Delete メソッド

現在の行を削除します。

プロトタイプ

```
' Visual Basic
    Public Sub Delete()

// C#
    public void Delete();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- StartSynchronizationDelete メソッド
- StopSynchronizationDelete メソッド
- AutoCommit フィールド

DeleteAllRows メソッド

テーブルのすべてのローを削除します。

プロトタイプ

```
' Visual Basic
    Public Sub DeleteAllRows()

// C#
    public void DeleteAllRows();
```

説明

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータ・セットをテーブルにダウンロードするほうが便利なことがあります。StopSynchronizationDelete メソッドを使用すると、統合データベースからは削除しないで Ultra Light データベースからローを削除できます。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- Truncate メソッド

- AutoCommit フィールド

Finalize メソッド

Finalize メソッドです。

プロトタイプ

```
' Visual Basic
    Overrides Protected Sub Finalize()

// C#
    protected override void Finalize();
```

FindBegin メソッド

このテーブルで新規に検索を実行する準備を行います。

プロトタイプ

```
' Visual Basic
    Public Sub FindBegin()

// C#
    public void FindBegin();
```

説明

検索する値は、このテーブルを開いたインデックス内のカラムで適切な set*Type* メソッドを呼び出して指定します。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindFirst メソッド
- FindFirst メソッド
- FindLast メソッド
- FindLast メソッド

FindFirst メソッド

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセット全体に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindFirst() As Boolean

// C#
    public bool FindFirst();
```

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド

FindFirst メソッド

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットの一部に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindFirst( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindFirst(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド

FindLast メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindLast() As Boolean

// C#
    public bool FindLast();
```

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド

FindLast メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に完全に一致するローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindLast( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindLast(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に FindBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindBegin メソッド
- FindNext メソッド
- FindPrevious メソッド

FindNext メソッド

現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindNext() As Boolean

// C#
    public bool FindNext();
```

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは次のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の `FindNext` の動作は不確定です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindFirst メソッド

FindNext メソッド

現在の位置からテーブルを順方向に移動しながら、次のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindFirst メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindNext( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool FindNext(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは次のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の `FindNext` の動作は不確定です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindFirst メソッド

FindPrevious メソッド

現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセット全体に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindPrevious() As Boolean

// C#
    public bool FindPrevious();
```

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは前のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の `FindPrevious` の動作は不確定です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindLast メソッド

FindPrevious メソッド

現在の位置からテーブルを逆方向に移動しながら、前のローが現在のインデックスの値かそのセットの一部に完全に一致するかどうかを調べて、FindLast メソッド 検索を続行します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function FindPrevious( _
      ByVal numColumns As Short _
    ) As Boolean
```

// C#
public bool **FindPrevious(**
short *numColumns*
);

パラメータ

- **numColumns** 複合インデックスのための、検索で使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を 1 に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

インデックスの値と完全に一致すると、カーソルは前のローで停止します。失敗すると、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索されるカラムの値がローの更新中に修正された場合の `FindPrevious` の動作は不確定です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- FindLast メソッド

Insert メソッド

現在のカラム値 (set メソッドを使用して指定されます) で新しいローを挿入します。

挿入を行う前に InsertBegin メソッド を呼び出してください。

プロトタイプ

' Visual Basic
Public Sub **Insert()**

// C#
public void **Insert();**

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- AutoCommit フィールド

InsertBegin メソッド

現在のすべてのカラムをデフォルト値に設定して、このテーブルに新しいローを挿入する準備を行います。

プロトタイプ

```
' Visual Basic
    Public Sub InsertBegin()

// C#
    public void InsertBegin();
```

説明

適切な Set*Type* メソッドまたは Append*Type* メソッドを呼び出して、挿入するデフォルト以外の値を指定します。

Insert メソッド が実行されないと、ローが実際に挿入されることも、ロー内のデータが実際に変更されることもありません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- Insert メソッド

LookupBackward メソッド

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より小さい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupBackward() As Boolean

// C#
    public bool LookupBackward();
```

戻り値　成功した場合は true、失敗した場合は false

説明　検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致する最初のローか、それより少ない値の最初のローで停止します。失敗した場合 (検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- LookupBegin メソッド

LookupBackward メソッド

テーブルを最初から逆方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より小さい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupBackward( _
      ByVal numColumns As Short _
    ) As Boolean

// C#
    public bool LookupBackward(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、ルックアップで使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を 1 に指定します。

戻り値　成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致する最初のローか、それより少ない値の最初のローで停止します。失敗した場合 (検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 (IsBOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- LookupBegin メソッド

LookupBegin メソッド

このテーブルで新規に検索を実行する準備を行います。検索する値は、このテーブルを開いたインデックス内のカラムで適切な set*Type* メソッドを呼び出して指定します。

プロトタイプ

```
' Visual Basic
    Public Sub LookupBegin()

// C#
    public void LookupBegin();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- LookupForward メソッド
- LookupForward メソッド
- LookupBackward メソッド
- LookupBackward メソッド

LookupForward メソッド

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセット全体に一致するか、その値より大きい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupForward() As Boolean

// C#
    public bool LookupForward();
```

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。失敗した場合 (検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- LookupBegin メソッド

LookupForward メソッド

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットの一部に一致するか、その値より大きい値を持つローを検索します。

プロトタイプ

```
' Visual Basic
    Overloads Public Function LookupForward( _
      ByVal numColumns As Short _
    ) As Boolean
```

```
// C#
    public bool LookupForward(
      short numColumns
    );
```

パラメータ

- **numColumns** 複合インデックスのための、ルックアップで使用するカラムの数。たとえば、3 つのカラムのインデックスがあり、最初のカラムにのみ基づいて一致する値を検索する場合は、最初のカラムに値を設定し、*numColumns* の値を `1` に指定します。

戻り値

成功した場合は true、失敗した場合は false

説明

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。失敗した場合 (検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ (IsEOF プロパティ) になります。

検索を行う前に LookupBegin メソッド を呼び出してください。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- LookupBegin メソッド

Open メソッド

プライマリ・キーを使用して、このテーブルをデータ・アクセス用に開きます。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Open()

// C#
    public void Open();
```

例外

- SQLException クラス - SQL エラーが発生しました。

Open メソッド

指定したインデックスを使用して、このテーブルをデータ・アクセス用に開きます。

プロトタイプ

```
' Visual Basic
    Overloads Public Sub Open( _
      ByVal index As String _
    )

// C#
    public void Open(
      string  index
    );
```

パラメータ

- **index** テーブルを開くインデックスの名前。`null` の場合は、プライマリ・キーが使用されます。

例外

- SQLException クラス - SQL エラーが発生しました。

SetBoolean メソッド

指定したカラムの値を、Boolean を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBoolean( _
      ByVal columnID As Short, _
      ByVal val As Boolean _
    )

// C#
    public void SetBoolean(
      short columnID,
      bool val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetBytes メソッド

指定したカラムの値を、Byte の配列を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetBytes( _
      ByVal columnID As Short, _
      ByVal val As Byte() _
    )

// C#
    public void SetBytes(
      short columnID,
      byte[] val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

SQLType 列挙 型または SQLType 列挙 型のカラム、あるいは val の長さが 16 の場合の SQLType 列挙 型のカラムにのみ適しています。ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド

- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetDouble メソッド

指定したカラムの値を、Double を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetDouble( _
      ByVal columnID As Short, _
      ByVal val As Double _
    )

// C#
    public void SetDouble(
      short columnID,
      double val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ

- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetFloat メソッド

指定したカラムの値を、Single を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetFloat( _
      ByVal columnID As Short, _
      ByVal val As Single _
    )
```

```
// C#
    public void SetFloat(
      short columnID,
      float  val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド

- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetInt メソッド

指定したカラムの値を、Int32 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetInt( _
      ByVal columnID As Short, _
      ByVal val As Integer _
    )

// C#
    public void SetInt(
      short columnID,
      int val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド

- SetDouble メソッド
- SetFloat メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetLong メソッド

指定したカラムの値を、Int64 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetLong( _
      ByVal columnID As Short, _
      ByVal val As Long _
    )

// C#
    public void SetLong(
      short columnID,
      long val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetNull メソッド

カラムを NULL に設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetNull( _
      ByVal columnID As Short _
    )

// C#
    public void SetNull(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。

説明

データは、Insert メソッド または Update メソッド を実行するまでは、実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド

- Schema プロパティ
- IsColumnNullable メソッド

SetShort メソッド

指定したカラムの値を、Int16 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetShort( _
      ByVal columnID As Short, _
      ByVal val As Short _
    )

// C#
    public void SetShort(
      short columnID,
      short val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド

- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetString メソッド

指定したカラムの値を、String を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetString( _
      ByVal columnID As Short, _
      ByVal val As String _
    )

// C#
    public void SetString(
      short columnID,
      string  val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetTime メソッド

指定したカラムの値を、TimeSpan を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetTime( _
      ByVal columnID As Short, _
      ByVal val As TimeSpan _
    )

// C#
    public void SetTime(
      short columnID,
      TimeSpan val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

設定された値は、ミリ秒の精度であり、0 ～ 24 時間の負でない値に正規化されます。ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetUInt メソッド

- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetTimestamp メソッド

指定したカラムの値を、DateTime を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetTimestamp( _
      ByVal columnID As Short, _
      ByVal val As Date _
    )

// C#
    public void SetTimestamp(
      short columnID,
      DateTime val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

設定された値の精度はミリ秒です。ローのデータは、Insert メソッドまたは Update メソッドが実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド

- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetToDefault メソッド

指定したカラムの値を、そのデフォルト値に設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetToDefault( _
      ByVal columnID As Short _
    )

// C#
    public void SetToDefault(
      short columnID
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetNull メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド
- GetColumnDefaultValue メソッド

SetUInt メソッド

指定したカラムの値を、UInt32 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUInt( _
      ByVal columnID As Short, _
      ByVal val As UInt32 _
    )

// C#
    public void SetUInt(
      short columnID,
      uint val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetULong メソッド

- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetULong メソッド

指定したカラムの値を、UInt64 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetULong( _
      ByVal columnID As Short, _
      ByVal val As UInt64 _
    )

// C#
    public void SetULong(
      short columnID,
      ulong val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド

- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetUShort メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetUShort メソッド

指定したカラムの値を、UInt16 を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUShort( _
      ByVal columnID As Short, _
      ByVal val As UInt16 _
    )

// C#
    public void SetUShort(
      short columnID,
      ushort val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド
- SetLong メソッド
- SetShort メソッド
- SetString メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUUID メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド

SetUUID メソッド

指定したカラムの値を、Guid を使用して設定します。

プロトタイプ

```
' Visual Basic
    Public Sub SetUUID( _
      ByVal columnID As Short, _
      ByVal val As Guid _
    )

// C#
    public void SetUUID(
      short columnID,
      Guid val
    );
```

パラメータ

- **columnID** カラムの ID 番号。テーブルの最初のカラムの ID 値は 1 です。
- **val** カラムの新しい値

説明

ローのデータは、Insert メソッド または Update メソッド が実行されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。SQLType 列挙 型のカラム、または長さが 16 の SQLType 列挙 型のカラムの場合にのみ有効です。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- GetNewUUID メソッド
- GetColumnID メソッド
- SetToDefault メソッド
- SetNull メソッド
- AppendBytes メソッド
- AppendChars メソッド
- SetBytes メソッド
- SetBoolean メソッド
- SetDouble メソッド
- SetFloat メソッド
- SetInt メソッド

- SetLong メソッド
- SetShort メソッド
- SetTime メソッド
- SetTimestamp メソッド
- SetUInt メソッド
- SetULong メソッド
- SetUShort メソッド
- FindBegin メソッド
- LookupBegin メソッド
- InsertBegin メソッド
- UpdateBegin メソッド
- Schema プロパティ
- GetColumnSQLType メソッド
- GetColumnSize メソッド

Truncate メソッド

テーブル内のすべてのローを削除し、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。

プロトタイプ

```
' Visual Basic
    Public Sub Truncate()

// C#
    public void Truncate();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- DeleteAllRows メソッド
- AutoCommit フィールド

Update メソッド

現在のカラム値 (set メソッドを使用して指定されます) で新しいローを更新します。

更新を行う前に UpdateBegin メソッド を呼び出してください。

プロトタイプ

```
' Visual Basic
    Public Sub Update()

// C#
    public void Update();
```

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- Table クラス
- Table メンバ
- AutoCommit フィールド

UpdateBegin メソッド

このテーブルの現在のローを更新する準備を行います。

プロトタイプ

```
' Visual Basic
    Public Sub UpdateBegin()

// C#
    public void UpdateBegin();
```

説明

カラム値は、適切な set*Type* メソッドまたは Append*Type* メソッドを呼び出すことによって修正します。カラムに対する最初の追加では、現在のカラム値がクリアされてから新しい値が追加されます。

ローのデータは、Update メソッド が呼び出されるまで実際には変更されません。また、その変更がコミットされないかぎり、永続化されません。

テーブルを開くのに使用されるインデックス内のカラムを修正すると、アクティブな検索処理に予期しない影響を及ぼします。テーブルのプライマリ・キー内のカラムは更新できません。

例外

- SQLException クラス - SQL エラーが発生しました。

TableSchema クラス

Ultra Light のテーブルのスキーマを表します。

プロトタイプ

```
' Visual Basic
   NotInheritable Public Class TableSchema
    Inherits CursorSchema

// C#
   public sealed class TableSchema :
    CursorSchema
```

説明

このクラスを直接インスタンス化することはできません。TableSchema クラス オブジェクトは、その Schema プロパティ としてテーブルにアタッチされます。

TableSchema メンバ

パブリック・インスタンス・プロパティ

メンバ	説明
ColumnCount プロパティ (継承 CursorSchema)	このカーソル内のカラム数を返します。
IndexCount プロパティ	このテーブルのインデックス数です。
IsNeverSynchronized プロパティ	このテーブルが、まったく同期されないようにマーク付けされているかどうかを示します。
IsOpen プロパティ (継承 CursorSchema)	カーソルのスキーマが現在開いているかどうかを確認します。
Name プロパティ	このテーブルの名前です。
PrimaryKey プロパティ	このテーブルのプライマリ・キーのインデックス・スキーマです。

メンバ	説明
UploadsUnchangedRows プロパティ	データベースが、変更されていないローをアップロードするかどうかを示します。

パブリック・インスタンス・メソッド

メンバ	説明
GetColumnDefaultValue メソッド	指定したカラムのデフォルト値です。
GetColumnID メソッド (継承 CursorSchema)	指定したカラムのカラム ID を返します。
GetColumnName メソッド (継承 CursorSchema)	指定したカラム ID で識別されたカラムの名前を返します。
GetColumnPartitionSize メソッド	指定したカラムに割り当てられているグローバル・オートインクリメントの分割サイズを返します。
GetColumnPrecision メソッド (継承 CursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの精度を返します。
GetColumnScale メソッド (継承 CursorSchema)	カラムが数値カラム (SQL NUMERIC 型) であれば、指定したカラムの位取りを返します。
GetColumnSize メソッド (継承 CursorSchema)	カラムがサイズ指定されたカラム (SQL BINARY 型または CHAR 型) であれば、指定したカラムのサイズを返します。
GetColumnSQLType メソッド (継承 CursorSchema)	指定したカラムの SQL データ型を取得します。
GetIndex メソッド	指定したインデックスのインデックス・スキーマを返します。

メンバ	説明
GetIndexName メソッド	指定したインデックス ID で識別されたインデックスの名前を返します。
GetOptimalIndex メソッド	指定したカラムを使用してテーブルを検索するために最適なインデックスです。
IsColumnAutoIncrement メソッド	指定したカラムのデフォルトがオートインクリメントに設定されているかどうかをチェックします。
IsColumnCurrentDate メソッド	指定したカラムのデフォルトが、現在の日付 (SQLType.DATE) に設定されているかどうかをチェックします。
IsColumnCurrentTime メソッド	指定したカラムのデフォルトが、現在の時刻 (SQLType.TIME) に設定されているかどうかをチェックします。
IsColumnCurrentTimestamp メソッド	指定したカラムのデフォルトが、現在のタイムスタンプ (SQLType.TIMESTAMP) に設定されているかどうかをチェックします。
IsColumnGlobalAutoIncrement メソッド	指定したカラムのデフォルトがグローバル・オートインクリメントに設定されているかどうかをチェックします。
IsColumnNewUUID メソッド	指定したカラムのデフォルトが新しい UUID (Guid) に設定されているかどうかをチェックします。
IsColumnNullable メソッド	指定したカラムが null 入力可能であるかどうかをチェックします。
IsInPublication メソッド	このテーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

IndexCount プロパティ

このテーブルのインデックス数です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IndexCount As Integer

// C#
    public int IndexCount {get;}
```

説明

インデックス ID の範囲は、1 ～ `IndexCount` です。

注意：インデックスの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

IsNeverSynchronized プロパティ

このテーブルが、まったく同期されないようにマーク付けされているかどうかを示します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property IsNeverSynchronized As Boolean

// C#
    public bool IsNeverSynchronized {get;}
```

プロパティ値

このテーブルが、まったく同期されないようにマーク付けされている場合は true

説明

まったく同期されないようにマーク付けされているテーブルは、パブリケーションに含まれているものであっても、まったく同期されていません。このようなテーブルは、「未同期」テーブルと呼ばれることもあります。

例外

- SQLException クラス - SQL エラーが発生しました。

Name プロパティ

このテーブルの名前です。

プロトタイプ

```
' Visual Basic
    Public Readonly Property Name As String

// C#
    public string  Name {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。

PrimaryKey プロパティ

このテーブルのプライマリ・キーのインデックス・スキーマです。

プロトタイプ

```
' Visual Basic
    Public Readonly Property PrimaryKey As IndexSchema

// C#
    public IndexSchema PrimaryKey {get;}
```

例外

- SQLException クラス - SQL エラーが発生しました。

UploadsUnchangedRows プロパティ

データベースが、変更されていないローをアップロードするかどうかを示します。

プロトタイプ

```
' Visual Basic
    Public Readonly Property UploadsUnchangedRows As Boolean

// C#
    public bool UploadsUnchangedRows {get;}
```

プロパティ値

このテーブルが、同期中に常にすべてのローをアップロードするようにマーク付けされている場合は true

説明

このようにマーク付けされているテーブルでは、その同期時に、変更済みのローと未変更のローがアップロードされます。このようなテーブルは、「完全同期」テーブルと呼ばれることもあります。

例外

- SQLException クラス - SQL エラーが発生しました。

GetColumnDefaultValue メソッド

指定したカラムのデフォルト値です。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnDefaultValue( _
      ByVal name As String _
    ) As String

// C#
    public string  GetColumnDefaultValue(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

指定したカラムの文字列としてのデフォルト値。または、デフォルト値が `null` の場合は null としてのデフォルト値

例外

- SQLException クラス - SQL エラーが発生しました。

GetColumnPartitionSize メソッド

指定したカラムに割り当てられているグローバル・オートインクリメントの分割サイズを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetColumnPartitionSize( _
      ByVal name As String _
    ) As UInt64

// C#
    public ulong GetColumnPartitionSize(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムのグローバル・オートインクリメントの分割サイズ

説明　テーブルのすべてのグローバル・オートインクリメント・カラムは、同じグローバル・オートインクリメントの分割サイズを共有します。

例外
- SQLException クラス - SQL エラーが発生しました。

参照
- TableSchema クラス
- TableSchema メンバ
- IsColumnGlobalAutoIncrement メソッド

GetIndex メソッド

指定したインデックスのインデックス・スキーマを返します。

プロトタイプ

```
' Visual Basic
    Public Function GetIndex( _
      ByVal name As String _
    ) As IndexSchema

// C#
    public IndexSchema GetIndex(
      string  name
    );
```

パラメータ
- **name** インデックスの名前

戻り値　指定したインデックスのインデックス・スキーマ

例外
- SQLException クラス - SQL エラーが発生しました。

GetIndexName メソッド

指定したインデックス ID で識別されたインデックスの名前を返します。

プロトタイプ

```
' Visual Basic
    Public Function GetIndexName( _
      ByVal indexID As Integer _
    ) As String
```

```
// C#
    public string  GetIndexName(
      int indexID
    );
```

パラメータ

- **indexID** インデックスの ID。*indexID* は、`[1,TableSchema.IndexCount]` の範囲内である必要があります。

戻り値

インデックスの名前

説明

インデックスの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後にリフレッシュします。

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- TableSchema クラス
- TableSchema メンバ
- IndexCount プロパティ

GetOptimalIndex メソッド

指定したカラムを使用してテーブルを検索するために最適なインデックスです。

プロトタイプ

```
' Visual Basic
    Public Function GetOptimalIndex( _
      ByVal name As String _
    ) As IndexSchema

// C#
    public IndexSchema GetOptimalIndex(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

指定したカラムのインデックス

説明　指定したカラムは、インデックス内の最初のカラムになりますが、インデックスには複数のカラムがある場合があります。

例外
- SQLException クラス - SQL エラーが発生しました。

IsColumnAutoIncrement メソッド

指定したカラムのデフォルトがオートインクリメントに設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnAutoIncrement( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnAutoIncrement(
      string  name
    );
```

パラメータ
- **name** カラムの名前

戻り値　カラムがオートインクリメントされる場合は true、オートインクリメントされない場合は false

例外
- SQLException クラス - SQL エラーが発生しました。

IsColumnCurrentDate メソッド

指定したカラムのデフォルトが、現在の日付 (SQLType.DATE) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentDate( _
      ByVal name As String _
    ) As Boolean
```

```
// C#
    public bool IsColumnCurrentDate(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムのデフォルトが現在の日付に設定されている場合は true、現在の日付に設定されていない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

IsColumnCurrentTime メソッド

指定したカラムのデフォルトが、現在の時刻 (SQLType.TIME) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentTime( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnCurrentTime(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムのデフォルトが現在の時刻に設定されている場合は true、現在の時刻に設定されていない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

IsColumnCurrentTimestamp メソッド

指定したカラムのデフォルトが、現在のタイムスタンプ (SQLType.TIMESTAMP) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnCurrentTimestamp( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnCurrentTimestamp(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムのデフォルトが現在のタイムスタンプに設定されている場合は true、現在のタイムスタンプに設定されていない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

IsColumnGlobalAutoIncrement メソッド

指定したカラムのデフォルトがグローバル・オートインクリメントに設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnGlobalAutoIncrement( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnGlobalAutoIncrement(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムがグローバル・オートインクリメントされる場合は true、グローバル・オートインクリメントされない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

参照

- TableSchema クラス
- TableSchema メンバ

- GetColumnPartitionSize メソッド
- DatabaseID プロパティ

IsColumnNewUUID メソッド

指定したカラムのデフォルトが新しい UUID (Guid) に設定されているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnNewUUID( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnNewUUID(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値

カラムのデフォルトが新しい UUID に設定されている場合は true、新しい UUID に設定されていない場合は false

例外

- SQLException クラス - SQL エラーが発生しました。

IsColumnNullable メソッド

指定したカラムが null 入力可能であるかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsColumnNullable( _
      ByVal name As String _
    ) As Boolean

// C#
    public bool IsColumnNullable(
      string  name
    );
```

パラメータ

- **name** カラムの名前

戻り値　カラムが null 入力可能であれば true、null 入力が不可であれば false

例外
- SQLException クラス - SQL エラーが発生しました。

IsInPublication メソッド

このテーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

プロトタイプ

```
' Visual Basic
    Public Function IsInPublication( _
      ByVal pubName As String _
    ) As Boolean

// C#
    public bool IsInPublication(
      string  pubName
    );
```

パラメータ
- **pubName** パブリケーションの名前

戻り値　テーブルがパブリケーション内にある場合は true、パブリケーション内にない場合は false

例外
- SQLException クラス - SQL エラーが発生しました。

索引

A

AcceptChangesDuringFill プロパティ
ネームスペース iAnywhere.Data.UltraLite 155
ActiveSyncInvoked メソッド
ネームスペース iAnywhere.Data.UltraLite 64
ネームスペース iAnywhere.UltraLite 432
ActiveSyncListener インタフェース
ネームスペース iAnywhere.UltraLite 432
ActiveSync 同期
Ultra Light.NET 59
AdditionalParms プロパティ
ネームスペース iAnywhere.Data.UltraLite 128
ネームスペース iAnywhere.UltraLite 516, 584
Add メソッド
ネームスペース iAnywhere.Data.UltraLite 272, 273, 274, 275, 276, 277
AppendBytesParameter メソッド
ネームスペース iAnywhere.UltraLite 547
AppendBytes メソッド
ネームスペース iAnywhere.Data.UltraLite 376
ネームスペース iAnywhere.UltraLite 677
AppendCharsParameter メソッド
ネームスペース iAnywhere.UltraLite 549
AppendChars メソッド
ネームスペース iAnywhere.Data.UltraLite 378
ネームスペース iAnywhere.UltraLite 679
ApplyFile メソッド
ネームスペース iAnywhere.Data.UltraLite 183, 184
ネームスペース iAnywhere.UltraLite 528, 529
Ultra Light.NET 開発 31
AuthenticationParms プロパティ
ネームスペース iAnywhere.Data.UltraLite 336
ネームスペース iAnywhere.UltraLite 628
AuthStatusCode 列挙
ネームスペース iAnywhere.UltraLite 436
AuthStatus プロパティ
ネームスペース iAnywhere.Data.UltraLite 363
ネームスペース iAnywhere.UltraLite 665
AuthValue プロパティ
ネームスペース iAnywhere.Data.UltraLite 364
ネームスペース iAnywhere.UltraLite 666
AutoCommit フィールド
ネームスペース iAnywhere.UltraLite 440
AutoCommit モード
Ultra Light.NET 54

B

BeginTransaction メソッド
ネームスペース iAnywhere.Data.UltraLite 102, 103

C

CacheSize プロパティ
ネームスペース iAnywhere.Data.UltraLite 132
ネームスペース iAnywhere.UltraLite 461
Cancel メソッド
ネームスペース iAnywhere.Data.UltraLite 81
ChangeDatabase メソッド
ネームスペース iAnywhere.Data.UltraLite 104
ChangeEncryptionKey メソッド
ネームスペース iAnywhere.Data.UltraLite 105
ネームスペース iAnywhere.UltraLite 447
CheckpointStore プロパティ
ネームスペース iAnywhere.Data.UltraLite 337
ネームスペース iAnywhere.UltraLite 629
Clear メソッド
ネームスペース iAnywhere.Data.UltraLite 278
Close メソッド

ネームスペース iAnywhere.Data.UltraLite 105, 200
ネームスペース iAnywhere.UltraLite 447, 473, 551, 577
CollationName プロパティ
ネームスペース iAnywhere.Data.UltraLite 180
ネームスペース iAnywhere.UltraLite 525
ColumnCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 141, 240
ネームスペース iAnywhere.UltraLite 499, 536
CommandText プロパティ
ネームスペース iAnywhere.Data.UltraLite 76
CommandTimeout プロパティ
ネームスペース iAnywhere.Data.UltraLite 76
CommandType プロパティ
ネームスペース iAnywhere.Data.UltraLite 77
Command プロパティ
ネームスペース iAnywhere.Data.UltraLite 294, 297
Commit メソッド
ネームスペース iAnywhere.Data.UltraLite 429
ネームスペース iAnywhere.UltraLite 448
commit メソッド
Ultra Light.NET 54
ConnectionName プロパティ
ネームスペース iAnywhere.Data.UltraLite 133
ネームスペース iAnywhere.UltraLite 462
ConnectionOpenParms プロパティ
ネームスペース iAnywhere.UltraLite 441
ConnectionParms クラス
ネームスペース iAnywhere.UltraLite 459
ConnectionParms コンストラクタ
ネームスペース iAnywhere.UltraLite 461
ConnectionString プロパティ
ネームスペース iAnywhere.Data.UltraLite 96
ConnectionTimeout プロパティ
ネームスペース iAnywhere.Data.UltraLite 97
Connection クラス
ネームスペース iAnywhere.UltraLite 437
Ultra Light.NET 32
Connection プロパティ
ネームスペース iAnywhere.Data.UltraLite 78, 428
Contains メソッド
ネームスペース iAnywhere.Data.UltraLite 279
ContinueUpdateOnError プロパティ
ネームスペース iAnywhere.Data.UltraLite 155
CopyFrom メソッド
ネームスペース iAnywhere.Data.UltraLite 347
ネームスペース iAnywhere.UltraLite 638
CopyTo メソッド
ネームスペース iAnywhere.Data.UltraLite 280
CountUploadRows メソッド
ネームスペース iAnywhere.Data.UltraLite 106, 107
ネームスペース iAnywhere.UltraLite 448, 449
Count プロパティ
ネームスペース iAnywhere.Data.UltraLite 270
CreateCommand メソッド
ネームスペース iAnywhere.Data.UltraLite 108
CreateDatabase メソッド
ネームスペース iAnywhere.UltraLite 508
CreateParameter メソッド
ネームスペース iAnywhere.Data.UltraLite 81
CreateParms クラス
ネームスペース iAnywhere.UltraLite 465
CreateParms コンストラクタ
ネームスペース iAnywhere.UltraLite 467
CursorSchema クラス
ネームスペース iAnywhere.UltraLite 498
Cursor クラス
ネームスペース iAnywhere.UltraLite 469

D

DatabaseID プロパティ
ネームスペース iAnywhere.Data.UltraLite 98
ネームスペース iAnywhere.UltraLite 441
DatabaseManager クラス
ネームスペース iAnywhere.UltraLite 505
Ultra Light.NET 32
DatabaseManager コンストラクタ
ネームスペース iAnywhere.UltraLite 506, 507
DatabaseManager プロパティ

ネームスペース iAnywhere.Data.UltraLite 98
DatabaseNameParms.UnusedEventHandler 委譲
ネームスペース iAnywhere.UltraLite 523
DatabaseNameParms クラス
ネームスペース iAnywhere.UltraLite 514
DatabaseNameParms コンストラクタ
ネームスペース iAnywhere.UltraLite 516
DatabaseOnCE プロパティ
ネームスペース iAnywhere.Data.UltraLite 133
ネームスペース iAnywhere.UltraLite 520
DatabaseOnDesktop プロパティ
ネームスペース iAnywhere.Data.UltraLite 134
ネームスペース iAnywhere.UltraLite 520
DatabaseSchema クラス
ネームスペース iAnywhere.UltraLite 524
iAnywhere.UltraLite.Net ネームスペース 55
Database プロパティ
ネームスペース iAnywhere.Data.UltraLite 97
DbType プロパティ
ネームスペース iAnywhere.Data.UltraLite 261
DeleteAllRows メソッド
ネームスペース iAnywhere.Data.UltraLite 380
ネームスペース iAnywhere.UltraLite 681
DeleteCommand プロパティ
ネームスペース iAnywhere.Data.UltraLite 156
Delete メソッド
ネームスペース iAnywhere.Data.UltraLite 379
ネームスペース iAnywhere.UltraLite 681
Depth プロパティ
ネームスペース iAnywhere.Data.UltraLite 194
Direction プロパティ
ネームスペース iAnywhere.Data.UltraLite 261
DisableConcurrency プロパティ
ネームスペース iAnywhere.Data.UltraLite 337
ネームスペース iAnywhere.UltraLite 629
Dispose メソッド
ネームスペース iAnywhere.Data.UltraLite 160, 200, 430
ネームスペース iAnywhere.UltraLite 521, 652
DML
Ultra Light.NET 38
DownloadOnly プロパティ
ネームスペース iAnywhere.Data.UltraLite 338
ネームスペース iAnywhere.UltraLite 630
DropDatabase メソッド
ネームスペース iAnywhere.Data.UltraLite 175
ネームスペース iAnywhere.UltraLite 509

E

EncryptionKey プロパティ
ネームスペース iAnywhere.Data.UltraLite 134
ネームスペース iAnywhere.UltraLite 462
ErrorCode プロパティ
ネームスペース iAnywhere.UltraLite 608
ErrorMessage プロパティ
ネームスペース iAnywhere.Data.UltraLite 349
ネームスペース iAnywhere.UltraLite 641
ExecuteNonQuery メソッド
ネームスペース iAnywhere.Data.UltraLite 82
ExecuteQuery メソッド
ネームスペース iAnywhere.UltraLite 551
ExecuteReader メソッド
ネームスペース iAnywhere.Data.UltraLite 83, 84
ExecuteScalar メソッド
ネームスペース iAnywhere.Data.UltraLite 85
ExecuteStatement メソッド
ネームスペース iAnywhere.UltraLite 551
ExecuteTable メソッド
ネームスペース iAnywhere.Data.UltraLite 86, 87, 108, 110, 111

F

FieldCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 195
FillError イベント
ネームスペース iAnywhere.Data.UltraLite 171
FillSchema メソッド
ネームスペース iAnywhere.Data.UltraLite 165, 166, 167
Fill メソッド

ネームスペース iAnywhere.Data.UltraLite 160, 162, 163, 164
ネームスペース iAnywhere.UltraLite 473
Finalize メソッド
ネームスペース iAnywhere.Data.UltraLite 142, 201, 367, 380
ネームスペース iAnywhere.UltraLite 474, 500, 552, 578, 668, 682
FinalProgressCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 301
ネームスペース iAnywhere.UltraLite 587
FindBegin メソッド
ネームスペース iAnywhere.Data.UltraLite 381
ネームスペース iAnywhere.UltraLite 682
FindFirst メソッド
ネームスペース iAnywhere.Data.UltraLite 381, 382
ネームスペース iAnywhere.UltraLite 682, 683
FindLast メソッド
ネームスペース iAnywhere.Data.UltraLite 383, 384
ネームスペース iAnywhere.UltraLite 684, 685
FindNext メソッド
ネームスペース iAnywhere.Data.UltraLite 385
ネームスペース iAnywhere.UltraLite 686, 687
FindPrevious メソッド
ネームスペース iAnywhere.Data.UltraLite 386, 387
ネームスペース iAnywhere.UltraLite 688
find メソッド
Ultra Light.NET 50

G

GetBoolean メソッド
ネームスペース iAnywhere.Data.UltraLite 201
ネームスペース iAnywhere.UltraLite 474
GetBytes メソッド
ネームスペース iAnywhere.Data.UltraLite 202, 204
ネームスペース iAnywhere.UltraLite 475, 476
GetByte メソッド
ネームスペース iAnywhere.Data.UltraLite 202
GetChars メソッド
ネームスペース iAnywhere.Data.UltraLite 206
ネームスペース iAnywhere.UltraLite 478
GetChar メソッド
ネームスペース iAnywhere.Data.UltraLite 205
GetColumnDefaultValue メソッド
ネームスペース iAnywhere.Data.UltraLite 418
ネームスペース iAnywhere.UltraLite 724
GetColumnID メソッド
ネームスペース iAnywhere.Data.UltraLite 143
ネームスペース iAnywhere.UltraLite 500
GetColumnName メソッド
ネームスペース iAnywhere.Data.UltraLite 144, 245
ネームスペース iAnywhere.UltraLite 501, 541
GetColumnPartitionSize メソッド
ネームスペース iAnywhere.Data.UltraLite 418
ネームスペース iAnywhere.UltraLite 724
GetColumnPrecision メソッド
ネームスペース iAnywhere.Data.UltraLite 145
ネームスペース iAnywhere.UltraLite 502
GetColumnScale メソッド
ネームスペース iAnywhere.Data.UltraLite 145
ネームスペース iAnywhere.UltraLite 503
GetColumnSize メソッド
ネームスペース iAnywhere.Data.UltraLite 146
ネームスペース iAnywhere.UltraLite 503
GetColumnSQLType メソッド
ネームスペース iAnywhere.UltraLite 504
GetColumnULDbType メソッド
ネームスペース iAnywhere.Data.UltraLite 147
GetDatabaseProperty メソッド
ネームスペース iAnywhere.Data.UltraLite 185
ネームスペース iAnywhere.UltraLite 529
GetDataTypeName メソッド
ネームスペース iAnywhere.Data.UltraLite 208
GetData メソッド
ネームスペース iAnywhere.Data.UltraLite 207
GetDateTime メソッド
ネームスペース iAnywhere.Data.UltraLite 209
GetDecimal メソッド

ネームスペース iAnywhere.Data.UltraLite 209
GetDouble メソッド
ネームスペース iAnywhere.Data.UltraLite 210
ネームスペース iAnywhere.UltraLite 480
GetEnumerator メソッド
ネームスペース iAnywhere.Data.UltraLite 281
GetErrorMessage メソッド
ネームスペース iAnywhere.UltraLite 653
GetFieldType メソッド
ネームスペース iAnywhere.Data.UltraLite 211
GetFillParameters メソッド
ネームスペース iAnywhere.Data.UltraLite 168
GetFloat メソッド
ネームスペース iAnywhere.Data.UltraLite 212
ネームスペース iAnywhere.UltraLite 481
GetGuid メソッド
ネームスペース iAnywhere.Data.UltraLite 212
GetIndexName メソッド
ネームスペース iAnywhere.Data.UltraLite 420
ネームスペース iAnywhere.UltraLite 725
GetIndex メソッド
ネームスペース iAnywhere.Data.UltraLite 419
ネームスペース iAnywhere.UltraLite 725
GetInt16 メソッド
ネームスペース iAnywhere.Data.UltraLite 213
GetInt32 メソッド
ネームスペース iAnywhere.Data.UltraLite 214
GetInt64 メソッド
ネームスペース iAnywhere.Data.UltraLite 215
GetInt メソッド
ネームスペース iAnywhere.UltraLite 482
GetLastDownloadTime メソッド
ネームスペース iAnywhere.Data.UltraLite 113
ネームスペース iAnywhere.UltraLite 450
GetLong メソッド
ネームスペース iAnywhere.UltraLite 483
GetMessage メソッド
ネームスペース iAnywhere.UltraLite 654
GetName メソッド
ネームスペース iAnywhere.Data.UltraLite 215
GetNewUUID メソッド
ネームスペース iAnywhere.Data.UltraLite 114
ネームスペース iAnywhere.UltraLite 451
GetObjectData メソッド
ネームスペース iAnywhere.Data.UltraLite 238
ネームスペース iAnywhere.UltraLite 608
GetOptimalIndex メソッド
ネームスペース iAnywhere.Data.UltraLite 420
ネームスペース iAnywhere.UltraLite 726
GetOrdinal メソッド
ネームスペース iAnywhere.Data.UltraLite 216
GetPublicationName メソッド
ネームスペース iAnywhere.Data.UltraLite 186
ネームスペース iAnywhere.UltraLite 531
GetPublicationSchema メソッド
ネームスペース iAnywhere.Data.UltraLite 187
ネームスペース iAnywhere.UltraLite 532
GetSchemaTable メソッド
ネームスペース iAnywhere.Data.UltraLite 147, 218
GetShort メソッド
ネームスペース iAnywhere.UltraLite 484
GetString メソッド
ネームスペース iAnywhere.Data.UltraLite 220
ネームスペース iAnywhere.UltraLite 485, 654
GetTableCountInPublications メソッド
ネームスペース iAnywhere.Data.UltraLite 188
ネームスペース iAnywhere.UltraLite 532
GetTableName メソッド
ネームスペース iAnywhere.Data.UltraLite 188
ネームスペース iAnywhere.UltraLite 533
GetTable メソッド
ネームスペース iAnywhere.UltraLite 451
GetTimeSpan メソッド
ネームスペース iAnywhere.Data.UltraLite 221
GetTimestamp メソッド
ネームスペース iAnywhere.UltraLite 488
GetTime メソッド
ネームスペース iAnywhere.UltraLite 486
GetUInt16 メソッド
ネームスペース iAnywhere.Data.UltraLite 222
GetUInt32 メソッド
ネームスペース iAnywhere.Data.UltraLite 222
GetUInt64 メソッド

ネームスペース iAnywhere.Data.UltraLite 223
GetUInt メソッド
ネームスペース iAnywhere.UltraLite 489
GetULong メソッド
ネームスペース iAnywhere.UltraLite 490
GetUShort メソッド
ネームスペース iAnywhere.UltraLite 491
GetUUID メソッド
ネームスペース iAnywhere.UltraLite 492
GetValues メソッド
ネームスペース iAnywhere.Data.UltraLite 224
GetValue メソッド
ネームスペース iAnywhere.Data.UltraLite 224
GlobalAutoIncrementUsage プロパティ
ネームスペース iAnywhere.Data.UltraLite 99
ネームスペース iAnywhere.UltraLite 442
GrantConnectTo メソッド
ネームスペース iAnywhere.Data.UltraLite 114
ネームスペース iAnywhere.UltraLite 452
grantConnectTo メソッド
Ultra Light.NET 開発 58

H

HasResultSet プロパティ
ネームスペース iAnywhere.UltraLite 546

I

iAnywhere.Data.UltraLite ネームスペース
ネームスペース iAnywhere.Data.UltraLite 63
説明 2
iAnywhere.UltraLite ネームスペース
ネームスペース iAnywhere.UltraLite 431
説明 2
IgnoredRows プロパティ
ネームスペース iAnywhere.Data.UltraLite 364
ネームスペース iAnywhere.UltraLite 666
IndexCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 416
ネームスペース iAnywhere.UltraLite 721
IndexName プロパティ
ネームスペース iAnywhere.Data.UltraLite 78
IndexOf メソッド
ネームスペース iAnywhere.Data.UltraLite 281, 282
IndexSchema クラス
ネームスペース iAnywhere.UltraLite 535
Ultra Light.NET 開発 55
InfoMessage イベント
ネームスペース iAnywhere.Data.UltraLite 121
InsertBegin メソッド
ネームスペース iAnywhere.Data.UltraLite 388
ネームスペース iAnywhere.UltraLite 690
InsertCommand プロパティ
ネームスペース iAnywhere.Data.UltraLite 157
Insert メソッド
ネームスペース iAnywhere.Data.UltraLite 282, 388
ネームスペース iAnywhere.UltraLite 689
INVALID_DATABASE_ID フィールド
ネームスペース iAnywhere.Data.UltraLite 95
ネームスペース iAnywhere.UltraLite 440
IsBOF プロパティ
ネームスペース iAnywhere.Data.UltraLite 195
ネームスペース iAnywhere.UltraLite 471
IsCaseSensitive プロパティ
ネームスペース iAnywhere.Data.UltraLite 181
ネームスペース iAnywhere.UltraLite 526
IsClosed プロパティ
ネームスペース iAnywhere.Data.UltraLite 196
IsColumnAutoIncrement メソッド
ネームスペース iAnywhere.Data.UltraLite 421
ネームスペース iAnywhere.UltraLite 727
IsColumnCurrentDate メソッド
ネームスペース iAnywhere.Data.UltraLite 422
ネームスペース iAnywhere.UltraLite 727
IsColumnCurrentTimestamp メソッド
ネームスペース iAnywhere.Data.UltraLite 423
ネームスペース iAnywhere.UltraLite 728
IsColumnCurrentTime メソッド
ネームスペース iAnywhere.Data.UltraLite 422
ネームスペース iAnywhere.UltraLite 728

IsColumnDescending メソッド
ネームスペース iAnywhere.Data.UltraLite 246
ネームスペース iAnywhere.UltraLite 541
IsColumnGlobalAutoIncrement メソッド
ネームスペース iAnywhere.Data.UltraLite 424
ネームスペース iAnywhere.UltraLite 729
IsColumnNewUUID メソッド
ネームスペース iAnywhere.Data.UltraLite 424
ネームスペース iAnywhere.UltraLite 730
IsColumnNullable メソッド
ネームスペース iAnywhere.Data.UltraLite 425
ネームスペース iAnywhere.UltraLite 730
IsDatabaseNew プロパティ
ネームスペース iAnywhere.UltraLite 442
IsDBNull メソッド
ネームスペース iAnywhere.Data.UltraLite 226
IsEOF プロパティ
ネームスペース iAnywhere.Data.UltraLite 196
ネームスペース iAnywhere.UltraLite 472
IsForeignKeyCheckOnCommit プロパティ
ネームスペース iAnywhere.Data.UltraLite 241
ネームスペース iAnywhere.UltraLite 537
IsForeignKeyNullable プロパティ
ネームスペース iAnywhere.Data.UltraLite 242
ネームスペース iAnywhere.UltraLite 538
IsForeignKey プロパティ
ネームスペース iAnywhere.Data.UltraLite 241
ネームスペース iAnywhere.UltraLite 537
IsInPublication メソッド
ネームスペース iAnywhere.Data.UltraLite 426
ネームスペース iAnywhere.UltraLite 731
IsLastCodeOK プロパティ
ネームスペース iAnywhere.UltraLite 443
IsLastFetchOK プロパティ
ネームスペース iAnywhere.UltraLite 443
IsNeverSynchronized プロパティ
ネームスペース iAnywhere.Data.UltraLite 416
ネームスペース iAnywhere.UltraLite 722
IsNullable プロパティ
ネームスペース iAnywhere.Data.UltraLite 262
IsNull メソッド
ネームスペース iAnywhere.UltraLite 493
IsolationLevel プロパティ
ネームスペース iAnywhere.Data.UltraLite 428
IsOpen プロパティ
ネームスペース iAnywhere.Data.UltraLite 142, 181, 242, 288
ネームスペース iAnywhere.UltraLite 444, 472, 499, 526, 538, 546, 572
IsPrimaryKey プロパティ
ネームスペース iAnywhere.Data.UltraLite 243
ネームスペース iAnywhere.UltraLite 538
IsUniqueIndex プロパティ
ネームスペース iAnywhere.Data.UltraLite 243
ネームスペース iAnywhere.UltraLite 539
IsUniqueKey プロパティ
ネームスペース iAnywhere.Data.UltraLite 243
ネームスペース iAnywhere.UltraLite 539
Item プロパティ
ネームスペース iAnywhere.Data.UltraLite 196, 197, 270, 271

K

KeepPartialDownload プロパティ
ネームスペース iAnywhere.Data.UltraLite 338
ネームスペース iAnywhere.UltraLite 630

L

LastIdentity プロパティ
ネームスペース iAnywhere.Data.UltraLite 99
ネームスペース iAnywhere.UltraLite 444
LastSQLCode プロパティ
ネームスペース iAnywhere.UltraLite 445
LookupBackward メソッド
ネームスペース iAnywhere.Data.UltraLite 389, 390
ネームスペース iAnywhere.UltraLite 690, 691
LookupBegin メソッド
ネームスペース iAnywhere.Data.UltraLite 391
ネームスペース iAnywhere.UltraLite 692
LookupForward メソッド

ネームスペース iAnywhere.Data.UltraLite 391, 392
ネームスペース iAnywhere.UltraLite 693
lookup メソッド
Ultra Light.NET 50

M

Mask プロパティ
ネームスペース iAnywhere.Data.UltraLite 288
ネームスペース iAnywhere.UltraLite 572
Message プロパティ
ネームスペース iAnywhere.Data.UltraLite 248
MissingMappingAction プロパティ
ネームスペース iAnywhere.Data.UltraLite 157
MissingSchemaAction プロパティ
ネームスペース iAnywhere.Data.UltraLite 158
MoveAfterLast メソッド
ネームスペース iAnywhere.Data.UltraLite 226
ネームスペース iAnywhere.UltraLite 494
MoveBeforeFirst メソッド
ネームスペース iAnywhere.Data.UltraLite 227
ネームスペース iAnywhere.UltraLite 494
MoveFirst メソッド
ネームスペース iAnywhere.Data.UltraLite 227
ネームスペース iAnywhere.UltraLite 495
moveFirst メソッド
Ultra Light.NET 開発 43
moveFirst メソッド (Table クラス)
Ultra Light.NET 開発 46
MoveLast メソッド
ネームスペース iAnywhere.Data.UltraLite 227
ネームスペース iAnywhere.UltraLite 495
MoveNext メソッド
ネームスペース iAnywhere.Data.UltraLite 228
ネームスペース iAnywhere.UltraLite 495
moveNext メソッド
Ultra Light.NET 開発 43
moveNext メソッド (Table クラス)
Ultra Light.NET 開発 46
MovePrevious メソッド
ネームスペース iAnywhere.Data.UltraLite 228
ネームスペース iAnywhere.UltraLite 496
MoveRelative メソッド
ネームスペース iAnywhere.Data.UltraLite 229
ネームスペース iAnywhere.UltraLite 496

N

Name プロパティ
ネームスペース iAnywhere.Data.UltraLite 142, 244, 289, 291, 417
ネームスペース iAnywhere.UltraLite 500, 540, 573, 580, 722
NativeError プロパティ
ネームスペース iAnywhere.Data.UltraLite 237, 249
NewPassword プロパティ
ネームスペース iAnywhere.Data.UltraLite 340
ネームスペース iAnywhere.UltraLite 632
NextResult メソッド
ネームスペース iAnywhere.Data.UltraLite 229

O

Offset プロパティ
ネームスペース iAnywhere.Data.UltraLite 262
OnClosing メソッド
ネームスペース iAnywhere.UltraLite 655
OpenConnection メソッド
ネームスペース iAnywhere.UltraLite 510
OpenWithCreate メソッド
ネームスペース iAnywhere.Data.UltraLite 116
Open メソッド
ネームスペース iAnywhere.Data.UltraLite 115
ネームスペース iAnywhere.UltraLite 694, 695

P

ParameterName プロパティ
ネームスペース iAnywhere.Data.UltraLite 263

Parameters プロパティ
ネームスペース iAnywhere.Data.UltraLite 79
ParmsUsed プロパティ
ネームスペース iAnywhere.UltraLite 520, 585
PartialDownloadRetained プロパティ
ネームスペース iAnywhere.Data.UltraLite 364
ネームスペース iAnywhere.UltraLite 666
Password プロパティ
ネームスペース iAnywhere.Data.UltraLite 135, 340
ネームスペース iAnywhere.UltraLite 463, 632
PingOnly プロパティ
ネームスペース iAnywhere.Data.UltraLite 341
ネームスペース iAnywhere.UltraLite 633
Plan プロパティ
ネームスペース iAnywhere.Data.UltraLite 79
ネームスペース iAnywhere.UltraLite 546
Precision プロパティ
ネームスペース iAnywhere.Data.UltraLite 263
PreparedStatement クラス
ネームスペース iAnywhere.UltraLite 543
iAnywhere.UltraLite.NET ネームスペース 38
PrepareStatement メソッド
ネームスペース iAnywhere.UltraLite 453
Prepare メソッド
ネームスペース iAnywhere.Data.UltraLite 88
PrimaryKey プロパティ
ネームスペース iAnywhere.Data.UltraLite 417
ネームスペース iAnywhere.UltraLite 723
ProgressCounter プロパティ
ネームスペース iAnywhere.Data.UltraLite 302
ネームスペース iAnywhere.UltraLite 588
PublicationCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 182
ネームスペース iAnywhere.UltraLite 526
PublicationMask プロパティ
ネームスペース iAnywhere.Data.UltraLite 341
ネームスペース iAnywhere.UltraLite 633
PublicationSchema クラス
ネームスペース iAnywhere.UltraLite 570
Ultra Light.NET 開発 56

R

Read メソッド
ネームスペース iAnywhere.Data.UltraLite 230
ReceivedBytes プロパティ
ネームスペース iAnywhere.Data.UltraLite 350
ネームスペース iAnywhere.UltraLite 642
ReceivedDeletes プロパティ
ネームスペース iAnywhere.Data.UltraLite 350
ネームスペース iAnywhere.UltraLite 642
ReceivedInserts プロパティ
ネームスペース iAnywhere.Data.UltraLite 350
ネームスペース iAnywhere.UltraLite 642
ReceivedUpdates プロパティ
ネームスペース iAnywhere.Data.UltraLite 351
ネームスペース iAnywhere.UltraLite 643
RecordsAffected プロパティ
ネームスペース iAnywhere.Data.UltraLite 198, 294
ReferencedIndexName プロパティ
ネームスペース iAnywhere.Data.UltraLite 244
ネームスペース iAnywhere.UltraLite 540
ReferencedTableName プロパティ
ネームスペース iAnywhere.Data.UltraLite 245
ネームスペース iAnywhere.UltraLite 540
RemoveAt メソッド
ネームスペース iAnywhere.Data.UltraLite 284, 285
Remove メソッド
ネームスペース iAnywhere.Data.UltraLite 283
ResetLastDownloadTime メソッド
ネームスペース iAnywhere.Data.UltraLite 117
ネームスペース iAnywhere.UltraLite 453
ResetValues メソッド
ネームスペース iAnywhere.UltraLite 655
ResultSetSchema クラス
ネームスペース iAnywhere.UltraLite 579
ResultSetSchema プロパティ
ネームスペース iAnywhere.UltraLite 547
ResultSet クラス
ネームスペース iAnywhere.UltraLite 574
ResumePartialDownload プロパティ
ネームスペース iAnywhere.Data.UltraLite 342

ネームスペース iAnywhere.UltraLite 634
RevokeConnectFrom メソッド
ネームスペース iAnywhere.Data.UltraLite 117
ネームスペース iAnywhere.UltraLite 454
revokeConnectionFrom メソッド
Ultra Light.NET 開発 58
RollbackPartialDownload メソッド
ネームスペース iAnywhere.Data.UltraLite 118
ネームスペース iAnywhere.UltraLite 455
Rollback メソッド
ネームスペース iAnywhere.Data.UltraLite 430
ネームスペース iAnywhere.UltraLite 454
rollback メソッド
Ultra Light.NET 54
RowCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 199
ネームスペース iAnywhere.UltraLite 472
RowUpdated イベント
ネームスペース iAnywhere.Data.UltraLite 172
RowUpdating イベント
ネームスペース iAnywhere.Data.UltraLite 172
RuntimeType プロパティ
ネームスペース iAnywhere.Data.UltraLite 174
ネームスペース iAnywhere.UltraLite 507
RuntimeType 列挙
ネームスペース iAnywhere.UltraLite 581

S

Scale プロパティ
ネームスペース iAnywhere.Data.UltraLite 264
SchemaOnCE プロパティ
ネームスペース iAnywhere.Data.UltraLite 136
ネームスペース iAnywhere.UltraLite 585
SchemaOnDesktop プロパティ
ネームスペース iAnywhere.Data.UltraLite 136
ネームスペース iAnywhere.UltraLite 585
SchemaParms クラス
ネームスペース iAnywhere.UltraLite 582
SchemaParms コンストラクタ
ネームスペース iAnywhere.UltraLite 583, 584
SchemaUpgradeData クラス
ネームスペース iAnywhere.UltraLite 587
SchemaUpgradeListener インタフェース
ネームスペース iAnywhere.UltraLite 590
SchemaUpgradeState 列挙
ネームスペース iAnywhere.UltraLite 592
SchemaUpgrading メソッド
ネームスペース iAnywhere.Data.UltraLite 304
ネームスペース iAnywhere.UltraLite 590
Schema プロパティ
ネームスペース iAnywhere.Data.UltraLite 100, 199, 376
ネームスペース iAnywhere.UltraLite 445, 467, 577, 676
SelectCommand プロパティ
ネームスペース iAnywhere.Data.UltraLite 158
SELECT 文
Ultra Light.NET 開発 43
SendColumnNames プロパティ
ネームスペース iAnywhere.Data.UltraLite 343
ネームスペース iAnywhere.UltraLite 634
SendDownloadAck プロパティ
ネームスペース iAnywhere.Data.UltraLite 343
ネームスペース iAnywhere.UltraLite 635
SentBytes プロパティ
ネームスペース iAnywhere.Data.UltraLite 351
ネームスペース iAnywhere.UltraLite 643
SentDeletes プロパティ
ネームスペース iAnywhere.Data.UltraLite 352
ネームスペース iAnywhere.UltraLite 644
SentInserts プロパティ
ネームスペース iAnywhere.Data.UltraLite 352
ネームスペース iAnywhere.UltraLite 644
SentUpdates プロパティ
ネームスペース iAnywhere.Data.UltraLite 353
ネームスペース iAnywhere.UltraLite 644
ServerSyncInvoked メソッド
ネームスペース iAnywhere.Data.UltraLite 307
ネームスペース iAnywhere.UltraLite 594
ServerSyncListener インタフェース
ネームスペース iAnywhere.UltraLite 594
SetActiveSyncListener メソッド

ネームスペース iAnywhere.Data.UltraLite 175
ネームスペース iAnywhere.UltraLite 511
SetBoolean メソッド
ネームスペース iAnywhere.Data.UltraLite 393
ネームスペース iAnywhere.UltraLite 695
SetBoolParameter メソッド
ネームスペース iAnywhere.UltraLite 552
SetBytesParameter メソッド
ネームスペース iAnywhere.UltraLite 553
SetBytes メソッド
ネームスペース iAnywhere.Data.UltraLite 395
ネームスペース iAnywhere.UltraLite 696
SetByte メソッド
ネームスペース iAnywhere.Data.UltraLite 394
SetDateTimeParameter メソッド
ネームスペース iAnywhere.UltraLite 555
SetDateTime メソッド
ネームスペース iAnywhere.Data.UltraLite 396
SetDBNull メソッド
ネームスペース iAnywhere.Data.UltraLite 397
SetDecimal メソッド
ネームスペース iAnywhere.Data.UltraLite 398
SetDoubleParameter メソッド
ネームスペース iAnywhere.UltraLite 556
SetDouble メソッド
ネームスペース iAnywhere.Data.UltraLite 399
ネームスペース iAnywhere.UltraLite 698
SetFloatParameter メソッド
ネームスペース iAnywhere.UltraLite 557
SetFloat メソッド
ネームスペース iAnywhere.Data.UltraLite 400
ネームスペース iAnywhere.UltraLite 699
SetGuid メソッド
ネームスペース iAnywhere.Data.UltraLite 401
SetInt16 メソッド
ネームスペース iAnywhere.Data.UltraLite 402
SetInt32 メソッド
ネームスペース iAnywhere.Data.UltraLite 403
SetInt64 メソッド
ネームスペース iAnywhere.Data.UltraLite 404
SetIntParameter メソッド
ネームスペース iAnywhere.UltraLite 558
SetInt メソッド
ネームスペース iAnywhere.UltraLite 701
SetLongParameter メソッド
ネームスペース iAnywhere.UltraLite 559
SetLong メソッド
ネームスペース iAnywhere.UltraLite 702
SetNullParameter メソッド
ネームスペース iAnywhere.UltraLite 560
SetNull メソッド
ネームスペース iAnywhere.UltraLite 703
SetServerSyncListener メソッド
ネームスペース iAnywhere.Data.UltraLite 176
ネームスペース iAnywhere.UltraLite 512
SetShortParameter メソッド
ネームスペース iAnywhere.UltraLite 561
SetShort メソッド
ネームスペース iAnywhere.UltraLite 705
SetStringParameter メソッド
ネームスペース iAnywhere.UltraLite 562
SetString メソッド
ネームスペース iAnywhere.Data.UltraLite 405
ネームスペース iAnywhere.UltraLite 706
SetTimeSpanParameter メソッド
ネームスペース iAnywhere.UltraLite 563
SetTimeSpan メソッド
ネームスペース iAnywhere.Data.UltraLite 406
SetTimestamp メソッド
ネームスペース iAnywhere.UltraLite 709
SetTime メソッド
ネームスペース iAnywhere.UltraLite 707
SetToDefault メソッド
ネームスペース iAnywhere.Data.UltraLite 407
ネームスペース iAnywhere.UltraLite 710
SetUInt16 メソッド
ネームスペース iAnywhere.Data.UltraLite 407
SetUInt32 メソッド
ネームスペース iAnywhere.Data.UltraLite 408
SetUInt64 メソッド
ネームスペース iAnywhere.Data.UltraLite 409
SetUIntParameter メソッド
ネームスペース iAnywhere.UltraLite 564
SetUInt メソッド

ネームスペース iAnywhere.UltraLite 711
SetULongParameter メソッド
ネームスペース iAnywhere.UltraLite 565
SetULong メソッド
ネームスペース iAnywhere.UltraLite 713
SetUShortParameter メソッド
ネームスペース iAnywhere.UltraLite 566
SetUShort メソッド
ネームスペース iAnywhere.UltraLite 714
SetUUIDParameter メソッド
ネームスペース iAnywhere.UltraLite 568
SetUUID メソッド
ネームスペース iAnywhere.UltraLite 716
ShowDialog メソッド
ネームスペース iAnywhere.UltraLite 655, 656
Signature プロパティ
ネームスペース iAnywhere.Data.UltraLite 182
ネームスペース iAnywhere.UltraLite 527
Size プロパティ
ネームスペース iAnywhere.Data.UltraLite 264
SourceColumn プロパティ
ネームスペース iAnywhere.Data.UltraLite 265
SourceVersion プロパティ
ネームスペース iAnywhere.Data.UltraLite 265
Source プロパティ
ネームスペース iAnywhere.Data.UltraLite 238, 249
SQL Anywhere Studio
マニュアル viii
SQLCode プロパティ
ネームスペース iAnywhere.Data.UltraLite 353
ネームスペース iAnywhere.UltraLite 645, 652
SQLCode 列挙
ネームスペース iAnywhere.UltraLite 598
SQLException クラス
ネームスペース iAnywhere.UltraLite 606
SQLException コンストラクタ
ネームスペース iAnywhere.UltraLite 607
SQLType 列挙
ネームスペース iAnywhere.UltraLite 610
StartSynchronizationDelete メソッド
ネームスペース iAnywhere.Data.UltraLite 119
ネームスペース iAnywhere.UltraLite 455
StateChange イベント
ネームスペース iAnywhere.Data.UltraLite 122
State プロパティ
ネームスペース iAnywhere.Data.UltraLite 100, 302, 354
ネームスペース iAnywhere.UltraLite 588, 645
StopSynchronizationDelete メソッド
ネームスペース iAnywhere.Data.UltraLite 119
ネームスペース iAnywhere.UltraLite 456
StreamErrorCode プロパティ
ネームスペース iAnywhere.Data.UltraLite 365
ネームスペース iAnywhere.UltraLite 667
StreamErrorCode 列挙
ネームスペース iAnywhere.UltraLite 614
StreamErrorContext プロパティ
ネームスペース iAnywhere.Data.UltraLite 365
ネームスペース iAnywhere.UltraLite 667
StreamErrorContext 列挙
ネームスペース iAnywhere.UltraLite 620
StreamErrorID プロパティ
ネームスペース iAnywhere.Data.UltraLite 366
ネームスペース iAnywhere.UltraLite 667
StreamErrorID 列挙
ネームスペース iAnywhere.UltraLite 622
StreamErrorSystem プロパティ
ネームスペース iAnywhere.Data.UltraLite 366
ネームスペース iAnywhere.UltraLite 668
StreamParms プロパティ
ネームスペース iAnywhere.Data.UltraLite 344
ネームスペース iAnywhere.UltraLite 636
StreamType 列挙
ネームスペース iAnywhere.UltraLite 624
Stream プロパティ
ネームスペース iAnywhere.Data.UltraLite 344
ネームスペース iAnywhere.UltraLite 635
SYNC_ALL_DB フィールド
ネームスペース iAnywhere.Data.UltraLite 287
ネームスペース iAnywhere.UltraLite 571
SYNC_ALL_PUBS フィールド
ネームスペース iAnywhere.Data.UltraLite 287
ネームスペース iAnywhere.UltraLite 571

Synchronize メソッド
ネームスペース iAnywhere.Data.UltraLite 120
ネームスペース iAnywhere.UltraLite 456, 457
SyncParms クラス
ネームスペース iAnywhere.UltraLite 626
SyncParms プロパティ
ネームスペース iAnywhere.Data.UltraLite 101, 354
ネームスペース iAnywhere.UltraLite 446, 645
SyncProgressData クラス
ネームスペース iAnywhere.UltraLite 640
SyncProgressDialog クラス
ネームスペース iAnywhere.UltraLite 649
SyncProgressDialog コンストラクタ
ネームスペース iAnywhere.UltraLite 651
SyncProgressed メソッド
ネームスペース iAnywhere.Data.UltraLite 357
ネームスペース iAnywhere.UltraLite 657, 659
SyncProgressListener インタフェース
ネームスペース iAnywhere.UltraLite 659
SyncProgressState 列挙
ネームスペース iAnywhere.UltraLite 661
SyncResult クラス
ネームスペース iAnywhere.UltraLite 664
SyncResult プロパティ
ネームスペース iAnywhere.Data.UltraLite 101, 355
ネームスペース iAnywhere.UltraLite 446, 646

T

TableCount プロパティ
ネームスペース iAnywhere.Data.UltraLite 183, 355
ネームスペース iAnywhere.UltraLite 527, 646
TableIndex プロパティ
ネームスペース iAnywhere.Data.UltraLite 356
ネームスペース iAnywhere.UltraLite 647
TableMappings プロパティ
ネームスペース iAnywhere.Data.UltraLite 159
TableSchema クラス
ネームスペース iAnywhere.UltraLite 719
Ultra Light.NET 開発 55
TableTotal プロパティ
ネームスペース iAnywhere.Data.UltraLite 356
ネームスペース iAnywhere.UltraLite 647
Table クラス
ネームスペース iAnywhere.UltraLite 670
Timestamp プロパティ
ネームスペース iAnywhere.Data.UltraLite 366
ネームスペース iAnywhere.UltraLite 668
ToString メソッド
ネームスペース iAnywhere.Data.UltraLite 137, 249, 267
ネームスペース iAnywhere.UltraLite 521, 586
Transaction プロパティ
ネームスペース iAnywhere.Data.UltraLite 80
Truncate メソッド
ネームスペース iAnywhere.Data.UltraLite 410
ネームスペース iAnywhere.UltraLite 717

U

ULActiveSyncListener インタフェース
ネームスペース iAnywhere.Data.UltraLite 64
ULAuthStatusCode 列挙
ネームスペース iAnywhere.Data.UltraLite 68
ULCommand クラス
ネームスペース iAnywhere.Data.UltraLite 69
iAnywhere.Data.UltraLite.NET ネームスペース 38
Ultra Light.NET 開発 43
ULCommand コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 72, 73, 74, 75
ULConnectionParms.UnusedEventHandler 委譲
ネームスペース iAnywhere.Data.UltraLite 139
ULConnectionParms クラス
ネームスペース iAnywhere.Data.UltraLite 124
ULConnectionParms コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 128
ULConnection クラス
ネームスペース iAnywhere.Data.UltraLite 89
ULConnection コンストラクタ

ネームスペース iAnywhere.Data.UltraLite 93, 94
ULCursorSchema クラス
ネームスペース iAnywhere.Data.UltraLite 140
ULDataAdapter クラス
ネームスペース iAnywhere.Data.UltraLite 149
ULDataAdapter コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 152, 153, 154
ULDatabaseManager クラス
ネームスペース iAnywhere.Data.UltraLite 173
ULDatabaseSchema クラス
ネームスペース iAnywhere.Data.UltraLite 179
iAnywhere.Data.UltraLite.NET ネームスペース 55
ULDataReader クラス
ネームスペース iAnywhere.Data.UltraLite 190
Ultra Light.NET 開発 43
ULDbType プロパティ
ネームスペース iAnywhere.Data.UltraLite 266
ULDbType 列挙
ネームスペース iAnywhere.Data.UltraLite 231
ULException クラス
ネームスペース iAnywhere.Data.UltraLite 235
ULException コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 236
ULIndexSchema クラス
ネームスペース iAnywhere.Data.UltraLite 239
ULInfoMessageEventArgs クラス
ネームスペース iAnywhere.Data.UltraLite 248
ULInfoMessageEventHandler 委譲
ネームスペース iAnywhere.Data.UltraLite 251
ULParameterCollection クラス
ネームスペース iAnywhere.Data.UltraLite 268
ULParameter クラス
ネームスペース iAnywhere.Data.UltraLite 252
ULParameter コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 254, 255, 256, 257, 258, 259
ULPublicationSchema クラス
ネームスペース iAnywhere.Data.UltraLite 286
ULResultSetSchema クラス
ネームスペース iAnywhere.Data.UltraLite 290
ULRowUpdatedEventArgs クラス
ネームスペース iAnywhere.Data.UltraLite 292
ULRowUpdatedEventArgs コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 293
ULRowUpdatedEventHandler 委譲
ネームスペース iAnywhere.Data.UltraLite 295
ULRowUpdatingEventArgs クラス
ネームスペース iAnywhere.Data.UltraLite 296
ULRowUpdatingEventArgs コンストラクタ
ネームスペース iAnywhere.Data.UltraLite 297
ULRowUpdatingEventHandler 委譲
ネームスペース iAnywhere.Data.UltraLite 299
ULRuntimeType 列挙
ネームスペース iAnywhere.Data.UltraLite 300
ULSchemaUpgradeData クラス
ネームスペース iAnywhere.Data.UltraLite 301
ULSchemaUpgradeListener インタフェース
ネームスペース iAnywhere.Data.UltraLite 304
ULSchemaUpgradeState 列挙
ネームスペース iAnywhere.Data.UltraLite 306
ULServerSyncListener インタフェース
ネームスペース iAnywhere.Data.UltraLite 307
ULSQLCode 列挙
ネームスペース iAnywhere.Data.UltraLite 311
ULStreamErrorCode 列挙
ネームスペース iAnywhere.Data.UltraLite 321
ULStreamErrorContext 列挙
ネームスペース iAnywhere.Data.UltraLite 327
ULStreamErrorID 列挙
ネームスペース iAnywhere.Data.UltraLite 329
ULStreamType 列挙
ネームスペース iAnywhere.Data.UltraLite 331
ULSyncParms クラス
ネームスペース iAnywhere.Data.UltraLite 333
ULSyncProgressData クラス
ネームスペース iAnywhere.Data.UltraLite 348
ULSyncProgressListener インタフェース
ネームスペース iAnywhere.Data.UltraLite 357
ULSyncProgressState 列挙
ネームスペース iAnywhere.Data.UltraLite 359
ULSyncResult クラス

ネームスペース iAnywhere.Data.UltraLite 362
ULTableSchema クラス
ネームスペース iAnywhere.Data.UltraLite 413
ULTable クラス
ネームスペース iAnywhere.Data.UltraLite 368
Ultra Light.NET
アーキテクチャ 5
暗号化 37
開発 29
サポートされるプラットフォーム 3
スキーマ・ファイルの作成 30
説明 1
チュートリアル 7
データベース・スキーマのアップグレード 31
テーブル API を使用したデータ操作 46
動的 SQL チュートリアル 20
動的 SQL を使用したデータ操作 38
配置 27
利点 2
ULTransaction クラス
ネームスペース iAnywhere.Data.UltraLite 427
UnusedEvent イベント
ネームスペース iAnywhere.Data.UltraLite 138
ネームスペース iAnywhere.UltraLite 522
UpdateBegin メソッド
ネームスペース iAnywhere.Data.UltraLite 411
ネームスペース iAnywhere.UltraLite 718
UpdateCommand プロパティ
ネームスペース iAnywhere.Data.UltraLite 159
UpdatedRowSource プロパティ
ネームスペース iAnywhere.Data.UltraLite 80
Update メソッド
ネームスペース iAnywhere.Data.UltraLite 168, 169, 170, 171, 411
ネームスペース iAnywhere.UltraLite 717
UpgradeOperations プロパティ
ネームスペース iAnywhere.Data.UltraLite 303
ネームスペース iAnywhere.UltraLite 589
UploadOK プロパティ
ネームスペース iAnywhere.Data.UltraLite 367
ネームスペース iAnywhere.UltraLite 668
UploadOnly プロパティ
ネームスペース iAnywhere.Data.UltraLite 345
ネームスペース iAnywhere.UltraLite 637
UploadsUnchangedRows プロパティ
ネームスペース iAnywhere.UltraLite 723
UploadUnchangedRows プロパティ
ネームスペース iAnywhere.Data.UltraLite 417
UserID プロパティ
ネームスペース iAnywhere.Data.UltraLite 136
ネームスペース iAnywhere.UltraLite 464
UserName プロパティ
ネームスペース iAnywhere.Data.UltraLite 346
ネームスペース iAnywhere.UltraLite 637
usm ファイル
Ultra Light.NET 30

V

Value プロパティ
ネームスペース iAnywhere.Data.UltraLite 266
Version プロパティ
ネームスペース iAnywhere.Data.UltraLite 346
ネームスペース iAnywhere.UltraLite 638

あ

アイコン
マニュアルで使用 xiii
値
Ultra Light.NET でのアクセス 48
アップグレード
Ultra Light.NET のデータベース・スキーマ 31
暗号化
Ultra Light.NET 開発 37

い

インデックス
Ultra Light.NET のスキーマ情報 55

え

エラー
Ultra Light.NET での処理　57
エラー処理
Ultra Light.NET　57

か

開発
Ultra Light.NET　29
開発プラットフォーム
Ultra Light.NET　3

き

規則
表記　xi
キャスト
Ultra Light.NET のデータ型　50

け

結果セット
Ultra Light.NET　44
結果セット・スキーマ
Ultra Light.NET　45
検索モード
Ultra Light.NET　48

こ

更新
Ultra Light.NET のロー　51
更新モード
Ultra Light.NET　48
コミット
Ultra Light.NET　54

さ

削除
Ultra Light.NET のロー　53
サポート
ニュースグループ　xvi
サポートされるプラットフォーム
Ultra Light.NET　3

し

準備文
Ultra Light.NET　38

す

スキーマ
Ultra Light.NET でのアクセス　55
Ultra Light.NET でのアップグレード　31
スキーマ・ファイル
Ultra Light.NET でのアップグレード　31
Ultra Light.NET での作成　30
スクロール
Ultra Light.NET　46
スレッド
マルチスレッドの Ultra Light.NET アプリケーション　33

せ

接続
Ultra Light.NET チュートリアル　16
Ultra Light.NET データベース　32

そ

挿入
Ultra Light.NET のロー　52
挿入モード
Ultra Light.NET　48

た

ターゲット・プラットフォーム
Ultra Light.NET 3

ち

チュートリアル
Ultra Light.NET の C# 7
Ultra Light.NET の Visual Basic 7

て

データ型
Ultra Light.NET でのアクセス 48
Ultra Light.NET でのキャスト 50
データ操作
Ultra Light.NET のテーブル API 46
Ultra Light.NET の動的 SQL 38
データベース
Ultra Light.NET での接続 32
Ultra Light.NET のスキーマ情報 55
データベース・スキーマ
Ultra Light.NET でのアクセス 55
Ultra Light.NET でのアップグレード 31
テーブル
Ultra Light.NET のスキーマ情報 55
テクニカル・サポート
ニュースグループ xvi

と

同期
Ultra Light.NET 59
Ultra Light.NET の ActiveSync 59
動的 SQL
Ultra Light.NET 開発 38
Ultra Light.NET チュートリアル 20
トランザクション
Ultra Light.NET 54
トランザクション処理
Ultra Light.NET 54

な

難読化
Ultra Light.NET 開発 37

に

ニュースグループ
テクニカル・サポート xvi

は

配置
Ultra Light.NET 27
パスワード
Ultra Light.NET での認証 58
パブリケーション
Ultra Light.NET のスキーマ情報 56

ひ

表記
規則 xi

ふ

フィードバック
提供 xvi
マニュアル xvi
プラットフォーム
Ultra Light.NET でのサポート 3

ま

マニュアル
SQL Anywhere Studio viii

マルチスレッドのアプリケーション
Ultra Light.NET 33

も

モード
Ultra Light.NET 48

ゆ

ユーザ
Ultra Light.NET での認証 58
ユーザ認証
Ultra Light.NET 開発 58

り

利点
Ultra Light.NET 2

る

ルックアップ・モード
Ultra Light.NET 48

ろ

ロー
Ultra Light.NET での現在のローの値へのアクセス 48
ロールバック
Ultra Light.NET 54