iAnywhere SOLUTIONS™
A SYBASE COMPANY

Ultra Light ActiveX
ユーザーズ・ガイド

パート番号：DC37123-01-0902-01
改訂：2005 年 3 月

目次

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 ActiveX について説明します。Ultra Light ActiveX を使用すると、Windows CE が稼働しているハンドヘルド、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 eMbedded Visual Basic および JScript アプリケーションの開発者を対象にしています。

eMbedded Visual Basic および JScript についての知識があることを前提とします。

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 拡張エージェント 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** }]

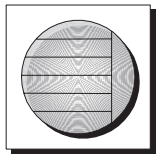
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

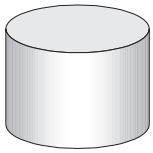
このマニュアルでは、次のアイコンを使用します。

- 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

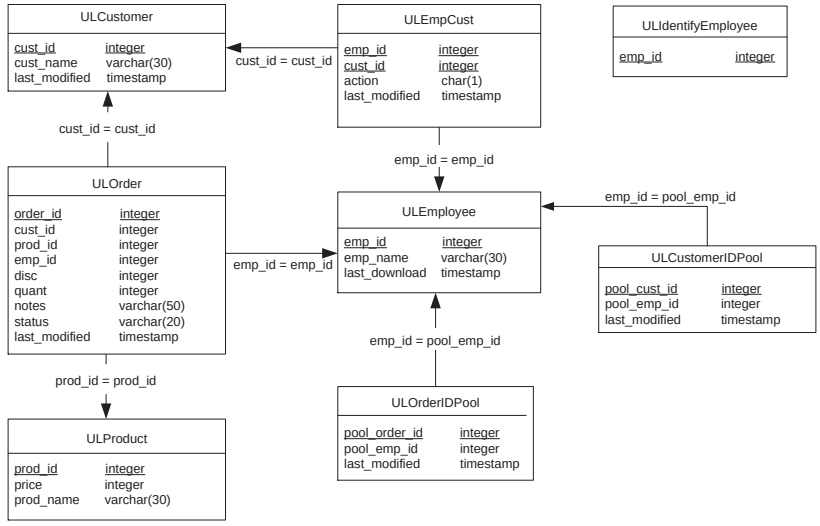
CustDB サンプル・データベース

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 ActiveX の概要

この章の内容

この章では、Ultra Light ActiveX 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 ActiveX の特徴

Ultra Light ActiveX は、モバイル・デバイスのためのリレーショナル・データ管理システムです。ビジネス・アプリケーションに必要なパフォーマンス、リソース効率、堅牢性、セキュリティを備えています。Ultra Light では、エンタープライズ・データ・ストアとの同期も提供されます。

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 ActiveX 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。

- 以下のいずれかが必要です。
  - eMbedded Visual Basic 3.0 (Visual Basic を使用した開発用)
  - Pocket Internet Explorer (JScript を使用した開発用)

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」を参照してください。

ターゲット・プラットフォーム

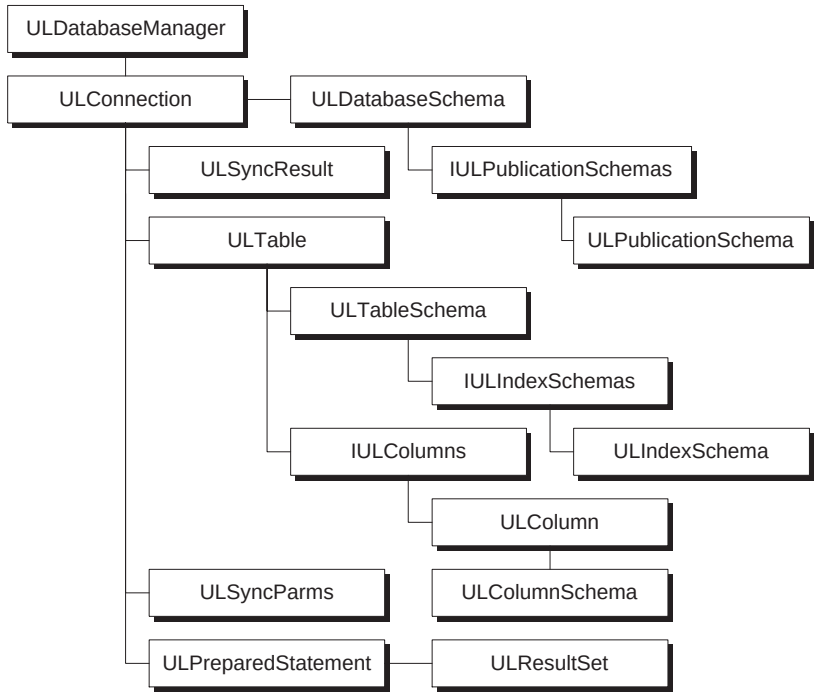
Ultra Light ActiveX は、次のターゲット・プラットフォームをサポートしています。

- ARM プロセッサと MIPS プロセッサの Pocket PC に搭載した Windows CE 3.0 以上。

詳細については、『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 ActiveX のアーキテクチャ

Ultra Light プログラミング・インタフェースは、Ultra Light データベースを使用したデータ操作のためのオブジェクト・セットを公開しています。次の図は、オブジェクト階層を示します。



次のリストは、よく使用される高度なオブジェクトの一部を示します。

- **ULDatabaseManager**　Ultra Light データベースへの接続を管理します。

  詳細については、「ULDatabaseManager クラス」126 ページを参照してください。

- **ULConnectionParms** 接続パラメータのセットを格納します。

  詳細については、「ULConnectionParms クラス」122 ページを参照してください。

- **ULConnection** データベース接続を表し、トランザクションを管理します。

  詳細については、「ULConnection クラス」112 ページを参照してください。

- **ULPreparedStatement、ULResultSet、ULResultSetSchema** SQL を使用してデータベース要求とその結果を管理します。

  詳細については、「ULPreparedStatement クラス」142 ページ、「ULResultSet クラス」147 ページ、「ULResultSetSchema クラス」153 ページを参照してください。

- **ULTable と ULColumn** テーブル・ベースの API を使用してデータを管理します。

  詳細については、「ULTable クラス」176 ページと「ULColumn クラス」104 ページを参照してください。

- **ULSyncParms と ULSyncResult** Mobile Link 同期サーバを介して同期を管理します。

  Mobile Link との同期の詳細については、『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。

第 2 章

Ultra Light ActiveX の開発の概要

この章の内容

この章では、Ultra Light ActiveX 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、「チュートリアル：Ultra Light ActiveX アプリケーションのサンプル」53 ページまたは『Ultra Light ActiveX ユーザーズ・ガイド』>「チュートリアル：Pocket IE 用の Ultra Light ActiveX アプリケーション」を参照してください。

Ultra Light ActiveX を使用するための準備

以下の手順では、Ultra Light ActiveX を使用してアプリケーションを構築する前に実行する必要のある操作を示します。

eMbedded Visual Basic 設計環境への Ultra Light ActiveX の追加

eMbedded Visual Basic プロジェクトから Ultra Light ActiveX コントロールにアクセスするには、Ultra Light ActiveX を設計環境に追加する必要があります。

次の手順が必要となるのは、eMbedded Visual Basic アプリケーションだけです。

❖ Ultra Light ActiveX への参照を追加するには、次の手順に従います。

1. eMbedded Visual Basic メニューから、[プロジェクト] ー [参照設定] を選択します。
2. [iAnywhere Solutions ActiveX for Ultra Light 9.0] コントロールが、利用可能な参照のリストに含まれている場合、それを選択して [OK] をクリックします。

   [iAnywhere Solutions ActiveX for Ultra Light 9.0] コントロールが利用可能な参照のリストにない場合は、次の手順に従います。

   - 利用可能な参照のリストにコントロールを追加します。

     SQL Anywhere 9.0 インストール環境の *UltraLite¥UltraLiteActiveX¥win32* サブディレクトリを検索します。*uldo9.dll* を開きます。

   - [iAnywhere Solutions ActiveX for Ultra Light 9.0] を選択し、[OK] をクリックします。

WindowsCE デバイスへの Ultra Light ActiveX の追加

エミュレータでアプリケーションをデバッグするには、エミュレータに Ultra Light ActiveX コントロールを追加します。Windows CE デバイスにアプリケーションを配備するには、デバイスに Ultra Light ActiveX コントロールを追加します。これらの作業は、Windows CE のコントロール・マネージャで行います。

次の手順は JScript と eMbedded Visual Basic の両方に適用されます。eMbedded Visual Basic がお使いのコンピュータにインストールされていない場合、適切な DLL をお使いのデバイスの *¥Windows* ディレクトリにコピーし、*regsvrce.exe* を使用して登録できます。お使いのデバイス上に *regsvrce.exe* がない場合、Microsoft Windows CE SDK からデバイスにコピーできます。

❖ Windows CE に Ultra Light ActiveX コントロールを追加するには、次の手順に従います。

1. eMbedded Visual Basic メニューから、[ツール] ー [リモートツール] ー [コントロール マネージャ] を選択します。
2. 左ウィンドウ枠で、Pocket PC など、開発するアプリケーションの対象デバイスを開きます。
3. [Pocket PC エミュレーション] など、配備先のデバイスを開きます。
4. [コントロール] ー [新しいコントロールを追加] を選択します。
5. SQL Anywhere インストール環境のサブディレクトリにある、プラットフォーム固有の次のファイルの 1 つを検索します。
   - **ARM** *UltraLite¥UltraLiteActiveX¥ce¥arm¥uldo9.dll*
   - **MIPS** *UltraLite¥UltraLiteActiveX¥ce¥mips¥uldo9.dll*
   - **エミュレータ** *UltraLite¥UltraLiteActiveX¥ce¥emulator30¥uldo9.dll*
   - **Intel 386** *UltraLite¥UltraLiteActiveX¥ce¥386¥uldo9.dll*

6 [開く] をクリックします。

Windows CE デバイスへのスキーマ・ファイルの配備

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

次の手順を使用して、Ultra Light スキーマ・ファイルを Windows CE デバイスに配備します。

Ultra Light スキーマ・ファイルの作成については、「Ultra Light データベース・スキーマ・ファイルの作成」11 ページを参照してください。

❖ Windows CE にスキーマ・ファイルを配備するには、次の手順に従います。

1 eMbedded Visual Basic メニューから、[ツール] ー [リモート ツール] ー [ファイル ビューア] を選択します。

2 左ウィンドウ枠にデバイスが表示されない場合は、次の手順に従ってデバイスに接続します。

- [接続] ー [接続の追加] を選択します。
- リストからデバイスを選択し、[OK] をクリックすると、接続が確立されます。

3 スキーマ・ファイルをデバイスにコピーします。

- デバイス上のコピー先ディレクトリを選択します。
- [ファイル] ー [ファイルをエクスポート] を選択します。
- スキーマ・ファイル (*.usm*) を選択します。
- [OK] をクリックすると、デバイスにファイルがエクスポートされます。

JScript の使用

次の項では、Ultra Light ActiveX のパフォーマンスを最適化するため、HTML ページを設定する方法を説明します。

キャッシュされたページ

Pocket IE がローカルにキャッシュされたページを使用するのを防ぎ、すべてのコンタクトが動的に再計算されるように、ページの有効期限を 0 に設定します。これにより、Pocket IE は常にページを再ロードします。HTML 文書の最初の行として次を使用します。

```
<META HTTP-EQUIV="Expires" CONTENT="0">
```

スクリプトの実行順

ページをロードする前にスクリプトを実行し、たとえば、動的な HTML コンテンツとなるデータベース接続を取得するには、目的のスクリプトを HTML ヘッダー部分に置きます。

```
<head> <script>... </script></head>
```

Pocket Internet Explorer の制限事項

Pocket IE がサポートする JScript 言語は、次の点で、Internet Explorer 4 以降の JScript 言語と互換性がありません。

- Pocket IE JScript は大文字と小文字を区別しません。Pocket IE 用に作成されたスクリプトは、適切な大文字と小文字を使用しないと IE で作動しない場合があります。
- BUTTON タグはサポートされていませんが、<INPUT TYPE="BUTTON"> を使用してボタンを作成できます。
- HTML タグの名前は、JScript 関数と同じネーム・スペースにあります。HTML 要素の名前が、現在の文書の関数名と競合しないことを確認してください。

  たとえば、このボタンを押すとエラーが発生します。

  ```
  <SCRIPT>
  function b_Done() { ... }
  </SCRIPT>
  <INPUT NAME="b_Done" TYPE="BUTTON" VALUE="Done"
  onClick="b_Done()">
  ```

- Pocket IE では動的 HTML はサポートされません。動的なコンテンツを取得するには、**document.write** を使って、または DIV コンテンツを設定して、HTML を発行できます。

データベース・スキーマの使用

スキーマは、データベースの構造です。スキーマは、データベース内のテーブル定義、インデックス定義、パブリケーション定義の集合であり、それらの間のすべての関係を示します。

Ultra Light データベースを作成するには、アプリケーションの関数を呼び出して、Ultra Light データベース・スキーマ・ファイルを作成し、このファイルをデータベースに適用します。

Ultra Light データベース・スキーマ・ファイルの作成の詳細については、以下を参照してください。

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。

データベースのスキーマの変更

既存のデータベースのスキーマを変更するには、新しいスキーマを使用してスキーマ・ファイルを作成し、このスキーマを既存のデータベースに適用します。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。

これらのメソッドの詳細については、「ApplyFile メソッド」138 ページと「ApplyFileWithParms メソッド」139 ページを参照してください。

新規スキーマ・ファイルの配備の準備については、『Ultra Light データベース・ユーザーズ・ガイド』>「Ultra Light データベース・スキーマのアップグレード」を参照してください。

例

次のコードで、新規スキーマ・ファイルが適用されます。

```
' eMbedded Visual Basic
Dim db As ULDatabaseManager
Dim conn As ULConnection
Set db = CreateObject("UltraLite.ULDatabaseManager")
Set conn = db.OpenConnection("dbf = ¥My
Documents¥mydb.udb")
conn.Schema.ApplyFile("¥My Documents¥myschema.usm")

// JScript
var db;
var conn;
db = new ActiveXObject("UltraLite.ULDatabaseManager");
conn = db.OpenConnection("dbf = ¥¥My
Documents¥¥mydb.udb");
conn.Schema.ApplyFile("¥¥My Documents¥¥myschema.usm");
```

Ultra Light 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。

ULConnection オブジェクトの使用

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

ULConnection オブジェクトの詳細については、「ULConnection クラス」112 ページを参照してください。

- **コミット動作**　デフォルトでは、Ultra Light 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.AutoCommit を false に設定し、アプリケーションにトランザクションを構築します。AutoCommit を off に設定しコミットを実行すると、アプリケーションのパフォーマンスを直接向上できます。

  詳細については、「Commit メソッド」114 ページを参照してください。

- **ユーザ認証**　GrantConnectTo メソッドと RevokeConnectFrom 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。

  詳細については、「ユーザの認証」46 ページを参照してください。

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

  詳細については、「データの同期」47 ページを参照してください。

- **テーブル**　ULConnection.GetTable メソッドを使用して、Ultra Light テーブルにアクセスします。

  詳細については、「GetTable メソッド」116 ページを参照してください。

データベースへの接続

ULConnectionParms オブジェクトまたは接続文字列を使用して、データベースに接続できます。ULConnectionParms オブジェクトを使用するメソッドでは、接続パラメータを簡単かつ正確に操作できます。接続文字列を使用するメソッドの場合、接続文字列を正しく作成することが要求されます。

次の手順では、ULConnectionParms オブジェクトを使用して Ultra Light データベースに接続します。

ULConnectionParms オブジェクトを使用した Ultra Light データベースへの接続の詳細については、「CreateDatabaseWithParms メソッド」128 ページと「OpenConnectionWithParms メソッド」135 ページを参照してください。

❖ ULConnectionParms を使用して 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 ULDatabaseManager オブジェクトを作成します。

DatabaseManager オブジェクトは、1 つのアプリケーションに 1 つだけ作成してください。このオブジェクトは、オブジェクト階層のルートにあります。そのため、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、またはクラスレベル変数として宣言するのが最も効果的です。

```
' eMbedded Visual Basic
 Dim DatabaseMgr As ULDatabaseManager
 Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")

// JScript
 var DatabaseMgr;
 DatabaseMgr = new ActiveXObject(
"UltraLite.ULDatabaseManager" );
```

2 ULConnection 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、接続を常時開いています。そのため、ULConnection 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。

```
' eMbedded Visual Basic
 Dim Connection As ULConnection

// JScript
 var Connection;
```

3　ULConnectionParms オブジェクトを作成します。

```
' eMbedded Visual Basic
 Dim LoginParms As ULConnectionParms
 Set LoginParms =
CreateObject("UltraLite.ULConnectionParms")

// JScript
 var LoginParms;
 LoginParms = new
ActiveXObject("UltraLite.ULConnectionParms");
```

4　ULConnectionParms オブジェクトの必須プロパティを設定します。

たとえば、次のコードは、Windows CE デバイス上のデータベースのロケーションを *¥tutorial¥tutorial.udb* と指定します。

```
' eMbedded Visual Basic
 LoginParms.DatabaseOnCE = "¥tutorial¥tutorial.udb"

// JScript
 LoginParms.DatabaseOnCE =
"¥¥tutorial¥¥tutorial.udb";
```

次のプロパティを使用して、CreateDatabaseWithParms のスキーマ・ファイルまたは OpenConnectionWithParms のデータベース・ファイルを指定します。追加プロパティの詳細については、「プロパティ」122 ページを参照してください。

キーワード	説明
DatabaseOnCE	Windows CE 上の Ultra Light データベースのパスとファイル名です。
DatabaseOnDesktop	デスクトップ・マシン上の Ultra Light データベースのパスとファイル名です。

キーワード	説明
SchmaOnCE	Windows CE 上の Ultra Light スキーマのパスとファイル名です。
SchmeOnDesktop	デスクトップ・マシン上の Ultra Light スキーマのパスとファイル名です。

5 データベースへの接続を開きます。

CreateDatabaseWithParms および OpenConnectionWithParms は、開いた接続を ULConnection オブジェクトとして返します。各メソッドは、1 つの ULConnectionParms オブジェクトを引数として取ります。

次のコードは、既存のデータベースに接続しようとします。データベースが存在しない場合、OpenConnectionWithParms はエラーを返します。これにより CreateDatabaseWithParms は、指定されたスキーマ・ファイルを使用してデータベースを作成します。

Crossfire では、GetOcx メソッドを ULConnectionParms オブジェクトに含めてください。

```
' eMbedded Visual Basic
 On Error Resume Next
 Set Connection =
DatabaseMgr.OpenConnectionWithParms( LoginParms )
 if Err.Number <> ULSQLCode.ulSQLE_NOERROR Then
   Set Connection =
DatabaseMgr.CreateDatabaseWithParms( LoginParms )
 End If

// JScript
 DatabaseMgr.ErrorResume = true;
 Connection = DatabaseMgr.OpenConnectionWithParms(
LoginParms );
 if ( DatabaseMgr.LastErrorCode !=
ULSQLCode.ulSQLE_NOERROR ) {
  Connection = DatabaseMgr.CreateDatabaseWithParms(
LoginParms );
 }
```

フレームによるアプリケーション状態の管理 (JScript)

Pocket IE では、HTML ページに埋め込まれた JScript を使用して、Ultra Light ActiveX にアクセスします。

次のコードでは、ULDatabaseManager オブジェクトが作成されます。このスクリプトを含むページがロードされると、新しい ULDatabaseManager オブジェクトが作成されます。ブラウザが別のページに移動すると、そのオブジェクトは廃棄されます。あるページでデータベース接続が取得された場合、そのページがアンロードされると、その接続は失われます。

```
<SCRIPT LANGUAGE="JScript">
 var DatabaseMgr;
 DatabaseMgr = new ActiveXObject(
"UltraLite.ULDatabaseManager" );
 </SCRIPT>
```

フォームが変更されるたびに、データベース接続などのアプリケーションの状態が失われると、速度が遅く負荷が高くなります。これを回避するため、Pocket IE アプリケーションは、HTML フレームを使用して、アプリケーションの状態を管理します。データベース接続は、FRAMESET 要素により維持されます。Pocket IE は、フレームセット内に少なくとも 2 つのフレームを必要とし、フレーム間に 3 ピクセルの境界を描きます。

注意
Pocket IE 1.1 より前のバージョンでは、フレームをサポートしません。

次のコードは 2 つのフレームを定義します。

```
// JScript
 <frameset rows="5%,*" BORDER=0>
 <frame SRC="topline.htm" NAME="TopLine" MARGINWIDTH=0
MARGINHEIGHT=0>
 <frame SRC="connect.htm" NAME="Connect" MARGINWIDTH=0
MARGINHEIGHT=0>
```

```
</frameset>
<SCRIPT LANGUAGE="JScript">
var Connection;      // UltraLite Connection
var DatabaseMgr;  // UltraLite Database Manager
</SCRIPT>
```

CustDB の例では、*topline.htm* で定義される最小フレームは、画面の一番上にアプリケーション名を表示するプレースホルダです。画面の残りの部分を使って、アプリケーションのほかのページが表示されます。*connect.htm* の HTML と JScript は、アプリケーションの初期フォームを提供します。これらのフォームは、すべて同じフレームセットにスワップされるので、**window.topwindow** を使用して、その親フレームセットのオブジェクトにアクセスできます。

次のコードは、新しい変数 conn を作成し、一番上のフレームで作成された接続をその変数に割り当てます。**top** にいる場合、その接続を **top.Connection** として参照するかどうかは任意です。

```
// JScript
 <SCRIPT>
 function usingConnection() {
   if ( top.Connection == null ) {
     return;     // not yet connected..
   }
   var conn = top.Connection;
   ...
 }
 </SCRIPT>
```

フォームをフレームの内外にスワップするのに、JScript 関数は **document.location.replace** メソッドを使用できます。次のコードは、現在の接続を閉じて、接続フォームを返します。

```
// JScript
 <SCRIPT>
 function exitApp() {
   if ( top.Connection != null ) {
     top.Connection.Close();
   }
   document.location.replace("connect.htm");
 }
 </SCRIPT>
 <INPUT NAME="b_Done" TYPE="BUTTON" VALUE="Done"
onClick="exitApp()">
```

暗号化と難読化

Ultra Light ActiveX を使用する場合、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 プロパティを設定します。CreateDatabaseWithParms を呼び出し、ConnectionParms オブジェクトを渡すと、作成されるデータベースは、指定されたキーを使用し暗号化されます。

EncryptionKey プロパティの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「EncryptionKey 接続パラメータ」と「ChangeEncryptionKey メソッド」114 ページを参照してください。

例

暗号化キーを変更するには、新しい暗号化キーを Connection オブジェクトで指定します。この例では「apricot」が暗号化キーです。

```
Connection.ChangeEncryptionKey("apricot")
```

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。

例

次のコードは、新しいデータベースを難読化します。

```
' eMbedded Visual Basic
 open_parms =
"ce_file=¥tutorial.udb;ce_schema=¥tutorial.usm;obfuscat
e=1"
 Set Connection = DatabaseMgr.CreateDatabase(
open_parms )
```

```
// JScript
 open_parms =
"ce_file=¥¥tutorial.udb;ce_schema=¥¥tutorial.usm;obfusc
ate=1";
 Connection = DatabaseMgr.CreateDatabase( open_parms );
```

動的 SQL を使用したデータの使用

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 の詳細については、「テーブル API を使用したデータの使用」30 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を説明します。

- 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
- テーブルのローの検索
- ローの挿入、削除、更新

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。

動的 SQL に必要な操作手順は、SQL の操作と変わりません。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。これらの操作は、ULPreparedStatement クラスのメンバである ExecuteStatement メソッドを使用して実行します。

ULPreparedStatement クラスの詳細については、「ULPreparedStatement クラス」142 ページを参照してください。

準備文でのパラメータの使用
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、各 ? は準備文での並び順を参照します。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 ULPreparedStatement オブジェクトを宣言します。

```
' eMbedded Visual Basic
 Dim PrepStmt As ULPreparedStatement

// JScript
 var PrepStmt;
```

2 INSERT 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
' eMbedded Visual Basic
 Set PrepStmt = Connection.PrepareStatement( _
     "INSERT into TableName(ColumnName) VALUES (?)")

// JScript
 PrepStmt = Connection.PrepareStatement(
    "INSERT into TableName(ColumnName) VALUES (?)");
```

3 その文にパラメータ値を割り当てます。

```
' eMbedded Visual Basic
 Dim NewValue As String
 NewValue = "Bob"
 PrepStmt.SetStringParameter 1, NewValue

// JScript
 var NewValue;
 NewValue = "Bob";
 PS.SetStringParameter(1, NewValue);
```

4 文を実行します。

```
' eMbedded Visual Basic
 PrepStmt.ExecuteStatement

// JScript
 PrepStmt.ExecuteStatement();
```

❖ ローを更新するには、次の手順に従います。

1　ULPreparedStatement オブジェクトを宣言します。

```
' eMbedded Visual Basic
 Dim PrepStmt As ULPreparedStatement

// JScript
 var PrepStmt;
```

2　UPDATE 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
' eMbedded Visual Basic
 Set PrepStmt = Connection.PrepareStatement( _
     "UPDATE TableName SET ColumnName = ? WHERE ID =
?")

// JScript
 PrepStmt = Connection.PrepareStatement(
     "UPDATE TableName SET ColumnName = ? WHERE ID =
?");
```

3　その文にパラメータ値を割り当てます。

```
' eMbedded Visual Basic
 Dim NewValue As String
 NewValue = "Bob"
 PrepStmt.SetParameter 1, NewValue
 PrepStmt.SetParameter 2, "6"

// JScript
 var NewValue;
 NewValue = "Bob";
 PrepStmt.SetParameter(1, NewValue);
 PrepStmt.SetParameter(2, "6");
```

4　文を実行します。

```
' eMbedded Visual Basic
 PrepStmt.ExecuteStatement

// JScript
 PrepStmt.ExecuteStatement();
```

❖ ローを削除するには、次の手順に従います。

1 ULPreparedStatement オブジェクトを宣言します。

```
' eMbedded Visual Basic
 Dim PrepStmt As ULPreparedStatement

// JScript
 var PrepStmt;
```

2 DELETE 文を準備文オブジェクトに割り当てます。

```
' eMbedded Visual Basic
 Set PrepStmt = Connection.PrepareStatement( _
     "DELETE FROM customer WHERE ID = ?")

// JScript
 PrepStmt = Connection.PrepareStatement( _
     "DELETE FROM customer WHERE ID = ?");
```

3 その文にパラメータ値を割り当てます。

```
' eMbedded Visual Basic
 Dim IDValue As String
 IDValue = "6"
 PrepStmt.SetParameter 1, IDValue

// JScript
 var IDValue;
 IDValue = "6";
 PrepStmt.SetParameter( 1, IDValue );
```

4 文を実行します。

```
'eMbedded Visual Basic
 PrepStmt.ExecuteStatement

// JScript
 PrepStmt.ExecuteStatement();
```

データ検索：SELECT

SELECT 文を実行すると、ULPreparedStatement.ExecuteQuery メソッドは ULResultSet オブジェクトを返します。

ULResultSet クラスには、結果セット内をナビゲーションするためのメソッドが含まれています。次に、ULResultSet Value プロパティを使用して、その値にアクセスします。

ULResultSet オブジェクトの詳細については、「ULResultSet クラス」147 ページを参照してください。

例

次のコードでは、SELECT クエリの結果に ULResultSet を使用してアクセスします。最初に割り当てられたとき、ULResultSet は最初のローの前に配置されます。次に ULResultSet.MoveFirst メソッドが呼び出され、結果セットの最初のレコードをナビゲーションします。

結果セットのナビゲーションの詳細については、「動的 SQL を使用したナビゲーション」28 ページを参照してください。

```
' eMbedded Visual Basic
 Dim MyResultSet As ULResultSet
 Dim PrepStmt As ULPreparedStatement
 Set PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 Set MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst

// JScript
 var MyResultSet;
 var PrepStmt;
 PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer");
 MyResultSet = PrepStmt.ExecuteQuery();
 MyResultSet.MoveFirst();
```

例

次のコードは、Value プロパティを使用して、現在のローのカラム値を取得する方法を説明します。これは、Value プロパティを使用して、Integer 値と String 値にアクセスします。Ultra Light ActiveX は、variant データ型を使用して、この柔軟性を達成します。

Value プロパティは次の構文を使用します。ここで、*Index* は SELECT 文でのカラム名の並び順です。

```
MyResultSetName.Value(Index)
```

データ修正の結果が反映されるように、MoveRelative(0) メソッドが呼び出され、結果セットの現在のバッファの内容をリフレッシュします。

```
' eMbedded Visual Basic
 If MyResultSet.RowCount = 0 Then
   lblID.Caption = ""
   txtName.Text = ""
 Else
   lblID.Caption = MyResultSet.Value(1)
   txtName.Text = MyResultSet.Value(2)
   MyResultSet.MoveRelative(0)
 End If

// JScript
 If ( MyResultSet.RowCount == 0 ) {
   lblID.Caption = "";
   txtName.Text = "";
 } Else {
   lblID.Caption = MyResultSet.Value(1);
   lblID.Text = MyResultSet.Value(2);
   MyResultSet.MoveRelative(0);
 }
```

次の手順では、SELECT 文を使用して、データベースから情報を取り出します。クエリの結果は、ULResultSet オブジェクトに割り当てられます。

❖ SELECT 文を実行するには、次の手順に従います。

1 ULPreparedStatement オブジェクトを宣言します。

```
' eMbedded Visual Basic
 Dim PrepStmt As ULPreparedStatement

// JScript
 var PrepStmt;
```

2 準備文を ULPreparedStatement オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
' eMbedded Visual Basic
 Set PrepStmt = Connection.PrepareStatement( _
     "SELECT ColumnName FROM TableName")

// JScript
 PrepStmt = Connection.PrepareStatement(
     "SELECT ColumnName FROM TableName")
```

3　クエリを実行します。

次の eMbedded Visual Basic コードでは、リストボックスが SELECT クエリの結果を取得します。

```
' eMbedded Visual Basic
 Dim MyResultSet As ULResultSet
 Set MyResultSet = PrepStmt.ExecuteQuery
 While MyResultSet.MoveNext
   listbox.AddItem y.Value(1)
 Wend
```

次の JScript コードでは、文字列が SELECT クエリの結果を HTML テーブルとして取得します。

```
// JScript
 var MyResultSet;
 var resultTable;
 MyResultSet = PrepStmt.ExecuteQuery();
 var ncols = MyResultSet.Schema.ColumnCount;
 var fld, line, nrows;
 resultTable = "<html><table cellpadding=0
cellspacing=0><tr>";
 resultTable = resultTable + "<th>" + "ColumnName" +
"</th>";
 resultTable = resultTable + "</tr><tr>";
 nrows = 0;
 while ( MyResultSet.MoveNext() ) {
   line = "<tr>";
     nrows++;
   if ( MyResultSet.IsNull(1) ) {
         fld = "(null)";
       } else {
         fld = MyResultSet.Value(1);
       }
```

```
            line = line + "<td>" + fld + "</td>";
        resultTable = resultTable + line + "</tr>";
    }
    resultTable = resultTable + _
        "</table><B>#Rows=" + nrows + "</B></html>";
```

動的 SQL を使用したナビゲーション

Ultra Light ActiveX は、幅広いナビゲーション作業を行うため、結果セットをナビゲーションする方法を多数提供します。

次の ULResultSet オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **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** いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再配置する場合は、0 が便利です。

例

次のコードは、MoveFirst メソッドを使用して、結果セット内をナビゲーションする方法を説明します。

```
' eMbedded Visual Basic
 Set PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 Set MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst
```

```
// JScript
 PrepStmt = Connection.PrepareStatement(
     "SELECT ID, Name FROM customer");
 MyResultSet = PrepStmt.ExecuteQuery();
 MyResultSet.MoveFirst();
```

すべての Move メソッドで同じ方法を使用できます。

これらのナビゲーション・メソッドの詳細については、「ULResultSet クラス」147 ページを参照してください。

ULResultSet schema プロパティ

ULResultSet.Schema プロパティを使うと、クエリのカラムに関する情報を取り出すことができます。ULResultSetSchema オブジェクトのプロパティは、ColumnName、ColumnCount、ColumnPrecision、ColumnScale、ColumnSize、ColumnSQLType です。

例

次の例は、ULResultSet.Schema を使用して、スキーマ情報を取得する方法を示しています。

```
' eMbedded Visual Basic
 Dim i As Integer
 Dim MySchema as ULResultSetSchema
 Set MySchema = MyResultSet.Schema
 For i = 1 To MySchema.ColumnCount
   cn = MySchema.ColumnName(i)
   ct = MySchema.ColumnSQLType(i)
   MsgBox cn, ct
 Next i

// JScript
 var i;
 var MySchema;
 For ( i = 1; i <= MySchema.ColumnCount; i++) {
   cn = MySchema.ColumnName(i);
   ct = MySchema.ColumnSQLType(i);
   alert ( cn + " " + ct );
 }
```

テーブル API を使用したデータの使用

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 の詳細については、「動的 SQL を使用したデータの使用」21 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 API を使用したナビゲーション

Ultra Light ActiveX 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。

次の ULTable オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **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** いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値はテーブル内を前に移動し、負のインデックス値はテーブル内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再配置する場合は、0 が便利です。

例

次のコードは、customer テーブルを開き、そのローをスクロールします。次に、各顧客の姓を示すメッセージ・ボックスまたは警告を表示します。

```
' eMbedded Visual Basic
 Dim tCustomer as ULTable
 Set tCustomer = Connection.GetTable( "customer" )
 tCustomer.Open
 ' the third column contains the last name of the
customer
 Set colLastName = tCustomer.Columns.Item(3)
 tCustomer.MoveBeforeFirst
 While tCustomer.MoveNext
     MsgBox colLastName.Value
 Wend

// JScript
 var tCustomer;
 tCustomer = Connection.GetTable( "customer" );
 tCustomer.Open();
 // the third column contains the last name of the
customer
 colLastName = tCustomer.Columns.Item(3);
 tCustomer.MoveBeforeFirst();
 While (tCustomer.MoveNext()) {
   alert( colLastName.Value );
 }
```

Columns コレクション

テーブルのカラムは、Columns コレクションに含まれます。インデックス番号 (スキーマ・ファイルで作成された順序) またはカラム名を使用して、カラムを指定できます。

詳細については、『Ultra Light ActiveX ユーザーズ・ガイド』> 「IULColumns コレクション」を参照してください。

例

次のコードは LastName カラムにアクセスします。

```
' eMbedded Visual Basic
 Set colLastName = tCustomer.Columns.( LastName )

// JScript
 colLastName = tCustomer.Columns.( LastName );
```

インデックスの指定

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・キー値の順に公開されますが、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_name インデックスで順序付けられた Customer テーブルの最初のローに移動します。

```
' eMbedded Visual Basic
 Set tCustomer = Connection.GetTable("customer")
 tCustomer.Open "ix_name"
 tCustomer.MoveFirst

// JScript
 tCustomer = Connection.GetTable("customer");
 tCustomer.Open("ix_name");
 tCustomer.MoveFirst();
```

現在のローの値へのアクセス

ULTable 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

ULTable オブジェクトがローの上に置かれている場合は、ULColumn.Value プロパティを使用して、現在のローのそのカラムの値を取得できます。

例

次のコードは、tCustomer ULTable オブジェクトから 3 つのカラムの値を取り出して、テキスト・ボックスに表示します。

```
' eMbedded Visual Basic
 Dim colID, colFirstName, colLastName As ULColumn
 Set colID = tCustomer.Columns.Item(1)
 Set colFirstName = tCustomer.Columns.Item(2)
 Set colLastName = tCustomer.Columns.Item(3)
 txtID.Text = colID.Value
 txtFirstName.Text = colFirstName.Value
 txtLastName.Text = colLastName.Value

// JScript
 var colID, colFirstName, colLastName;
 colID = tCustomer.Columns.Item(1);
 colFirstName = tCustomer.Columns.Item(2);
 colLastName = tCustomer.Columns.Item(3);
 txtID.Text = colID.Value;
 txtFirstName.value = colFirstName.Value;
 txtLastName.value = colLastName.Value;
```

Value プロパティを使用して、値を設定することもできます。

```
' eMbedded Visual Basic
 colLastName.Value = "Kaminski"

// JScript
 colLastName.Value = "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。

位置がテーブルの最初のローの前または最後のローの後ろにある場合でも、プロパティに値を割り当てることができます。しかし、カラムから値を取得することはできません。たとえば、次のコードはエラーを生成します。

```
' This eMbedded Visual Basic code is incorrect
 tCustomer.MoveBeforeFirst
 id = colID.Value

// This JScript code is incorrect
 tCustomer.MoveBeforeFirst();
 id = colID.Value();
```

値のキャスト

Value メソッドから variant 型が返されると、それを使用して任意のデータ型にアクセスできます。

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。ULTable 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** ULTable 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。

  find メソッドの詳細については、「FindBegin メソッド」178 ページを参照してください。

- **lookup メソッド** ULTable 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。

  lookup メソッドの詳細については、「LookupBackward メソッド」182 ページを参照してください。

❖ 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。

FindBegin メソッドまたは LookupBegin メソッドを呼び出します。たとえば、次のコードは ULTable.FindBegin を呼び出します。

```
' eMbedded Visual Basic
 tCustomer.FindBegin

// JScript
 tCustomer.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。これらの値を設定すると、バッファに影響しますが、データベースには影響しません。たとえば、次のコードは、バッファの姓のカラムを Kaminski に設定します。

```
' eMbedded Visual Basic
 tCustomer.Columns.Item(3).Value = "Kaminski"

// JScript
 tCustomer.Columns.Item(3).Value = "Kaminski";
```

マルチカラム・インデックスの場合は、最初のカラムの値が必要ですが、ほかのカラムは省略できます。

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。

```
' eMbedded Visual Basic
 tCustomer.FindFirst

// JScript
 tCustomer.FindFirst();
```

ローの挿入、更新、削除

Ultra Light は、テーブルのローを一度に 1 つずつアプリケーションに公開します。ULTable オブジェクトにはカレント・ポジションがあります。カレント・ポジションは、テーブルのローの上、最初のローの前、または最後のローの後ろになります。

アプリケーションがロケーションを変更すると、Ultra Light はバッファにそのローのコピーを作成します。値を取得または設定する操作はすべて、このバッファにあるデータのコピーにのみ影響します。データベースのデータには影響しません。

例　次の文は、バッファの ID カラムの値を 3 に変更します。

```
' eMbedded Visual Basic
 colID.Value = 3

// JScript
 colID.Value = 3;
```

Ultra Light のモードの使用

Ultra Light モードによって、バッファ内の値を使用する目的が決まります。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** ULTable.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** ULTable.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** ULTable.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 ULTable.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。

❖ 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 メソッドや lookup メソッドを使用して検索し、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、tCustomer テーブル上で更新モードを開始します。

```
' eMbedded Visual Basic
 tCustomer.UpdateBegin

// JScript
 tCustomer.UpdateBegin();
```

3 更新するローの新しい値を設定します。

たとえば、次の指示は新しい値を Elizabeth に設定します。

```
' eMbedded Visual Basic
 ColFirstName.Value = "Elizabeth"

// JScript
 ColFirstName.Value = "Elizabeth";
```

4　Update を実行します。

```
'eMbedded Visual Basic
 tCustomer.Update

// JScript
 tCustomer.Update();
```

更新操作が終了すると、直前に更新したローが現在のローになります。ULTable 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、現在の位置は不確定です。

デフォルトでは、Ultra Light 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」40 ページを参照してください。

警告
ローのプライマリ・キーを更新しないでください。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、テーブル内の特定のローにあらかじめ指定する必要はありません。ローは、テーブルを開くときに使用したインデックスで自動的にソートされます。

❖ ローを挿入するには、次の手順に従います。

1　挿入モードを開始します。

たとえば、次の指示は、CustomerTable テーブル上で更新モードを開始します。

```
' eMbedded Visual Basic
 CustomerTable.InsertBegin

// JScript
 CustomerTable.InsertBegin();
```

2 新しいローの値を設定します。

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値がない場合は、NULL が使用されます。カラムが NULL を許可しない場合は、次のデフォルトが使用されます。

- 数値カラムの場合は 0
- 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。

```
' eMbedded Visual Basic
 CustomerTable.Columns.Item("Fname").Value = fname
 CustomerTable.Columns.Item("Lname").Value = lname

// JScript
 CustomerTable.Columns("Fname").Value = fname;
 CustomerTable.Columns("Lname").Value = lname;
```

3 挿入を実行します。

挿入されたローは、Commit を実行したときに永続的にデータベースに保存されます。AutoCommit モードでは、Insert メソッドの一部として Commit が実行されます。

```
'eMbedded Visual Basic
 CustomerTable.Insert

// JScript
 CustomerTable.Insert();
```

ローの削除

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。

次のプロシージャは、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　削除を実行します。

```
'eMbedded Visual Basic
 tCustomer.Delete

// JScript
 tCustomer.Delete();
```

BLOB データの処理

GetByteChunk メソッドを使用して、BINARY または LONG BINARY と宣言された、カラムの BLOB データをフェッチできます。

詳細については、「GetByteChunk メソッド」106 ページを参照してください。

例

次のコードは、ULColumn.GetByteChunk メソッドを使用して、BLOB データを取得する方法を説明します。

```
' eMbedded Visual Basic
 Dim offset As Integer
 Dim nBytes As Integer
 Dim chunk(1024) As Byte
 ll = col.GetByteChunk( 0, chunk, 1024)
 If nBytes <> 1024 Then
    ' only got nBytes bytes, expected 1024
 Else
    ' have 1024 bytes
 End if

// JScript
 var offset;
 var nBytes;
 var chunk = new Array();
 nBytes = col.GetByteChunk( 0, chunk, 1024);
 if ( nBytes != 1024 ) {
   // only got nBytes bytes, expected 1024
 } else {
   // have 1024 bytes
 }
```

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 は AutoCommit モードで動作します。AutoCommit モードでは、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。

ULConnection.AutoCommit プロパティを false に設定すると、複数文のトランザクションを使用できます。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、振り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が、1 つのトランザクションを構成します。AutoCommit が false に設定されている場合は、ULConnection.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、ULConnection.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もかまいません。AutoCommit をオフにすると、パフォーマンスが向上します。

注意
Autocommit を False に設定していても、同期はコミットを実行します。

スキーマ情報へのアクセス

ULConnection、ULTable、ULColumn オブジェクトにはそれぞれ、スキーマ・プロパティが含まれます。これらのスキーマ・オブジェクトは、データベース内のテーブル、カラム、インデックス、パブリケーションに関する情報を提供します。

注意
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、「データベースのスキーマの変更」12 ページを参照してください。

- **ULDatabaseSchema**　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。

  ULDatabaseSchema オブジェクトを取得するには、ULConnection.Schema プロパティにアクセスします。

- **ULTableSchema**　テーブル内のカラムの数と名前、テーブルの Indexes コレクション。

  ULTableSchema オブジェクトを取得するには、ULTable.Schema プロパティにアクセスします。

- **ULColumnSchema**　デフォルト値、名前、オートインクリメントかどうかなど、個々のカラムに関する情報。

  ULColumnSchema オブジェクトを取得するには、ULColumn.Schema プロパティにアクセスします。

- **ULIndexSchema**　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対応するデータがないので、個別の ULIndex オブジェクトはなく、ULIndexSchema オブジェクトだけが存在します。

  ULIndexSchema オブジェクトは、ULTableSchema.Indexes コレクションの一部として利用可能です。

- **ULPublicationSchema** パブリケーションに含まれるテーブルとカラムの数と名前。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、ULPublication オブジェクトではなく、ULPublicationSchema オブジェクトが存在します。

  ULPublicationSchema オブジェクトは、ULDatabaseSchema.Publications コレクションの一部として利用可能です。

- **ULResultSetSchema** 結果セットのカラムの数と名前。

  ULResultSetSchema オブジェクトは、ULPreparedStatement.Schema プロパティを使用してアクセスできます。

エラー処理

通常の操作では、Ultra Light ActiveX はエラーをスローできます。エラーは SQLCODE 値として表現され、負の数字は特定の種類のエラーを示します。

Ultra Light ActiveX によってスローされるエラー・コードのリストについては、「ULSQLCode 列挙」155 ページを参照してください。

UltraLite ActiveX は、ULDatabaseManager オブジェクトと ULConnection オブジェクトからしか、エラーをスローしません。ULDatabaseManager の次のメソッドは、エラーをスローできます。

- CreateDatabase
- CreateDatabaseWithParms
- DropDatabase
- DropDatabaseWithParms
- OpenConnection
- OpenConnectionWithParms

Ultra Light ActiveX 内での他のエラーや例外はすべて、ULConnection オブジェクトを経由します。

ULDatabaseManager オブジェクトと ULConnection オブジェクトからのエラー番号へのアクセスの詳細については、「ULConnection クラス」112 ページと 「ULDatabaseManager クラス」126 ページを参照してください。

以下の項では、eMbedded Visual Basic と JScript でエラー処理を実装する方法を説明します。

eMbedded Visual Basic のエラー処理

eMbedded Visual Basic 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。エラー処理を有効にするには、On Error Resume Next 文を使用します。On Error Goto 0 を使用すると、エラーが発生し、コードの実行が停止します。

Err.Number プロパティは、エラーの SQLCODE 値を保持します。ULConnection.LastErrorCode プロパティを使用して、最後に発生したエラーを取得することもできます。

エラーを取得する必要がある一般的な場面は、存在しないデータベース・ファイルを処理する時です。次の例では、On Error Resume Next が有効なので、制御は、エラーを発生した文に続く次の文に進みます。指定したデータベースが存在しない場合、Err オブジェクトには、エラー番号と説明がロードされます。

```
' eMbedded Visual Basic
 On Error Resume Next
 Err.Clear     ' Clear any previous errors
 Set Connection = DBMgr.OpenConnection(udb)
 If Err.Number <> 0 Then   ' Connection failed, no
database?
   Err.Clear
   Set Connection = DBMgr.CreateDatabase(udb & usm)
   If Err.Number <> 0 Then
     MsgBox "Connect with " & udb & usm & " failed: " &
Err.Description
     App.End
   End If
 End If
```

JScript のエラー処理

Pocket IE でサポートされる JScript のバージョンには、エラー処理はありません。デフォルトの Pocket IE の設定では、スクリプト・エラーは無視され、何も通知されずにスクリプトは終了されます。新しい JScript を開発する場合、これによって開発が困難になります。

ShowScriptErrors レジストリ・キーの設定

eMbedded Visual Basic に付属しているレジストリ・エディタなどの、リモートのレジストリ・エディタを使用して、CE デバイスでこのキーを作成します。

```
[HKEY_CURRENT_USER¥Software¥Microsoft¥Internet
Explorer¥Main]ShowScriptErrors=dword:00000001
```

このキー・セットを使用して、Pocket Internet Explorer は、失敗した JScript コード用のエラー通知メッセージを提供します。レポートされる行番号は信頼できない場合があることに注意してください。

ErrorResume プロパティの設定

Ultra Light ActiveX のエラーを表示しないように、ULDatabaseManager オブジェクトと ULConnection オブジェクトは次の 2 つのプロパティを提供します。

- **ErrorResume** True に設定すると、後続のエラーのスローを無効にします。
- **LastErrorCode** 最後の操作によって設定されたエラー・コードを返します。

エラーを取得する必要がある一般的な場面は、スキーマから作成されることになっている、存在しないデータベース・ファイルを処理する時です。次の例では、ULDatabaseManager.ErrorResume プロパティが True なので、制御は、エラーを発生した文に続く次の文に進みます。指定したデータベースが存在しない場合、LastErrorCode プロパティにエラー番号がロードされます。

```
// JScript
 DBMgr.ErrorResume = true;   // Do not throw errors
 Connection = DBMgr.OpenConnection( udb );
 if ( DBMgr.LastErrorCode != 0 ) {
   Connection = DBMgr.CreateDatabase( udb + usm );
   if ( DBMgr.LastErrorCode != 0 ) {
     alert("Connect with " + udb + usm + failed: " +
DBMgr.LastErrorCode );
   }
 }
```

ユーザの認証

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。

接続権限の付与または取り消しの詳細については、「GrantConnectTo メソッド」116 ページと「RevokeConnectFrom メソッド」117 ページ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 希望するパスワードでユーザに接続権限を付与します。

```
'eMbedded Visual Basic
 conn.GrantConnectTo("Robert", "newPassword")

// JScript
 conn.GrantConnectTo("Robert", "newPassword");
```

❖ 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 次のように、ユーザの接続権限を取り消します。

```
'eMbedded Visual Basic
 conn.RevokeConnectFrom("Robert")

// JScript
 conn.RevokeConnectFrom("Robert");
```

データの同期

Ultra Light アプリケーションは、統合データベースと同期できます。同期には、Mobile Link 同期サーバと適切なライセンスが必要です。

この項では、同期の概要について簡単に説明し、Ultra Light ActiveX 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。eMbedded Visual Basic の場合、このサンプルは「チュートリアル：Ultra Light ActiveX アプリケーションのサンプル」53 ページで説明されています。JScript の場合、このサンプルは『Ultra Light ActiveX ユーザーズ・ガイド』> 「チュートリアル：Pocket IE 用の Ultra Light ActiveX アプリケーション」で説明されています。

Ultra Light ActiveX 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、ULConnection オブジェクトのメソッドとプロパティを使用して同期を制御します。

注意
暗号化された同期 (HTTPS) を使用して同期する、または TCP/IP で暗号化を使用するには、別途ライセンスを取得できるセキュリティ・オプションが必要です。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。

❖ TCP/IP または HTTP で同期するには、次の手順に従います。

1　同期情報を準備します。

ULConnection.SyncParms オブジェクトの必須プロパティに値を割り当てます。

設定するプロパティと値の詳細については、『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。

2 同期を実行します。

ULConnection.Synchronize メソッドを呼び出します。

同期の進行状況のモニタ

この項は、eMbedded Visual Basic にのみ適用されます。JScript を使用して同期の進行状況をモニタすることはできません。

同期の進行状況をモニタするには、プロジェクトに同期ダイアログを追加し、必要に応じてコーディングします。

次の手順では、*ULSyncStatus.ebf* のダイアログがステータス・メッセージを表示します。

❖ プロジェクトに同期ステータス・ダイアログを追加するには、次の手順に従います (eMbedded Visual Basic)。

1 SQL Anywhere インストール環境の *Samples¥UltraLite-ActiveX¥dbview.evb* サブディレクトリを検索します。

2 プロジェクトに *ULSyncStatus.bas* と *ULSyncStatus.ebf* をコピーし追加します。

3 Ultra Light ActiveX プロジェクトにコードを追加します。

- ULDatabase Manager をインスタンス化するのに、CreateObject を使用する代わりに、次の指示を使用します。

```
Set DBMgr = CreateObjectWithEvents( _
   "UltraLite.ULDatabaseManager", "UL_")
```

- 次のように、同期呼び出しがコールバックを受け入れることを確認します。

```
connection.Synchronize(True)
```

- [同期進行] ダイアログでイベント通知を取得するコードを作成します。

  次のメソッドは、統合データベースへのデータ送信時の、データの挿入、更新、削除をユーザに表示します。

```
Private Sub UL_OnSend(ByVal nBytes As Long,
ByVal _
     nInserts As Long, _
     ByVal nUpdates As Long,  _
     ByVal nDeletes As Long)_
     prLine "OnSend " & nBytes & " bytes, " &
nInserts &_
         " inserts, " & nUpdates & " updates, "
&_
         nDeletes & " deletes"_
 End Sub
```

  次のメソッドは、統合データベースでのデータ受信時の、データの挿入、更新、削除をユーザに表示します。

```
Private Sub UL_OnReceive(ByVal nBytes As Long,
ByVal _
     nInserts As Long, _
     ByVal nUpdates As Long, _
     ByVal nDeletes As Long)
     prLine "OnReceive " & _
        nBytes & " bytes, " & _
        nInserts  & " inserts, " & _
        nUpdates & " updates, " & _
        nDeletes & " deletes"
 End Sub
```

  次のメソッドは、同期ステータスがいつ変更されるかをユーザに表示します。

```
Private Sub UL_OnStateChange(ByVal newState As
Long, _
  ByVal oldState As Long)
     prLine "OnStateChange new:" & _
            newState & ", old: " & oldState
 End Sub
```

  次のメソッドは、現在同期中のテーブルがいつ変更されるかをユーザに表示します。

```
Private Sub UL_OnTableChange(ByVal
newTableIndex As _
  Long, ByVal numTables As Long)
     prLine "OnTableChange index:" &
newTableIndex & _
            ", #tables=" & numTables
 End Sub
```

Ultra Light アプリケーションの配備

アプリケーションを完了したら、またはアプリケーションをテストしたい場合、アプリケーションをデバイスに配備する必要があります。この項では、デバイスに Ultra Light アプリケーションを配備するための手順について説明します。

Windows CE への Ultra Light ActiveX アプリケーションの配備

Ultra Light アプリケーションを Windows CE に配備するには、次の手順を実行する必要があります。

- アプリケーションと Ultra Light コンポーネントを配備します。
- Ultra Light データベースまたはスキーマの初期コピーを配備します。

  多くの場合、Ultra Light スキーマ・ファイルのみを配備すれば十分です。データベース・ファイルは、初回の接続試行時にスキーマから作成されます。このため、同期を使用してデータの初期コピーをロードできます。

  データベースまたはスキーマ・ファイルは、アプリケーションが特定できるロケーションに配備する必要があります。このロケーションは、Database On CE および Schema On CE 接続パラメータによって定義されます。

  『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnCE 接続パラメータ」と『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnCE 接続パラメータ」を参照してください。
- ActiveSync 用 Mobile Link 同期コンジットを配備します。

  この手順が必要なのは、ActiveSync を使用してアプリケーションを同期する場合のみです。

  手順については、『Mobile Link クライアント』>「ActiveSync 用 Mobile Link プロバイダのインストール」を参照してください。

第 3 章

チュートリアル：Ultra Light ActiveX アプリケーションのサンプル

この章の内容

この章はチュートリアル形式で、PocketPC デバイス用の Ultra Light ActiveX アプリケーションを構築するプロセスを解説します。

このチュートリアルでは、eMbedded Visual Basic を使用します。JScript を使用した Ultra Light ActiveX アプリケーションの例については、『Ultra Light ActiveX ユーザーズ・ガイド』>「チュートリアル：Pocket IE 用の Ultra Light ActiveX アプリケーション」を参照してください。

概要

このチュートリアルでは、テーブル API を使用して Ultra Light ActiveX アプリケーションを構築するプロセスを解説します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、中央のデータベースと同期します。

テーブル API の詳細については、『Ultra Light ActiveX ユーザーズ・ガイド』>「Ultra Light ActiveX API リファレンス」を参照してください。

所要時間

このチュートリアルは、コードをコピーして貼り付ける場合、約 30 分で終了します。自分でコードを入力する場合、これよ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。

- Microsoft eMbedded Visual Tools がコンピュータにインストールされている
- eMbedded Visual Basic アプリケーションの作成、テスト、トラブルシューティングができる
- 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』>「Ultra Light スキーマ・ペインタ」を参照してください。
- AppForge Booster がインストールされている

  Booster がない場合は、http://www.appforge.com/booster.html から入手できます。

注意
SQL Anywhere Studio がなくても、このチュートリアルの大部分を実行できます。このチュートリアルの同期の項では、SQL Anywhere Studio が必要です。

目的

このチュートリアルの目的は、Ultra Light 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：プロジェクト・アーキテクチャの作成

最初の手順では、Ultra Light データベース・スキーマを作成する方法を説明します。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、パブリケーションとそれらの間のすべての関係を記述します。

データベース・スキーマの詳細については、「Ultra Light データベース・スキーマ・ファイルの作成」11 ページを参照してください。

❖ Ultra Light データベース・スキーマを作成するには、次の手順に従います。

1 このチュートリアル用のディレクトリを作成します。

このチュートリアルでは、保存先ディレクトリを *c:¥Tutorial¥evb* 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、*c:¥Tutorial¥evb* の代わりにそのディレクトリを使用してください。

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

データベース・スキーマの作成の詳細については、『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。

- **スキーマのファイル名** tutcustomer.usm
- **テーブル名** customer
- **customer のカラム**

カラム名	データ型 (サイズ)	カラムの NULL 値の許可	デフォルト値
id	integer	いいえ	オートインクリメント
fname	char (15)	いいえ	なし

カラム名	データ型 (サイズ)	カラムの NULL 値の許可	デフォルト値
lname	char (20)	いいえ	なし
city	char (20)	はい	なし
phone	char (12)	はい	555-1234

- **プライマリ・キー** 昇順 id

eMbedded Visual Basic プロジェクトの作成

eMbedded Visual Basic 開発用の Ultra Light コンポーネントは、Ultra Light ActiveX です。

次の手順では、アプリケーションの eMbedded Visual Basic プロジェクトを作成し、Ultra Light ActiveX コントロールへの参照を追加します。

❖ Ultra Light ActiveX コントロールへの参照を追加するには、次の手順に従います。

1　eMbedded Visual Basic を起動します。

- [スタート] ー [プログラム] ー [Microsoft eMbedded Visual Tools] ー [eMbedded Visual Basic 3.0] を選択します。

[新規プロジェクト] ウィンドウが表示されます。

2　新規プロジェクトを作成します。

アプリケーションの設計ターゲットを選択し、[開く] をクリックします。これ以降は、PocketPC プロジェクトに Windows CE を選択したことを前提に説明します。

3　Ultra Light ActiveX に参照を追加します。

- [プロジェクト] ー [参照] を選択します。

- [iAnywhere Solutions, ActiveX for Ultra Light 9.0] を選択し、[OK] をクリックします。

  コントロールが使用可能な参照のリストに表示されない場合は、コントロールを使用可能な参照のリストに追加します。

  - SQL Anywhere インストール環境の *UltraLite¥UltraLiteActiveX¥win32¥* サブディレクトリを検索します。
  - *uldo9.dll* を選択し、[OK] をクリックします。

4 プロジェクトを保存します。

- [ファイル] ー [プロジェクトの保存] を選択します。
- フォームを *c:¥tutorial¥evb¥Tutorial.ebf* として保存します。
- プロジェクトを *c:¥tutorial¥evb¥Tutorial.ebp* として保存します。

レッスン 2：フォームの作成

「レッスン 1：プロジェクト・アーキテクチャの作成」56 ページの手順を完了すると、プロジェクトにフォームが 1 つ表示されます。次の手順は、フォームを使用してユーザ・インタフェースを作成します。この例では、実際のアプリケーションと同様に、ラベルを出力として、テキスト・ボックスとボタンを入力として使用します。

❖ プロジェクトにコントロールを追加するには、次の手順に従います。

1　次の表のコントロールとプロパティをフォームに追加します。

タイプ	名前	キャプションまたはテキスト
TextBox	txtfname	
TextBox	txtlname	
TextBox	txtcity	
TextBox	txtphone	
Label	lblID	
Button	btnInsert	Insert
Button	btnUpdate	Update
Button	btnDelete	Delete
Button	btnNext	Next
Button	btnPrevious	Previous
Button	btnSync	Synchoronize
Button	btnDone	End

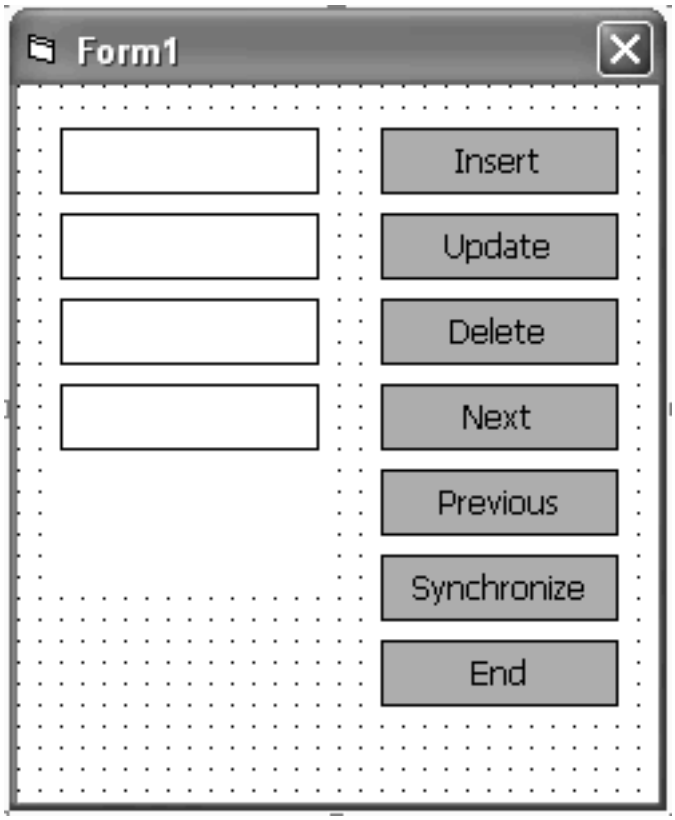
2　アプリケーションを確認します。

- [実行] ー [実行] を選択します。

Windows CE エミュレータにアプリケーションが表示されます。

- フォームの右上にある [OK] ボタンをクリックしてアプリケーションを終了します。

フォームは、次の図のようになります。



Ultra Light ActiveX をサポートするエミュレータの設定

アプリケーションへの Ultra Light オブジェクトの追加が完了したら、アプリケーションのデバッグとテストのために、エミュレータに Ultra Light ActiveX コントロールを追加します。

❖ エミュレータに Ultra Light ActiveX コントロールを追加するには、次の手順に従います。

1　コントロール・マネージャを起動します。

- eMbedded Visual Basic で、[ツール] ー [リモート ツール] ー [コントロール マネージャ] を選択します。

2　ターゲット・エミュレータを選択します。

- 左ウィンドウ枠で、[Pocket PC] を開き、[Pocket PC エミュレーション] を選択します。

3　Ultra Light コントロールを追加します。

- [コントロール] ー [新しいコントロールを追加] を選択します。
- SQL Anywhere ディレクトリにある、*ultralite¥UltraLiteActiveX¥ce¥emulator30¥uldo9.dll* を検索します。
- [OK] をクリックします。

  uldo9.dll は、ActiveX ファイルであり、コントロールではないので、デバイスのコントロールの一覧に表示されないことに注意してください。

データベース・スキーマの配備

Ultra Light コントロールに加え、データベース・スキーマもエミュレータに配備します。次の手順は、アプリケーションが初めてデータベースに接続するとき、そのスキーマを使用してデータベース・ファイルを作成します。

❖ エミュレータにデータベース・スキーマ・ファイルを配備するには、次の手順に従います。

1　Windows CE のファイル・ビューアを起動します。

- eMbedded Visual Basic で、[ツール] － [リモート ツール] － [ファイル ビューア] を選択します。
- ファイル・ビューアが自動的にエミュレータに接続しない場合は、[接続] － [接続を追加] を選択して、[Pocket PC エミュレーション] を選択します。

2 アプリケーションを格納するフォルダを作成します。

- [Program Files] フォルダを開きます。
- ファイル・ビューアで、[ファイル] － [新しいフォルダ] を選択します。
- *tutorial* という名前のフォルダを作成します。このフォルダにアプリケーション・ファイルを格納します。
- *tutorial* をクリックして、このフォルダに移動します。

3 スキーマ・ファイルをエミュレータに配備します。

- [ファイル] － [ファイルをエクスポート] を選択します。
- *c:¥tutorial¥evb* に移動し、*tutcustomer.usm* をダブルクリックします。

レッスン 3：サンプル・コードの作成

この章では、データベースに接続し、データベース内をナビゲーションし、データベース内のデータを操作するための、eMbedded Visual Basic コードを作成するプロセスを解説します。

このレッスンには、アプリケーションを Adaptive Server Anywhere データベースと同期するための指示も含まれています。レッスンのこの部分はオプションで、SQL Anywhere Studio を必要とします。

データベースとの接続のためのコードの作成

このアプリケーションでは、Form_Load イベントの間にデータベースに接続します。一般のモジュールを使用しても、データベースに接続できます。

この例は、データベースに接続するのに ULConnectionParms オブジェクトを使用します。接続文字列も使用できます。

参照情報については、「ULConnectionParms クラス」122 ページを参照してください。

❖ Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。

1. フォームをダブルクリックして、[コード] ウィンドウを開きます。
2. 必要な Ultra Light オブジェクトを宣言します。

   フォームの [一般] 領域に、次のコードを入力します。

```
Dim DatabaseMgr As ULDatabaseManager
Dim Connection As ULConnection
Dim CustomerTable As ULTable
Dim colID, colFirstName, colLastName As ULColumn
```

3. データベースに接続するためにのコードを Form_Load イベントに追加します。

下のコードでは、CreateObject メソッドを使用して、最初のデータベース・マネージャ・オブジェクトを作成します。データベース・マネージャは、ULConnectionParms1 オブジェクトで指定されるデータベースへの接続を開こうとします。データベースが存在しない場合は、指定されたスキーマを使用して新しいデータベースを作成します。

```
Sub Form_Load()
   ' declare variables
        Dim open_parms As String
   Dim schema_parms As String
   ' enable error handling
   On Error Resume Next
   ' specify the schema and database file
   ' locations
   open_parms = ";ce_file=¥Program
Files¥tutorial¥tutCustomer.udb"
   schema_parms = open_parms & ";ce_schema=¥Program
Files¥tutorial¥tutCustomer.usm"
   ' create a database manager and try to
   ' connect to an existing database
   Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")
   Set Connection =
DatabaseMgr.OpenConnection(open_parms)
   ' if the database does not exist, create one
   If Err.Number = UlSQLCode.ulSQLE_NOERROR Then
     MsgBox "Connected to an existing database."
   ElseIf Err.Number =
UlSQLCode.ulSQLE_ULTRALITE_DATABASE_NOT_FOUND Then
     Err.Clear
     Set Connection =
DatabaseMgr.CreateDatabase(schema_parms)
     If Err.Number <> 0 Then
       MsgBox Err.Description
     Else
       MsgBox "Connected to a new database"
     End If
   End If
 End Sub
```

4 [終了] ボタンをクリックされたときに、アプリケーションを終了して接続を閉じるコードを追加します。

```
Sub btnDone_Click()
     Connection.Close
     End
 End Sub
```

5　アプリケーションを実行します。

- [実行] ー [実行] を選択します。
- 最初のメッセージ・ボックスの後に、フォームがロードされます。
- 右上にある [OK] ボタンをクリックしてアプリケーションを終了します。

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。

❖ テーブルを開くには、次の手順に従います。

1　テーブルを初期化して最初のローに移動するためのコードを記述します。

このコードは、データベースの customer テーブルを CustomerTable 変数に割り当てます。Open を呼び出すとテーブルが開き、テーブル・データの読み込みや操作ができます。このコードは、アプリケーションをテーブル内にある最初のローの前に置きます。

次のコードを、Form_Load イベントの End Sub 命令の直前に挿入します。

```
Set CustomerTable = Connection.GetTable("customer")
 CustomerTable.Open
 If Err.Number <> ULSQLCode.ulSQLE_NOERROR Then
     MsgBox Err.Description
 End If
```

2　新しいプロシージャ DisplayCurrentRow を作成し、次のように実装します。

テーブルにローが存在しない場合は、次のプロシージャによって、アプリケーションは空のコントロールを表示します。ローが存在する場合は、データベースの現在のローの各カラムに格納された値を表示します。

```
Private Sub DisplayCurrentRow()
   If CustomerTable.RowCount = 0 Then
     txtFname.Text = ""
     txtLname.Text = ""
     txtCity.Text = ""
     txtPhone.Text = ""
     lblID.Caption = ""
   Else
     lblID.Caption =
CustomerTable.Columns("ID").Value
     txtFname.Text =
CustomerTable.Columns("Fname").Value
     txtLname.Text =
CustomerTable.Columns("Lname").Value
     txtCity.Text =
CustomerTable.Columns("City").Value
     txtPhone.Text =
CustomerTable.Columns("Phone").Value
   End If
 End Sub
```

3 Form_Activate オブジェクトから DisplayCurrentRow を呼び出します。この関数を呼び出すと、アプリケーションの起動時にフィールドが更新されます。

```
Private Sub Form_Activate()
     DisplayCurrentRow
End Sub
```

❖ テーブルにローを挿入するには、次の手順に従います。

1 [挿入] ボタンを実装するためのコードを記述します。

次のプロシージャでは、InsertBegin を呼び出すと、アプリケーションが挿入モードになり、ローのすべての値がデフォルトに設定されます。たとえば、ID カラムは、次のオートインクリメント値を受け取ります。カラム値が設定されると、新しいローが挿入されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnInsert_Click()
   Dim fname As String
   Dim lname As String
   Dim city As String
   Dim phone As String
   fname = txtFname.Text
   lname = txtLname.Text
   city = txtCity.Text
   phone = txtPhone.Text
   CustomerTable.InsertBegin
   CustomerTable.Columns("Fname").Value = fname
   CustomerTable.Columns("Lname").Value = lname
   If Len(city) > 0 Then
     CustomerTable.Columns("City").Value = city
   End If
   If Len(phone) > 0 Then
      CustomerTable.Columns("Phone").Value = phone
   End If
   CustomerTable.Insert
   CustomerTable.MoveLast
   DisplayCurrentRow
End Sub
```

2　アプリケーションを実行します。

最初のメッセージ・ボックスの後にフォームが表示されます。

3　2 つのローをデータベースに挿入します。

- 先頭のテキスト・ボックスに名前 Jane を入力し、2 つめのテキスト・ボックスに姓 Doe を入力します。[挿入] をクリックします。

  テーブルに、これらの値を持つローが追加されます。アプリケーションはテーブルの最後のローに移動し、そのローを表示します。ラベルには、Ultra Light がローに割り当てた ID カラムの自動的にインクリメントされた値が表示されます。

- 先頭のテキスト・ボックスに名前 John を入力し、2 つめのテキスト・ボックスに姓 Smith を入力します。[挿入] をクリックします。

4　[OK] をクリックしてプログラムを終了します。

❖ テーブルのローを移動するには、次の手順に従います。

1　[次へ] と [前へ] の各ボタンを実装するためのコードを記述します。

次のプロシージャをフォームに追加します。

```
Private Sub btnNext_Click()
     If Not CustomerTable.MoveNext Then
         CustomerTable.MoveLast
     End If
     DisplayCurrentRow
 End Sub
 Private Sub btnPrevious_Click()
     If Not CustomerTable.MovePrevious Then
         CustomerTable.MoveFirst
     End If
     DisplayCurrentRow
 End Sub
```

2　アプリケーションを実行します。

最初にフォームが表示されると、現在位置が最初のローの前にあるため、コントロールは空です。

フォームが表示されたら、[次へ] と [前へ] をクリックして、テーブルのローの間を移動します。

❖ テーブルのローを更新、削除するには、次の手順に従います。

1　[更新] ボタンを実装するためのコードを記述します。

下のコードでは、UpdateBegin を呼び出すと、アプリケーションが更新モードに設定されます。Update が呼び出されると、カラム値が更新された後にロー自体が更新されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnUpdate_Click()
     Dim fname As String
     Dim lname As String
     Dim city As String
     Dim phone As String

     fname = txtFname.Text
     lname = txtLname.Text
     city = txtCity.Text
     phone = txtPhone.Text
     CustomerTable.UpdateBegin
     CustomerTable.Columns("Fname").Value = _
         fname
     CustomerTable.Columns("Lname").Value = _
         lname
     If Len(city) > 0 Then
         CustomerTable.Columns("City").Value = _
         city
     End If
     If Len(phone) > 0 Then
        CustomerTable.Columns("Phone").Value = _
         phone
     End If
     CustomerTable.Update
     DisplayCurrentRow
     Exit Sub
 End Sub
```

2　[削除] ボタンを実装するためのコードを記述します。

下のコードでは、Delete を呼び出すと、アプリケーションの現在のローが削除されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnDelete_Click()
     If CustomerTable.RowCount = 0 Then
         Exit Sub
     End If
     CustomerTable.Delete
     CustomerTable.MoveRelative 0
     DisplayCurrentRow
 End Sub
```

3　アプリケーションを実行します。

同期のためのコードの記述

次の手順は、同期を実装します。同期には、SQL Anywhere Studio が必要です。チュートリアルのこの部分はオプションです。

❖ [同期] ボタンのコードを記述するには、次の手順に従います。

- [同期] ボタンを実装するためのコードを記述します。

  下のコードでは、ULSyncParms オブジェクトに同期パラメータが含まれています。たとえば、ULSyncParms.UserName プロパティは、Mobile Link が起動したら新しいユーザを追加することを指定します。ULSyncParms.SendColumnNames プロパティは、カラム名が Mobile Link に送信されることを指定し、アップロード・スクリプトとダウンロード・スクリプトを生成できるようにします。

  次のプロシージャをフォームに追加します。

```
Private Sub btnSync_Click()
     Dim parms As ULSyncParms
     Dim result As ULSyncResult
     On Error Resume Next
     Set parms = Connection.SyncParms
     Set result = Connection.SyncResult
     parms.UserName = "ULevbUser"
     parms.Stream = ULStreamType.ulTCPIP
     parms.Version = "ul_default"
     parms.SendColumnNames = True
     Connection.Synchronize (False)
     If Err.Number <> UlSQLCode.ulSQLE_NOERROR Then
         MsgBox result.StreamErrorCode
     End If
 End Sub
```

アプリケーションの同期

ASA 9.0 サンプル・データベースの Customer テーブルは、作成した Ultra Light データベースの **customer** テーブルとカラムが一致しています。次の手順は、データベースを ASA 9.0 サンプル・データベースと同期します。

❖ アプリケーションを同期するには、次の手順に従います。

1. コマンド・プロンプトから、次のコマンド・ラインを実行して、Mobile Link 同期サーバを起動します。

   ```
   dbmlsrv9 -c "dsn=ASA 9.0 Sample" -v+ -zu+ -za
   ```

   `-zu+` と `-za` コマンド・ライン・オプションによって、ユーザの追加と同期スクリプトの生成が自動的に行われます。これらのオプションの詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバのオプション」を参照してください。

2. Ultra Light アプリケーションを起動します。

3. テーブルのすべてのローを削除します。

   テーブルにローがあると、ASA 9.0 サンプル・データベースの Customer テーブルに、ローがアップロードされます。

4. アプリケーションを同期します。

   [同期] をクリックします。

   Mobile Link 同期サーバのウィンドウでは、同期の進行状況が表示されます。

5. 同期が完了したら、[次へ] と [前へ] をクリックしてテーブルのローの間を移動します。

レッスン 4：デバイスへの配備

手動でも、アプリケーション・インストール・ウィザードを使用しても、デバイスにアプリケーションを配備できます。以下の項では 2 つの手順について説明します。

リモート・デバイスへの手動での配備

次の手順は、アプリケーションを Windows CE デバイスに手動で配備します。

❖ デバイスに Ultra Light ActiveX コントロールを追加するには、次の手順に従います。

1 Windows CE コントロール・マネージャを起動します。

[ツール] ー [リモート ツール] ー [コントロール マネージャ] を選択します。

2 左ウィンドウ枠で、デバイスのタイプをダブルクリックし、デバイスを選択します。デバイスは接続している必要があります。

右ウィンドウ枠に、選択したデバイスで利用可能なコントロールが表示されます。

3 [コントロール] ー [新しいコントロールを追加] を選択します。

4 SQL Anywhere インストール環境で、プラットフォーム固有の次のサブディレクトリにある、Ultra Light ActiveX コントロールの適切なバージョンを検索します。

- **ARM** *ultralite¥UltraLiteActiveX¥ce¥arm¥uldo9.dll*
- **MIPS** *ultralite¥UltraLiteActiveX¥ce¥mips¥uldo9.dll*

5 [OK] をクリックします。

または、DLL をデバイスの *¥Windows* ディレクトリにコピーし、*regsvrce.exe* を使用してその DLL を登録してもかまいません。

❖ データベース・スキーマ・ファイルをデバイスに配備するには、次の手順に従います。

1　Windows CE のファイル・ビューアを起動します。

eMbedded Visual Basic で、[ツール] ― [リモート ツール] ― [ファイル ビューア] を選択します。

2　ファイル・ビューアが自動的にデバイスに接続しない場合は、[接続] ― [接続を追加] を選択して、お使いのデバイスを選択します。

3　アプリケーションを格納するフォルダを作成します。

- ファイル・ビューアで、[Program Files] フォルダを開きます。
- [ファイル] ― [新規フォルダ] を選択します。
- *tutorial* という名前のフォルダを作成します。このフォルダにアプリケーション・ファイルを格納します。
- *tutorial* をクリックして、このフォルダに移動します。

4　スキーマ・ファイルをデバイスに配備します。

- [ファイル] ― [ファイルをエクスポート] を選択します。
- *c:¥tutorial¥evb* に移動し、*tutcustomer.usm* をダブルクリックします。

❖ アプリケーションをリモート・デバイスに配備するには、次の手順に従います。

1　[ファイル] ― [作成] を選択します。

2 *c:¥tutorial¥evb* に移動します。プロジェクトに Tutorial という名前を付けます。[OK] をクリックします。

3 *.vb* ファイルをデバイスに配備します。

- ファイル・ビューアで *Program Files¥tutorial* に移動します。
- [ファイル] ー [ファイルをエクスポート] を選択します。
- *c:¥tutorial¥evb* に移動し、*Tutorial.vb* をダブルクリックします。

Program Files¥tutorial に移動し、*Tutorial.vb* を起動すれば、リモート・デバイスでサンプル・アプリケーションを起動できます。

アプリケーション・インストール・ウィザードを使用してのリモート・デバイスへの配備

次の手順は、アプリケーション・インストール・ウィザードを使用して、アプリケーションをモバイル・デバイスに配備します。アプリケーション・インストール・ウィザードは、実行ファイルと、配備に必要なファイルを含むキャビネット (*.cab*) ファイルを作成します。

❖ アプリケーション・インストール・ウィザードを使用して配備するには、次の手順に従います。

1 アプリケーション・インストール・ウィザードを使用する前に、プロジェクトを完成し実行して、ランタイム・エラーが発生しないことを確認します。

2 [ファイル] ー [作成] を選択します。

3 *c:¥tutorial¥evb* に移動します。プロジェクトに Tutorial という名前を付けます。[OK] をクリックします。

4 [ツール] ー [リモート ツール] ー [アプリケーション インストール] ウィザードを選択します。

[アプリケーション インストール] ウィザードが表示されます。[次へ] をクリックします。

5　*c:¥tutorial¥evb¥Tutorial.ebp* に移動します。[次へ] をクリックします。

6　*c:¥tutorial¥evb¥Tutorial.vb* に移動します。[次へ] をクリックします。

7　*c:¥tutorial¥evb¥* に移動します。[次へ] をクリックします。

8　アプリケーションがサポートするプロセッサを選択します。複数のプロセッサを選択すると、それぞれのプロセッサは別々の *.cab* ファイルに含まれます。[次へ] をクリックします。

9　[iAnywhere Solutions, ActiveX for UltraLite] を選択します。アプリケーションをインストールすると、Ultra Light ActiveX コントロールは自動的に登録されます。[次へ] をクリックします。

10　スキーマ・ファイルをデバイスに配備します。

- [追加] をクリックします。
- *c:¥tutorial¥tutcustomer.usm* に移動します。
- [開く] をクリックします。
- システム・ファイルかどうかをたずねるプロンプトが表示される場合は、[いいえ] をクリックします。
- そのデバイスのランタイム・ライブラリをインストール環境に含めたい場合は、[Include Device Runtimes in Cab file] をクリックします。デバイスのランタイム・ライブラリは必ずしも必要ありません。含めると、*.cab* ファイルが非常に大きくなる場合があります。アプリケーションが現在インストールされているランタイム・ライブラリで動作する場合は、このオプションをチェックする必要はありません。
- [次へ] をクリックします。

11 各フィールドに **tutorial** と入力します。[次へ] をクリックします。

12 [Create Install] をクリックし、*.cab* ファイルとセットアップ実行プログラムを作成します。

ウィザードは、出力ファイルを格納するのに指定したディレクトリに、フォルダ *CD1* を作成します。*CD1* には、アプリケーションを配布するのに必要なファイルがすべて含まれます。

13 [終了] をクリックします。

14 デスクトップ・コンピュータで *c:¥tutorial¥evb¥CD1¥setup.exe* を実行し、デスクトップ・コンピュータに接続したデバイスにアプリケーションをインストールします。

Program Files¥tutorial に移動し、*Tutorial.vb* を起動すれば、デバイスでアプリケーションを起動できます。

まとめ

学習の成果

このチュートリアルでは、以下の作業を行いました。

- データベース・スキーマの作成
- Ultra Light ActiveX アプリケーションの作成
- Ultra Light リモート・データベースと Adaptive Server Anywhere 統合データベースの同期
- Ultra Light ActiveX アプリケーションの開発プロセスに関する知識の取得

その他のサンプル

その他のサンプル・アプリケーションとユーティリティについては、iAnywhere CodeXchange を参照してください。

第 4 章

チュートリアル：Pocket IE 用の Ultra Light ActiveX アプリケーション

この章の内容

この章は、Pocket Internet Explorer 用の Ultra Light ActiveX アプリケーションを構築するプロセスについて理解していただくためのチュートリアルです。そのアプリケーションは、HTML ページに埋め込まれた JScript を使用して、Ultra Light ActiveX パッケージにアクセスします。

eMbedded Visual Basic を使用した Ultra Light ActiveX アプリケーションの例については、「チュートリアル：Ultra Light ActiveX アプリケーションのサンプル」53 ページを参照してください。

概要

このチュートリアルは、Ultra Light ActiveX アプリケーションを構築するプロセスを説明します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、デスクトップ・コンピュータで稼働するデータベースと同期します。

このチュートリアルでは、CustDB サンプル・アプリケーションを説明します。このアプリケーションは、Pocket Internet Explorer (Pocket IE) から使用される Ultra Light ActiveX の機能を説明します。このアプリケーションは、Windows CE デバイス上で動作します。

CustDB サンプル・アプリケーションは、高機能顧客アプリケーションです。CustDB サンプル・アプリケーションを使用して、Ultra Light ActiveX アプリケーションを開発するときに必要な、多くの方法の実行例を紹介します。

CustDB サンプル・アプリケーションは、JScript と HTML で記述されています。このアプリケーションのコードは、SQL Anywhere 9 インストール環境の *Samples¥UltraLiteActiveX¥pie* サブディレクトリにあります。

注意
このアプリケーションは、フレームを使用するので、Pocket IE 1.1 以降のバージョンでしか動作しません。

所要時間

チュートリアルはおよそ 50 分で終了します。

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。

- JScript と Pocket Internet Explorer に精通している
    - JScript アプリケーションの作成、テスト、トラブルシューティングができる

このチュートリアルの同期の項では、次のことが必要です。

- コマンド・ライン・オプションとパラメータを使用できる

目的

このチュートリアルの目的は、Ultra Light ActiveX 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：Ultra Light ActiveX パッケージのインストール

JScript 開発用の Ultra Light コンポーネントは、Ultra Light ActiveX です。次のプロシージャは、リモート・デバイスに Ultra Light ActiveX コントロールを登録します。

❖ Ultra Light ActiveX コントロールを登録するには、次の手順に従います。

1. eMbedded Visual Basic 3.0 を起動します。

   [スタート] ー [プログラム] ー [Microsoft eMbedded Visual Tools] ー [eMbedded Visual Basic 3.0] を選択します。

   [新しいプロジェクト] ウィンドウが表示されます。

2. [キャンセル] をクリックします。

3. [ツール] ー [リモート ツール] ー [コントロール マネージャ] を選択します。

4. 左ウィンドウ枠で、デバイスのタイプに対応するフォルダを開きます。デバイスを選択します。

   コントロール・マネージャがデバイスに接続します。

5. [コントロール] ー [新しいコントロールを追加] を選択します。

6. SQL Anywhere 9 インストール環境のサブディレクトリにある、プラットフォーム固有の次のファイルの 1 つを検索します。

   - **ARM** *UltraLite¥UltraLiteActiveX¥ce¥arm¥uldo9.dll*
   - **MIPS** *UltraLite¥UltraLiteActiveX¥ce¥mips¥uldo9.dll*
   - **エミュレータ** *UltraLite¥UltraLiteActiveX¥ce¥386¥uldo9.dll*

7　[開く] をクリックします。

ULConnectionParms クラスと ULDatabaseManager クラスがデバイスに追加されます。

または、デバイスに適した DLL を *¥Windows¥uldo9.dll* にコピーし、*regsvrce.exe* を使用してその DLL を登録できます。お使いのデバイス上に *regsvrce.exe* がない場合、Microsoft Windows CE SDK からデバイスにコピーできます。

レッスン 2：デバイスへの配備

次のプロシージャは、埋め込まれた JScript を含む HTML ファイルをリモート・デバイスにコピーします。または、Web サーバとインターネット接続を使用して、サーバまたはハード・ドライブから、リモートでファイルにアクセスできます。

❖ HTML ファイルをデバイスにコピーするには、次の手順に従います。

1 ファイル・ビューアを起動します。

eMbedded Visual Basic 3.0 から、[ツール] ー [リモート ツール] ー [ファイル ビューア] を選択します。

2 左ウィンドウ枠で、デバイスのタイプに対応するフォルダを開きます。デバイスを選択します。

ファイル・ビューアがデバイスに接続します。

3 [ファイル] ー [フォルダの新規作成] を選択します。デバイスのルートに *pie* という名前のフォルダを作成します。

4 [ファイル] ー [ファイルのエクスポート] を選択します。

SQL Anywhere 9 インストール環境の *Samples¥UltraLiteActiveX¥pie* サブディレクトリの内容を、デバイスの *pie* ディレクトリにコピーします。

レッスン 3 ：Ultra Light データベース・スキーマの作成と配備

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、パブリケーションとそれらの間のすべての関係を記述します。

データベース・スキーマは、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、作成できます。次のプロシージャは、ulinit を使用して、Adaptive Server Anywhere データベースに基づいたデータベース・スキーマを作成します。

❖ データベース・スキーマを作成するには、次の手順に従います。

1. コンピュータに、このチュートリアル用のディレクトリを作成します。

   チュートリアルの後半では、このディレクトリが *c:¥tutorial¥pie* であることを前提に説明します。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、*c:¥tutorial¥pie* の代わりにそのディレクトリを使用してください。

2. SQL Anywhere 9 インストール環境の *Samples¥UltraLiteActiveX¥pie* サブディレクトリの内容を、*c:¥tutorial¥pie* ディレクトリにコピーします。

3. 統合データベースを作成します。

   コマンド・プロンプトを開き、*c:¥tutorial¥pie* に移動します。次のコマンドを実行します。

   ```
   dbinit custdbsrv.db
   ```

4. Mobile Link 同期サーバを起動します。

   次のコマンドを実行します。

   ```
   runml.bat
   ```

5 ulinit ユーティリティを使用して Ultra Light スキーマを作成します。

ulinit ユーティリティの詳細については、『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。

次のコマンドを実行します。

```
makeschemas.bat
```

❖ データベース・スキーマをデバイスに配備するには、次の手順に従います。

1 ファイル・ビューアを起動します。

eMbedded Visual Basic 3.0 から、[ツール] ー [リモート ツール] ー [ファイル ビューア] を選択します。

2 ファイル・ビューアがデバイスに接続します。

- [接続] ー [接続の追加] を選択します。
- デバイスのタイプに対応するフォルダを開きます。
- デバイスを選択します。

3 [ファイル] ー [フォルダの新規作成] を選択します。*UltraLiteDB* という名前のフォルダを作成します。

4 [ファイル] ー [ファイルのエクスポート] を選択します。

5 *c:¥tutorial¥pie¥ul_custdb.usm* を検索します。[開く] をクリックすると、デバイスにファイルがエクスポートされます。

レッスン 4：フォーム・インタフェースの作成

フォーム・インタフェースは、アプリケーションの視覚的な要素で構成されています。次のプロシージャは、*main.htm* のフォーム・インタフェースを説明します。新しいテキスト・ファイルを作成して、自分でフォーム・インタフェースを作成できます。

❖ プロジェクトにコントロールを追加するには、次の手順に従います。

1　テキスト・エディタで *c:¥tutorial¥pie¥main.htm* を開きます。

2　*main.htm* の次の行までスクロールします。

```
<br>
Customer: <INPUT TYPE="text" NAME="txt_Custname"
SIZE=15 MAXLENGTH=40>
```

これらの行は、下の表で指定されたテキスト・ボックスを作成します。

キャプション	入力タイプ	名前
Customer:	Text	txt_Custname
Product:	Text	txt_Prodname
Quantity:	Text	txt_Quant
Prince:	Text	txt_Price
Discount:	Text	txt_Discount
Status:	Text	txt_Status
Notes:	Text	txt_Notes

3　*main.htm* の次の行までスクロールします。

```
<br>
 <INPUT NAME="b_Previous" TYPE="BUTTON" VALUE="
<Back " onClick="MoveBack()">
```

これらの行は、下の表で示されたテキスト・ボックスを作成します。

入力タイプ	名前	値	OnClick
Button	b_Previous	< Back	MoveBack()
Button	b_Next	Next >	MoveForward()
Button	b_Approve	Approve...	ApproveOrder()
Button	b_Deny	Deny...	DenyOrder()
Button	b_Add	Add...	addNewOrder()
Button	b_Delete	Delete	deleteOrder()
Button	b_Synchronize	Synchronize	syncData()

レッスン 5：JScript サンプル・コードの作成

この章では、データベースに接続し、データベース内をナビゲーションし、データベース内のデータを操作するための、JScript コードを作成するプロセスを解説します。

このレッスンの同期の項では、SQL Anywhere Studio が必要です。

Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。

Pocket IE では、HTML ページに埋め込まれた JScript を使用して、Ultra Light ActiveX にアクセスします。接続スクリプトを含むページがロードされると、新しいデータベース接続が作成されます。ブラウザが他のページに移動すると、前のページで作成されたオブジェクトはすべて廃棄されます。したがって、そのページがアンロードされると、データベース接続は失われます。

フォームが変更されるたびに、データベース接続などのアプリケーションの状態が失われると、速度が遅く負荷が高くなります。これを回避するため、Pocket IE アプリケーションが、HTML フレームを使用して、アプリケーションの状態を管理することをおすすめします。データベース接続は、FRAMESET 要素により維持されます。Pocket IE は、フレームセット内に少なくとも 2 つのフレームを必要とし、フレーム間に 3 ピクセルの境界を描きます。

CustDB のサンプルでは、*topline.htm* で定義される最小フレームは、画面の一番上にアプリケーション名を表示するプレースホルダです。画面の残りの部分を使って、アプリケーションのほかのページが表示されます。

フォームをフレームの内外にスワップするのに、JScript 関数は **document.location.replace** メソッドを使用できます。たとえば、次のコード・フラグメントは、現在の接続を閉じて、接続フォームを返します。

```
<SCRIPT>
 function exitApp() {
   if ( top.Connection != null ) {
     top.Connection.Close();
   }
   document.location.replace("connect.htm");
 }
 </SCRIPT>
 <INPUT NAME="b_Done" TYPE="BUTTON" VALUE="Done"
onClick="exitApp()">
```

次のプロシージャは、Ultra Light データベースに接続します。Ultra Light データベースへの接続の詳細については、「Ultra Light データベースへの接続」13 ページを参照してください。

❖ 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 Pocket Internet Explorer を起動します。

2 *login.htm* を開きます。

3 ハイパーリンクを選択します。

ハイパーリンクは、*frames.htm* にリンクします。

frames.htm のコードは、新しいデータベース・マネージャ・オブジェクトを作成し、それを使用して、接続文字列が指定し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、新しいデータベースを作成します。

詳細については、「Ultra Light データベースへの接続」13 ページを参照してください。

```
<SCRIPT LANGUAGE="JScript">
 var Connection;       // UltraLite Connection
 var DatabaseMgr;  // UltraLite Database Manager
 var CS;         // Current State object

 // Initialize connection. Create database if not
already present.
 function initDatabase() {
   var udb, usm;
```

```
   udb = "file_name=¥¥UltraLiteDB¥¥ul_custapi.udb";
   usm =
";schema_file=¥¥UltraLiteDB¥¥ul_custdb.usm";
   CS = new CurrState( 0 );
   DatabaseMgr = new ActiveXObject(
"UltraLite.ULDatabaseManager" );
   var bval = DatabaseMgr.ErrorResume;
   DatabaseMgr.ErrorResume = true;
   Connection = DatabaseMgr.OpenConnection( udb );
   if ( DatabaseMgr.LastErrorCode != 0 ) {
       Connection = DatabaseMgr.CreateDatabase( udb
+ usm );
   }
 }
 // Initialize the database.
 initDatabase();
 </SCRIPT>
```

4　スクリプトが、従業員 ID の入力を要求します。デフォルト値 50 を受け入れます。

データベースを同期するためのコードの作成

製品のリスト (リモート・データベースでは変更できない) と最初の注文のリストを取得するには、データベースを同期する必要があります。

次のプロシージャはデータベースを同期します。

❖ サンプル・データベースを同期するには、次の手順に従います。

1　[Synchronize] を選択します。

下のコードが実行されます。

ULConnection.Synchronize メソッドは、1 つのパラメータ *show-progress* を取ります。JScript アプリケーションでは、Pocket Internet Explorer がメッセージの表示を許可しないので、*show-progress* は false に設定されています。

Ultra Light データベースの同期の詳細については、「データの同期」47 ページを参照してください。

```
function syncData() {
   var conn = top.Connection;
   var parms = conn.SyncParms;
   // Set Sync Params
   parms.Stream = 2;  // ulTCPIP = 2, ulHTTP = 1,
ulHTTPS = 3
   parms.StreamParams = "";
   parms.UserName = "50";       // m_EmpIDStr;
   parms.Version = "custdb 9.0";
   conn.ErrorResume = true;
   conn.Synchronize(false);
   conn.ErrorResume = false;
   if ( conn.LastErrorCode != 0 ) {
       SetStatus("Sync failed: " +
conn.LastErrorDescription );
       return;
   }
   conn.Commit();         // Save updates.
   SetStatus("Synchronized");
   SkipToValidOrder();
   SetOrderData();
 }
```

2 同期が完了すると、画面の下部に [Synchronize] という語が表示されます。

注文情報を表示するためのコードの作成

main.htm がロードされると、次の関数が実行され、注文情報を取得、表示します。

GetTable メソッドの詳細については、「GetTable メソッド」116 ページを参照してください。

```
function OpenGetOrder() {
  var conn = top.Connection;
  var cs = top.CS;
  var empid;
  if ( conn == null ) {
     alert("Not yet connected!");
   return;
```

```
  }
  conn.AutoCommit = false;
  var table =
conn.GetTable("ULIdentifyEmployee_nosync");
  table.Open();
  if ( table.RowCount == 0 ) {
     empid = window.prompt("Enter employee ID #: ",
"50");
    table.InsertBegin();
     table.Columns("emp_id").value = empid;
    table.Insert();
     conn.Commit();
  }

table.MoveFirst();
  cs.SetEmployeeID( table.Columns("emp_id").value );
  table.Close();
  ProductList = conn.GetTable("ULProduct");
  ProductList.Open();
  CustomerList = conn.GetTable("ULCustomer");
  CustomerList.Open();
  OrderList = conn.GetTable("ULOrder");
  OrderList.Open();
  SetOrderData();
  SkipToValidOrder();
 }

function UpdateForm() {
  var cs = top.CS;
  if ( cs.GetNoOrder() ) {
     txt_Custname.value = "";
     txt_Prodname.value = "";
     txt_Quant.value = "";
     txt_Price.value = "";
     txt_Discount.value = "";
     txt_Status.value = "";
     txt_Notes.value = "";
   return;
  }

txt_Custname.value = cs.GetCustName();
  txt_Prodname.value = cs.GetProdName();
  txt_Quant.value = cs.GetQuantity();
  txt_Price.value = cs.GetPrice();
  txt_Discount.value = cs.GetDiscount();
  txt_Status.value = cs.GetStatus();
  txt_Notes.value = cs.GetNotes();
```

```
 if ( cs.FirstOrder() == cs.GetOrderID() ) {
    SetStatus("First Order");
 } else if ( cs.LastOrder() == cs.GetOrderID() ) {
    SetStatus("Last Order");
 }
}
```

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。

❖ 注文のリストをスクロールするには、次の手順に従います。

- [Back] または [Next] を選択します。

  以下の関数のいずれかが実行されます。

```
function MoveBack() {
   var cs = top.CS;
   var posn = cs.GetBookmark();
   if ( MoveOrder( -1 ) ) {
       cs.SetBookmark(posn - 1);
       UpdateForm();
   }
 }
 function MoveNext() {
   var cs = top.CS;
   var posn = cs.GetBookmark();
   if ( MoveOrder(1) ) {
     cs.SetBookmark(posn + 1);
     UpdateForm();
   }
 }
```

❖ データベースに注文を追加するには、次の手順に従います。

1 [Add] を選択します。

  次のスクリプトが実行され、*main.htm* をフレーム内の *add.htm* で置き換えます。

```
function addNewOrder() {
     document.location.replace("add.htm");
 }
```

2　フォームに必要な値を入力します。[OK] をクリックします。

次のスクリプトが実行されます。

ローの挿入の詳細については、「ローの挿入、更新、削除」35 ページを参照してください。

```
function addCustomer() {
   var custName = window.prompt("Enter new customer
name: ", "");
   if ( custName == "" ) return;
   var conn = top.Connection;
   var custlist = conn.GetTable( "ULCustomer" )
   custlist.Open("ULCustomerName");
   custlist.FindBegin();
   custlist.Columns("cust_name").value = custName;
   if ( custlist.FindFirst() ) {  // already
present. Done.
     custlist.Close();
       return;
   }

var custid = NextCustomerID();
   if ( custid == -1 ) {
       alert("No more customer IDs, cannot add.
Replenish ULCustomerIDPool");
       custlist.Close();
       return;
   }
   custlist.InsertBegin();
   custlist.Columns("cust_id").value = custid;
   custlist.Columns("cust_name").Value = custName;
   custlist.Insert();
   custlist.Close();
   conn.Commit();
  // Only way to show new customer is to reload this
form!
   document.location.reload();
 }
```

❖ 注文を承認または拒否するには、次の手順に従います。

1 [Approve] または [Deny] を選択します。

次のスクリプトの 1 つが実行され、フレームに *approve.htm* または *deny.htm* をロードします。

```
function ApproveOrder() {
   var cs = top.CS;
   document.location.replace("approve.htm");
 }
 function DenyOrder() {
   document.location.replace("deny.htm");
 }
```

2 ノートを入力して [OK] をクリックします。

注文を承認する場合は、次のコードが実行されます。注文を拒否する場合、コードは同じですが、ステータスが Denied に設定されます。

ローの更新の詳細については、「ローの挿入、更新、削除」35 ページを参照してください。

```
function OK_Approve() {
   var cs = top.CS;
   var conn = top.Connection;
   var notes = txt_Notes.value;
   var order = conn.GetTable("ULOrder");
   order.Open();
   order.FindBegin();
   order.Columns("order_id").value =
cs.GetOrderID();
   if ( order.FindFirst() ) {
     order.UpdateBegin();
     order.Columns("status").value = "Approved";
     order.Columns("notes").value = notes;
     order.Update();
     if (conn.LastErrorCode != 0 ) {
       alert("Failed: " + conn.LastErrorDescription
);
     }
     conn.Commit();
   } else {
```

```
    alert("Cannot find order " + cs.GetOrderID() );
  }
  order.Close();
  document.location.replace("main.htm");
}
```

第 5 章

Ultra Light ActiveX API リファレンス

この章の内容

この章では、UltraLite ActiveX API について説明します。これは、Ultra Light データベースを使用するアプリケーション用の eMbedded Visual Basic または JScript のコードを作成するためのクラスとメソッドのセットです。各トピックには、個別のクラス、メソッド、定数、または列挙に関する情報が含まれます。リファレンスはクラス別に編成されており、そのクラスに関連するメソッドが続けて示されます。

IULColumns コレクション

カラムに関するメタデータを提供する ULColumn オブジェクトのコレクション。

プロパティ

プロトタイプ	説明
Count as long (読み込み専用)	このコレクション内のカラム数を取得する。
Item (Index) as ULColumn (読み込み専用)	このコレクションから値を取得する。インデックスは、1 ～ count までの数字またはカラム名です。

例

eMbedded Visual Basic の For Each 文または JScript の for ... in 文を使用して、IULColumns コレクションのカラムを列挙できます。

```
' eMbedded Visual Basic
Dim col As ULColumn
For Each col In table.Columns
    If col.IsNull Then
    MsgBox col.Schema.Name & "is null"
    End If
Next

// JScript
var col : UltraLite.ULColumn;
var collection = table.Columns;
for ( col in collection ) {
  if ( col.IsNull ) {
    alert ( col.Schema.Name + "is null" );
  }
}
```

IULIndexSchemas コレクション

ULIndexSchema の情報を提供する ULIndexSchema オブジェクトのコレクション。

プロパティ

プロトタイプ	説明
Count as long (読み込み専用)	このコレクション内のインデックス数を取得する。
Item (Index) as ULIndexSchema (読み込み専用)	このコレクションからインデックスを取得する。項目には、1 から始まるインデックスを使用してインデックスが付けられています。インデックスは、1 ～ count までの数字です。

例

eMbedded Visual Basic の For Each 文または JScript の for ... in 文を使用して、テーブルのインデックスを列挙できます。

```
' eMbedded Visual Basic
Dim index As ULIndexSchema
For Each index In TableSchema.Indexes
    'use index
Next

// JScript
var index : UltraLite.ULIndexSchema;
var collection = TableSchema.Indexes;
for ( index in collection ) {
  ' use index
}
```

IULPublicationSchemas コレクション

ULPublicationSchema に関する情報を提供する ULPublicationSchema オブジェクトのコレクション。

プロパティ

プロトタイプ	説明
Count as long (読み込み専用)	このコレクション内のパブリケーション数を取得する。
Item (Index) as ULPublicationSchema (読み込み専用)	このコレクションからパブリケーションを取得する。

例

eMbedded Visual Basic の For Each 文または JScript の for ... in 文を使用して、すべてのパブリケーションを列挙できます。

```
' eMbedded Visual Basic
Dim pub As ULPublicationSchema
For Each pub In connection.schema.publications
    'use pub
Next

// JScript
var ps : UltraLite.ULPublicationSchema;
var collection = connection.schema.publications;
for ( pub in collection ) {
  ' use pub
}
```

ULAuthStatusCode 列挙

ULAuthStatusCode は、ULSyncResult オブジェクトで使用される auth_status 同期パラメータです。

定数	値
ulAuthUnknown	0
ulAuthValid	1000
ulAuthValidButExpiresSoon	2000
ulAuthExpired	3000
ulAuthInvalid	4000
ulAuthInUse	5000

ULColumn クラス

ULColumn オブジェクトでは、データベース内のテーブルの値を取得、設定できます。各カラム・オブジェクトは、テーブル内の特定の値を表します。ローは、ULTable オブジェクトによって決定されます。

Ultra Light データベースの型を Visual Basic の型に変換する場合の注意

Ultra Light では、データベースのカラムのデータ型を Visual Basic のデータ型に変換しようとします。変換が成功しなかった場合は、ulSQLE_CONVERSION_ERROR が発生します。

テーブル・オブジェクトの詳細については、「ULTable クラス」176 ページを参照してください。

プロパティ

プロトタイプ	説明
IsNull As Boolean (読み込み専用)	カラム値が NULL かどうかを示す。
Schema As ULColumnSchema (読み込み専用)	カラムのスキーマを表すオブジェクトを取得する。
Value As Variant	現在のローのこのカラムのデータ値を Variant として取得または設定する。

AppendByteChunk メソッド

プロトタイプ

```
AppendByteChunk( _
    byteArray, _
    [ chunkSize ] _
) As Boolean
Member of UltraliteActiveX.ULColumn
```

説明　カラムの型が ulTypeLongBinary または TypeBinary の場合、バイトをローのカラムに追加します。

パラメータ　**byteArray**　追加するバイトの配列。

chunkSize　追加するバイトの数。指定されていない場合は、byteArray の長さが使用されます。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。

エラー・セット　**ulSQLE_CONVERSION_ERROR**　カラムのデータ型が LONG BINARY または BINARY でない場合、エラーが発生します。

例　次の例では、データが edata カラムに追加されます。

```
' eMbedded Visual Basic
Dim data (512) As Byte
' ...
table.Columns("edata").AppendByteChunk(data)

// JScript
var data = new Array();
// ...
table.Columns("edata").AppendByteChunk(data);
```

AppendStringChunk メソッド

プロトタイプ　**AppendStringChunk(** *chunk* **)**
Member of **UltraLiteActiveX.ULColumn**

説明　カラムの型が TypeLongString または TypeString の場合、カラムに文字列を追加します。

パラメータ　**data**　テーブル内の既存の文字列に追加する文字列。

エラー・セット　**ulSQLE_CONVERSION_ERROR**　カラムのデータ型が VARCHAR でない場合、エラーが発生します。

GetByteChunk メソッド

プロトタイプ

```
GetByteChunk (_
    offset As Long, _
    pByteArray, _
    [ chunkSize ] _
) As Long
Member of UltraliteActiveX.ULColumn
```

説明

渡されたバッファ (配列) に、カラム内のバイナリ・データを入れます。BLOB に適しています。

パラメータ

offset 基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、ulSQLE_INVALID_PARAMETER エラーが発生します。

pByteArray variant 型。配列データは、参照で配列として渡されます。

chunkSize Long 型として表現されたバイト配列を表すオプションのパラメータ。

検査結果

読み込まれたバイト数。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合は、エラーが発生します。

カラムのデータ型が LONG BINARY でオフセットが 1 より小さい、またはデータ長が 0 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。

```
' eMbedded Visual Basic
Dim data (512) As Byte
' ...
table.Columns.Item("edata").GetByteChunk(0,data)
```

```
// JScript
 var data = new Array();
 // ...
 table.Columns.Item("edata").GetByteChunk(0, data);
```

GetStringChunk メソッド

プロトタイプ

GetStringChunk(_
offset As Long, _
pStringObj, _
[*chunkSize*] _
) As Long
Member of **UltraliteActiveX.ULColumn**

説明

渡された文字列に、カラムからのバイナリ・データを挿入します。LONG VARCHAR 型のカラムに適しています。

パラメータ

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

pStringObj　挿入される文字列の配列。この variant 型は参照で渡されます。

chunkSize　取得する文字数を表す、オプションのパラメータ。

検査結果

コピーされた文字数。末尾の null 終了文字を挿入するための領域が残され、この文字は長さに含まれていません。

エラー

ulSQLE_CONVERSION_ERROR　カラムのデータ型が TypeString でも TypeLongString でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。

src_offset が 1 より小さいか、文字列の長さが 0 より小さい場合も、エラーが発生します。

例

```
' eMbedded Visual Basic
 Dim cd As ULColumn
 Dim S As Strong
 Dim l, offset As Long
 S=String(512, vbNulChar)
 offset=0
 Do
     l=cd.GetStringChunk(offset, S, 512)
     If l=0 then Exit Do
     'use string ins
 Loop

// JScript
 var cd;
 var s;
 var l, offset;
 l = 0;
 While (!l) {
   l = cd.GetStringChunk(offset, s, 512);
 }
```

SetByteChunk メソッド

プロトタイプ

SetByteChunk (_
ByteArray, _
[*length*] _
) As Boolean
Member of **UltraliteActiveX.ULColumn**

説明

データベース内のカラムの値を、データ・フィールド内のバイト配列に設定します。

パラメータ

ByteArray variant 型のバイト配列。

length 配列の長さ。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER　データ長が 0 より小さいか 64K より大きい場合は、エラーが発生します。

例

次の例では、edata はカラム名で、データ変数の最初の 232 バイトはデータベースに格納されています。

```
' eMbedded Visual Basic
 Dim data (1 to 512) As Byte
 ' ...
 table.Columns.Item("edata").SetByteChunk(data,232)

// JScript
 var data = new Array();
 // ...
 table.Columns.Item("edata").SetByteChunk(data,232);
```

SetToDefault メソッド

プロトタイプ

SetToDefault( )
Member of **UltraliteActiveX.IColumn**

説明

現在のカラムを、データベース・スキーマで定義されているデフォルト値に設定します。たとえば、オートインクリメント・カラムの場合は、次に使用可能な値が割り当てられます。

ULColumnSchema クラス

ULColumnSchema オブジェクトを使用して、テーブル内のメタデータ、つまりカラムの属性を取得できます。属性は、テーブルのデータに依存しません。

プロパティ

プロトタイプ	説明
AutoIncrement As Boolean (読み込み専用)	このカラムのデフォルトがオートインクリメント値かどうかを示す。オートインクリメントの場合は true です。
DefaultValue As String (読み込み専用)	ローの挿入時に値が指定されていない場合に使用される値を取得する。
GlobalAutoIncrement As Boolean (読み込み専用)	この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値かどうかを示す。
ID As Long(読み込み専用)	カラムの ID を取得する。
Name As String (読み込み専用)	カラム名を取得する。
Nullable As Boolean (読み込み専用)	カラムが NULL を許可するかどうかを示す。
OptimalIndex As ULIndexSchema (読み込み専用)	最初のカラムとしてこのカラムを持つインデックスを取得する。
Precision As Integer (読み込み専用)	型が ulTypeNumeric である場合は、カラムの精度値を取得する。
Scale As Long (読み込み専用)	カラムの位取りの値を取得する。
Size As Long (読み込み専用)	バイナリ、数値、char データ型のカラム・サイズを取得する。

プロトタイプ	説明
SQLType As ULSQLType (読み込み専用)	作成時にカラムに割り当てられた SQL 型を取得する。

ULConnection クラス

ULConnection オブジェクトは、Ultra Light データベース接続を表します。これは、テーブルなどのデータベース・オブジェクトを取得するメソッドと同期のメソッドを提供します。

プロパティ

ULConnection のプロパティは次のとおりです。

プロトタイプ	説明
AutoCommit As Boolean	AutoCommit 値を示す。true の場合、データを変更するとその直後にすべてのデータ変更がコミットされます。それ以外の場合、Commit を呼び出すまで、変更はデータベースにコミットされません。デフォルトでは、このプロパティは True です。
CollationName As String (読み込み専用)	データベース文字セットとソート順を取得する。
DatabaseID As Long	グローバル・オートインクリメントのカラムの開始値を決定するデータベース ID を取得または設定する。 データベース ID が設定されていない場合、値は -1 です。
DatabaseManager As ULDatabaseManager (読み込み専用)	所有しているデータベース・マネージャ・オブジェクトを取得する。
DatabaseNew As Boolean (読み込み専用)	この接続に対してデータベースが新しく作成されたかどうかを示す。
ErrorResume As Boolean	エラー処理方法を示す。
GlobalAutoIncrementUsage As Long (読み込み専用)	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 (%) を取得する。

プロトタイプ	説明
IsCaseSensitive As Boolean (読み込み専用)	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示す。
LastErrorCode As SQLCodeConstants	最後のエラー番号を取得し、以前のエラー・コードをクリアできるようにする。
LastErrorDescription As String (読み込み専用)	最後のエラーの説明を取得する。
LastIdentity As Long (読み込み専用)	デフォルトでオートインクリメントまたはグローバル・オートインクリメントに設定されたカラムに最後に挿入された値を取得する。
OpenParms As String (読み込み専用)	データベースへの接続を開くために使用される文字列を取得する。
Schema As ULDatabaseSchema (読み込み専用)	データベースの定義を表す ULDatabaseSchema オブジェクトを取得する。
SQLErrorOffset As Integer (読み込み専用)	PrepareStatement がエラーを発生した場合、エラーが記録された SQL 文で 1 から始まるオフセットを示す。この値が 0 以下の場合、オフセット情報はありません。
SyncParms As ULSyncParms (読み込み専用)	同期パラメータ・オブジェクトを取得する。
SyncResult As ULSyncResult (読み込み専用)	最後の同期の結果を取得する。

CancelSynchronize メソッド

プロトタイプ

CancelSynchronize( )
Member of **UltraliteActiveX.ULConnection**

説明

同期処理中に呼び出された場合、このメソッドは同期をキャンセルします。ユーザがこのメソッドを呼び出せるのは、いずれかの同期イベントの実行時だけです。

ChangeEncryptionKey メソッド

プロトタイプ

ChangeEncryptionKey(*newkey*As String **)**
Member of **UltraliteActiveX.ULConnection**

説明

指定されたキーを使用してデータベースを暗号化します。

パラメータ

newkey データベースの新しい暗号化キーの値。

例

CreateDatabaseWithParms を呼び出し parms オブジェクトを渡す場合、EncryptionKey に値が指定されていれば、データベースは暗号化されて作成されます。暗号化キーを変更する別の方法は、新しい暗号化キーを ULConnection オブジェクトで指定することです。この例では "apricot" がキーです。

```
Connection.ChangeEncryptionKey( "apricot" )
```

OpenConnectionWithParms のようなデータベースへの接続は、データベースが暗号化されてから、EncryptionKey プロパティにも *apricot* を指定します。そうしないと、接続は失敗します。

Close メソッド

プロトタイプ

Close( )
Member of **UltraliteActiveX.ULConnection**

説明

データベースとの接続を閉じます。この接続で ULConnection オブジェクトなどのデータベース・オブジェクトのメソッドは、このメソッドを呼び出す前に呼び出します。接続が明示的に閉じられていない場合は、アプリケーションの終了時に暗黙的に閉じられます。

Commit メソッド

プロトタイプ

Commit( )
Member of **UltraliteActiveX.ULConnection**

説明

未処理の変更をデータベースにコミットします。AutoCommit が false の場合にのみ役立ちます。

詳細については、ULConnection 「プロパティ」112 ページの Autocommit を参照してください。

CountUploadRows メソッド

プロトタイプ

```
CountUploadRows(
    [ mask As Long = 0 ], _
    [ threshold As Long = -1 ] _
) As Long
Member of UltraliteActiveX.ULConnection
```

説明

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。

パラメータ

mask　チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。指定されていない場合は、値は 0 になります。

threshold　カウントするローの最大数を表す、オプションのパラメータ。最大数がないことを指定するには、-1 を使用します。指定されていない場合、この値は -1 です。

検査結果

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。

GetNewUUID メソッド

プロトタイプ

```
GetNewUUID( ) As String
    Member of UltraliteActiveX.ULConnection
```

説明

新しいユニバーサル・ユニーク識別子を返します。値は *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列であり、通常はデータ型 UNIQUEIDENTIFIER のカラムに格納されます。

検査結果

呼び出すたびに新しい UUID が返されます。

GetTable メソッド

プロトタイプ

GetTable(*name* As String **)** As ULTable
Member of **UltraliteActiveX.ULConnection**

説明

指定されたテーブルの **ULTable** オブジェクトを返します。テーブルを開いてから、テーブルのデータを読み込みます。

パラメータ

name 調べるテーブルの名前。

検査結果

ULTable オブジェクトを返します。

GrantConnectTo メソッド

プロトタイプ

GrantConnectTo(
userid As String, _
password As String _
)
Member of **UltraliteActiveX.ULConnection**

説明

指定したパスワードを使用してデータベースに接続するパーミッションを特定のユーザに付与します。

パラメータ

userid 接続する権限を付与されるユーザ ID。

password ユーザ ID が接続するのに指定するパスワード。

LastDownloadTime メソッド

プロトタイプ

LastDownloadTime([*mask* As Long = 0] As Date
Member of **UltraliteActiveX.ULConnection**

説明

パブリケーションの最終ダウンロード時間を返します。

パラメータ

mask チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。このパラメータが省略されている場合は、0 が使用されます。

検査結果　日付形式で表した最後のダウンロード時刻。

PrepareStatement メソッド

プロトタイプ　**PrepareStatement(** *sqlStatement* As String **)** As ULPreparedStatement
Member of **UltraliteActiveX.ULConnection**

説明　SQL 文の実行を準備します。

パラメータ　**sqlStatement**　準備する SQL 文。

検査結果　ULPreparedStatement を返します。文を準備するときに問題が起こった場合は、エラーが発生します。エラーが発生した文へのオフセットは、SQLErrorOffset プロパティから判断できます。

ResetLastDownloadTime メソッド

プロトタイプ　**ResetLastDownloadTime(** [*mask* As Long] **)**
Member of **UltraliteActiveX.ULConnection**

説明　マスクで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リセットします。

パラメータ　**mask**　リセットするパブリケーションのマスク。デフォルトは 0 で、すべてのパブリケーションを指定します。

RevokeConnectFrom メソッド

プロトタイプ　**RevokeConnectFrom(** *userID* As String **)**
Member of **UltraliteActiveX.ULConnection**

説明　指定されたユーザがデータベースに接続できないようにします。

パラメータ　**userid**　接続する権限を取り消されるユーザ ID。

Rollback メソッド

プロトタイプ

Rollback( )
Member of **UltraliteActiveX.ULConnection**

説明

未処理の変更をデータベースにロールバックします。AutoCommit が false の場合にのみ役立ちます。

RollbackPartialDownload メソッド

失敗した同期から変更をロールバックします。

プロトタイプ

RollbackPartialDownload ( )
Member of **UltraliteActiveX.ULConnection**

説明

同期のダウンロード時に通信エラーが発生した場合、Ultra Light では、ダウンロードした変更を適用し、同期が中断した時点から同期を再開することができます。ダウンロードした変更が不要な場合 (ダウンロードが中断した時点での再開を望まない場合)、RollbackPartialDownload を使用することで、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します。

参照

- 『Mobile Link 管理ガイド』> 「失敗したダウンロードの再開」
- 『Mobile Link クライアント』> 「Keep Partial Download 同期パラメータ」
- 『Mobile Link クライアント』> 「Partial Download Retained 同期パラメータ」
- 『Mobile Link クライアント』> 「Resume Partial Download 同期パラメータ」

StartSynchronizationDelete メソッド

プロトタイプ

StartSynchronizationDelete( )
Member of **UltraliteActiveX.ULConnection**

説明

StartSynchronizationDelete が呼び出されると、すべての削除操作がもう一度同期されます。

StopSynchronizationDelete メソッド

プロトタイプ

StopSynchronizationDelete( )
Member of **UltraliteActiveX.ULConnection**

説明

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。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使用すると便利です。

StringToUUID メソッド

プロトタイプ

StringToUUID(*s_uuid* As String **)**
Member of **UltraliteActiveX.ULConnection**

説明

形式が xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の文字列として表されるユニバーサル・ユニーク識別子を 16 バイトのバイト配列に変換します。

新しいデータベースでは不要

バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、StringToUUID 関数は不要です。

詳細については、『ASA SQL リファレンス・マニュアル』>「UNIQUEIDENTIFIER データ型 [バイナリ]」を参照してください。

パラメータ

s_uuid　文字列として渡されるユニバーサル・ユニーク識別子。GetNewUUID を使用して新しい文字列 UUID を取得できます。

例

次の例は、文字列形式の UUID 0a141e28-323c-4650-5a64-6e78828c96a0 をバイナリ配列に変換します。

```
' eMbedded Visual Basic
 Dim buff(1 to 16) As Byte
 conn.StringToUUID( "0a141e28-323c-4650-5a64-
6e78828c96a0", VarPtr(buff(1)) )
```

Synchronize メソッド

プロトタイプ

Synchronize([*show-progress* As Boolean] **)**
Member of **UltraliteActiveX.ULConnection**

説明

Mobile Link を使用して統合データベースの同期を行います。この関数は、同期が完了するまで戻りませんが、接続が WithEvents で宣言されている場合はイベントを通知できます。

パラメータ

show-progress 値が true または false である、オプションのパラメータ。発生した同期の進行状況を表示するには true を設定します。デフォルトは false です。

UUIDToString メソッド

プロトタイプ

UUIDToString(*buffer_16_bytes* **)** As String
Member of **UltraliteActiveX.ULConnection**

説明

16 バイトのバッファへの VarPtr を予想します。このバッファを、*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の形式の文字列に変換します。バッファは、(1 to 16) As Byte (つまり、16 バイトの配列) として宣言します。Visual Basic は、このバッファの境界をチェックできないため、大きさが十分でない場合は、アプリケーションがメモリを上書きすることがあります。

新しいデータベースでは不要
バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、UUIDToString 関数は不要です。

詳細については、『ASA SQL リファレンス・マニュアル』>「UNIQUEIDENTIFIER データ型 [バイナリ]」を参照してください。

パラメータ　**buffer_16_bytes**　UUID を含む 16 バイトの配列。

検査結果　呼び出しごとに、*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の形式の文字列が返されます。

ULConnectionParms クラス

ULConnectionParms オブジェクトを使用すると、ユーザ ID、パスワード、スキーマ・ファイル、デスクトップ上のファイルなど、接続を指定する多くのパラメータの設定ができます。

プロパティ

ULConnectionParms クラスは、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。

Ultra Light ActiveX では、コードの中で ULConnectionParms オブジェクトを使用して、接続プロパティを設定できます。
ULConnectionParms オブジェクトは、
ULDatabaseManager.CreateDatabaseWithParms メソッドと
ULDatabaseManager.OpenConnectionWithParms メソッドとともに使用します。

注意
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。

これらのパラメータの意味の詳細については、『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。

プロトタイプ	説明
AdditionalParms As String (読み込み／書き込み)	セミコロンで区切られた**名前** = *値*の組み合わせとして指定される追加のパラメータ。 『Ultra Light データベース・ユーザーズ・ガイド』>「AdditionalParms 接続パラメータ」を参照してください。

プロトタイプ	説明
CacheSize As String (読み込み／書き込み)	キャッシュのサイズ。CacheSize 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。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。 『Ultra Light データベース・ユーザーズ・ガイド』>「CacheSize 接続パラメータ」を参照してください。
ConnectionName As String (読み込み／書き込み)	接続の名前。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。 『Ultra Light データベース・ユーザーズ・ガイド』>「ConnectionName 接続パラメータ」を参照してください。
DatabaseOnCE As String (読み込み／書き込み)	PocketPC に配備されるデータベースのファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnCE 接続パラメータ」を参照してください。
DatabaseOnDesktop As String (読み込み／書き込み)	開発中のデータベースのファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnDesktop 接続パラメータ」を参照してください。

プロトタイプ	説明
EncryptionKey As String (読み込み／書き込み)	データベースを暗号化するためのキー。OpenConnection と OpenConnectionWithParms は、データベース作成中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。キーは以下の条件を満たしていることが推奨されます。 1. 任意の長い文字列を選択します。 2. キーを見破られる可能性を減らすため、多様な数字、文字、特殊文字を使用した文字列を選択します。 『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。
ParmsUsed As String (読み込み専用)	ULDatabaseManager が使用するパラメータ。デバッグに便利です。
Password As String (読み込み／書き込み)	認証ユーザのパスワード。データベースは最初、1 つの認証されたユーザ・パスワード *SQL* を使用して、作成されます。データベースの大文字と小文字を区別しない場合は、パスワードの大文字と小文字を区別せず、データベースの大文字と小文字を区別する場合は、パスワードの大文字と小文字も区別します。デフォルト値は *SQL* です。 『Ultra Light データベース・ユーザーズ・ガイド』>「Password 接続パラメータ」を参照してください。
ReserveSize As Integer (読み込み／書き込み)	Ultra Light の永続的データの保管に使用するため予約するファイル・システム領域の大きさ。 『Ultra Light データベース・ユーザーズ・ガイド』>「Reserve Size 接続パラメータ」を参照してください。

プロトタイプ	説明
SchemaOnCE As String (読み込み／書き込み)	PocketPC に配備されるスキーマ・ファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnCE 接続パラメータ」を参照してください。
SchemaOnDesktop As String (読み込み／書き込み)	開発中のスキーマ・ファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnDesktop 接続パラメータ」を参照してください。
UserID As String (読み込み／書き込み)	データベースで認証されたユーザ。データベースは最初、1 つの認証されたユーザ DBA を使用して、作成されます。データベースの大文字と小文字を区別しない場合は、UserID の大文字と小文字を区別せず、データベースの大文字と小文字を区別する場合は、UserID の大文字と小文字も区別します。デフォルト値は *DBA* です。 『Ultra Light データベース・ユーザーズ・ガイド』>「User ID 接続パラメータ」を参照してください。

ULDatabaseManager クラス

ULDatabaseManager クラスを使用して、接続とデータベースを管理します。アプリケーションは、このオブジェクトのインスタンスを 1 つだけ持ちます。データベースを作成し、そのデータベースへの接続を確立することは、Ultra Light の使用に必要な最初の手順です。CreateDatabaseWithParms、OpenConnectionWithParms、DropDatabaseWithParms を使用し、正しく接続してからデータ操作言語でデータベースを操作するように、コードに検査制約を含めることをおすすめします。

ULConnectionParms の使用の有無

データベースへの接続を作成、開始、削除するには、2 種類のメソッドがあります。WithParms メソッドと ULConnectionParms オブジェクトを使用しないメソッドです。WithParms メソッドは、ULConnectionParms オブジェクトを使用するメソッドで、簡単かつ正確に接続パラメータを操作できます。ULConnectionParms オブジェクトを使用しないメソッドでは、接続文字列を正しく作成し、その接続文字列を CreateDatabase、OpenConnection、または DropDatabase メソッドで使用することが要求されます。

プロパティ

ULDatabaseManager のプロパティは次のとおりです。

プロトタイプ	説明
ErrorResume As Boolean	エラー処理方法。デフォルトは false です。true に設定すると、ULDatabaseManager メソッドが失敗したときに、エラーが発生しません。
LastErrorCode As SQLCodeConstants	最後のエラー番号を取得し、以前のエラー・コードをクリアできるようにする。
Version As String (読み込み専用)	Ultra Light コンポーネントのバージョン文字列を取得する。

CreateDatabase メソッド

CreateDatabase は、新規データベースを作成し、そのデータベースへの接続を返します。

プロトタイプ

CreateDatabase(*parms* As String **)** As ULConnection
Member of **UltraliteActiveX.ULDatabaseManager**

説明

新規データベースを作成し、そのデータベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。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。既存のデータベースのスキーマを変更するには、ULDatabaseSchema ApplyFile メソッドを使用します。

警告
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。

ApplyFile の詳細については、「ULDatabaseSchema クラス」137 ページと「ApplyFile メソッド」138 ページを参照してください。

パラメータ

parms　セミコロンで区切られたデータベース作成パラメータのリスト。

接続パラメータの詳細については、『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。

検査結果

新しく作成された Ultra Light データベースへの接続を返します。

例

次の例では、CreateObject を使用して新しいデータベースを作成し、開きます。

```
' eMbedded Visual Basic
open_parms = "file_name=¥tutCustomer.udb"
schema_parms = open_parms & ";" &
"schema_name=¥tutCustomer.usm"
Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")
Set Connection =
DatabaseMgr.CreateDatabase(schema_parms)
```

```
// JScript
 open_parms = "file_name=¥¥tutCustomer.udb";
 schema parms = open_parms + ";" +
"schema_name=¥¥tutCustomer.usm"
 DatabaseMgr = new
ActiveXObject("UltraLite.ULDatabaseManager")
 Set Connection =
DatabaseMgr.CreateDatabase(schema_parms);
```

次の例は、イベントを持つ ULDatabaseManager の作成方法を示します。この方法は、同期の進行状況を表示する場合に使用します。

この機能は、eMbedded Visual Basic でのみ利用できます。

同期の進行状況の表示の詳細については、『Ultra Light ActiveX ユーザーズ・ガイド』> 「同期の進行状況のモニタ」を参照してください。

```
'eMbedded Visual Basic
 Set DBMgr =
CreateObjectWithEvents("UltraLite.ULDatabaseManager",
"UL_")
```

接続パラメータの詳細については、「OpenConnection メソッド」134 ページを参照してください。

CreateDatabaseWithParms メソッド

CreateDatabaseWithParms は、接続パラメータ・オブジェクトを使用して、新しいデータベースを作成し、そのデータベースへの接続を返します。

プロトタイプ

CreateDatabaseWithParms(*parms* As ULConnectionParms **)**
As ULConnection
Member of **UltraliteActiveX.ULDatabaseManager**

説明

新規データベースを作成し、そのデータベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。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。既存のデータベースのスキーマを変更するには、**ULDatabaseSchema.ApplyFileWithParms** メソッドを使用します。

警告
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。

パラメータ

parms　一連の接続パラメータを格納する ULConnectionParms オブジェクト。

検査結果

新しく作成された Ultra Light データベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。

例

次の例は、フォーム上に ULConnectionParms オブジェクトを配置しており、それに **LoginParms** という名前を付け、[Connection parms] プロパティ・ウィンドウでデータベースのロケーションとスキーマのロケーションを指定していることを前提としています。

次の例では、CreateDatabaseWithParms を使用して新しいデータベースを作成し、開きます。

```
' eMbedded Visual Basic
 ' Use CreateObject in to get an instance of the
ULDatabaseManager object
 Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")
 ' Use CreateObject to get an instance of the
ULConnectionParms object
 Dim LoginParms As ULConnectionParms
 Set LoginParms =
CreateObject("UltraLite.ULConnectionParms")
 LoginParms.DatabaseOnCE = "/tutorial/tutorial.udb"
 LoginParms.SchemaOnCE = "/tutorial/tutorial.usm"
 ' Drop the existing database and create a new database
 Call DatabaseMgr.DropDatabaseWithParms( LoginParms )
 Set Connection =
DatabaseMgr.CreateDatabaseWithParms(LoginParms)

// JScript
 ' get an instance of the ULDatabaseManager object
 DatabaseMgr = new
ActiveXObject("UltraLite.ULDatabaseManager");
 var LoginParms;
 LoginParms = new
```

```
ActiveXObject("UltraLite.ULConnectionParms");
 LoginParms.DatabaseOnCE = "//tutorial//tutorial.udb";
 LoginParms.SchemaOnCE = "//tutorial//tutorial.usm";
 ' Drop the existing database and create a new database
 DatabaseMgr.DropDatabaseWithParms( LoginParms );
 Connection = DatabaseMgr.CreateDatabaseWithParms(
LoginParms );
```

DropDatabase メソッド

DropDatabase メソッドは、データベース・ファイルを削除します。

プロトタイプ **DropDatabase(** *parms* As String **)**
Member of **UltraliteActiveX.ULDatabaseManager**

説明 データベース・ファイルを削除します。データベース・ファイルの情報はすべて失われます。指定されたデータベースが存在しない場合、または DropDatabase の実行時にオープン接続が存在する場合は、失敗します。

パラメータ **parms** データベースのファイル名。

例 次の例は、データベースを削除します。

```
' eMbedded Visual Basic
 open_parms = "ce_file=¥tutCustomer.udb"
 DropDatabase(open_parms)

// JScript
 open_parms = "ce_file=¥¥tutCustomer.udb";
 DropDatabase( open_parms );
```

DropDatabaseWithParms メソッド

DropDatabaseWithParms メソッドは、データベース・ファイルを削除します。

プロトタイプ **DropDatabaseWithParms(** *parms* As ULConnectionParms **)**
Member of **UltraliteActiveX.ULDatabaseManager**

説明　データベース・ファイルを削除します。データベース・ファイルの情報はすべて失われます。

パラメータ　**parms**　重要な接続パラメータを含む ULConnectionParms オブジェクト。

例　次の例は、**LoginParms** という名前の ULConnectionParms オブジェクトを宣言しインスタンス化しており、そのオブジェクトを使用してデータベースのロケーションを指定していることを前提としています。

```
' eMbedded Visual Basic
 Call DatabaseMgr.DropDatabaseWithParms( LoginParms )

// JScript
 DatabaseMgr.DropDatabseWithParms( LoginParms );
```

OnReceive イベント

プロトタイプ

OnReceive(
nBytes As Long, _
nInserts As Long, _
nUpdates As Long, _
nDeletes As Long _
)
Member of **UltraliteActiveX.ULDatabaseManager**

説明　Mobile Link を介した、統合データベースからアプリケーションへのダウンロード情報をレポートします。このイベントは、複数回呼び出すことができます。

パラメータ　**nBytes**　受信した累積バイト数。

nInserts　リモート・アプリケーションで受信した、統合データベースからの挿入の累積回数。

nUpdates　リモート・アプリケーションで受信した、統合データベースからの更新の累積回数。

nDeletes リモート・アプリケーションで受信した、統合データベースからの削除の累積回数。

例

```
'eMbedded Visual Basic
Private Sub OnReceive(ByVal nBytes As Long, ByVal
nInserts As Long, ByVal nUpdates As Long, ByVal
nDeletes As Long)
     prLine "OnReceive " & nBytes & " bytes, " & Inserts
& " inserts, " & nUpdates & " updates, " & nDeletes & "
deletes"
End Sub
```

OnSend イベント

プロトタイプ

OnSend(
nBytes As Long,
nInserts As Long,
nUpdates As Long,
nDeletes As Long
)
Member of **UltraliteActiveX.ULDatabaseManager**

説明

Mobile Link を介した、リモート・データベースから統合データベースへのアップロード情報をレポートします。このイベントは、複数回呼び出すことができます。

パラメータ

nBytes Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した累積バイト数。

nInserts Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した挿入の累積回数。

nUpdates Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した更新の累積回数。

nDeletes Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した削除の累積回数。

例

```
' eMbedded Visual Basic
 Private Sub Connection_OnSend(ByVal nBytes As Long, _
     ByVal nInserts As Long, ByVal nUpdates As Long,  _
     ByVal nDeletes As Long)
   send_count = send_count + nBytes
   DisplaySyncStatus
 End Sub
```

OnStateChange イベント

プロトタイプ

OnStateChange(
newState As ULSyncState, _
oldState As ULSyncState _
)
Member of **UltraliteActiveX.ULDatabaseManager**

説明

このイベントは、同期ステータスが変更されるたびに呼び出されます。

パラメータ

newState　直後に開始される同期処理のステータス。

oldState　直前に完了した同期処理のステータス。

例

```
' eMbedded Visual Basic
 Private Sub _OnStateChange(ByVal newState As Long,
ByVal oldState As Long)
     prLine "OnStateChange new:" & newState & ", old: "
& oldState
 End Sub
```

OnTableChange イベント

プロトタイプ

OnTableChange(
newTableIndex As Long, _
numTables As Long _
)
Member of **UltraliteActiveX.ULDatabaseManager**

説明

このイベントは、同期処理が次のテーブルの同期を開始するたびに呼び出されます。

パラメータ

newTableIndex 現在同期が行われているテーブルのインデックス番号。この番号はテーブル ID とは異なるため、ULDatabaseSchema.GetTableName メソッドでは使用できません。

numTables 同期が行われるテーブル数。

例

```
' eMbedded Visual Basic
 Private Sub _OnTableChange(ByVal newTableIndex As
Long, ByVal numTables As Long)
     prLine "OnTableChange index:" & newTableIndex & ",
#tables=" & numTables
 End Sub
```

OpenConnection メソッド

プロトタイプ

OpenConnection(*connparms* As string **)** As ULConnection
Member of **UltraliteActiveX.ULDatabaseManager**

説明

データベースが存在する場合は、このメソッドを使用して接続を受信します。データベースが存在しない場合、または接続パラメータが無効な場合は、この呼び出しは失敗します。エラー・オブジェクトを使用して、呼び出しが失敗した理由を判別します。

この関数は、指定した Ultra Light データベースとのオープン接続を提供する ULConnection オブジェクトを返します。データベース・ファイル名は、connparms 文字列を使用して指定します。パラメータは、一連の**名前** = *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

パラメータ

connparms データベースへの接続を確立するために使用されるパラメータ。パラメータは、一連の**キーワード** = *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

検査結果

接続に成功した場合は、ULConnection オブジェクトが返されます。

例

次の例は、OpenConnection メソッドでの接続パラメータの使用方法を示します。

```
' eMbedded Visual Basic
 open_parms = "ce_file = ¥tutCustomer.udb"
 Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")
 Set Connection =
DatabaseMgr.OpenConnection(open_parms)

// JScript
 open_parms = "ce_file = ¥¥tutCustomer.udb";
 DatabaseMgr = new
ActiveXObject("UltraLite.ULDatabaseManager");
 Connection = DatabaseMgr.OpenConnection(open_parms);
```

OpenConnectionWithParms メソッド

プロトタイプ

OpenConnectionWithParms(*connparms* As ULConnectionParms**)**
As ULConnection
Member of **UltraliteActiveX.ULDatabaseManager**

説明

データベースが存在する場合は、このメソッドを使用して接続を受信します。データベースが存在しない場合、または接続パラメータが無効な場合は、この呼び出しは失敗します。エラー・オブジェクトを使用して、呼び出しが失敗した理由を判別します。

この関数は、指定した Ultra Light データベースとのオープン接続を提供する ULConnection オブジェクトを返します。データベース・ファイル名は、connparms オブジェクトを使用して指定します。パラメータは、一連の**名前** = *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

パラメータ

connparms　この接続を定義するパラメータ。

検査結果

接続に成功した場合は、ULConnection オブジェクトが返されます。

例

次の例は、ULConnectionParms オブジェクト **LoginParms** を作成し、データベースのロケーションとスキーマのロケーションを指定していることを前提としています。

```
' eMbedded Visual Basic
 Set DatabaseMgr =
CreateObject("UltraLite.ULDatabaseManager")
 Set Connection =
DatabaseMgr.OpenConnection(LoginParms)

// JScript
 DatabaseMgr = new
ActiveXObject("UltraLite.ULDatabaseManager");
 Connection = DatabaseMgr.OpenConnection(LoginParms);
```

ULDatabaseSchema クラス

ULDatabaseSchema オブジェクトを使用すると、接続先データベースの属性を取得できます。

プロパティ

ULDatabaseSchema のプロパティは次のとおりです。

プロトタイプ	説明
DateFormat As String (読み込み専用)	データベースから取り出した日付のフォーマットを取得する。デフォルトは、'YYYY-MM-DD' です。取得される日付のフォーマットは、スキーマ・ファイルの作成時に使用したフォーマットに対応しています。
DateOrder As String (読み込み専用)	日付フォーマットの解釈を示す。有効な値は 'MDY'、'YMD'、または 'DMY' です。
NearestCentury As String (読み込み専用)	文字列から日付への変換で、2 桁の年の解釈を示す。これは、ロールオーバ・ポイントとして動作する数値です。この値より小さい 2 桁の年は 20yy に変換され、この値以上の年は 19yy に変換されます。デフォルトは 50 です。
Precision As String (読み込み専用)	10 進法計算での結果の最大桁数を取得する。
Publications As IULPublicationSchemas (読み込み専用)	パブリケーション・スキーマ・オブジェクトのコレクションを取得する。
Scale (読み込み専用)	データベースの数値の位取りを取得する。

プロトタイプ	説明
Signature As Variant (読み込み専用)	データベースのシグニチャを取得する。これは、データベース・スキーマを表す内部識別子です。
TableCount As Long (読み込み専用)	接続したデータベース内のテーブル数を取得する。
TimeFormat As String (読み込み専用)	データベースから取り出した時刻のフォーマットを取得する。
TimestampFormat As String (読み込み専用)	データベースから取り出したタイムスタンプのフォーマットを取得する。
TimestampIncrement (読み込み専用)	データベースのタイムスタンプの増分を取得する。

ApplyFile メソッド

プロトタイプ

ApplyFile(*parms* As String **)** As Boolean
Member of **UltraliteActiveX.ULDatabaseSchema**

説明

このデータベースのスキーマを変更します。*Parms* は、データベースに適用しているスキーマ・ファイルを指します。このメソッドは、既存のデータベース構造を変更する場合にのみ役立ちます。

警告
ApplyFile は非常に安全ですが、次のような場合にデータの消失が起こることがあります。(1) カラムが削除された場合、(2) カラムのデータ型が互換性のない型に変更された場合、(3) Ultra Light 9.0 で ApplyFile を使用して 8.0.2 のデータベースを更新した場合です。

パラメータ

parms 変更するデータベース・スキーマを含むファイル。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

例

```
DatabaseSchema.ApplyFile( _
 "schema_file=MySchemaFile.usm;palm_schema=MySchema" )
```

ApplyFileWithParms メソッド

プロトタイプ

ApplyFileWithParms(*parms* As ULConnectionParms) As Boolean
Member of **UltraliteActiveX.ULDatabaseSchema**

説明

パラメータ・オブジェクト *Parms* を使用して、このデータベースのスキーマを更新します。オブジェクトは、データベースに適用するスキーマ・ファイルを指しています。このメソッドは、既存のデータベース構造を変更する場合にのみ役立ちます。

警告
ApplyFile は非常に安全ですが、ApplyFileWithParms を使用すると、次のような場合にデータの消失が起こることがあります。(1) カラムが削除された場合、(2) カラムのデータ型が互換性のない型に変更された場合、(3) Ultra Light 9.0 で ApplyFile を使用して 8.0.2 のデータベースを更新した場合です。

パラメータ

parms　適用するスキーマ・ファイルを示すオブジェクト。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

ULIndexSchema クラス

ULIndexSchema オブジェクトを使用すると、インデックスの属性を取得できます。インデックスは順序付きカラムのセットであり、これを使用してテーブル内のデータをソートします。インデックスはおもに、1 つ以上のカラムでテーブルのデータを並べ替えるために使用されます。

インデックスに外部キーを使用することができ、この場合、データベースの参照整合性が維持されます。

プロパティ

プロトタイプ	説明
ColumnCount As Long (読み込み専用)	インデックス内のカラム数を取得する。
ColumnName(position As Long) As String (読み込み専用)	インデックスの位置にあるカラムの名前を取得する。
ForeignKey As Boolean (読み込み専用)	外部キーかどうかを示す。
IsColumnDescending(position As Long) As Boolean (読み込み専用)	カラムが降順であるかどうかを示す。昇順の場合は false です。
Name As String (読み込み専用)	インデックスの名前を取得する。
PrimaryKey As Boolean (読み込み専用)	このテーブルのプライマリ・キーかどうかを取得する。
ReferencedIndexName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するインデックスの名前を取得する。
ReferencedTableName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するテーブルの名前を取得する。
UniqueIndex As Boolean(読み込み専用)	インデックスの値がユニークである必要があるかどうかを示す。

プロトタイプ	説明
UniqueKey As Boolean (読み込み専用)	インデックスがテーブルの一意性制約かどうかを示す。True の場合は、インデックス内のカラムはユニークであり、NULL 値を使用できません。

ULPreparedStatement クラス

ULPreparedStatement は、実行の準備ができたコンパイル済みの SQL 文を表します。準備文を使用して、SQL クエリを実行できます。ULPreparedStatement を使用すれば、多くの入力パラメータを使用して、同じ文を複数回実行することもできます。準備文はコンパイル済みなので、最初の実行以後の追加については、わずかな処理で済みます。複数のローで比較的速いデータ操作言語が必要な場合は、ULPreparedStatement と Dynamic SQL を使用します。

プロパティ

プロトタイプ	説明
HasResultSet As Boolean (読み込み専用)	準備文が結果セットを生成するかどうかを示す。 文が結果セットを持つ場合は true、持たない場合は false です。 true の場合、ExecuteStatement ではなく ExecuteQuery を呼び出します。
Schema As ULResultSetSchema (読み込み専用)	文の結果を説明するスキーマを取得する。

AppendByteChunk メソッド

プロトタイプ

```
AppendByteChunk( _
    parameter_id As Long, _
    Array, _
    [ chunkSize ] _
) As Boolean
Member of UltraliteActiveX.ULPreparedStatement
```

説明

カラムの型が ulTypeLongBinary の場合、バイトのバッファをローのカラムに追加します。

パラメータ　**parameter_id**　1 から始まるパラメータ番号を設定します。

data　追加するバイトの配列。

chunkSize　追加するバイト数。指定されていない場合は、byteArray の長さが使用されます。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。

エラー・セット　**ulSQLE_CONVERSION_ERROR**　カラムのデータ型が LONG BINARY または BINARY でない場合、エラーが発生します。

AppendStringChunk メソッド

プロトタイプ

AppendStringChunk(
parameter_id As Long **,**
chunk **)**
Member of **UltraLiteActiveX.ULPreparedStatement**

説明　カラムの型が ulTypeLongString の場合、カラムに文字列を追加します。

パラメータ　**parameter_id**　1 から始まるパラメータ番号を設定します。

chunk　テーブル内の既存の文字列に追加する文字列。

エラー・セット　**ulSQLE_CONVERSION_ERROR**　カラムのデータ型が VARCHAR でない場合、エラーが発生します。

Close メソッド

プロトタイプ

Close( )
Member of **UltraLiteActiveX.ULPreparedStatement**

説明　ULPreparedStatement に関連付けられているリソースを解放します。

ExecuteQuery メソッド

プロトタイプ
ExecuteQuery( ) As ULResultSet
Member of **UltraliteActiveX.ULPreparedStatement**

説明
クエリを実行し結果セットを返します。

検査結果
ULResultSet オブジェクト。ULResultSet は、SELECT 文で要求したデータです。クエリに関する詳細については、「ULResultSetSchema クラス」153 ページを参照してください。

ExecuteStatement メソッド

プロトタイプ
ExecuteStatement( ) As Long
Member of **UltraliteActiveX.ULPreparedStatement**

説明
文を実行します。

検査結果
更新されたローの数。

SetNullParameter メソッド

プロトタイプ
SetNullParameter(*parameter_id* As Long **)**
Member of **UltraliteActiveX.ULPreparedStatement**

説明
パラメータを NL に設定します。

パラメータ
parameter_id 1 から始まるパラメータ番号を設定します。

SetParameter メソッド

プロトタイプ
SetParameter(
parameter_id As Long
val
)
Member of **UltraliteActiveX.ULPreparedStatement**

説明　実行パラメータを、渡された値に設定します。

パラメータ　**parameter_id**　1 から始まるパラメータ番号を設定します。

val　実行パラメータに使用する値。

ULPublicationSchema クラス

ULPublicationSchema オブジェクトを使用すると、パブリケーションの属性を取得できます。

プロパティ

プロトタイプ	説明
Mask As Long (読み込み専用)	パブリケーションのマスクを取得する。
Name As String (読み込み専用)	パブリケーションの名前を取得する。

ULResultSet クラス

ULResultSet オブジェクトは、SQL クエリが返したローを移動します。ULResultSet オブジェクトには、クエリが返したデータが含まれているので、INSERT、UPDATE、または DELETE のようなデータ操作言語の操作を行った後では、クエリの結果セットをリフレッシュする必要があります。このためには、ExecuteStatement を実行してから、ExecuteQuery を実行します。

プロパティ

プロトタイプ	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。最後のローの後であれば、EOF は true であり、そうでなければ false となる。
IsNull(columnID As Long) As Boolean (読み込み専用)	指定されたカラムの値が SQL NULL かどうかを示す。カラムが null の場合は true、そうでなければ IsNull は false となる。
RowCount As Long (読み込み専用)	結果セット内のロー数。
Schema As ULResultSetSchema (読み込み専用)	この結果セットのスキーマの説明。
Value(columnID As Long) (読み込み専用)	指定されたカラムの値。

Close メソッド

プロトタイプ

Close()
Member of **UltraliteActiveX.ULResultSet**

説明

このオブジェクトに関連付けられているリソースを解放します。

GetByteChunk メソッド

プロトタイプ

GetByteChunk(
index As Long, _
offset As Long, _
data, _
[*data_len* As Long] _
) As Long
Member of **UltraliteActiveX.ULResultSet**

説明

渡されたバッファ (配列) に、カラム内のバイナリ・データを入れます。BLOB に適しています。

パラメータ

index バイナリ・データを含むカラムの 1 から始まる序数。

offset 基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、SQLE_INVALID_PARAMETER エラーが発生します。64K を超えるバッファも許されます。

data バイトの配列へのポインタ。

data_len バッファ (配列) の長さ。data_len は 0 以上であることが必要です。

検査結果

読み込まれたバイト数。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合は、エラーが発生します。

カラムのデータ型が LONG BINARY で、オフセットが 1 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。

```
' eMbedded Visual Basic
Dim data (512) As Byte
...
table.Columns.Item("edata").GetByteChunk(0,data)

// JScript
var data = new Array();
// ...
table.Columns.Item("edata").GetByteChunk(0,data);
```

GetStringChunk メソッド

プロトタイプ

GetStringChunk(
index As Long, _
offset As Long, _
pStringObj, _
[*chunkSize*] **)** As Long
Member of **UltraliteActiveX.ULResultSet**

説明

渡された文字列に、カラム内のバイナリ・データを挿入します。Long Varchar に適しています。

パラメータ

index　ターゲット・カラムの 1 から始まるカラム ID。

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

pStringObj　返される文字列。この variant 型は参照で渡されます。

chunkSize　取得する文字数を表す、オプションのパラメータ。

検査結果

コピーされた文字数。末尾の null 終了文字を挿入するための領域が残され、この文字は長さに含まれていません。

BLOB データを binary または long binary のカラムから取得します。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。

オフセットが 1 より小さいか、文字列の長さが 0 より小さい場合、エラーが発生します。

MoveAfterLast メソッド

プロトタイプ

MoveAfterLast( )
Member of **UltraliteActiveX.ULResultSet**

説明

ULResultSet の最後のローの後に移動します。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveBeforeFirst メソッド

プロトタイプ

MoveBeforeFirst( )
Member of **UltraliteActiveX.ULResultSet**

説明

最初のローの前に移動します。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveFirst メソッド

プロトタイプ

MoveFirst( ) As Boolean
Member of **UltraliteActiveX.ULResultSet**

説明　最初のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveLast メソッド

プロトタイプ　**MoveLast( )**
Member of **UltraliteActiveX.ULResultSet**

説明　最後のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveNext メソッド

プロトタイプ　**MoveNext( )** As Boolean
Member of **UltraliteActiveX.ULResultSet**

説明　次のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MovePrevious メソッド

プロトタイプ　**MovePrevious( )** As Boolean
Member of **UltraliteActiveX.ULResultSet**

説明　前のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveRelative メソッド

プロトタイプ　**MoveRelative(** *index* As Long **)** As Boolean
Member of **UltraliteActiveX.ULResultSet**

説明　いくつかのローを、現在のローを基準にして相対的に移動します。結果セットでのカーソルの現在位置を基準にして、正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。

パラメータ　**index**　移動するローの数。値は、正、負、または 0 です。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

IsNull メソッド

プロトタイプ　**IsNull(** *index* As Integer **)** As Boolean
Member of **UltraliteActiveX.ULResultSet**

説明　このカラムに null 値が含まれているかどうかを示します。

パラメータ　**index**　カラムのインデックス値。

検査結果　値が Null の場合は true。

ULResultSetSchema クラス

ULResultSetSchema は、結果セットのスキーマに関する情報を提供します。

プロパティ

プロトタイプ	説明
ColumnCount As Long (読み込み専用)	結果セット内のカラム数を取得する。
ColumnName As String (読み込み専用)	結果セットに含まれるカラムの名前を取得する。
ColumnPrecision As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の精度を取得する。
ColumnScale As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の位取りを取得する。
ColumnSize As Integer (読み込み専用)	カラムのデータのサイズを取得する。
ColumnSQLType As ULSQLType (読み込み専用)	カラムの ULSQLType を取得する。
Name (column_id As Long) as String (読み込み専用)	序数 ID からカラム名が指定される場合、パブリケーションの名前を取得する。
Precision(columnID As Long) As Long (読み込み専用)	カラムの数値精度を取得する。

GetColumnID メソッド

プロトタイプ　**GetColumnID(** *col_name* As String **)** As Long
Member of **UltraliteActiveX.ULResultSetSchema**

説明　指定されたカラムのカラム id を取得します。

パラメータ **col_name** id が検索されるカラムの名前。

検査結果 GetColumnID は、指定したカラムのカラム ID を返します。

ULSQLCode 列挙

ULSQLCode 定数は、Ultra Light によってレポートされる SQL コードを示します。

エラーの説明については、『*Adaptive Server Anywhere エラー・メッセージ*』マニュアルを参照してください。

定数	値
ulSQLE_AGGREGATES_NOT_ALLOWED	-150
ulSQLE_ALIAS_NOT_UNIQUE	-830
ulSQLE_ALIAS_NOT_YET_DEFINED	-831
ulSQLE_BAD_ENCRYPTION_KEY	-840
ulSQLE_BAD_PARAM_INDEX	-689
ulSQLE_CANNOT_ACCESS_FILE	-602
ulSQLE_CANNOT_CHANGE_USER_NAME	-867
ulSQLE_CANNOT_MODIFY	-191
ulSQLE_CANNOT_EXECUTE_STMT	-111
ulSQLE_COLUMN_AMBIGUOUS	-144
ulSQLE_COLUMN_CANNOT_BE_NL	-195
ulSQLE_COLUMN_IN_INDEX	-127
ulSQLE_COLUMN_NOT_FOUND	-143
ulSQLE_COMMUNICATIONS_ERROR	-85
ulSQLE_CONNECTION_NOT_FOUND	-108
ulSQLE_CONVERSION_ERROR	-157
ulSQLE_CURSOROP_NOT_ALLOWED	-187
ulSQLE_CURSOR_ALREADY_OPEN	-172
ulSQLE_CURSOR_NOT_OPEN	-180

定数	値
ulSQLE_DATABASE_ERROR	-301
ulSQLE_DATABASE_NEW	123
ulSQLE_DATABASE_NOT_CREATED	-645
ulSQLE_DATABASE_NOT_FOUND	-83
ulSQLE_DATABASE_UPGRADE_FAILED	-672
ulSQLE_DATABASE_UPGRADE_NOT_POSSIBLE	-673
ulSQLE_DATATYPE_NOT_ALLOWED	-624
ulSQLE_DBSPACE_FL	-604
ulSQLE_DIV_ZERO_ERROR	-628
ulSQLE_DOWNLOAD_CONFLICT	-839
ulSQLE_DROP_DATABASE_FAILED	-651
ulSQLE_DYNAMIC_MEMORY_EXHAUSTED	-78
ulSQLE_ENGINE_ALREADY_RUNNING	-96
ulSQLE_ENGINE_NOT_MTIUSER	-89
ulSQLE_ERROR	-300
ulSQLE_ERROR_CALLING_FUNCTION	-622
ulSQLE_EXPRESSION_ERROR	-156
ulSQLE_IDENTIFIER_TOO_LONG	-250
ulSQLE_INDEX_NOT_FOUND	-183
ulSQLE_INDEX_NOT_UNIQUE	-196
ulSQLE_INTERRUPTED	-299
ulSQLE_INVALID_AGGREGATE_PLACEMENT	-862
ulSQLE_INVALID_FOREIGN_KEY	-194
ulSQLE_INVALID_FOREIGN_KEY_DEF	-113
ulSQLE_INVALID_GROUP_SELECT	-149

定数	値
ulSQLE_INVALID_LOGON	-103
ulSQLE_INVALID_OPTION_SETTING	-201
ulSQLE_INVALID_ORDER	-152
ulSQLE_INVALID_ORDERBY_COLUMN	-854
ulSQLE_INVALID_PARAMETER	-735
ulSQLE_INVALID_SQL_IDENTIFIER	-760
ulSQLE_INVALID_STATEMENT	-130
ulSQLE_LOCKED	-210,
ulSQLE_MEMORY_ERROR	-309
ulSQLE_METHOD_CANNOT_BE_CALLED	-669
ulSQLE_NAME_NOT_UNIQUE	-110
ulSQLE_NOERR	0
ulSQLE_NOTFOUND	100
ulSQLE_NOT_IMPLEMENTED	-134
ulSQLE_NO_CURRENT_ROW	-197
ulSQLE_NO_INDICATOR	-181
ulSQLE_OVERFLOW_ERROR	-158
ulSQLE_PERMISSION_DENIED	-121
ulSQLE_PRIMARY_KEY_NOT_UNIQUE	-193
ulSQLE_PRIMARY_KEY_VALUE_REF	-198
ulSQLE_PUBLICATION_NOT_FOUND	-280
ulSQLE_RESOURCE_GOVERNOR_EXCEEDED	-685
ulSQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE	130
ulSQLE_SERVER_SYNCHRONIZATION_ERROR	-857
ulSQLE_START_STOP_DATABASE_DENIED	-75

定数	値
ulSQLE_STATEMENT_ERROR	-132
ulSQLE_SYNTAX_ERROR	-131
ulSQLE_STRING_RIGHT_TRUNCATION	-638
ulSQLE_TABLE_HAS_PUBLICATIONS	-281
ulSQLE_TABLE_IN_USE	-214
ulSQLE_TABLE_NOT_FOUND	-141
ulSQLE_TOO_MANY_CONNECTIONS	-102
ulSQLE_TRALITE_OBJ_CLOSED	-908
ulSQLE_UNABLE_TO_CONNECT_OR_START	-764
ulSQLE_UNABLE_TO_START_DATABASE	-82
ulSQLE_UNCOMMITTED_TRANSACTIONS	-755
ulSQLE_UNKNOWN_FUNC	-148
ulSQLE_UNKNOWN_USERID	-140
ulSQLE_UNSUPPORTED_CHARACTER_SET_ERROR	-869
ulSQLE_UPLOAD_FAILED_AT_SERVER	-794
ulSQLE_WRONG_PARAMETER_COUNT	-154

ULSQLType 列挙

ULSQLType は、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リストします。

定数	Ultra Light データベースの型	値
ulTypeLong	Integer	1
ulTypeUnsignedLong	SmallInt	2
ulTypeShort	UnsignedInteger	3
ulTypeUnsignedShort	UnsignedSmallInt	4
ulTypeBig	Big	5
ulTypeUnsignedBig	UnsignedBig	6
ulTypeByte	Byte	7
ulTypeBit	Bit	8
ulTypeDateTime	Time	9
ulTypeDate	Date	10
ulTypeTime	Timestamp	11
ulTypeDouble	Double	12
ulTypeReal	Real	13
ulTypeNumeric	(Var)Binary	14
ulTypeBinary	LongBinary	15
ulTypeString	(Var)Char	16
ulTypeLongString	LongVarchar	17
ulTypeLongBinary	Numeric	18

ULStreamErrorCode 列挙

ULStreamErrorCode 定数は、同期時の通信エラーを識別します。

これらのエラーの詳細については、『ASA エラー・メッセージ』>「Mobile Link 通信エラー・メッセージ」を参照してください。

定数	値
ulStreamErrorCodeNone	0
ulStreamErrorCodeParameter	1
ulStreamErrorCodeParameterNotUint32	2
ulStreamErrorCodeParameterNotUint32Range	3
ulStreamErrorCodeParameterNotBoolean	4
ulStreamErrorCodeParameterNotHex	5
ulStreamErrorCodeMemoryAllocation	6
ulStreamErrorCodeParse	7
ulStreamErrorCodeRead	8
ulStreamErrorCodeWrite	9
ulStreamErrorCodeEndWrite	10
ulStreamErrorCodeEndRead	11
ulStreamErrorCodeNotImplemented	12
ulStreamErrorCodeWouldBlock	13
ulStreamErrorCodeGenerateRandom	14
ulStreamErrorCodeInitRandom	15
ulStreamErrorCodeSeedRandom	16
ulStreamErrorCodeCreateRandomObject	17
ulStreamErrorCodeShuttingDown	18

定数	値
ulStreamErrorCodeDequeuingConnection	19
ulStreamErrorCodeSecureCertificateRoot	20
ulStreamErrorCodeSecureCertificateCompanyName	21
ulStreamErrorCodeSecureCertificateChainLength	22
ulStreamErrorCodeSecureCertificateRef	23
ulStreamErrorCodeSecureCertificateNotTrusted	24
ulStreamErrorCodeSecureDuplicateContext	25
ulStreamErrorCodeSecureSetIo	26
ulStreamErrorCodeSecureSetIoSemantics	27
ulStreamErrorCodeSecureCertificateChainFunc	28
ulStreamErrorCodeSecureCertificateChainRef	29
ulStreamErrorCodeSecureEnableNonBlocking	30
ulStreamErrorCodeSecureSetCipherSuites	31
ulStreamErrorCodeSecureSetChainNumber	32
ulStreamErrorCodeSecureCertificateFileNotFound	33
ulStreamErrorCodeSecureReadCertificate	34
ulStreamErrorCodeSecureReadPrivateKey	35
ulStreamErrorCodeSecureSetPrivateKey	36
ulStreamErrorCodeSecureCertificateExpiryDate	37
ulStreamErrorCodeSecureExportCertificate	38
ulStreamErrorCodeSecureAddCertificate	39
ulStreamErrorCodeSecureTrustedCertificateFileNotFound	40
ulStreamErrorCodeSecureTrustedCertificateRead	41
ulStreamErrorCodeSecureCertificateCount	42

定数	値
ulStreamErrorCodeSecureCreateCertificate	43
ulStreamErrorCodeSecureImportCertificate	44
ulStreamErrorCodeSecureSetRandomRef	45
ulStreamErrorCodeSecureSetRandomFunc	46
ulStreamErrorCodeSecureSetProtocolSide	47
ulStreamErrorCodeSecureAddTrustedCertificate	48
ulStreamErrorCodeSecureCreatePrivateKeyObject	49
ulStreamErrorCodeSecureCertificateExpired	50
ulStreamErrorCodeSecureCertificateCompanyUnit	51
ulStreamErrorCodeSecureCertificateCommonName	52
ulStreamErrorCodeSecureHandshake	53
ulStreamErrorCodeHttpVersion	54
ulStreamErrorCodeSecureSetReadFunc	55
ulStreamErrorCodeSecureSetWriteFunc	56
ulStreamErrorCodeSocketHostNameNotFound	57
ulStreamErrorCodeSocketGetHostByAddr	58
ulStreamErrorCodeSocketLocalhostNameNotFound	59
ulStreamErrorCodeSocketCreateTcpip	60
ulStreamErrorCodeSocketCreateUdp	61
ulStreamErrorCodeSocketBind	62
ulStreamErrorCodeSocketCleanup	63
ulStreamErrorCodeSocketClose	64
ulStreamErrorCodeSocketConnect	65
ulStreamErrorCodeSocketGetName	66

定数	値
ulStreamErrorCodeSocketGetOption	67
ulStreamErrorCodeSocketSetOption	68
ulStreamErrorCodeSocketListen	69
ulStreamErrorCodeSocketShutdown	70
ulStreamErrorCodeSocketSelect	71
ulStreamErrorCodeSocketStartup	72
ulStreamErrorCodeSocketPortOutOfRange	73
ulStreamErrorCodeLoadNetworkLibrary	74
ulStreamErrorCodeActsyncNoPort	75
ulStreamErrorCodeHttpExpectedPost	89

ULStreamErrorContext 列挙

ULStreamErrorContext 定数は、ULStreamErrorContext の指定に使用できる定数を識別します。ULStreamErrorContext は、ストリーム・エラーが発生したときに実行されるネットワーク・オペレーションです。

定数	値
ulStreamErrorContextUnknown	0
ulStreamErrorContextRegister	1
ulStreamErrorContextUnregister	2
ulStreamErrorContextCreate	3
ulStreamErrorContextDestroy	4
ulStreamErrorContextOpen	5
ulStreamErrorContextClose	6
ulStreamErrorContextRead	7
ulStreamErrorContextWrite	8
ulStreamErrorContextWriteFlush	9
ulStreamErrorContextEndWrite	10
ulStreamErrorContextEndRead	11
ulStreamErrorContextYield	12
ulStreamErrorContextSoftshutdown	13

ULStreamErrorID 列挙

ULStreamErrorID は、同期が失敗したときにおそらくエラーの原因となった、ネットワーク・レイヤの列挙です。

定数	値
ulStreamErrorIDTcpip	0
ulStreamErrorIDSerial	1
ulStreamErrorIDFake	2
ulStreamErrorIDNettech	5
ulStreamErrorIDRimbb	6
ulStreamErrorIDHttp	7
ulStreamErrorIDHttps	8
ulStreamErrorIDDhCast	9
ulStreamErrorIDSecure	10
ulStreamErrorIDCerticom	11
ulStreamErrorIDJavaCerticom	12
ulStreamErrorIDCerticomSsl	13
ulStreamErrorIDCerticomTls	14
ulStreamErrorIDWirestrm	15
ulStreamErrorIDWireless	16
ulStreamErrorIDReplay	17
ulStreamErrorIDStrm	18
ulStreamErrorIDUdp	19
ulStreamErrorIDEmail	20
ulStreamErrorIDFile	21

定数	値
ulStreamErrorIDActivesync	22
ulStreamErrorIDRsaTls	23
ulStreamErrorIDJavaRsa	24

ULStreamType 列挙

ULStreamType 定数は、ストリーム・タイプの指定に使用できる定数を識別します。これらの定数は、同期に使用できる、Mobile Link 同期ストリームのタイプを表します。

定数	値	説明
ulTCPIP	1	TCP/IP ストリーム
ulHTTP	2	HTTP ストリーム
ulHTTPS	3	HTTPS 同期

ULSyncParms クラス

ULSyncParms オブジェクトの属性セットは、データベースと統合データベースまたはデスクトップ・データベースとの同期方法を決定します。読み込み専用の属性は、最後の同期ステータスを反映します。

プロパティ

ULSyncParms のプロパティは次のとおりです。

プロトタイプ	説明
CheckpointStore As Boolean	true の場合は、同期中にデータベースのチェックポイントを追加して、同期処理中にデータベースが大きくなりすぎないように制限する。これは、多くの更新を伴う大量のダウンロードに最適です。 『Mobile Link クライアント』>「Checkpoint Store 同期パラメータ」を参照してください。
DownloadOnly As Boolean	同期はデータのダウンロードのみを行うかどうかを示す。 『Mobile Link クライアント』>「Download Only 同期パラメータ」を参照してください。
KeepPartialDownload As Boolean	ダウンロード時の通信エラーが原因で同期が失敗した場合、すべての変更をロールバックしないで、正常にダウンロードされた変更を適用する。 『Mobile Link クライアント』>「Keep Partial Download 同期パラメータ」を参照してください。

プロトタイプ	説明
NewPassword As String	次の同期で、この新しいパスワードにユーザ・パスワードを変更する。 『Mobile Link クライアント』>「New Password 同期パラメータ」を参照してください。
Password As String	特定のユーザ名に対応したパスワード。 『Mobile Link クライアント』>「Password 同期パラメータ」を参照してください。
PublicationMask As Long	同期させるパブリケーションを指定する。デフォルトでは、すべてのデータを同期する。 『Mobile Link クライアント』>「publication 同期パラメータ」を参照してください。
ResumePartialDownload As Boolean	同期が失敗したときにダウンロードされる予定だった変更のみを適用し、ダウンロード時の通信エラーが原因で失敗した同期を再開する。 『Mobile Link クライアント』>「Resume Partial Download 同期パラメータ」を参照してください。
SendColumnNames As Boolean	SendColumnNames が true の場合、Mobile Link 同期サーバにカラム名を送信する。自動スクリプト生成を行うためには、Mobile Link 同期サーバにカラム名を送信する必要がある。 『Mobile Link クライアント』>「Send Column Names 同期パラメータ」を参照してください。

プロトタイプ	説明
SendDownloadAck As Boolean	SendDownloadAck が true の場合、同期中にダウンロード確認を送信する。 『Mobile Link クライアント』> 「Send Download Acknowledgement 同期パラメータ」を参照してください。
Stream As ULStreamType constants	同期中に使用するストリームのタイプを設定する。 『Mobile Link クライアント』> 「Stream Type 同期パラメータ」を参照してください。
StreamParms As String	特定のストリーム・タイプのネットワーク・プロトコル・オプションを設定する。 『Mobile Link クライアント』> 「Stream Parameters 同期パラメータ」と『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。
UploadOnly As Boolean	同期はデータのアップロードのみを行うかどうかを示す。 『Mobile Link クライアント』> 「Upload Only 同期パラメータ」を参照してください。
UserName As String	同期用の Mobile Link ユーザ名。 『Mobile Link クライアント』> 「User Name 同期パラメータ」を参照してください。
Version As String	実行する同期スクリプト・バージョン。 『Mobile Link クライアント』> 「Version 同期パラメータ」を参照してください。

AddAuthenticationParm メソッド

プロトタイプ　**AddAuthenticationParm(** BSTR parm **)**
Member of **UltraliteActiveX.ULSyncParms**

説明　authenticate_parms Mobile Link 同期スクリプトに渡すパラメータを追加します。

パラメータ　**parm**　追加するパラメータ。

検査結果　戻り値なし。

参照　『Mobile Link クライアント』>「Authentication Parameters 同期パラメータ」

『Mobile Link 管理ガイド』>「authenticate_parameters 接続イベント」

ClearAuthenticationParms メソッド

プロトタイプ　**ClearAuthenticationParms( )**
Member of **UltraliteActiveX.ULSyncParms**

説明　authenticate_parms Mobile Link 同期スクリプトに渡すパラメータをすべてクリアします。

検査結果　戻り値なし。

参照　『Mobile Link クライアント』>「Authentication Parameters 同期パラメータ」

『Mobile Link 管理ガイド』>「authenticate_parameters 接続イベント」

ULSyncResult クラス

ULSyncResult オブジェクトの属性は、最後の同期の結果を格納します。

プロパティ

次は、ULSyncResult のプロパティです。

プロトタイプ	説明
AuthStatus As ULAuthStatusCode (読み込み専用)	前回行われた同期の認証ステータス・コードを取得する。 『Mobile Link クライアント』> 「Authentication Status 同期パラメータ」を参照してください。
AuthValue As Long(読み込み専用)	Mobile Link 認証値を取得する。 『Mobile Link クライアント』> 「Authentication Value 同期パラメータ」を参照してください。
PartialDownloadRetained (読み込み専用)	ダウンロード時に同期が失敗し、部分的にダウンロードが保持されていることを示す。 『Mobile Link クライアント』> 「Partial Download Retained 同期パラメータ」を参照してください。
IgnoredRows As Boolean (読み込み専用)	前回行われた同期でローが無視されたかどうかを示す。 『Mobile Link クライアント』> 「Ignored Rows 同期パラメータ」を参照してください。
StreamErrorCode As ULStreamErrorCode(読み込み専用)	同期ストリームによってレポートされるエラー・コードを取得する。

プロトタイプ	説明
StreamErrorContext As ULStreamErrorContext (読み込み専用)	実行される基本的なネットワーク・オペレーションを取得する。
StreamErrorID As ULStreamErrorID (読み込み専用)	エラーをレポートするネットワーク・レイヤを取得する。
StreamErrorSystem As Long (読み込み専用)	ストリーム・エラー・システム固有のコードを取得する。
Timestamp as Variant (読み込み専用)	最後の同期のタイムスタンプを取得する。
UploadOK As Boolean (読み込み専用)	前回行われた同期でデータが正常にアップロードされたかどうかを示す。 『Mobile Link クライアント』>「Version 同期パラメータ」を参照してください。

ULSyncState 列挙

定数	説明
ulSyncStateStarting	同期処理はまだ行われていない。
ulSyncStateConnecting	同期ストリームは構築されたが、まだ開かれていない。
ulSyncStateSendingHeader	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る。
ulSyncStateSendingTable	テーブルの送信中。
ulSyncStateSendingData	現在のテーブルのデータの送信中。
ulSyncStateFinishingUpload	アップロードが完了。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。
ulSyncStateReceivingUploadAck	アップロード完了の確認の受信中。
ulSyncStateReceivingTable	テーブルの受信中。
ulSyncStateReceivingData	現在のテーブルのデータの受信中。
ulSyncStateCommittingDownload	ダウンロードのコミット中。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。
ulSyncStateSendingDownloadAck	ダウンロード完了の確認の送信中。
ulSyncStateDisconnecting	同期ストリームが閉じられようとしている。
ulSyncStateDone	同期が正常に完了した。SyncResult オブジェクトが更新されました。
ulSyncStateError	同期は完了したが、エラーが発生した。SyncResult と QLCode の詳細を確認してください。

定数	説明
ulSyncStateRollingBackDownload	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る。後続の ulSyncStateError 進行状況レポートにエラーがレポートされます。
ulSyncStateCancelled	同期がキャンセルされた。

ULTable クラス

ULTable クラスは、テーブルのデータの格納、削除、更新、読み込みに使用します。

Open メソッドを呼び出さないと、テーブルのデータを操作できません。ULTable ではテーブル・モードを使用してテーブルを操作します。

モード	説明
FindBegin	検索モードを開始
InsertBegin	挿入モードを開始
LookupBegin	ルックアップ・モードを開始
UpdateBegin	更新モードを開始

プロパティ

プロトタイプ	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
Columns As IColumns (読み込み専用)	カラム・オブジェクトのコレクションを取得する。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
IsOpen As Boolean (読み込み専用)	テーブルが現在開いているかどうかを示す。
RowCount As Long (読み込み専用)	テーブルのローの数を取得する。

プロトタイプ	説明
Schema As ULTableSchema (読み込み専用)	テーブル・スキーマに関する情報を取得する。

Close メソッド

プロトタイプ

Close( )
Member of **UltraliteActiveX.ULTable**

説明

テーブルに関連付けられているリソースを解放します。このメソッドは、テーブルに関するすべての処理が完了した後で呼び出してください。

Delete メソッド

プロトタイプ

Delete( )
Member of **UltraliteActiveX.ULTable**

説明

現在のローをテーブルから削除します。

DeleteAllRows メソッド

プロトタイプ

DeleteAllRows( )
Member of **UltraliteActiveX.ULTable**

説明

テーブルのすべてのローを削除します。

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。**ULConnection.StopSynchronizationDelete** メソッドを使用するか、**DeleteAllRows** の代わりに **Truncate** を呼び出すかすれば、ローは、統合データベースから削除しなくても、Ultra Light データベースから削除できます。

FindBegin メソッド

プロトタイプ

FindBegin( )
Member of **UltraliteActiveX.ULTable**

説明

検索のためにテーブルを準備します。

FindFirst メソッド

プロトタイプ

FindFirst([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

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に完全に一致するローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

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 (**EOF**) になります。

注意 : FindBegin を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns FindFirst で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindFirst に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindFirst で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

FindLast メソッド

プロトタイプ

FindLast([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

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に一致する最初のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されます。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」186 ページを参照してください。

検索する値を指定するには、値を検索する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後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 (**BOF**) になります。

注意
FindBegin を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns　FindLast で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindLast に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindLast で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

FindNext メソッド

プロトタイプ

FindNext([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

現在の位置からテーブルを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に完全に一致する次のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」186 ページを参照してください。

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 (**EOF**) になります。

注意 : FindFirst または FindLast を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns　FindNext で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindNext に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindNext で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合 (EOF) は **False** です。

FindPrevious メソッド

プロトタイプ

FindPrevious([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

現在の位置からテーブルを逆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に完全に一致する前のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」186 ページを参照してください。

失敗すると、カーソル位置は最初のローの前 (**BOF**) になります。

パラメータ　**num_columns**　FindPrevious で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindPrevious に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindPrevious で使用されます。

検査結果　成功の場合は **True** です。

失敗の場合 (BOF) は **False** です。

Insert メソッド

プロトタイプ　**Insert( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　直前に実行された **Set** メソッドで指定された値を使って、テーブルにローを挿入します。**InsertBegin** を呼び出してから、このメソッドを使用してください。各 ULColumn オブジェクトに対して設定します。

検査結果　成功の場合は **True** です。

失敗の場合 (BOF) は **False** です。

InsertBegin メソッド

プロトタイプ　**InsertBegin( )**
Member of **UltraliteActiveX.ULTable**

説明　カラム値をデフォルトに設定し、新しいローを挿入できるようにテーブルを準備します。

例　次の例では、InsertBegin は挿入モードに設定して、CustomerTable カラムへのデータ値の代入を開始できるようにします。

```
' eMbedded Visual Basic
 CustomerTable.InsertBegin
 CustomerTable.Columns("Fname").Value = fname
 CustomerTable.Columns("Lname").Value = lname
 If Len(city) > 0 Then
```

```
    CustomerTable.Columns("City").Value = city
  End If
  If Len(phone) > 0 Then
    CustomerTable.Columns("phone").Value = phone
  End If
  CustomerTable.Insert

// JScript
  CustomerTable.InsertBegin();
  CustomerTable.Columns("Fname").Value = fname;
  CustomerTable.Columns("Lname").Value = lname;
  If ( Len(city) > 0 ) {
    CustomerTable.Columns("City").Value = city;
  }
  If ( Len(phone) > 0 ) {
    CustomerTable.Columns("phone").Value = phone;
  }
  CustomerTable.Insert();
```

参照 「UpdateBegin メソッド」187 ページ

LookupBackward メソッド

プロトタイプ

LookupBackward([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

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を検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」186 ページを参照してください。

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、それより少ない値の最後のローで停止します。検索が失敗した場合 (つまり、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 (**BOF**) になります。

パラメータ　**num_columns**　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。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。

LookupBegin メソッド

プロトタイプ　**LookupBegin( )**
Member of **UltraliteActiveX.ULTable**

説明　ルックアップのためにテーブルを準備します。

LookupForward メソッド

プロトタイプ　**LookupForward(** [*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　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に一致するか、その値より大きい値を持つ最初のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」186 ページを参照してください。

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。検索が失敗した場合 (つまり、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 (**EOF**) になります。

パラメータ　**num_columns**　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。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。

MoveAfterLast メソッド

プロトタイプ　**MoveAfterLast( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　最後のローの後に移動します。

検査結果　成功の場合は **True** です。

処理が失敗した場合は **False** です。

MoveBeforeFirst メソッド

プロトタイプ　**MoveBeforeFirst( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　最初のローの前に移動します。

検査結果　成功の場合は **True** です。

処理が失敗した場合は **False** です。

MoveFirst メソッド

プロトタイプ　**MoveFirst( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　最初のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。

MoveLast メソッド

プロトタイプ　**MoveLast( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　最後のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。

MoveNext メソッド

プロトタイプ　**MoveNext( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　次のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。たとえば、ローがこれ以上ない場合 MoveNext は失敗します。

MovePrevious メソッド

プロトタイプ　**MovePrevious( )** As Boolean
Member of **UltraliteActiveX.ULTable**

説明　前のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。たとえば、ローがこれ以上ない場合 MovePrevious は失敗します。

MoveRelative メソッド

プロトタイプ

MoveRelative(*index* As Long **)** As Boolean
Member of **UltraliteActiveX.ULTable**

説明

いくつかのローを、現在のローを基準にして相対的に移動します。

パラメータ

index 移動するローの数。値は、正、負、または 0 です。ロー・バッファを再投入する場合は、0 が便利です。

検査結果

成功の場合は **True** です。

処理が失敗した場合、たとえばカーソルが最初のローの前、または最後のローの後ろにある場合は **False** です。

Open メソッド

プロトタイプ

Open(
[*index_id*], _**)**
Member of **UltraliteActiveX.ULTable**

説明

テーブルを開き、読み込みまたは操作ができるようにします。デフォルトでは、ローはプライマリ・キーによって順序付けられます。インデックスを指定すると、ローを別の方法で並べ替えることができます。

カーソルは、テーブル内にある最初のローの前に置かれます。

パラメータ

indexID インデックスの ID を参照するオプションのパラメータ。

Truncate メソッド

プロトタイプ

Truncate( )
Member of **UltraliteActiveX.ULTable**

説明

このテーブルからデータをすべて削除します。この変更の同期は行われないため、同期時に統合データベース内のデータには影響しません。

詳細については、「StartSynchronizationDelete メソッド」118 ページを参照してください。

Update メソッド

プロトタイプ

Update( )
Member of **UltraliteActiveX.ULTable**

説明

ULColumn メソッドで指定された値を使って、テーブルのローを更新します。

注意 : 先に UpdateBegin を呼び出してください。

UpdateBegin メソッド

プロトタイプ

UpdateBegin( )
Member of **UltraliteActiveX.ULTable**

説明

現在のローの内容を変更するためにテーブルを準備します。

例

```
' eMbedded Visual Basic
 Table.UpdateBegin
 Table.Columns( "ColName" ).Value = "New Value"
 Table.Update

// JScript
 Table.UpdateBegin();
 Table.Columns("ColName").Values = "NewValue";
 Table.Update();
```

ULTableSchema クラス

ULTableSchema オブジェクトを使用すると、テーブルの属性を取得できます。

プロパティ

ULTableSchema は、テーブルに関するメタデータを表します。ULTableSchema クラスのプロパティは次のとおりです。

プロトタイプ	説明
ColumnCount As Integer (読み込み専用)	このテーブル内のカラム数。
Indexes As IIndexSchemas	このテーブル内のインデックス数。
Name As String (読み込み専用)	このテーブルの名前。
NeverSynchronized As Boolean (読み込み専用)	テーブルが同期から常に除外されるかどうかを示す。
PrimaryKey As ULIndexSchema (読み込み専用)	このテーブルのプライマリ・キー。
UploadUnchangedRows As Boolean (読み込み専用)	前回の同期以降に変更されたローだけでなく、同期に対してテーブルのすべてのローをアップロードするかどうかを示す。

InPublication メソッド

プロトタイプ

InPublication(*publicationName* **)** As Boolean
Member of **UltraliteActiveX.ULTableSchema**

説明

このテーブルが、指定されたパブリケーションの一部であるかどうかを示します。

パラメータ

publicationName 確認するパブリケーションの名前。

検査結果

True テーブルがパブリケーションの一部である場合

False テーブルがパブリケーションの一部でない場合

索引

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

T

U

V

W

あ

い

は

ひ

ふ

ま

も

ゆ

ら

る

ろ