Getting Started for Offline RIA Magic xpa

OUTPERFORM THE FUTURE™

本書に記載の内容は、将来予告なしに変更することがあります。これらの情報について MSE（Magic Software Enterprises Ltd.）および MSJ（Magic Software Japan K.K.）は、いかなる責任も負いません。

本書の内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE および MSJ はいかなる責任、債務も負いません。

MSE および MSJ は、この製品の商業価値や特定の用途に対する適合性の保証を含め、この製品に関する明示的、あるいは黙示的な保証は一切していません。

本書に記載のソフトウェアは、製品の使用許諾契約書に記載の条件に同意をされたライセンス所有者に対してのみ供給されるものです。同ライセンスの許可する条件のもとでのみ、使用または複製することが許されます。当該ライセンスが特に許可している場合を除いては、いかなる媒体へも複製することはできません。

ライセンス所有者自身の個人使用目的で行う場合を除き、MSE または MSJ の書面による事前の許可なしでは、いかなる条件下でも、本書のいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対する互換性に関しての情報提供のみを目的としてなされるものです。

本書において、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

一般に、会社名、製品名は各社の商標または登録商標です。

MSE および MSJ は、本製品の使用またはその使用によってもたらされる結果に関する保証や告知は一切していません。この製品のもたらす結果およびパフォーマンスに関する危険性は、すべてユーザが責任を負うものとします。

この製品を使用した結果、または使用不可能な結果生じた間接的、偶発的、副次的な損害（営利損失、業務中断、業務情報の損失などの損害も含む）に関し、事前に損害の可能性が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負いません。

第 2 版 2014 年 9 月 26 日

目次

第 1 章 Magic xpa オフライン RIA 入門コースについて

この入門コースは、Magic xpa を使用して既にリッチインターネットアプリケーション（RIA）の作成を経験していて、初めてオフライン対応の RIA を作成する方を対象にまとめられています。

このため、Magic xpa の基本的なコンセプトや開発する上で利用するさまざまなツール機能についての説明は行っていません。Magic xpa を始めて使用する方は、事前に『Magic xpa RIA 入門コース』で学んでいただくことを推奨します。

本書では、Windows OS のクライアントを前提としています。モバイル（iOS や Android）をクライアントとした場合、画面設計やメニュー定義などに制限が発生する場合があります。

入門コースで取扱う内容

本コースでは、次のことを学びます。

- オフライン RIA の基本構成
- 次のような特徴を持つ基本的なオフライン RIA の作成
  - RIA インタフェース
  - リソース／データソースの同期処理
  - SQLite ／ Local データベースの利用
  - 一対一、または一対多のリレーションを持つデータ
  - サーバプログラムの呼び出し

入門コースの構成

本コースの主目的は、学習者がオフライン RIA を開発できるようになることです。『Magic xpa RIA 入門コース』で提供されている通常のリッチインターネットアプリケーションをオフライン用に変更していくことで、Magic xpa のオフライン RIA のプログラミング方法の理解を深めることができます。

本コースは実際に作業しながら学習を進めてください。各章には実践的例題があり、それらを実際にやってみることで課題アプリケーションが完成してゆきます。問題の指示に従って注意深くプログラミングすることが大切です。本コースではスクリーンショットを豊富に用意していますので、参考にしてください。

入門コースの前提条件

本コースは、オフライン RIA を前提としたプログラミングの自習書です。Magic xpa（または、Magic uniPaaS）で通常の RIA のプログラムを行った経験があることを前提としています。

本コーステキスト表記上の約束

本コーステキストでは、特定文字列の表現において、下表のように取り決めています。

表現	意味
＜文字列＞	本コースの説明にしたがって学習者がキーボードから入力する文字列です。必ずしも本テキストと同じでなくともよいです。
＜文字列＞	学習者がすでに入力した文字列で、上記の内容に対応するものであることを意味します。実際に入力された値が本コーステキストと異なる場合は、読み換えてください。
「文字列」	Magic xpa の操作において選択肢となる文字列であることを示します。アプリケーション上での選択肢も含まれます。
［文字列］	Magic xpa の操作画面上に表示される文字列で、システムにより既定のものであることを意味しています。
文字列／文字列	メニューやヘルプなどのように、階層構造を持つものは、／で上位と下位を区切って表示します。

第 2 章 オフライン RIA プログラムの作成

Magic xpa ではプロジェクトベースの開発を行います。本章では既存のオンライン RIA をオフライン化する手順について学び、プログラム作成を行います。

キーワード

- ［プログラム］リポジトリ
- プログラム
- 変数項目

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- RIA プログラムのオフライン化
- 変数項目の定義
- オフライン RIA プログラムの実行

参照

Magic ヘルプの［Magic xpa リファレンス／プロジェクトとアプリケーション］

同様に、［Magic xpa リファレンス／プログラム］

１．はじめに

プログラムはプロジェクトの基本的要素です。開発ツールや開発言語の基本は、プログラムを記述することです。

本章では、Magic xpa を使用して、既存のオンライン RIA プログラムをオフライン RIA プログラムに変更します。そのプログラムを動作させ、結果を確認してみましょう。

２．Magic xpa RIA プロジェクトの準備

提供するプロジェクトは、二つあります。一つ目（Magic_xpa_Offline_RIA 入門 Before）は、変更前の（『Getting Started for RIA』で提供されている）オンライン RIA プロジェクトです。もう一つ（Magic_xpa_Offline_RIA 入門 After）は、オフライン用に修正済みの RIA プロジェクトです。

フォルダのコピー

Magic_xpa_Offline_RIA 入門 Before フォルダを**＜ Magic_xpa_Offline_RIA 入門＞**フォルダとしてコピーした上で本セミナーを進めてください。

環境ファイル

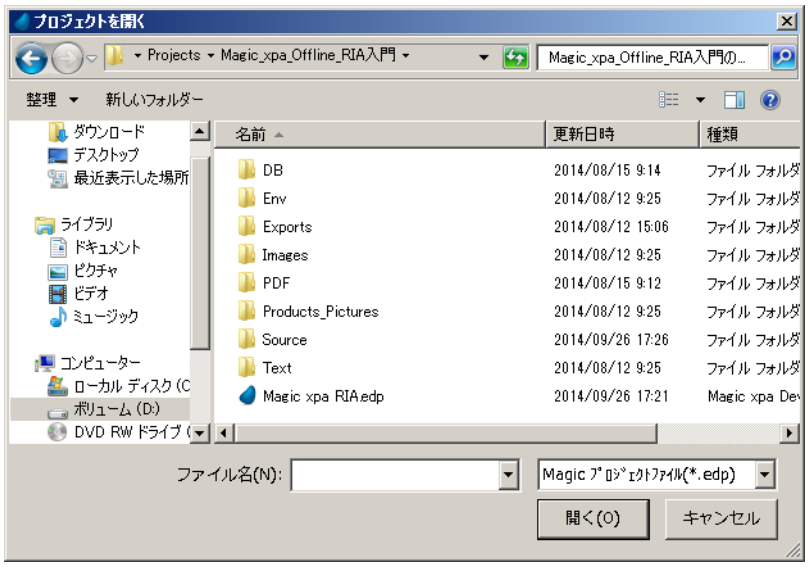
フォルダ内には、”add.ini” という環境ファイルがあります。これを読み込むことで必要な環境設定があらかじめ設定された状態になります。以下のようなコマンドラインパラメータを設定したショートカットを作成すると便利です。

＜ Magic xpa のインストールフォルダ＞ ¥MgxpaStudio.exe / @ ＜プロジェクトのコピーフォルダ＞ ¥Magic_xpa_Offline_RIA 入門 ¥add.ini

３．Magic xpa RIA プロジェクトを開く

Magic xpa RIA プロジェクトを開き、オフライン RIA プログラムに変更して行きましょう。

1. 開始画面の［ファイル］メニューから［プロジェクトを開く］（Ctrl+O）を選択し、＜ Magic_xpa_Offline_RIA 入門＞フォルダ内の Magic xpa RIA.edp ファイルを選択し、［開く］ボタンをクリックします。

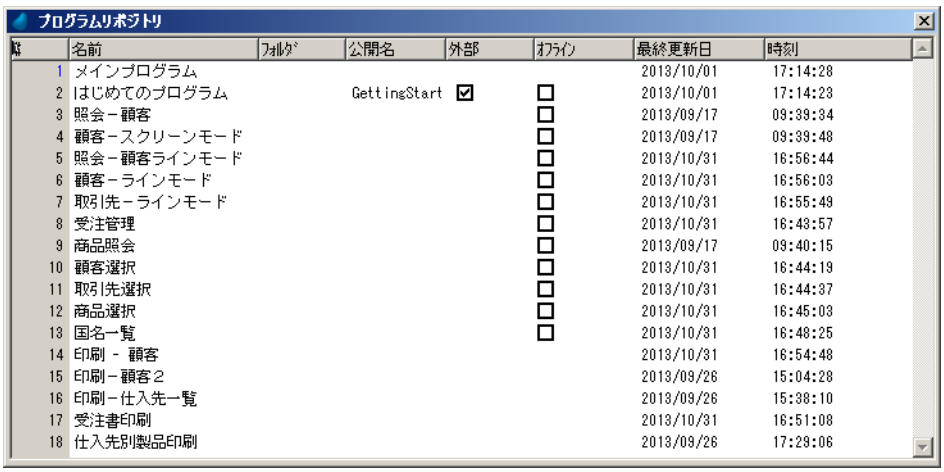


４．プログラムの変更

読み込まれたプロジェクトは、サーバ側で動作する通常のリッチクライアント対応（オンライン）のものです。簡単な RIA プログラムをオフライン用に変更してみましょう。

［プログラム］リポジトリ

Magic xpa RIA プロジェクトを読み込んだ直後の［プログラム］リポジトリは以下のように表示されているはずです。

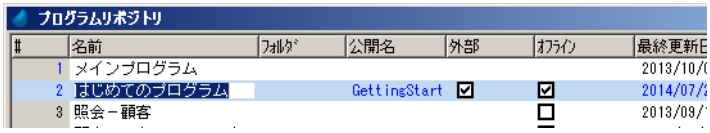


	名前	フォルダ	公開名	外部	オフライン	最終更新日	時刻
1	メインプログラム					2013/10/01	17:14:28
2	はじめてのプログラム		GettingStart	☑	☐	2013/10/01	17:14:23
3	照会－顧客				☐	2013/09/17	09:39:34
4	顧客－スクリーンモード				☐	2013/09/17	09:39:48
5	照会－顧客ラインモード				☐	2013/10/31	16:56:44
6	顧客－ラインモード				☐	2013/10/31	16:56:03
7	取引先－ラインモード				☐	2013/10/31	16:55:49
8	受注管理				☐	2013/10/31	16:43:57
9	商品照会				☐	2013/09/17	09:40:15
10	顧客選択				☐	2013/10/31	16:44:19
11	取引先選択				☐	2013/10/31	16:44:37
12	商品選択				☐	2013/10/31	16:45:03
13	国名一覧				☐	2013/10/31	16:48:25
14	印刷 - 顧客					2013/10/31	16:54:48
15	印刷－顧客２					2013/09/26	15:04:28
16	印刷－仕入先一覧					2013/09/26	15:38:10
17	受注書印刷					2013/10/31	16:51:08
18	仕入先別製品印刷					2013/09/26	17:29:06

オフライン RIA プログラムの設定

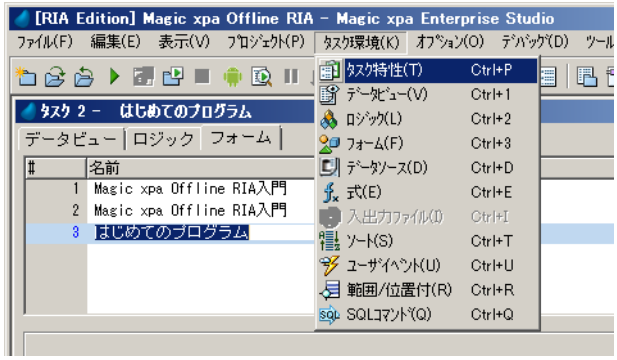
［プログラム］リポジトリには、［オフライン］というカラムがあります。これをチェックすることでその RIA プログラムはオフラインで動作するようになります。この設定は、そのプログラムの［タスク特性］ダイアログでも行うことができます。

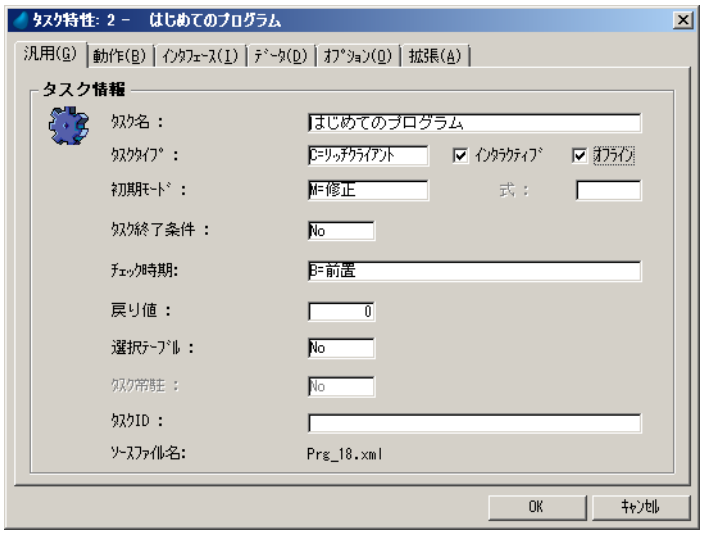
1. ＜はじめてのプログラム＞にカーソルをパークしてください。
2. ［オフライン］カラムをチェックします。



3. F5 キーを押下するか、マウスのダブルクリックにより、プログラム内部にズームしましょう。

4. ［タスク環境］メニューから［タスク特性］（Ctrl+P）を選択して［タスク特性］ダイアログを表示させましょう。





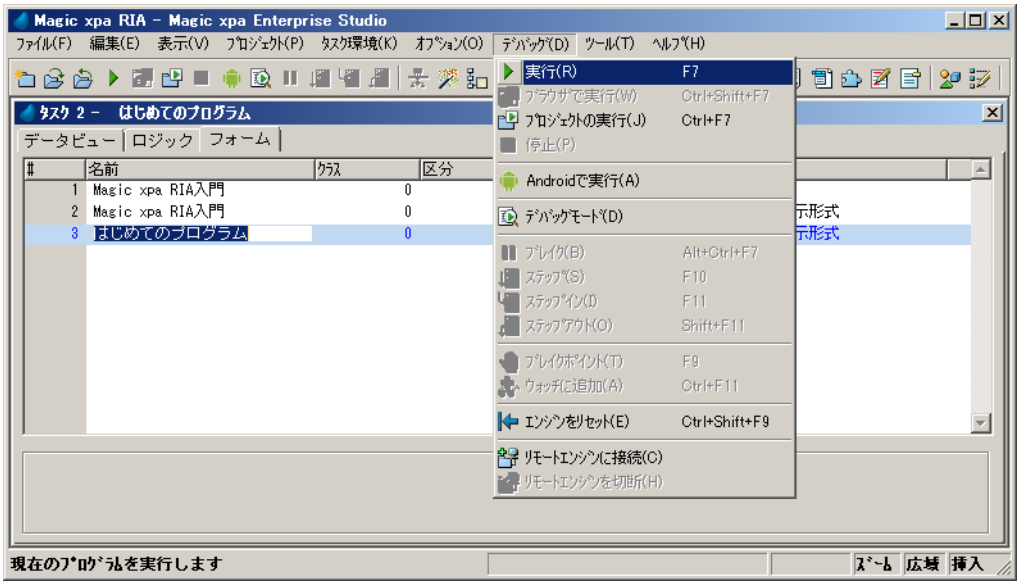
ここにも［オフライン］特性が表示されており、チェックされた状態になっています。オフラインの設定は、どちらでもできるようになっています。

これで、＜はじめてのプログラム＞は、オフライン RIA プログラムとなりました。では実行させてみましょう。

５．プログラムの実行

ここまでの作業で、Magic xpa Enterprise Studio 環境でオフライン RIA プログラムを動かしてみる準備が整いました。

1. ［プログラム］リポジトリで、＜はじめてのプログラム＞を選択します。
2. ［デバッグ］メニューから［実行］（F7）を選択します。

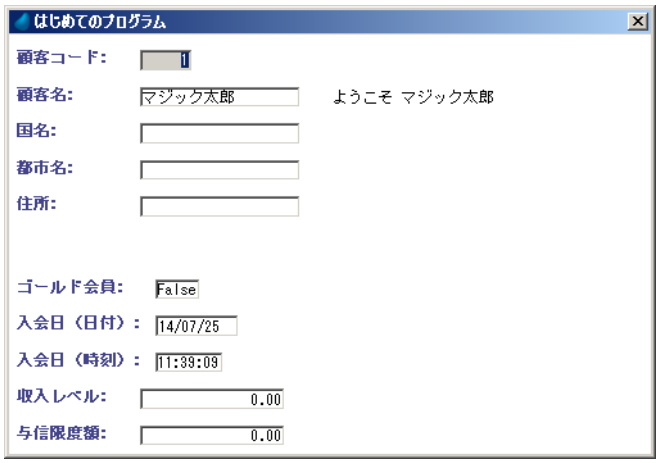


3. Magic xpa の実行ウィンドウが開き、RIA プログラムのメインフォームが表示されます。

   変数項目が表示され、データの入力ができることを確認しましょう。

   このプログラムでは変数項目に値を入力するようになっています。変数項目はプログラム実行中のみ値を保持し、プログラムが終了するとその値は破棄されます。

4. プログラムを終了し再度実行させると、変数項目はクリアされて初期状態に戻っています。



このように、今までのオンライン RIA のプログラムを簡単な作業でオフライン RIA プログラムに変更することができました。変数項目のみを使用する場合は、オンライン RIA とあまり変わりがありません。開発環境で実行させているため、オフラインとして動作しているのかどうかの区別がつきません。これについては、運用環境での動作させる場合の説明（第 12 章「アプリケーションの公開」）の中で解説します。

６．要約

本章では、Magic xpa で、オンライン RIA プログラムをオフライン RIA プログラムに変更しました。

実習を通して次のようなことについて理解を深めました。

- RIA プログラムをオフライン用に変更する設定方法

第 3 章 フォームの修正

Magic xpa はフォームの外形を整えるためのいろいろなツールを提供しています。フォームの色、壁紙、コントロールの色、フォント、スタイル、サイズ、また位置などを設定できます。

本章ではフォーム、およびコントロールの特性をオフライン用に設定する方法について学習します。オフライン RIA を作成する上で注意する点は、画像を表示させる場合、サーバ上の画像ファイルをクライアントにコピーする必要があることです。

キーワード

- フォーム
- 壁紙
- 論理名
- リソースのコピー
- ServerFileToClient() 関数

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- サーバ上のリソース（背景ファイル）をローカル側にコピーする処理

参照

Magic ヘルプの［Magic xpa リファレンス／フォームエディタ］

１．はじめに

本章では、Magic xpa でのオフラインアプリケーションのフォーム設定を行う上での注意点を学び、フォームを通常のオンライン RIA と同じように表示させるようにしてみましょう。

コントロールの設定方法は、基本的にはオンライン RIA と同じです。壁紙の設定を行う場合に事前に画像ファイルをクライアント側にコピーする処理を行うようにプログラムを変更してみましょう。

２．フォームの背景

Magic xpa のフォームに背景画像を設定することができます。これは、ウィンドウの背景表示として使用するイメージファイルです。

フォームの背景画像はフォームの［背景］特性で指定します。

このセクションでは、背景画像として使用するファイルを指定し、その特性について学んでみましょう。

ファイルのコピー

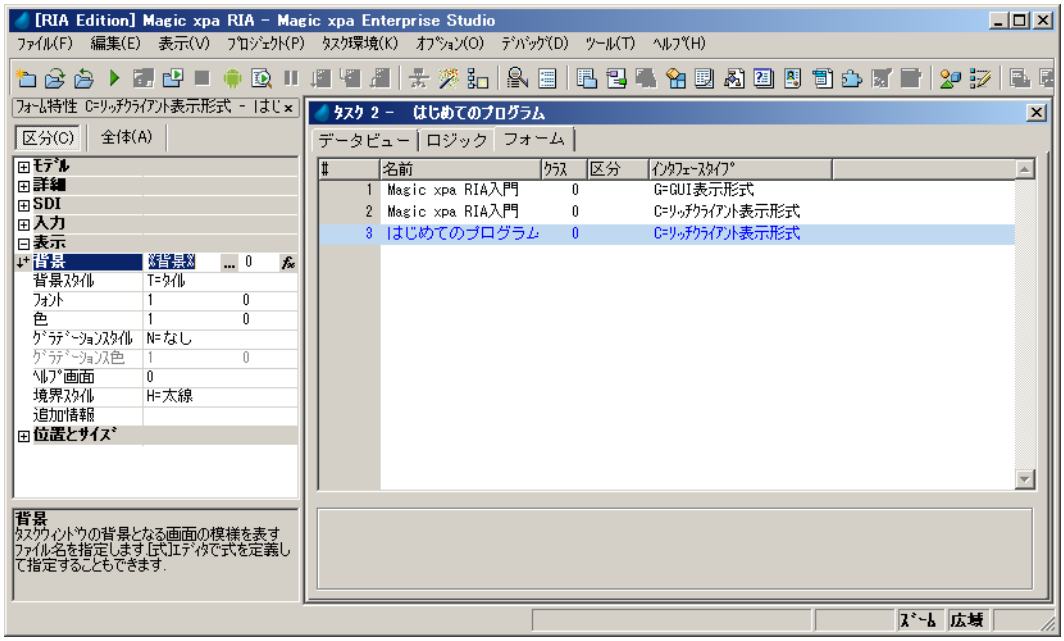
本コースのためにいくつかのフォルダとデータを用意しています。プロジェクトフォルダ内に以下のフォルダが存在していることを確認してください。ない場合は、本コースのデータフォルダからコピーしてください。

- Exports
- Images
- Products_Pictures
- Text
- PDF

フォームの［背景］特性設定

［フォーム特性］の［背景］特性を確認してください。

1. ＜はじめてのプログラム＞にズームします。
2. ［フォーム］エディタを選択します。
3. ＜はじめてのプログラム＞フォーム行にパークします。
4. ［編集］メニューから［特性］（Alt+Enter）を選択します。
5. ［背景］特性には＜ % 背景 % ＞が設定されているはずです。

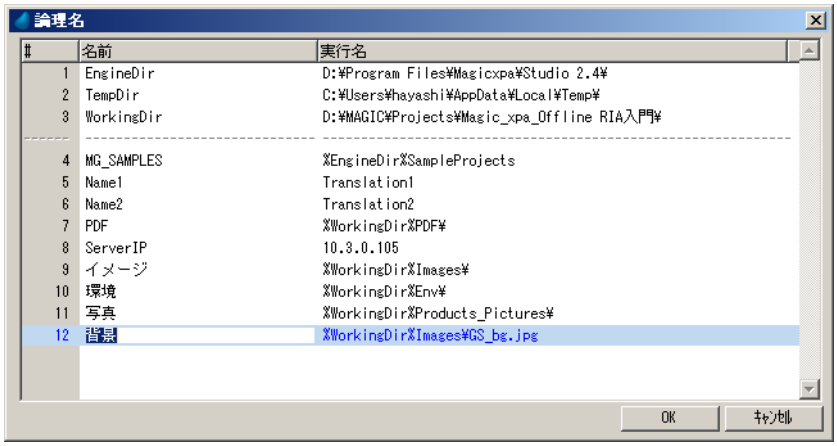


論理名の設定

本コースのデータフォルダにある Add.ini を起動持に読み込ませることで、使用する論理名が全て設定された状態になります。

ここで背景のファイルを指定する［論理名］を使用してみましょう。

1. ［オプション］メニューから［設定／論理名］を選択し、［論理名］テーブルを開きます。
2. 最後の行にパークし、F4 キーを押下して一行作成します。
3. ［名前］カラムに、**＜背景＞**と入力してください。
4. ［実行名］カラムに、**＜ %WorkingDir%images¥GS_bg.jpg ＞**と入力してください。

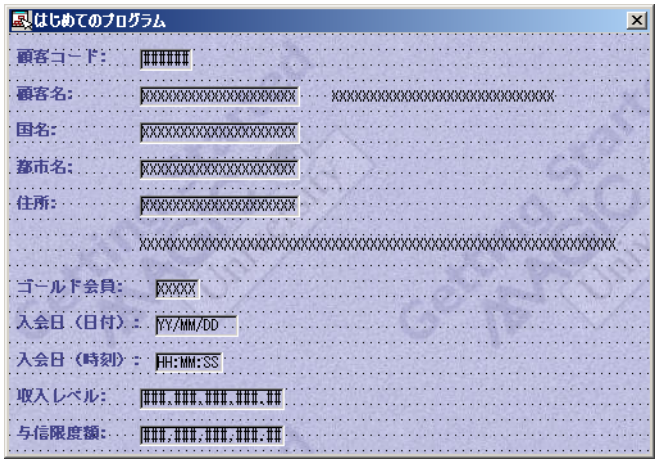


5. さらに、F4 キーを押下して一行作成します。
6. ［名前］カラムに、**＜環境＞**と入力してください。
7. ［実行名］カラムに、**＜ %WorkingDir%Env¥ ＞**と入力してください。これは、環境ファイル用の論理名です。
8. ［OK］ボタンをクリックして［論理名］テーブルを閉じます。

［フォーム］エディタで背景の確認

論理名が定義された状態で、もう一度［フォーム］エディタを開いてみましょう。

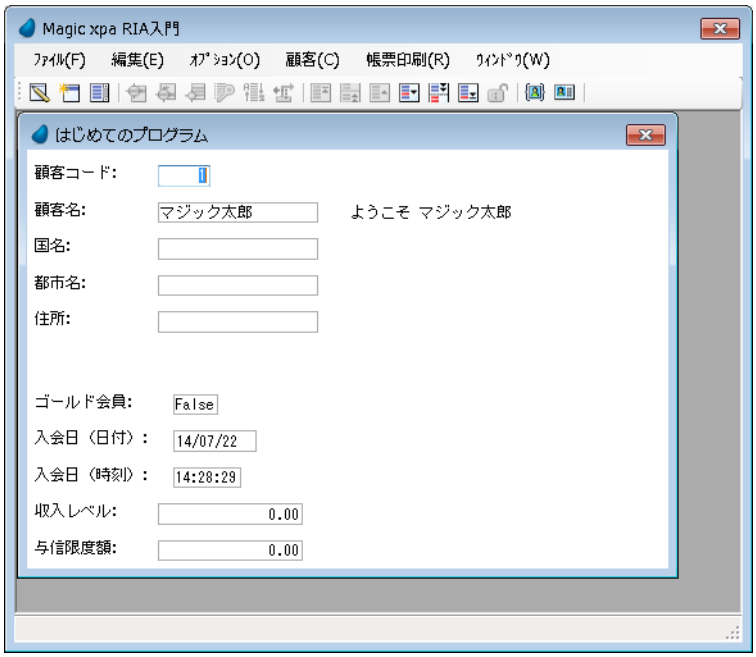
1. ＜はじめてのプログラム＞にズームします。
2. ［フォーム］エディタを選択します。
3. ＜はじめてのプログラム＞フォーム行にパークします。
4. ここからズーム（F5）して［フォーム］エディタを開くと、設定された背景が表示されます。



プログラムの実行と確認

これまでの作業でフォームのデザインは終了です。プログラムを動かして表示させてみましょう。

1. F7 キーを押下して＜はじめてのプログラム＞を起動させます。
2. 顧客情報を適当に入力してみてください。



3. プログラムは動作しますが、背景画像が表示されません。これは、画像ファイルがサーバにあるためオフライン RIA プログラムではアクセスできないからです。オフライン環境で表示させるには、プログラムを実行する前に画像ファイルをクライアント側にコピーする必要があります。

３．画像データの同期処理

オフラインアプリケーションでは、アプリケーションの起動時に画像ファイルなどのリソースをクライアントのローカル環境にコピーして同期させる必要があります。この処理は、ServerFileToClient() 関数をメインプログラムで実行させることで可能になります。

リソースの同期処理の追加

リソースの同期処理を［メインプログラム］に追加します。［syncResources］というユーザイベントを定義し、このイベントに対応する［イベント］ロジックユニット内で ServerFileToClient() 関数を実行するように定義します。

1. ［メインプログラム］にズームします。
2. Ctrl+U を押下して［イベント］テーブルを表示します。
3. 一行作成（F4）して、以下のようにイベントを定義します。

名前	トリガタイプ	強制終了
syncResources	N= なし	N= なし

4. プログラムの［ロジック］エディタに戻り Ctrl+H を押下してヘッダ行を追加します。以下のように設定します。

ユニットタイプ	イベント	条件
E= イベント	syncResources	Yes

5. ［イベント］ロジックユニットのヘッダ行にパークしている状態で、二行作成（F4）します。

6. 次の処理コマンドを定義します。

コマンド	項目	式	条件
アクション	E= 式	ServerFileToClient('%WorkingDir%images¥')	Yes
アクション	E= 式	ServerFileToClient('%WorkingDir%Products_Pictures¥')	Yes

ServerFileToClient() 関数のパラメータには、サーバ上の画像ファイルが保存されているフォルダ名を指定します。これで定義されたフォルダ内のファイルがコピーされます。

指定されたフォルダ内にサブフォルダがある場合は、コピーされません。別途このフォルダを指定した関数を実行させる必要があります。

同期処理の呼び出し

前述で定義したユーザイベント（syncResources）を［メインプログラム］の［タスク前］で発行するようにします。

1. ［メインプログラム］にズームします。
2. ［ロジック］エディタを選択します。
3. Ctrl+H を押下してヘッダ行を追加し、以下のように［タスク前］ロジックユニットを作成します。

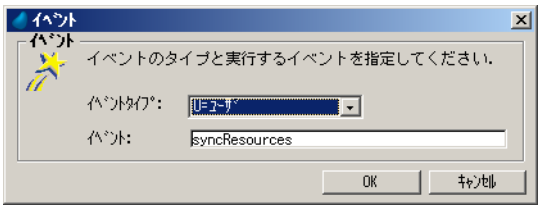
ユニットタイプ	イベント	条件
T= タスク	P= 前	

4. ［タスク前］のヘッダ行にパークします。一行作成（F4）して以下の処理を設定します。

処理コマンド	イベント	ウェイト	条件
イベント実行	syncResources	No	Yes

［イベント］カラムでズームすると［イベント］ダイアログが表示されます。ここで以下のように選択します。

- イベントタイプ …… U= ユーザ
- イベント …… syncResource



Magic xpa では以下のようなタスク上の制限があります。

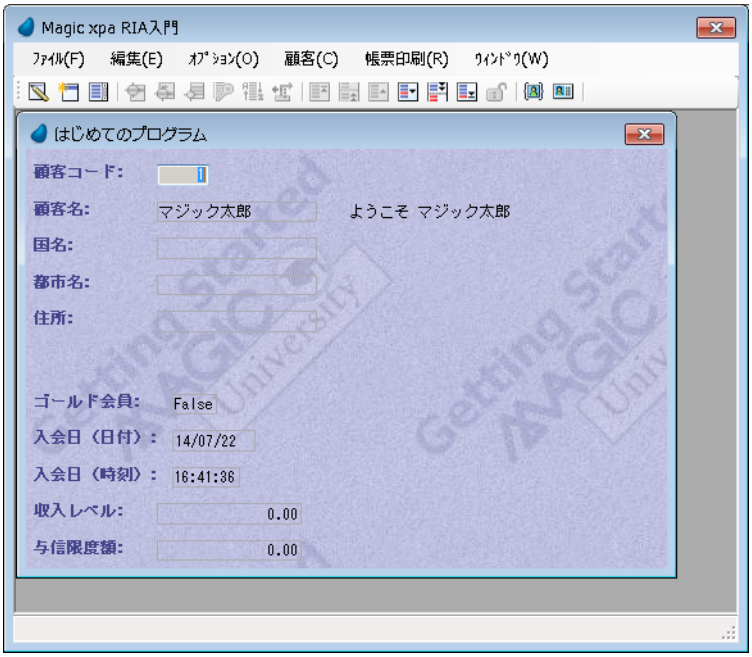
- **［メインプログラム］の［タスク前］には、クライアントの処理は定義できません。このため、ここから［コールプログラム］処理コマンドでオフライン RIA プログラムを呼び出すことができません。**
- **オフライン RIA プログラムから［コールプログラム］処理コマンドでオンライン RIA プログラムを呼び出すことができません。**

ServerFileToClient() 関数はクライアントでのみ動作する関数のため、［タスク前］で実行させるには、イベントを利用する必要があります。

プログラムの実行と確認

ここまでの状態でプログラムを動かして表示させてみましょう。

1. F7 キーを押下して＜はじめてのプログラム＞を起動させます。
2. まだ、背景が表示されないかもしれません。一旦終了します。
3. もう一度 F7 キーを押下して＜はじめてのプログラム＞を起動させます。今度は、背景が表示されます。



画像ファイルの確認

背景画像のファイルがどこにコピーされているか、確認してみましょう。

1. 以下のフォルダを開いてみてください。

   %TEMP%MgxpaRIACache¥＜クライアントのホスト名＞

%TEMP% は、Windows の環境変数です。コマンドプロンプトから set コマンドを実行することで確認できます。デフォルトは、以下のように設定されているはずです。

```
C:¥Users¥<User id>¥AppData¥Local¥Temp
```

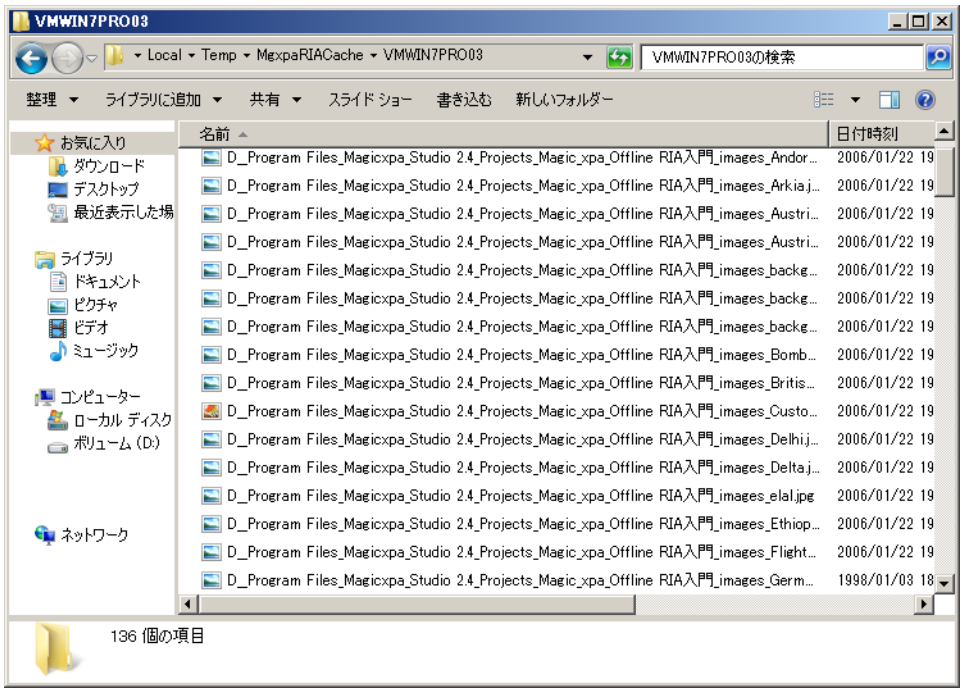
［フォーム特性］に定義された論理名＜％背景％＞は、＜%WorkingDir%images¥GS_bg.jpg＞という実行名が設定されています。たとえば、サーバ側の実際の保存先が以下のような場合を想定します。

```
D:¥Program Files¥Magicxpa¥Studio 2.4¥Projects¥Magic xpa Offline RIA 入門¥images¥GS_bg.jpg
```

この場合、クライアントのファイルは、以下のファイル名でコピーされます。

```
D__Program_Files_Magicxpa_Studio_2.4_Projects_Magic_xpa_Offline_RIA 入門_images_GS_bg.jpg
```

コロン（:）やバックスペース（¥）、空白は全てアンダーバー（_）に変換された名前になります。



最初の起動時に背景が表示されないのは何故でしょうか？

F7 キーで＜はじめてのプログラム＞が実行される前に［メインプログラム］が実行され、その［タスク前］で画像ファイルがローカルにコピーするようにしていますが、表示が間に合いませんでした。このため、初期画面を表示させるタスクがロードされる前にコピー処理を完了させる必要があります。

［メインプログラム］から、初期画面用のプログラムを起動するような場合は、処理を遅らせる必要があるかもしれません。また、初期画面表示の前に同期処理などを行う非インタラクティブなプログラムを作成し、これをアプリケーションの入り口にすることも対応の一つになります。

４．要約

本章では、以下のことを学びました。

- フォームの背景設定やコントロールの外観に関する特性
- 画像ファイルの同期処理の方法

第 4 章 データソースの内容参照

本章では、オフライン RIA でのデータソース定義と、単純なプログラムを使用してその内容を表示する方法を学習します。

キーワード

- ローカルデータベース
- ローカルデータソース
- カラム

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- オフライン RIA で利用できる DBMS
- ローカルデータベースの定義方法
- オフライン RIA 用のテーブルカラムの定義について
- レコードの更新／削除を行うためのプログラムについて

参照

Magic ヘルプの［Magic xpa リファレンス／データソース，［データビュー］エディタ］の各セクション

注意事項

この章では、ローカルデータソースの操作方法やそのためのプログラムについての説明を行っています。本章で説明しているデータソースは、マスタ系のデータとして利用するもので、通常はサーバ上で管理し、クライアント側と同期処理を行って利用します。同期処理については、第 9 章「サーバ側とのデータ同期」で説明します。

１．［データベース］テーブル

通常のオンライン RIA は、Magic xpa でサポートされる DBMS を利用した任意のデータベースを定義することができます。

Magic xpa でオフライン用のデータソースを利用する場合は、”Local” という名前の専用の DBMS を使用します。オフライン上では、これ以外の DBMS にアクセスすることはできません。Local DBMS は、SQLite DBMS とデータ互換があり、サーバ上の SQLite 用 DB ファイルをローカルにコピーして利用することもできます。

サーバ上のデータベースと連携させる場合の処理は、後の章で説明します。ここでは、オフライン環境で利用する場合の定義方法のみ説明します。

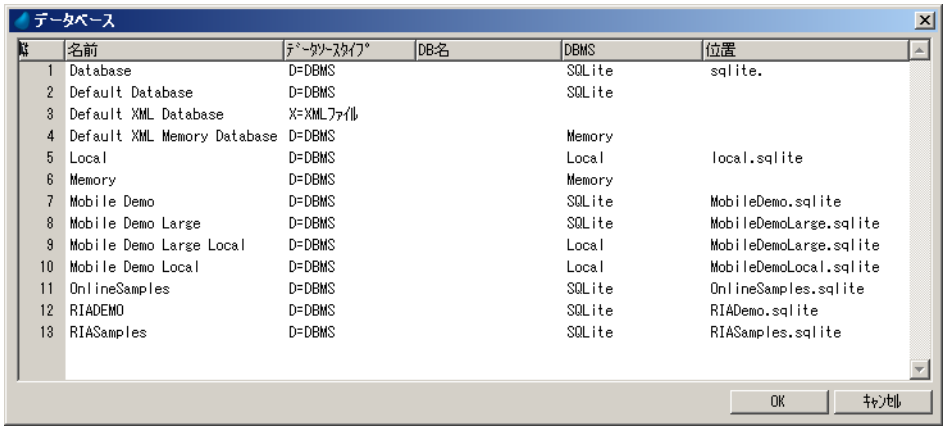
［データベース］テーブルには、現在の Magic xpa インストレーション環境で使用できる物理的なデータベース情報の詳細が登録されています。

SQLite は、シングルユーザ用の DBMS です。サンプルアプリケーションやオフラインアプリケーションとの互換性のために提供しています。通常のアプリケーションでの利用は、推奨できません。

［データベース］テーブルには、異なるデータベースあるいは同じデータベースの複数のエントリを定義することができます。各エントリには、DBMS 情報、データベース名、ユーザ名、パスワード、その他の追加情報など、Magic xpa がそのデータベースに実際に接続するときに必要となる情報が定義されます。

デフォルトの状態として、以下のようなデータベースが登録されています。

- Default Database …… デフォルトで選択されるデータベースです。新しくデータソースを作成したときに、特にデータベースを指定しないと、これが選択されます。
- Default XML Database …… 新しく XML データソースを定義したときにデフォルトで選択されます。
- Memory …… メモリテーブルが必要な場合、開発者または Magic xpa が一時的に保持する内部使用目的で選択します。
- Default XML Memory Database …… Default XML Database の一時データベースとして使用されるメモリデータベース。
- Local …… オフライン RIA 用のローカルデータベース
- RIADEMO/Mobile XXX /RIASample/OnlineSample……Magic xpa に添付されているデモアプリケーション用のデータベース。SQLite DBMS を使用しています。



＜ Local/Server ＞データベースの定義

本コースのプロジェクトは、ローカル用に Local DBMS 、サーバ用に SQLite DBMS を使用して開発します。

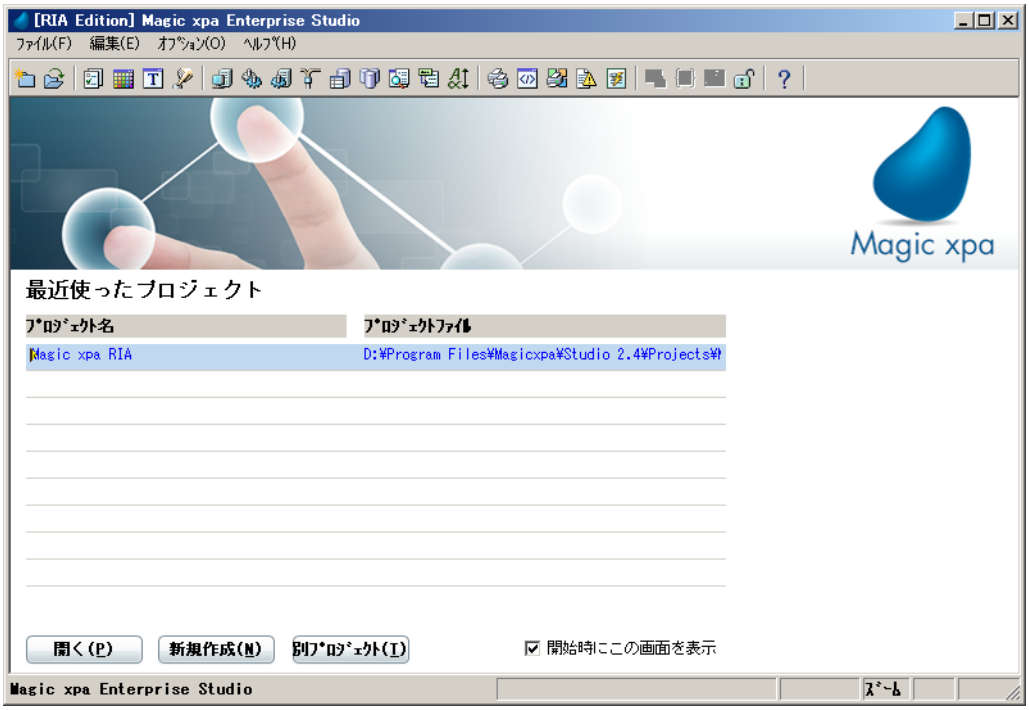
本コースは、＜Local＞データベースと、＜Server＞データベースに関連するテーブルが作成されることを前提としています。

これらのテーブルにアクセスするには、あらかじめ［データベース］テーブルの定義をしなければなりません。Local データベースは、インストール時のデフォルト（「Local」）を使用しますが、設定内容について説明します。また、サーバ用の「Server」データベースもここで定義します。

本コースのデータフォルダにある Add.ini を起動持に読み込ませることで、使用するデータベースが全て設定された状態になります。

［データベース］テーブルを開くには、次のようにします。

1. すべてのプロジェクトを閉じた状態にします。プロジェクトが開いているときは、［ファイル］メニューから［プロジェクトを閉じる］を選択します。
2. ［オプション］メニューから［設定／データベース］を選択し、［データベース］テーブルを開きます。



3. 「Local」データベースが定義されていることを確認して下さい。以下のように定義されているはずです。

#	名前	データソースタイプ	DB 名	DBMS	位置
5	Local	D=DBMS		Local	local.sqlite

［位置］カラムには、作成される物理ファイルの名前を指定します。上記の場合、ローカル側に ”local.sqlite” という DB ファイルが作成されます。

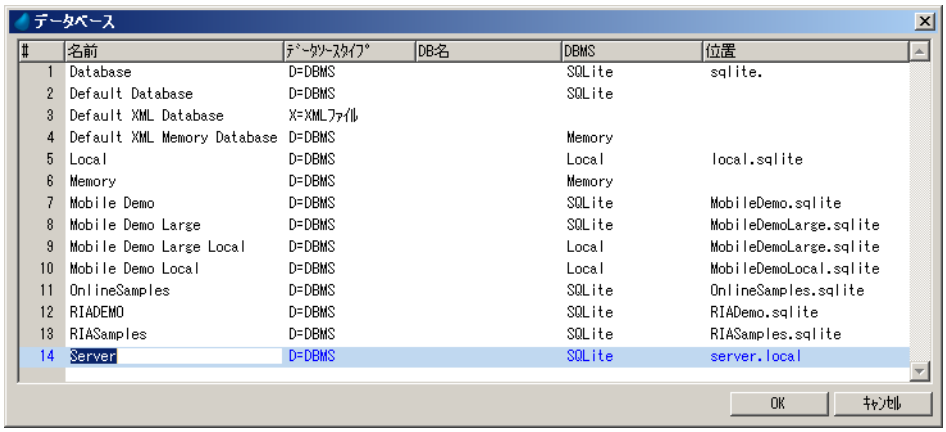
次にサーバ側のデータベースを定義します。

4. 最後の行にパークします。
5. F4 キーで一行作成します。
6. 次のようにパラメータを設定してください。

#	名前	データソースタイプ	DB 名	DBMS	位置
14	Server	D=DBMS		SQLite	server.sqlite

この場合、サーバ側のアプリケーションの作業フォルダに ”server.sqlite” という DB ファイルが作成されます。

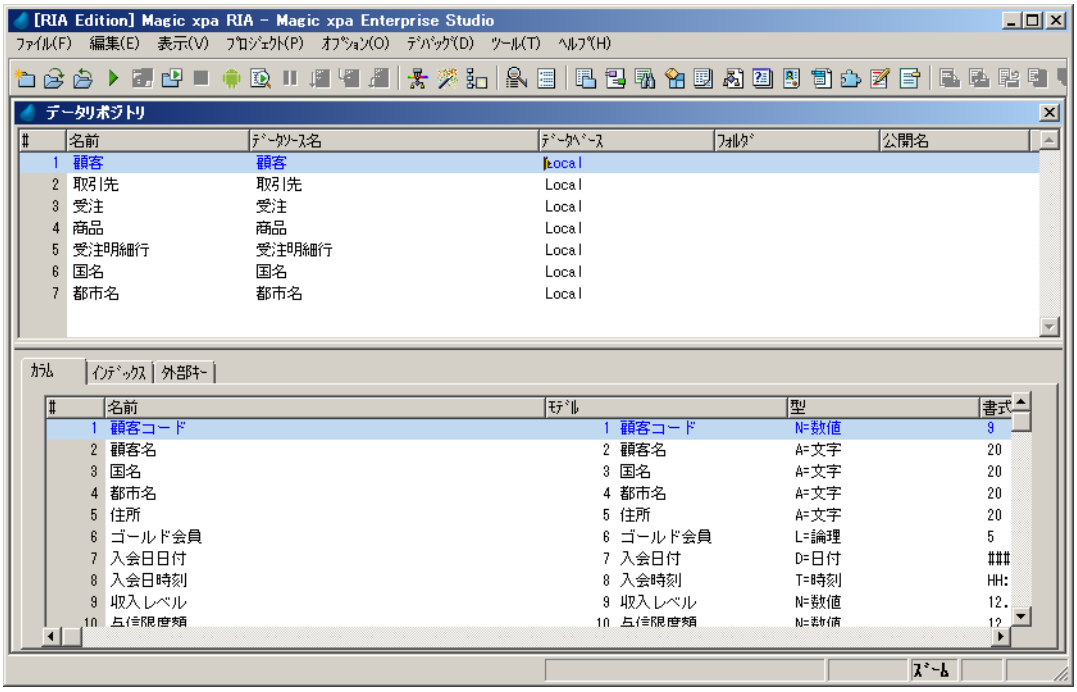
DBMS の選択は、［DBMS］カラムからズームして DBMS 一覧を表示し、その中から選択します。



２．データソースの変更

次のステップは、データソースの定義です。サンプルアプリケーションには既にデータソースが定義されており、＜ MSSQL ＞が定義されています。これを全て＜ Local ＞に変更してローカル用データソースとします。

1. ［プロジェクト］メニューから［データ］（Shift+F2）を選択し、［データ］リポジトリを開きます。
2. ［データベース］カラムにパークして、ズーム（F5）します。
3. ［データベース一覧］が開くので、ここから「Local」を選択して［選択］ボタンをクリックします。



次に、オフライン処理に必要なカラムをすべてのデータソースに対して追加します。

＜顧客＞データソースのカラムの追加

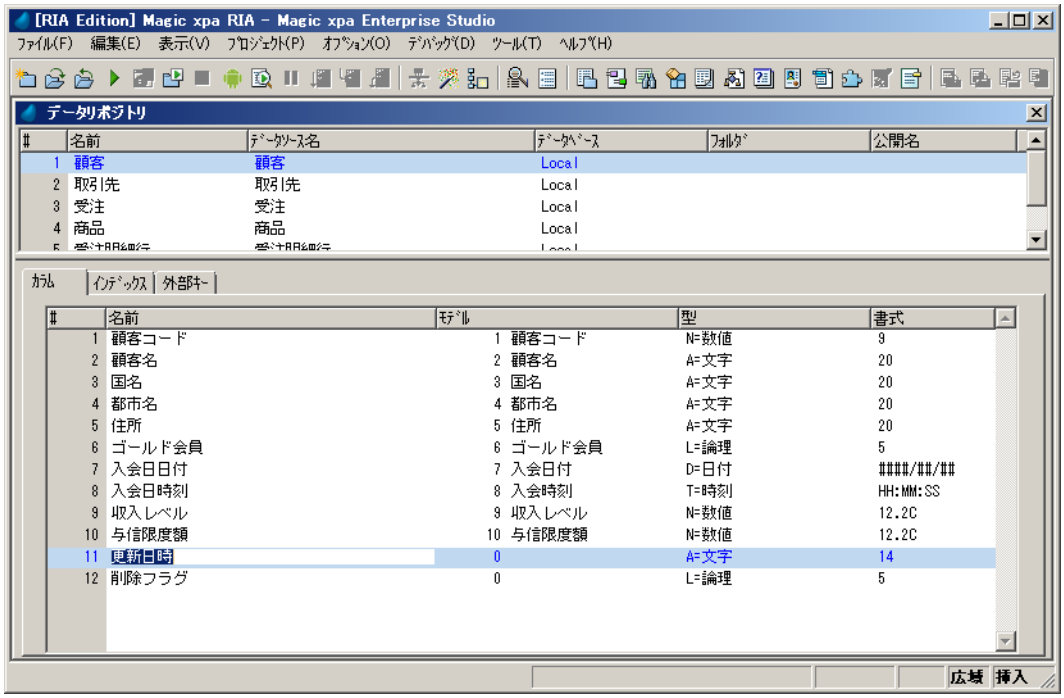
それでは、これから＜顧客＞データソースにカラムを追加します。

1. ［プロジェクト］メニューから［データ］（Shift+F2）を選択し、［データ］リポジトリを開きます。
2. 上部ペインの＜顧客＞データソースにパークします。
3. 下部ペインの［カラム］タブをクリックします。
4. ［カラム］テーブルの最後（[与信限度額]）にパークして、F4 キーで一行作成します。

5. 以下のカラムを追加します。

#	名前	モデル	型	書式
11	更新日時	0	A= 文字	14
12	削除フラグ	0	L= 論理	5

6. ［顧客］データソースのカラムは以下のようになります。

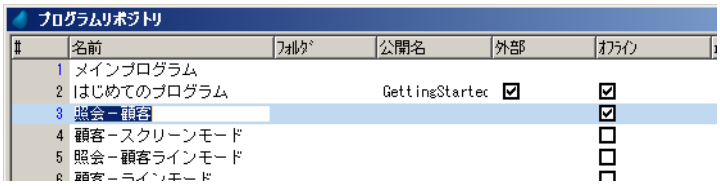


［更新日時］と［削除フラグ］は、オフラインアプリケーション で必要なカラムです。使い方は、後の章で説明します。

３．＜顧客＞プログラムの修正

＜顧客＞データソースのデータ内容の表示と操作（登録、修正など）は、＜照会ー顧客＞プログラムで行うことができます。このプログラムをオフライン用に変更して動作を確認してみましょう。

1. ［プログラム］リポジトリで＜照会ー顧客＞プログラムの行にパークします。
2. ［オフライン］カラムをチェックします。



3. F5 キーを押下してプログラムが開きます。
4. ［データビュー］エディタを開きます。ここには、＜顧客＞データソースのカラムが定義されていますが、先ほど追加したカラムも追加します。
5. テーブルの最後（[与信限度額]）にパークして一行作成（F4）します。
6. ［更新日時］カラム（#11）を定義します。さらに、一行作成（F4）します。
7. ［削除フラグ］カラム（#12）を定義します。

これで、オフライン用 プログラムになりました。実行してみましょう。

４．＜照会－顧客＞プログラムの実行

このセクションでは、生成されたプログラムを動かし、オフライン RIA プログラムのデータの操作方法を学びます。

1. ［プログラム］リポジトリで＜照会－顧客＞プログラムの行にパークし、F7 キーを押下して実行させます。

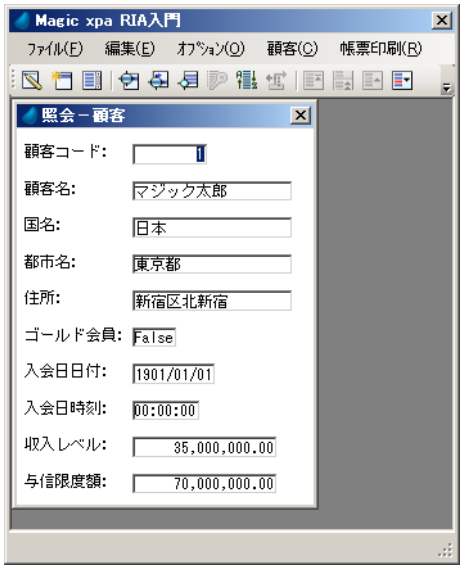
顧客データの登録

＜照会－顧客＞プログラムは、＜顧客＞データソースの内容を表示します。

このプログラムの初期モードは「Q= 照会」ですので、もしデータソースに一つでもレコードがあれば、その最初のレコードが表示されます。

しかし＜顧客＞データソースには最初はデータがないので、Magic xpa はタスクモードを「C= 登録」に自動的に変更し、エンドユーザのデータ登録を促します。

2. 顧客データを適当に入力しましょう。（イメージは参考入力です。）



このスクリーンモードでのデータナビゲーションについて次に説明します。

項目間のカーソル移動

Magic xpa では次のようにしてフォーム上の項目間のカーソル移動を行います。

スクリーンモードでは下記の方法が利用できます。

- 特定の項目をクリック
- Tab キーを押下して、次項目に移動
- Shift ＋ Tab キーを押下して、前項目に移動
- ↓キーを押下して、次項目に移動
- ↑キーを押下して、前項目に移動

顧客データの追加登録

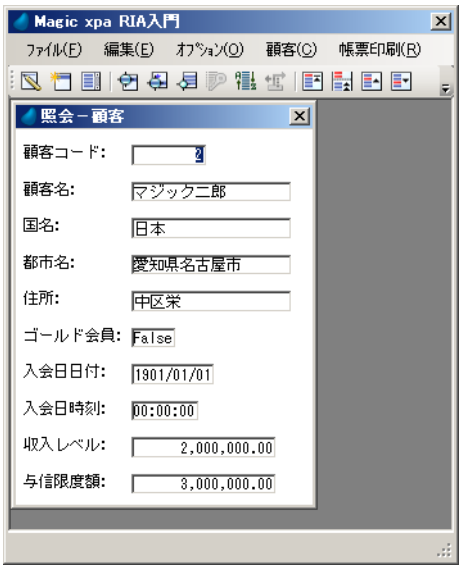
顧客レコードを一つ入力しましたが、まだデータベースには保存されていません。

顧客データの保存は、もう一つ別の顧客データを追加するなど、現在のレコードから他へ移動するアクションを実行したときに行われます。

では次のようにして顧客データをさらに追加してみましょう。

3. PageDown キーを押下します。

4. 顧客データを適当に追加しましょう。



顧客データの照会

先ほどの操作で登録した顧客データを、照会モードに切り替えて、表示させてみましょう。

照会モードに切り替えることにより、Magic xpa はレコードデータを保存し、同じレコードを照会モードで表示します。

1. Ctrl+Q を押下します。
2. 顧客データ間を次のキー操作を行って移動してみましょう。

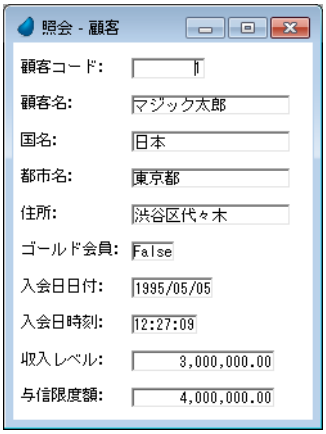
移動キー	説明	［オプション］メニュー
PageDown	次レコードへ移動	次画面
PageUp	前レコードへ移動	前画面
↑	前の項目へ移動	
↓	次の項目へ移動	
Ctrl+Home	最初のレコードへ移動	テーブル先頭
Ctrl+End	最後のレコードへ移動	テーブル末尾

モーダルウィンドウで作成されたプログラムでは、MDI に表示されている［オプション］などのプルダウンメニューにアクセスできません。この場合の操作は、キーボードで行ってください。

顧客データの修正

このセクションでは、修正モードでの操作を行います。

修正モードではデータの編集と削除を行うことができます。



1. 最初のレコード（Ctrl+Home）へ移動します。
2. Ctrl+M を押下します。
3. ［入会日日付］項目にパークします。

4. 日付を変更してみてください。

この例では［入会日日付］を変更しましたが、同様に他の項目も変更することができます。

コントロールの内容編集について

コントロールの内容を編集する方法は、以下のように二通りあります。

- **コントロールの内容がマークされている場合**
  これはデフォルトの方法です。コントロールにカーソルがパークするとき、コントロールの内容はマークされています

（青色で）。

この場合、入力を開始すると、既存の内容は削除され、新しくデータを入れ直すことになります。

- **コントロールの内容がマークされていない場合**
  コントロールにパークしているけれども、内容がマークされていない場合の処理です。

  この場合は、既存の内容に対して、書式を超えない範囲で文字を追加したり、既存のデータの一部（または全部）を削除してから追加したりできます。

修正内容のキャンセル

Ctrl+F2 を押下することで、レコードに加えた編集内容をキャンセルすることができます。

５．＜照会ー顧客＞プログラムの修正

オフライン RIA プログラムで更新されるデータはローカル用のデータのため、最終的には、サーバ側にアップする必要があります。そのための判断条件として［データソース］テーブルで追加された［更新日時］カラムを使用します。また、レコードの削除行う場合、物理レコードの削除は行わず［削除フラグ］を設定するだけにします。

ここでは、そのような処理に対応したプログラムの修正内容について説明します。

更新日時の更新

［更新日時］カラムには、更新時の日時を UTC（協定世界時刻）を基にした日付と時刻を文字型に変換した値にして格納します。これは、クライアントの実行場所にかかわらず同じタイムゾーンで管理する必要があるからです。

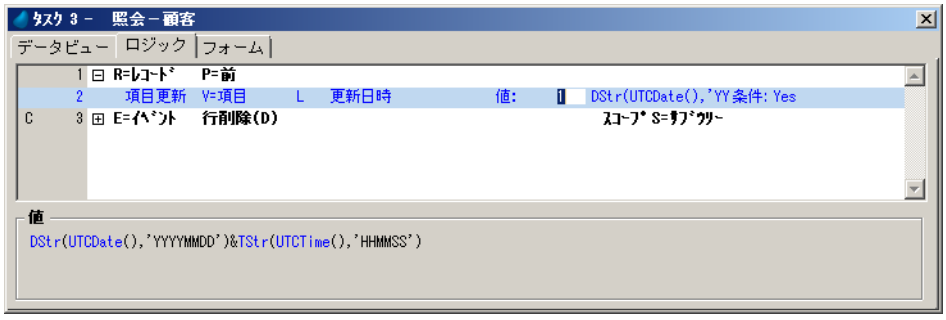
レコードが更新されると［レコード後］が必ず実行されるため、この処理はここに定義します。

1. ＜リッチクライアント - 顧客＞にズームします。
2. ［ロジック］エディタを選択します。
3. Ctrl+H を押下してヘッダ行を追加します。以下のように［レコード後］ロジックユニットを定義します。

タイプ	名前	スコープ	戻り値
R= レコード	S= 後		

4. ［レコード後］ロジックユニットのヘッダ行にパークしている状態で、一行作成（F4）します。
5. 次の処理コマンドを定義します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes



顧客データの削除

オフラインアプリケーションでの削除処理は、［削除フラグ］カラムを True に設定し［更新日時］カラムの更新を行います。

ここでは、F3 キーを押下することで［確認］ダイアログが表示され、「はい」を選択することによりレコードの［削除フラグ］カラムを True に設定するプログラムを作成してみましょう。

1. ＜照会 - 顧客＞にズームします。
2. ［データビュー］エディタを選択します。
3. 最終行に位置付けして一行作成（F4）し、以下のように変数項目を定義します。

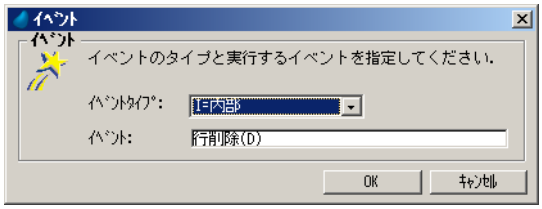
#	名前	モデル	型	書式
14	v_ 戻り値	0	N= 数値	1

4. ［ロジック］エディタを選択します。
5. Ctrl+H を押下してヘッダ行を追加します。以下のように［イベント］ロジックユニットを定義します。

タイプ	名前	スコープ	戻り値
E= イベント	行削除（D）	S= サブツリー	

［イベント］カラムでズームすると［イベント］ダイアログが表示されます。ここで以下のように選択します。

- **イベントタイプ** …… I= 内部
- **イベント** …… 行削除（D）



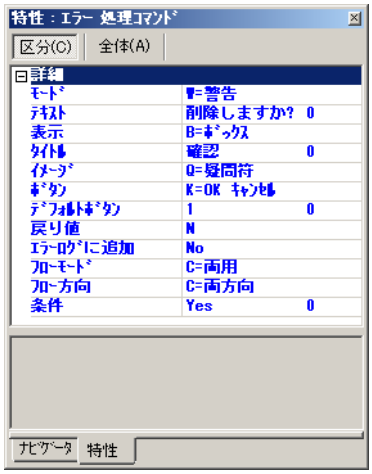
6. ［イベント］ロジックユニットのヘッダ行にパークしている状態で、一行作成（F4）します。
7. 次の処理コマンドを定義します。

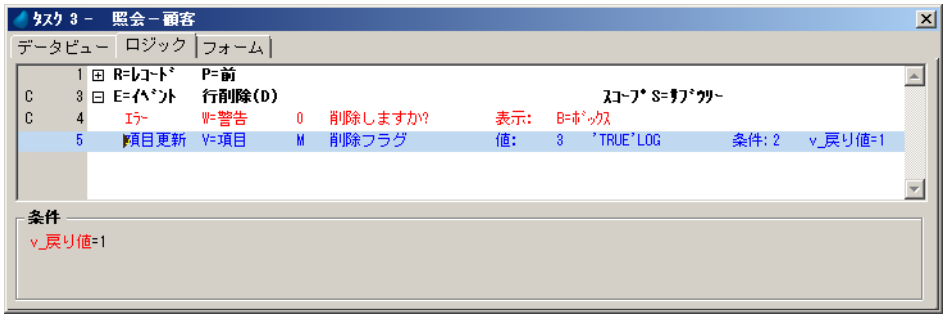
処理コマンド			式	条件
エラー	W= 警告	削除しますか？		Yes
項目更新	V= 項目	削除フラグ	'TRUE'LOG	＜ v_ 戻り値＞ =1

［エラー］処理コマンド行で Ctrl+P を押下し、［特性］シートを開き以下のように特性を設定します。

- **タイトル** …… ＜確認＞
- **イメージ** …… Q= 疑問符
- **ボタン** …… K=OK キャンセル
- **戻り値** …… ＜ v_ 戻り値＞

［エラー］処理コマンドの実行時に「OK」が選択されると、戻り値として「1」が返ります。



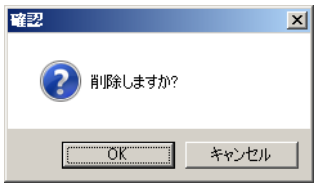


8. 動作確認用にフォームに［更新日付］と［削除フラグ］のカラムを配置してください。

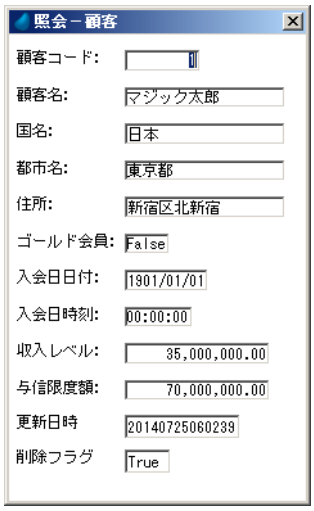
修正したプログラムを実行して確認してみましょう。

1. 最初のレコードに移動します。
2. タスクモードが「修正」であることを確認しましょう。
3. データを入力して PageDown と PageUp キーを押下します。
4. ＜更新日付＞が更新されていることを確認してください。

5. F3 キーを押下します。



6. ［確認］ダイアログで「OK」をクリックします。



この操作でレコードは削除されずフォーム上に表示されている［更新日時］と［削除フラグ］が更新されることが確認できます。

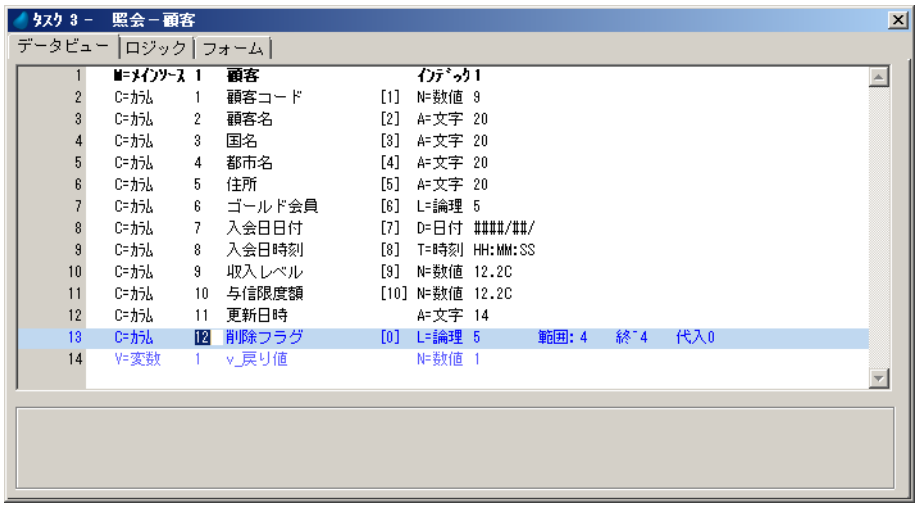
これでデータ操作についての一通りの説明は終了です。

［更新日付］と［削除フラグ］は、通常、フォームには表示しません。［フォーム］エディタを開いて削除しておいてください。

削除レコードを非表示にする

削除レコードは通常は表示しないようにしなければいけません。このための設定がプログラムに必要になります。メインソースの［削除フラグ］カラムが False の場合のみ表示するようにプログラムを修正してみましょう。

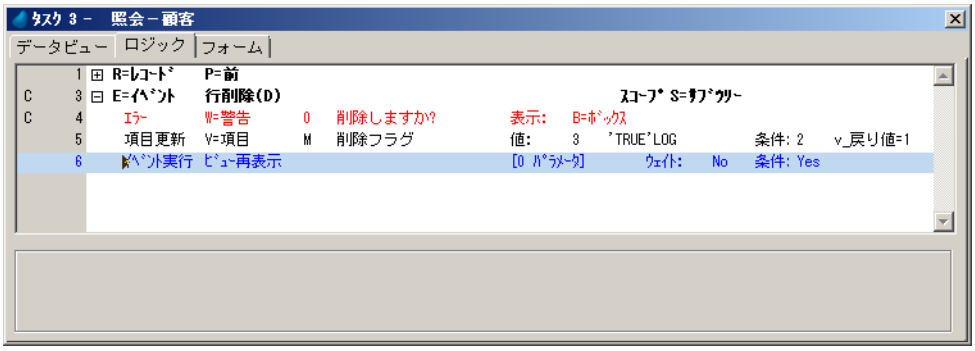
1. ＜照会 - 顧客＞にズームします。
2. ［データビュー］エディタを選択します。
3. ［削除フラグ］カラムの［範囲：最小］／［範囲：最大］の各特性に、＜ **'FALSE'LOG** ＞を設定します。



4. ［ロジック］エディタを選択し、［行削除］イベントの［イベント］ロジックユニットの最終行にパークします。
5. 一行追加（F4）して以下のように［イベント実行］処理コマンドを定義します。

処理コマンド	項目	式	条件
イベント実行	ビュー再表示		＜ v_ 戻り値＞ =1

この処理により、削除操作の結果が即時に反映されることになります。



削除操作を行っても実際にレコードが削除された訳ではないため、同じ インデックス情報を持つレコードを登録しようとすると重複エラーが発生します。実際のアプリケーション では、自動的に採番するなどして重複レコードが発生することを防止するためのプログラ ムの工夫が必要になります。

削除レコードはいつまで保存するのでしょうか？

クライアントのレコードは、削除フラグを設定するだけと説明いたしました。ではいつまで保存しておくのでしょうか？サーバにアップした後は、物理的に削除しても構わないため、サーバへの同期処理のタイミングで削除するか、手動で削除するように別途プログラムを用意する必要があります。

削除タイミングは、アプリケーションの仕様として任意に決定してください。

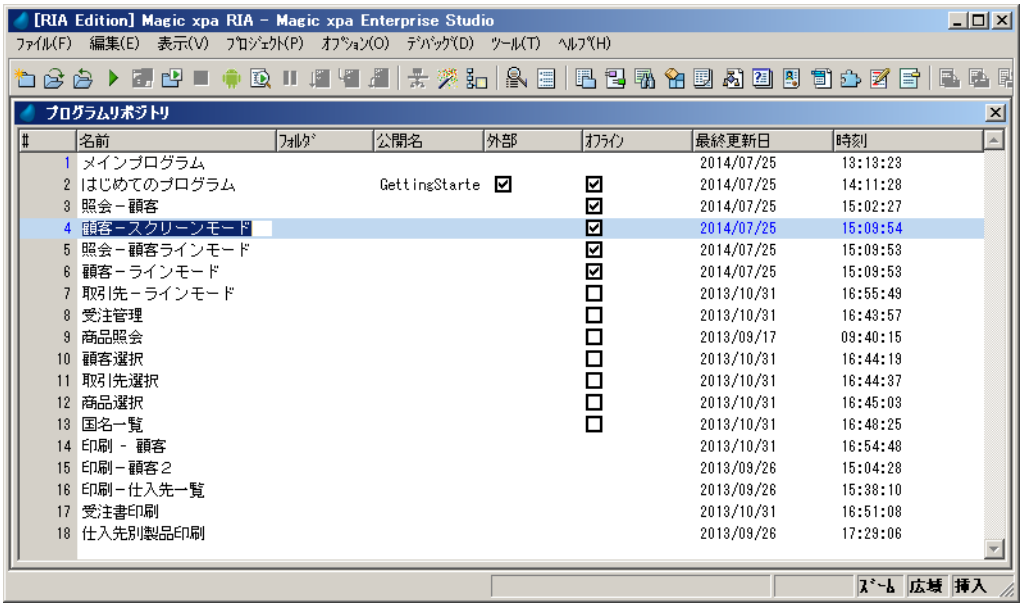
次のセクションからは、手動で同じプログラムを作成していきます。そのために現在実行中のプログラムを終了します。実行プログラムを終了すると、Magic xpa の実行エンジンが終了することに留意してください。

６．他のプログラムも修正

［プログラム］リポジトリには、＜顧客＞データソースを利用するプログラムとして、以下のものがあります。全てオフライン RIA プログラムに修正して確認してみましょう。

- 顧客－スクリーンモード
- 照会－顧客ラインモード
- 顧客－ラインモード

1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. 上記のプログラム行にパークし、［オフライン］カラムをチェックします。



プログラムの修正

［プログラム］リポジトリの上記の各プログラムに対してデータビューを追加します。

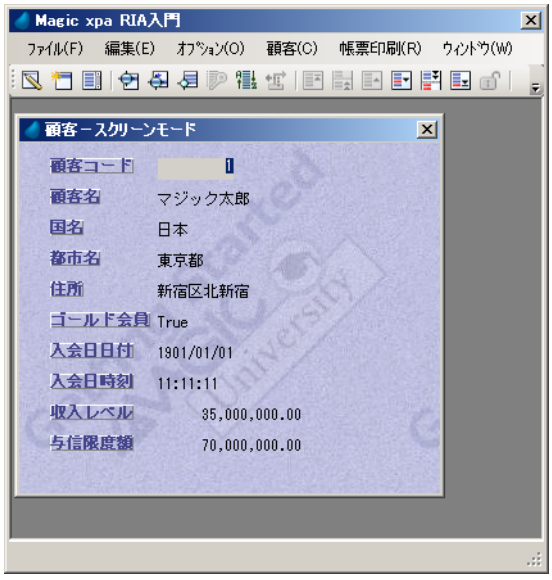
1. ズーム（F5）して、各プログラムの［データビュー］エディタを開きます。
2. ［更新日付］と［削除フラグ］のデータビューを追加します。
3. 変数項目＜**v_ 戻り値**＞（1 桁の数値型）を追加します。

4. ［ロジック］エディタに更新時の処理や削除の処理も追加しておいてください。

＜顧客－スクリーンモード＞プログラムの実行

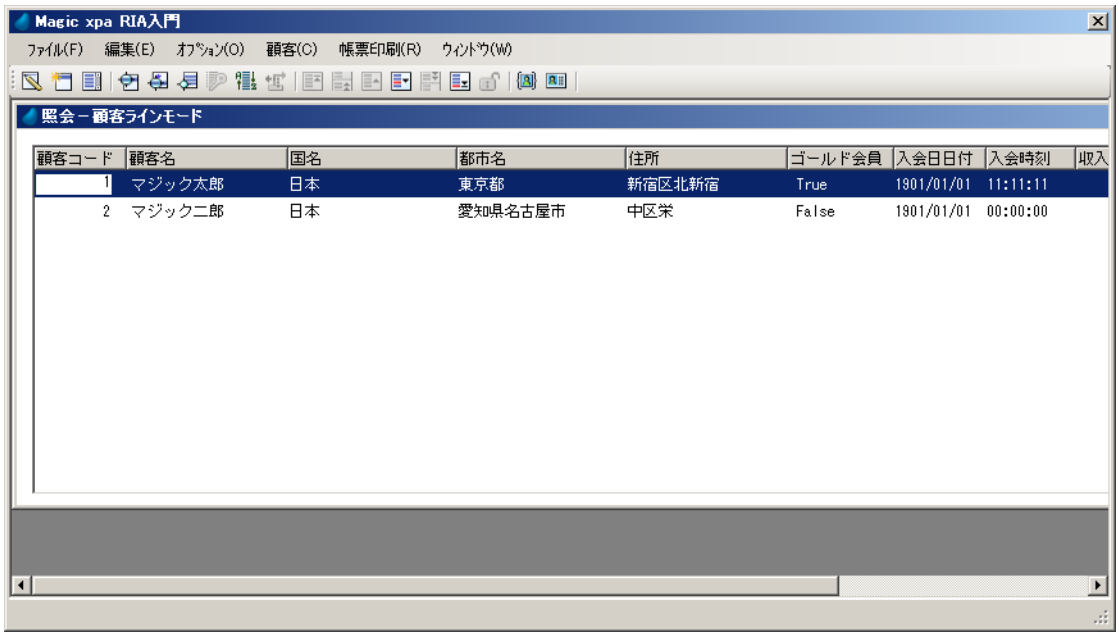
F7 キーを押下して、修正したプログラムを実行しましょう。

同じように動作し、＜照会－顧客＞プログラムで入力したデータが表示されることが確認できるはずです。また、背景イメージも表示されます。これは、すでにイメージデータがクライアント側にコピーされているためです。



＜照会－顧客ラインモード＞プログラムの実行

F7 キーを押下して、修正したプログラムを実行しましょう。これはテーブル形式のオフライン RIA プログラムですが、同じように動作し、＜照会－顧客＞プログラムで入力したデータが表示されることが確認できるはずです。



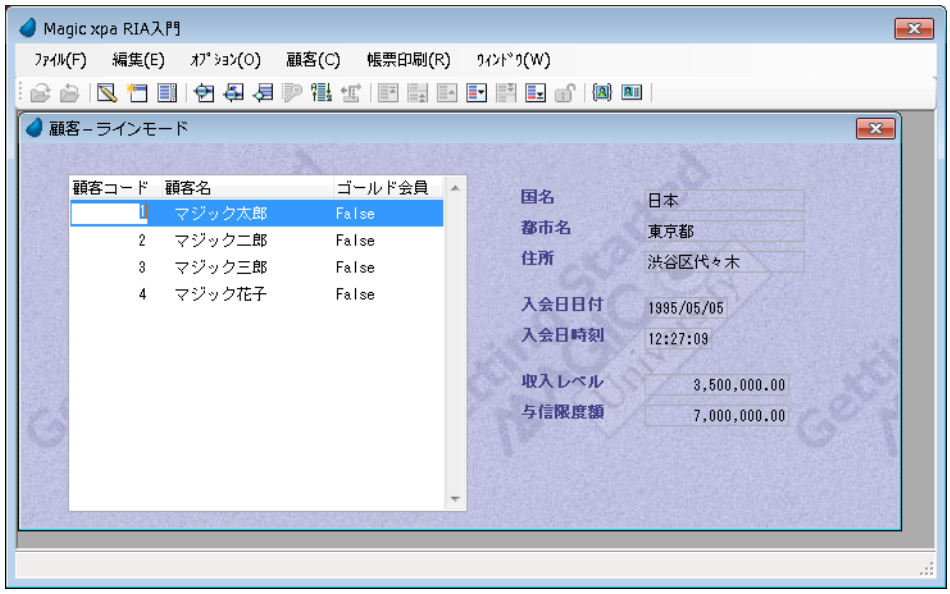
＜顧客ラインモード＞プログラムの実行

［プログラム］リポジトリで＜顧客－ラインモード＞プログラムの行（#6）にパークし、F7 キーを押下して実行してみましょう。

カーソルがテーブルのいずれかの項目にパークしているとき、'↑'、'↓'のキーを押下することで顧客レコードを移動することができます。

テーブルの外に配置してある項目データの表示内容は、そのときテーブルでカーソルがパークしている顧客レコードの内容に対応しています。

前のセクションで行ったのと同様の方法で、レコードの追加、内容の編集ができます。



７．ここまでの要約

ここまでの実習では、［データ］リポジトリ定義されているデータソース（テーブル）をローカル用に変更し、そのテーブル内容を参照するプログラムもオフライン RIA プログラムに変更しました。

データソースには、［更新日付］と［削除フラグ］の二つのカラムを追加します。

オフライン RIA プログラムには、追加した二つのカラムをデータビューとして追加します。

更新処理と削除処理が行われた場合、［更新日付］に UTC で更新日時を格納します。削除処理が行われた場合は、さらに削除フラグ］を True に設定し、実際のレコードは削除しないようにします。

オフライン RIA プログラムの制限事項

現在、オフライン RIA プログラムではレコードの位置付け、範囲指定、ソートの各処理を実行することができません。

８．練習問題

1. データソース＜取引先＞を同じように修正してください。

#	名前	データソース名	データベース	フォルダ	公開名
2	取引先	取引先	Local		

2. カラムを追加してください。

#	名前	モデル	型	書式
8	更新日時	0	A= 文字	14
9	削除フラグ	0	L= 論理	5

＜取引先－ラインモード＞プログラムを以下の手順で修正しましょう。

3. ［プログラム］リポジトリで＜照会－顧客＞プログラムの行にパークします。
4. ［オフライン］カラムをチェックします。
5. ［データビュー］エディタで、＜取引先＞データソースの［更新日付］と［削除フラグ］のカラムをデータビューに追加します。
6. ［削除フラグ］カラムの［範囲：最小］／［範囲：最大］の各特性に、**＜ 'FALSE'LOG ＞**を設定します。

7. また、変数項目として、**＜v_ 戻り値＞**（1 桁の数値型）を追加します。

8. ［ロジック］エディタを開き、［レコード後］ロジックユニットを作成し、以下の処理を追加します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes

9. ［行削除（D)］イベントに対する［イベント］ロジックユニットを作成し、以下の処理を追加します。

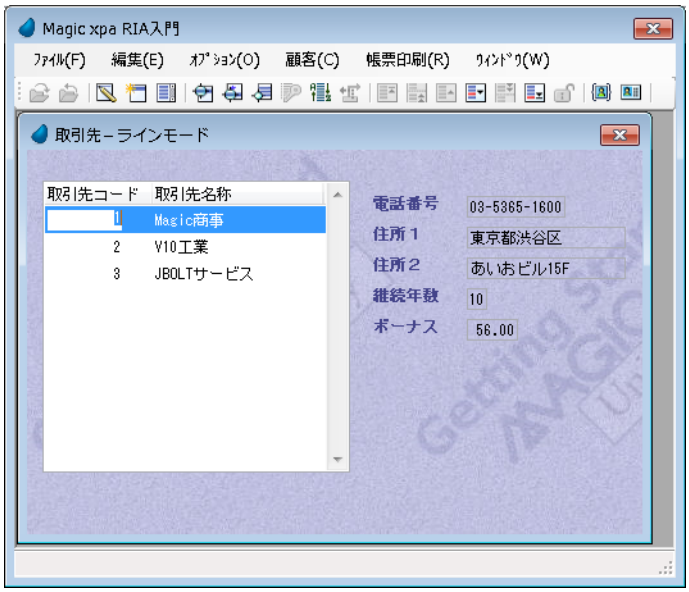
処理コマンド			式	条件
エラー	W= 警告	削除しますか？		Yes
項目更新	V= 項目	削除フラグ	'TRUE'LOG	＜v_ 戻り値＞=1
イベント実行	ビュー再表示			＜v_ 戻り値＞=1

［エラー］処理コマンド行で Ctrl+P を押下し、［特性］シートを開き以下のように特性を設定します。

- タイトル …… ＜**確認**＞
- イメージ …… Q= 疑問符
- ボタン …… K=OK キャンセル
- 戻り値 …… ＜v_ 戻り値＞

10. ＜取引先－ラインモード＞プログラムを実行してみましょう。
11. 次のような取引先データを登録しましょう。

取引先コード	取引先名称	電話番号	住所１	住所２	継続年数	ボーナス
1	Magic 商事	03-5365-1600	東京都渋谷区	あいおビル１５Ｆ	10	56.00
2	V10 工業	048-910-5510	埼玉県さいたま市		19	39.00
3	JBOLT サービス	06-8139-3939	大阪府大阪市	JBOLT ビル 12F	16	78.00



９．要約

本章では、次のことを学びました。

- ［データベース］テーブルに Local と SQLite の各 DBMS を使用するエントリを作成しました。
- ローカル用のデータソースを定義し、データソースの内容を表示し操作しました。
- また、ローカル用のデータソースに［更新日付］と［削除フラグ］のカラムを追加しました。
- RIA プログラムをオフライン用に変更し、更新／削除用の処理を追加しました。
- オフライン RIA でのデータの操作を実習しました。
  - レコードの登録
  - レコードの修正
  - レコードの削除

次章では、ローカルデータソースのリンク処理について学習します。

第 5 章 1 対 1 のデータリレーションシップ

本章では、1 対 1 のデータ リレーションシップを使用したオフライン RIA プログラムを作成する方法を実習します。

キーワード

- 照会リンク
- 登録リンク
- 書出リンク
- メインソース
- リンク評価

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- オフライン RIA プログラムで 1 対 1 のデータリレーションシップを実現する方法

参照

Magic ヘルプの［Magic xpa リファレンス／［データビュー］エディタ／リンク定義］

１．はじめに

本章では、オフライン RIA プログラムでの［リンク］コマンドの実習を通して、１対１のデータリレーションシップを実現する方法を学びます。

２．データソースへのリンク

Magic xpa では、［データビュー］エディタにリンクヘッダ行を定義することでデータソースを接続し、１対１データリレーションシップを確立します。

二つのデータソースを接続するためには、どちらにも同じカラムが含まれていることが必要です。通常の場合、共通となるカラムはリンク先となるデータソースのインデックスセグメントになっています。

タスク実行の間、Magic xpa はメインソースの各レコードに対し、リンク先のデータソースから 1 レコードずつ読み込みます。

リンクの種類

Magic xpa の［リンク］コマンドには 5 種類あり、それぞれ動作が異なります。

- L= 照会リンク …… リンク先データソースの既存レコードへの接続と読込みを行いたい時に使用します。対応する既存レコードが存在しないときは、データビューには何も読込まれません。
- W= 書出リンク …… 照会リンクと似ていますが、既存レコードが存在している時には読込みを、存在しない時にはリンク先のデータソースにレコードを作成したい場合に使用します。
- C= 登録リンク …… リンク先データソースに新規レコードを作成したい場合に使用します。

以下の二つは、Local データベースでは使用できません。

- I= 結合リンク …… プログラムのデータビューには、リンク先データソースに対応する既存レコードが存在するメインソースのレコードのみが読込まれます。
- O= 外部リンク …… このコマンドは、［照会リンク］と同じ動作をしますが、データビューの扱いが異なります。プログラムのデータビューには、リンク先データソース内に既存レコードが存在するか否かにかかわらず、メインソースの全てのレコードが読込まれます。

リンク定義の制限事項

オフライン RIA プログラムは、サーバ側のデータソースにアクセスできないため、オフライン用データソース間のリンクのみ定義することができます。また、同じタスク内にオンライン用のデータソースとオフライン用のデータソースを混在することができない制約があり、オンライン RIA プログラム内でも、オンライン用とオフライン用のデータソース間のリンクを定義することができません。

３．受注管理プログラムの実習

このセクションでは、＜受注管理＞プログラムをオフライン用に作成し、＜受注＞データソースのデータと＜顧客＞データソースから［リンク］コマンドによって取得した追加データとを表示させてみましょう。

この実習は二つのステップで構成されます。

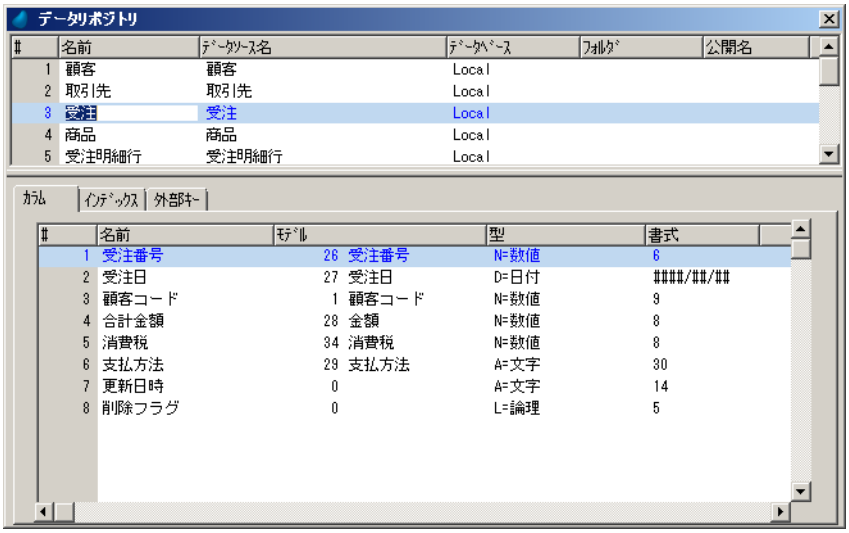
- ＜受注＞データソースの修正
- ＜受注管理＞プログラムの修正

＜受注＞データソースの修正

1. ［プロジェクト］メニューから［データ］（Shift+F2）を選択し、［データ］リポジトリを開きます。
2. ＜受注＞データソースの行（#3）にパークします。
3. ［データベース］カラムが＜ Local ＞になっていることを確認します。

4. 以下のカラムを追加します。

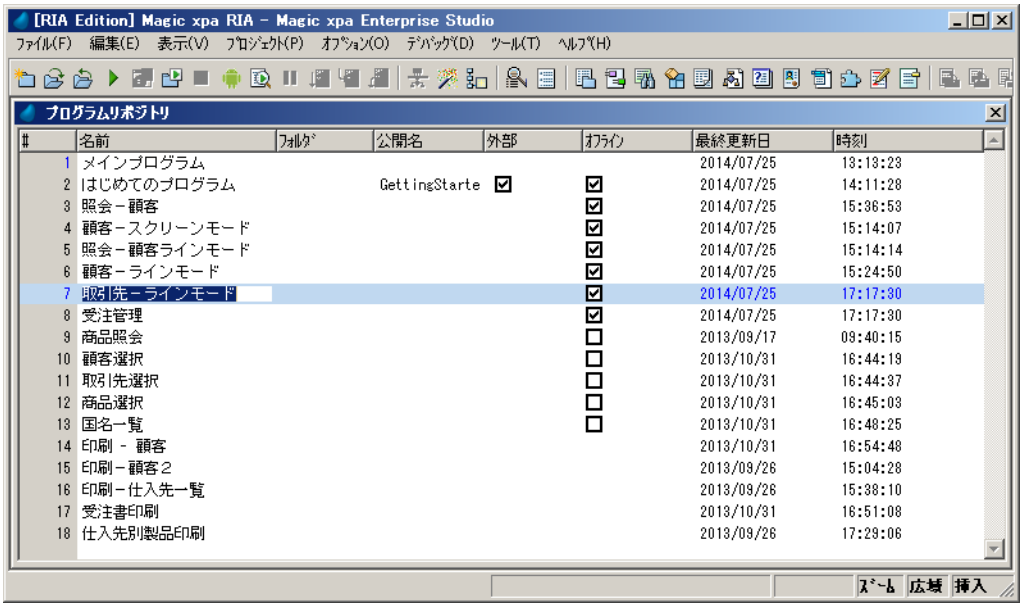
#	名前	モデル	型	書式
7	更新日時	0	A= 文字	14
8	削除フラグ	0	L= 論理	5



これで、＜受注＞データソースの修正が終了しました。

＜受注管理＞プログラムの修正

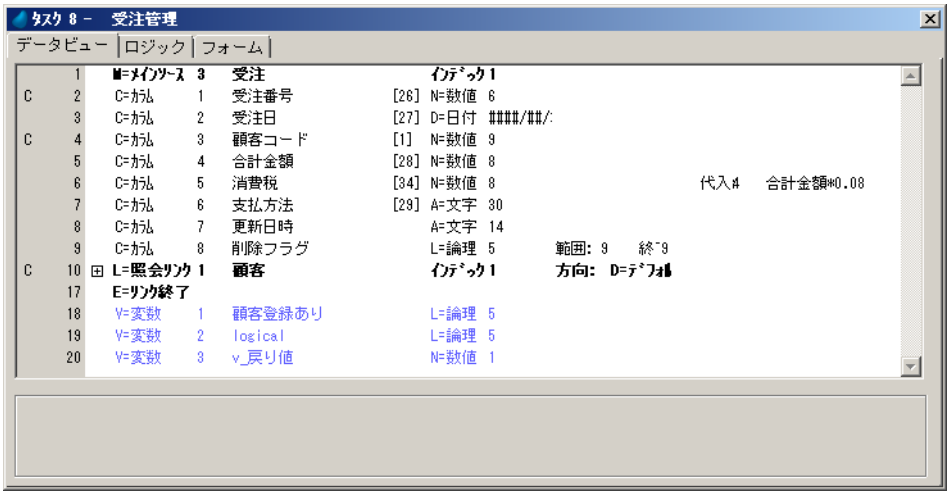
1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. ＜受注管理＞の行にパークし、［オフライン］カラムをチェックします。



3. ズームして＜受注管理＞プログラムを開きます。

データビューの追加

1. ［データビュー］エディタを開きます。
2. メインソースに対する［更新日付］と［削除フラグ］のカラムをデータビューとして追加します。
3. ［削除フラグ］カラムの［範囲：最小］／［範囲：最大］の各特性に、**＜ 'FALSE'LOG ＞**を設定します。
4. また、変数項目**＜ v_ 戻り値＞**を追加します（1 桁の数値型）。



修正／削除処理の追加

5. ［ロジック］エディタを開き、［レコード後］ロジックユニットに以下の処理を追加します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes

6. ［行削除（D)］イベントに対する［イベント］ロジックユニットを作成し、以下の処理を追加します。

処理コマンド			式	条件
エラー	W= 警告	削除しますか？		Yes
項目更新	V= 項目	削除フラグ	'TRUE'LOG	＜ v_ 戻り値＞ =1
イベント実行	ビュー再表示			＜ v_ 戻り値＞ =1

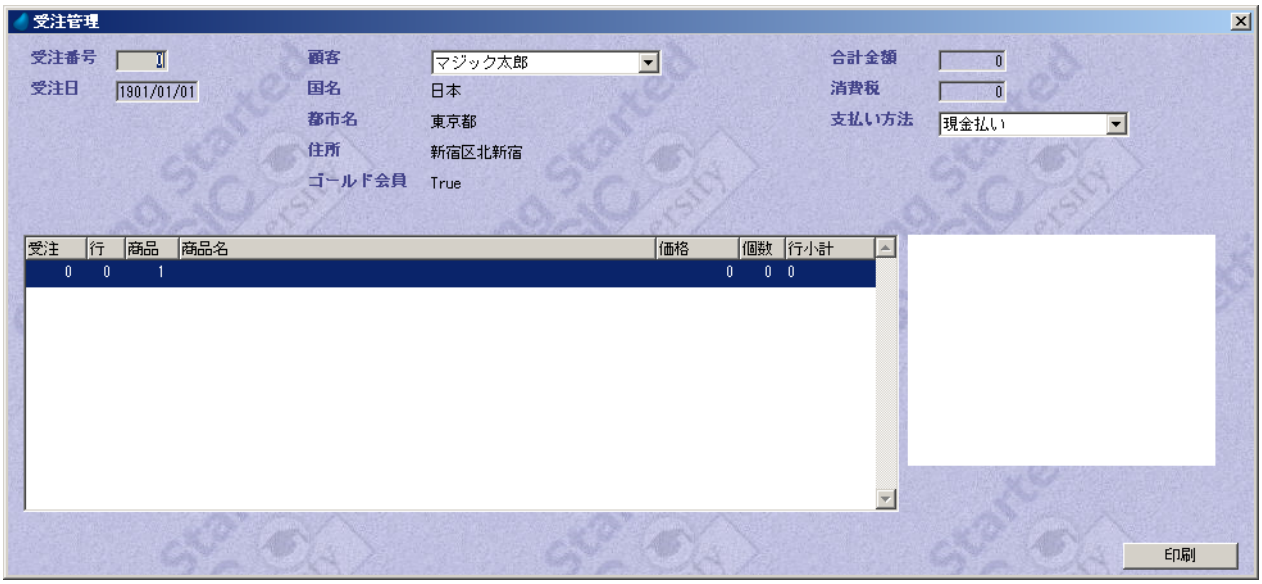
［エラー］処理コマンド行で Ctrl+P を押下し、［特性］シートを開き以下のように特性を設定します。

- **タイトル …… ＜確認＞**
- **イメージ …… Q= 疑問符**
- **ボタン …… K=OK キャンセル**
- **戻り値 …… 変数＜ v_ 戻り値＞**

7. Enter キーを押下してプログラムを閉じ、［プログラム］リポジトリに戻ります。

＜受注管理＞プログラムの実行

1. F7 キーを押下して＜受注管理＞プログラムを実行しましょう。
2. ＜受注番号＞欄に **＜ 1 ＞** と入力してください。
3. ＜受注日＞欄に、例えば今日の日付を入れてください。
4. ＜顧客＞欄のコンボボックスを選択してください。
5. Tab キーを押下して、サブフォームに移動します。



４．これまでの要約

作成した＜受注管理＞プログラムのデータビューには二つのデータソースのカラムが定義してあります。

- ＜受注＞データソースのカラム
- ＜顧客＞データソースのカラム

この二つのデータソースは、どちらもローカルのデータベースを使用して共通の値を持つカラム、すなわち＜顧客コード＞を含んでいます。

これらのデータソースのデータ関係は、＜顧客コード＞が＜顧客＞データソースではユニークな項目で、値が重複しないため、１対１のリレーションであるといえます。

プログラムでは、照会リンクを用いて顧客詳細情報を読み込みました。(ただし、対応する顧客データが存在する場合のみ)

＜受注管理＞プログラムのフォームでは＜受注＞情報詳細と、＜顧客＞情報の一部を表示するようになっています。

ここまでのまとめ

ローカルデータソース同士であれば［リンク］コマンドを利用して、オンライン RIA と同じように連携させることができます。

５．練習問題

この練習問題では、まず最初に＜商品＞データソースを修正します。

次に商品データを表示させるための＜商品照会＞プログラムを作成し､[リンク]コマンドを利用して＜取引先＞の詳細データをデータビューに読込み、表示させるようにします。

その際、取引先が存在するかどうかの確認を［リンク］コマンドの［戻り値］特性を使用して行うようにします。

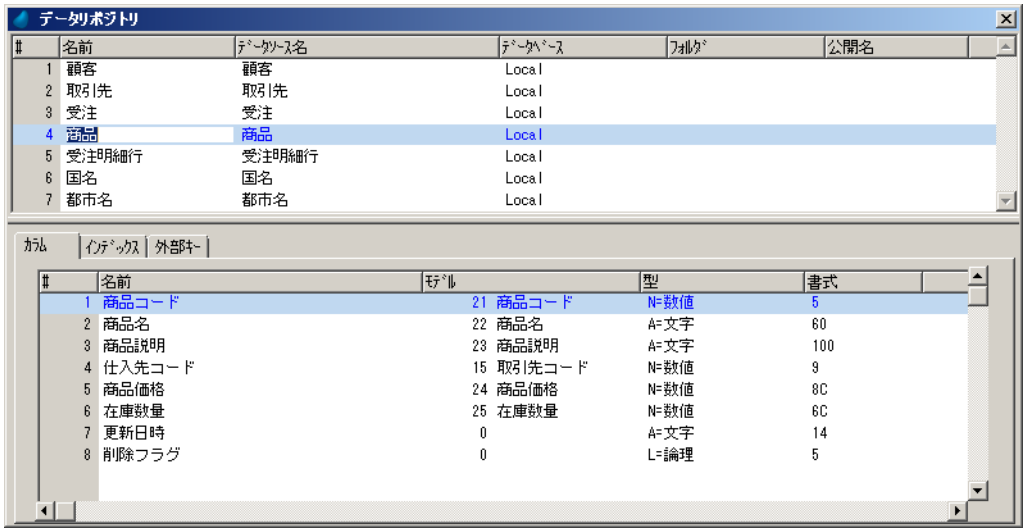
＜商品＞データソースの定義

1. ［データ］リポジトリに＜商品＞データソースを次のように定義してください。

#	名前	データソース名	データベース	フォルダ	公開名
4	商品	商品	Local		

2. カラムを追加してください。

#	名前	モデル	型	書式
7	更新日時	0	A= 文字	14
8	削除フラグ	0	L= 論理	5

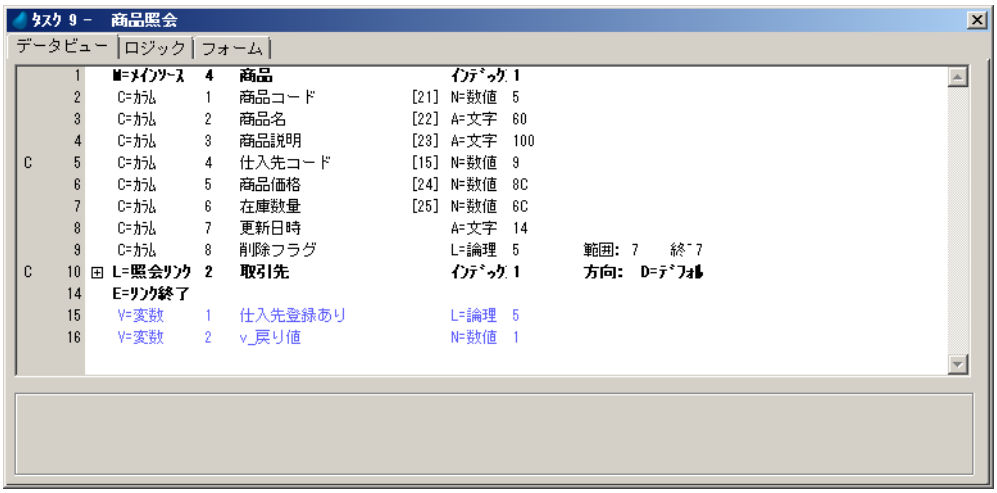


＜商品照会＞プログラムの修正

以下の手順にしたがって、＜商品照会＞プログラムを修正してください。

1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. ＜商品照会＞の行にパークし、［オフライン］カラムをチェックします。

3. ズームしてプログラムを開きます。
4. ［データビュー］エディタを開きます。
5. メインソースに対する［更新日付］と［削除フラグ］のカラムをデータビューとして追加します。
6. ［削除フラグ］カラムの［範囲：最小］／［範囲：最大］の各特性に、**＜ 'FALSE'LOG ＞**を設定します。
7. 変数項目**＜ v_ 戻り値＞**を追加します（1 桁の数値型）。



修正／削除処理の追加

8. ［ロジック］エディタを開き、［レコード後］ロジックユニットに以下の処理を追加します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes

9. ［行削除（D)］イベントに対する［イベント］ロジックユニットを作成し、以下の処理を追加します。

処理コマンド	項目	式		条件
エラー	W= 警告	削除しますか？		Yes
項目更新	V= 項目	削除フラグ	'TRUE'LOG	＜ v_ 戻り値＞ =1
イベント実行	ビュー再表示			＜ v_ 戻り値＞ =1

［エラー］処理コマンド行で Ctrl+P を押下し、［特性］シートを開き以下のように特性を設定します。

- タイトル …… ＜確認＞
- イメージ …… Q= 疑問符
- ボタン …… K=OK キャンセル
- 戻り値 …… ＜ v_ 戻り値＞

10.Enter キーを押下してプログラムを閉じ、［プログラム］リポジトリに戻ります。

プログラムの実行

1. F7 キーを押下して＜商品照会＞プログラムを実行させてください。
2. タスクの初期モードは「修正」モードですが、最初は商品データが存在しないので、「登録」モードでプログラムが動作しています。

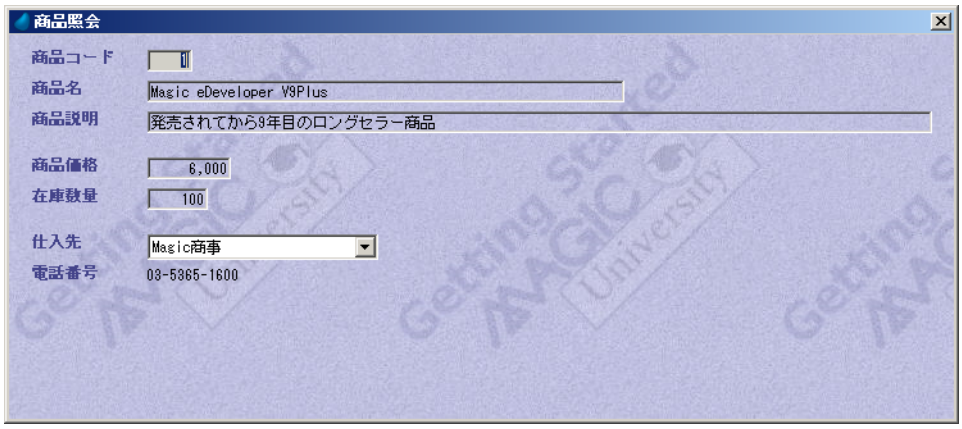
3. たとえば、次のようにデータを登録してゆきましょう。（必ずしも実際の商品ではありません）

商品コード	商品名	商品説明	商品価格	在庫数量	仕入先
1	Magic eDeveloper V9Plus	発売されてから 9 年目のロングセラー商品	60,000	1,000	Magic 商事
2	Magic xpa Studio	モバイルに対応しましたした	600,000	100	Magic 商事
3	Magic uniPaaS V1Plus	RIA の開発ができます。	600,000	500	Magic 商事
4	Magic Studio V10	2006 年末デビューしました	600,000	1,000	V10 工業
5	Magic xpi V3.2	SAP Certified Integration	2,000,000	20	jBOLT サービス
5	Apple 2 GB iPod Nano	Apple 2 GB iPod nano	4,000	20	

4. Page Down/Page Up キーを押下して、レコードを移動してみましょう。

このとき、＜仕入先名＞と＜電話番号＞が一緒に変更されることに留意してください。
＜仕入先コード＞を変更して、［リンク式の再計算］が行われることを確認してみましょう。

＜仕入先コード＞を指定しないとエラーとなることを確認してください。



６．要約

本章では、オフライン RIA プログラムにおける 1 対 1 のデータ リレーションシップの実習プログラムを作成しました。

このタイプのリレーションは、プログラムのメインソースのレコードと、他のデータソースの特定のデータとの関連付けを行うために使用されます。

練習問題では＜商品照会＞プログラムを作成し、商品の詳細データと、［リンク］コマンドを使用することで、仕入先データの一部を取得し、表示するようにしました。また戻り値を利用して、＜仕入先コード＞がすでに登録済かどうかの確認処理を追加しました。

このように、通常のオンライン RIA と同じようにプログラムを作成することができます。

第 6 章 １対多のデータリレーションシップ

本章では１対多のデータリレーションシップを利用したオフライン RIA のプログラミング方法を実習します。

キーワード

- １対多のデータリレーションシップ
- タスクとプログラム
- 親タスク
- サブタスク
- ［サブフォーム］コントロール

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- １対多のリレーションを持つ二つのローカルデータソースを処理するプログラム

参照

Magic ヘルプの［Magic xpa リファレンス／表示フォームとコントロール／表示コントロール／サブフォームコントロール］

１．はじめに

第 5 章では１対１のデータリレーションシップについて学びました。

もう一つの一般的なデータリレーションシップとして、１対多のデータリレーションシップがあります。

１対多のデータリレーションシップは、あるデータソースの１レコードに対し、別のデータソースの複数レコードが対応関係にある状態を言います。

Magic xpa では、１対多のデータリレーションシップは、二つのタスクまたはプログラムによって構成します。

この方法では、最初のタスク（通常、親タスクと呼びます）がメインとなるレコードを表示し、二番目以降のタスク（サブタスク）が、対応する”多くの”レコードを表示します。

Magic xpa では［サブフォーム］コントロールを使用することで、親タスクのフォームのなかにサブタスクのフォームを表示させ、親タスク側で共通項目が変わるたびにサブタスク側の表示を再描画させることができます。

本章ではオフライン RIA プログラムによる１対多のデータリレーションシップの例として、＜受注管理＞プログラムをオフライン用に修正してみましょう。

２．商品選択プログラムの修正

１対多のデータリレーションシップの実習に取り組む前に、まず＜商品＞データソースのための選択プログラムを修正します。

商品選択プログラムは、前の章で実習した方法と同じように修正します。

写真の論理名を定義

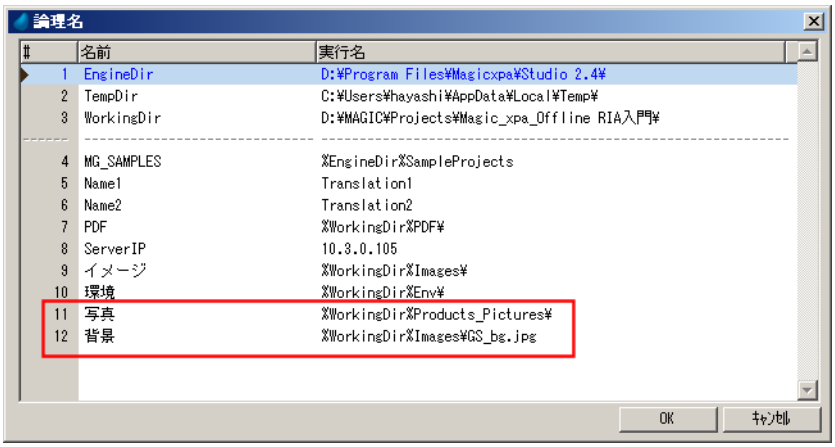
本コースのデータフォルダ内には商品のイメージデータファイルの入ったサブフォルダ（Products_Pictures）があります。

各イメージファイルの名前は、商品名と同じになっています。

商品のイメージを表示させるために、論理名を一つ定義し、イメージファイルの入っているフォルダの場所を指すようにします。

1. ［オプション］メニューから［設定／論理名］を選択します。
2. 最終行にパークします。
3. 一行作成（F4）します。

4. ［名前］カラムに**＜写真＞**と入力します。
5. ［実行名］カラムに**＜ %WorkingDir%Products_Pictures¥ ＞**と入力します。
6. ［OK］ボタンをクリックします。



＜商品選択＞プログラムの修正

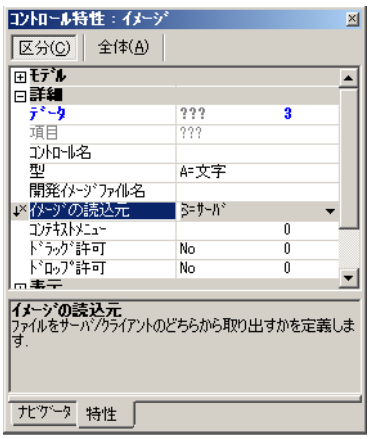
7. ［プログラム］リポジトリ（Shift+F3）を開きます。
8. ＜商品選択＞行にパークし、［オフライン］カラムをチェックします。
9. ズーム（F5）して、＜商品選択＞プログラムを開きます。

フォームの［イメージ］コントロールについて

オンライン RIA プログラムでは［イメージ］コントロールはサーバ上の画像データを表示しますが、オフライン RIA プログラムはクライアント側のキャッシュフォルダ内のデータを表示することになります。このため画像データの同期処理を追加する必要があります。

［イメージ］コントロールの特性

［イメージ］コントロールの特性の［イメージの読込元］特性は＜ S= サーバ＞のままにしておいてください。



［イメージの読込元］特性を＜ C= クライアント ＞にすると、指定された画像ファイルをキャッシュフォルダ内のファイルとして解釈されなくなります。

画像データの同期処理の追加

この処理は、「第 3 章 : 画像データの同期処理」（10 ページ）で既に反映されています。

３．受注明細行管理タスクの修正

このセクションでは、＜受注管理＞プログラムのサブタスクを修正してみましょう。

サブタスクとして＜受注明細行管理＞タスクは、＜受注＞データソースと＜受注明細行＞データソースとの間に 1 対多のデータリレーションシップを構築します。

まず＜受注明細行＞データソースの定義をしましょう。

受注明細行データソースの修正

1. ［データ］リポジトリ（Shift+F2）を開きます。
2. #5 ＜受注明細行＞にパークします。
3. ［データベース］カラムが＜ Local ＞になっていることを確認して下さい。

＜受注明細行＞カラムの追加

4. 次のようにカラムを追加します。

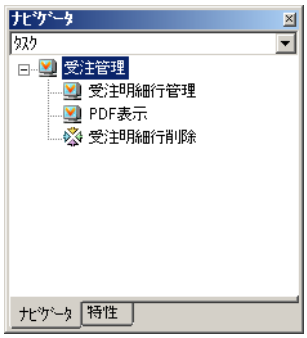
#	名前	モデル	型	書式
7	更新日時	0	A= 文字	14
8	削除フラグ	0	L= 論理	5

（サブ）タスクの修正

ここでは<受注管理>タスクのサブタスクとなる<受注明細行管理>を修正します。

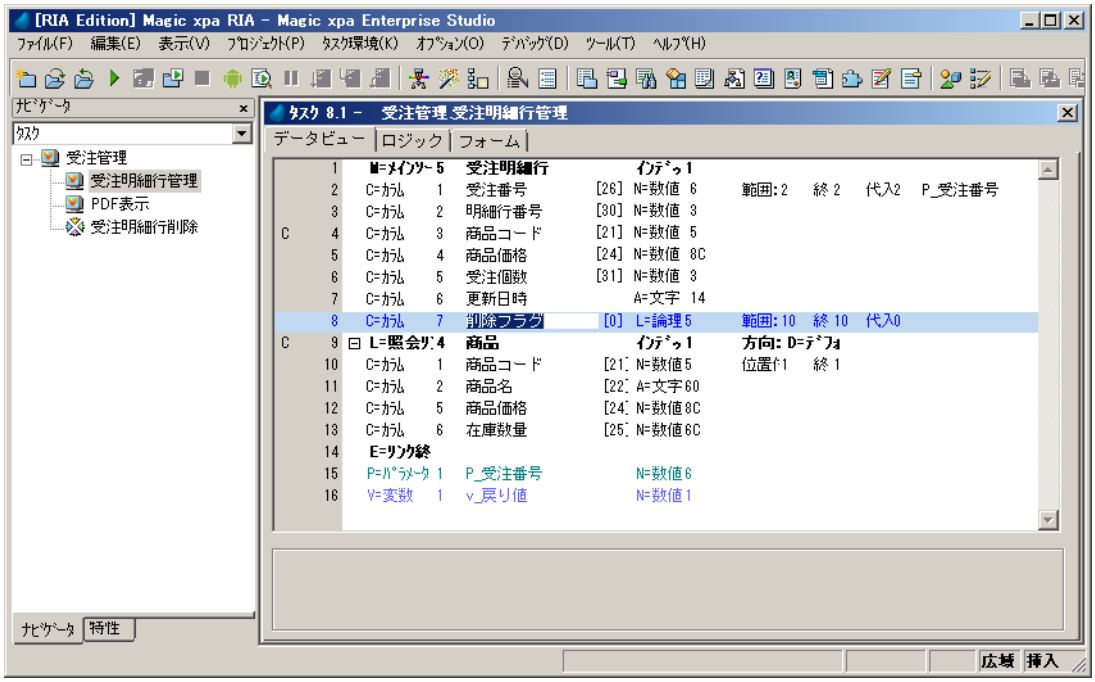
タスク特性の設定

1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. <受注管理>プログラムにズームします。
3. ［表示］メニューから［ナビゲータ］を選択し、［ナビゲータ］ペインを表示させます。
4. ［ナビゲータ］ペインで、<受注明細行管理>エントリをクリックしてタスクを開きます。



サブタスクのデータビュー定義

1. ［データビュー］エディタを選択します。
2. ＜受注明細行＞（#5） に追加した［更新日時］と［削除フラグ］カラムをデータビューに追加します。
3. ［削除フラグ］カラムの［範囲：最小］／［範囲：最大］の各特性に、＜ **'FALSE'LOG** ＞を設定します。
4. また、変数項目＜ **v_ 戻り値**＞を追加します（1 桁の数値型）。



修正／削除処理の追加

5. ［ロジック］エディタを開き、［レコード後］ロジックユニットに以下の処理を追加します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes

6. ［行削除（D)］イベントに対する［イベント］ロジックユニットを作成し、以下の処理を追加します。

処理コマンド			式	条件
エラー	W= 警告	削除しますか？		Yes
項目更新	V= 項目	削除フラグ	'TRUE'LOG	＜ v_ 戻り値＞ =1
イベント実行	ビュー再表示			＜ v_ 戻り値＞ =1

［エラー］処理コマンド行で Ctrl+P を押下し、［特性］シートを開き以下のように特性を設定します。

- タイトル …… **＜確認＞**
- イメージ …… Q= 疑問符
- ボタン …… K=OK キャンセル
- 戻り値 …… ＜ v_ 戻り値＞

7. Enter キーを押下してプログラムを閉じ、［プログラム］リポジトリに戻ります。

＜受注管理＞プログラムの動作確認

ここまでの作業で、＜受注明細行管理＞サブタスクを作成し、そのフォームを＜受注管理＞フォーム内に組込みました。
これで 1 対多のデータリレーションシップを取り扱うプログラムが完成したことになります。

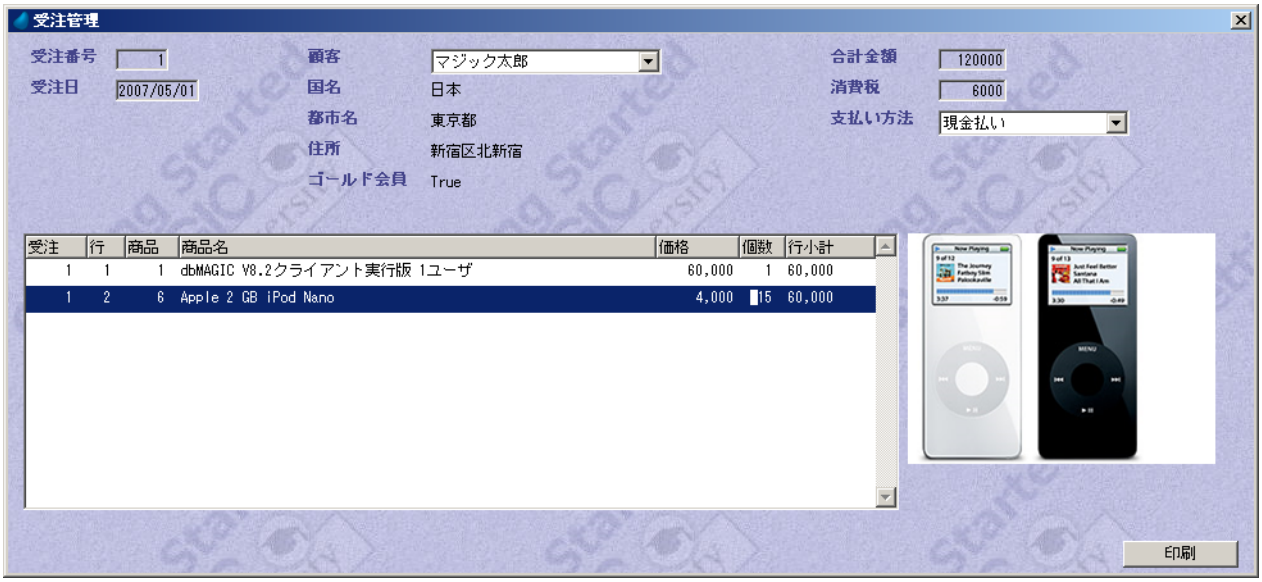
これから＜受注管理＞プログラムを動かして、1 対多のデータリレーションシップがどのようなものか確認しましょう。

1. ［プログラム］リポジトリで＜受注管理＞プログラム（#8）にパークし、F7 キーを押下してプログラムを実行しましょう。
2. Tab キーを押下して、［サブフォーム］コントロール内にカーソルを移動させます。
3. 下記のように商品を追加してみましょう。

受注	行	商品	商品名	価格	個数	行小計
1	1	1	Magic eDeveloper V9Plus	45,000	1	(45,000)
1	2	6	Apple 2 GB iPod Nano	4,000	15	(60,000)

＜合計金額＞や＜消費税＞は、自動的に計算されます。

4. Tab キーを押下して＜支払い方法＞コントロールに移動してください。
5. ＜現金払い＞を選択しましょう。



6. プログラムを終了しましょう。

４．練習問題

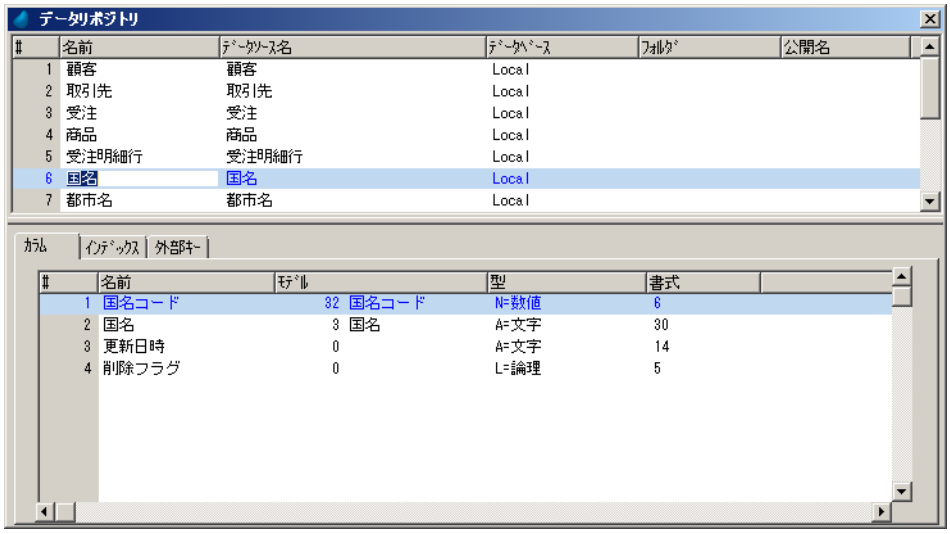
国名と都市名とは 1 対多のデータリレーションシップがあります。

練習問題として、これまで学習したことを応用し、国名とその国の都市名とを一覧表示させるプログラムを作成しましょう。

国名データソースの修正

1. ［データ］リポジトリ（Shift+F2）を開きます。
2. ＜国名＞行にパークします。［データベース］カラムが＜ Local ＞に設定されていることを確認します。
3. 次のようにカラムを追加します。

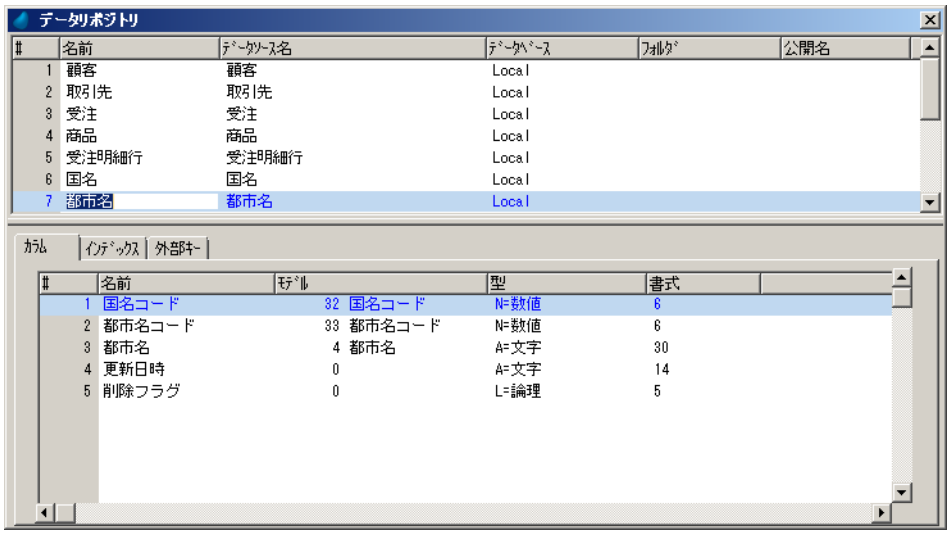
#	名前	モデル	型	書式
3	更新日時	0	A= 文字	14
4	削除フラグ	0	L= 論理	5



都市名データソースの修正

1. ［データ］リポジトリ（Shift+F2）を開きます。
2. ＜都市名＞行にパークします。［データベース］カラムが＜ Local ＞に設定されていることを確認します。
3. 次のようにカラムを追加します。

#	名前	モデル	型	書式
4	更新日時	0	A= 文字	14
5	削除フラグ	0	L= 論理	5



＜国名一覧＞プログラムの修正

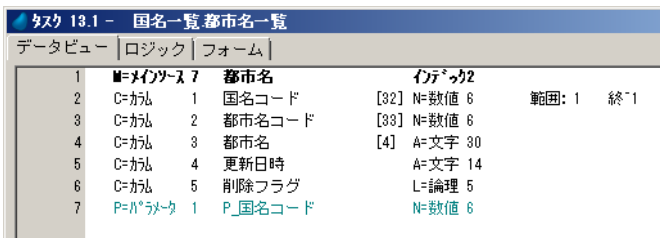
データビューの設定

1. ［プログラム］リポジトリを開きます。
2. ＜国名一覧＞行にパークし、［オフライン］カラムをチェックします。

3. ［データビュー］エディタでデータビューを追加します。

サブタスク＜都市名一覧＞の修正

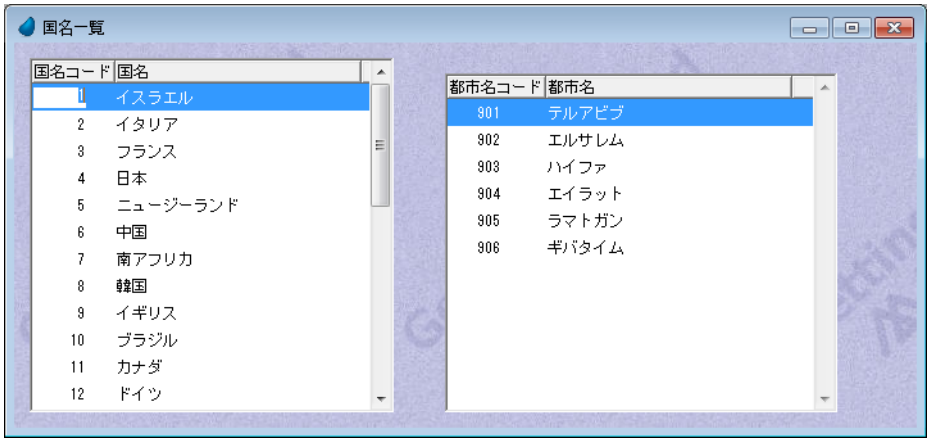
1. ［表示］メニューから［ナビゲータ］を選択し、［ナビゲータ］ペインを表示させます。
2. ［ナビゲータ］ペインで＜都市名一覧＞タスクアイコンをクリックしてタスクを開きます。
3. ［データビュー］エディタでデータビューを追加します。



＜国名一覧＞プログラムの実行

1. ［プログラム］リポジトリで＜国名一覧＞（#13）にパークし、F7 キーを押下して実行しましょう。
2. 適当に国と都市とを登録してください。

3. カーソルを動かして、国名と都市名が連動して表示されることを確認しましょう。



レコードの修正／削除の処理も反映してみてください。

５．要約

本章では、オフライン RIA プログラムにおける 1 対多のデータリレーションシップについて学習しました。

オンライン RIA と同じように 1 対多のデータリレーションシップを利用して、データソース定義の効率や保守性を高めることができることが分かりました。

また実際のプログラムを作成し、親タスクとサブタスクとで 1 対多のデータリレーションシップを実現しました。

次章では、オフライン RIA プログラムにおけるプログラムの呼び出し方法について説明します。

第 7 章 オフライン RIA プログラムの呼び出し

本章では、オフライン RIA プログラムの選択肢の一覧からエンドユーザが値を選択できるようにする選択テーブルプログラムを修正することで、オフライン RIA プログラムからのプログラム呼び出しについて学習します。

キーワード

- オンライン RIA プログラム
- エラー表示

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- オフライン RIA プログラムからのプログラム呼び出しについて

参照

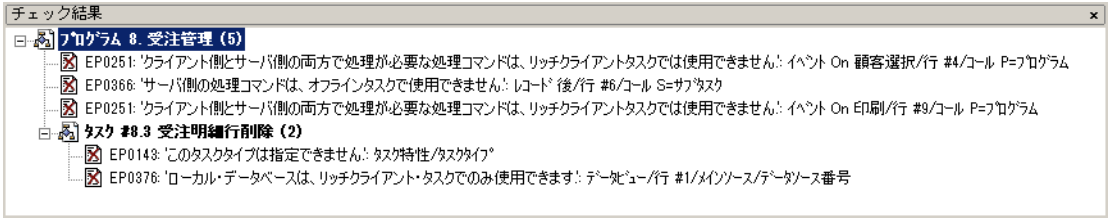
Magic ヘルプの［Magic xpa リファレンス／プログラム／タスク特性］

同様に、［Magic xpa リファレンス／表示フォームとコントロール／表示コントロール］

１．はじめに

前章で修正した＜受注管理＞プログラムの［ロジック］エディタを開いてみてください。行番号欄の左側に＜ E ＞と表示されている行があります。これは、この行がエラー箇所であることを示しています。

プログラムを閉じて構文チェック（F8）を実行してみてください。［チェック結果］ペインに以下のようなエラーが表示されます。



EP0251 のエラーは、オフライン RIA プログラムからオンライン RIA プログラムを呼び出すように定義された場合に表示されます。このエラーを修正するには、＜顧客選択＞プログラムをオフライン化する必要があります。

この章では、＜顧客選択＞プログラムの呼び出しに対するエラーを修正して、顧客データの選択処理ができるようにします。

以下のエラーについては、次章で対応します。

- EP0366 …… サーバ側で実行されるバッチタスクを呼び出している場合に発生します。
- EP0143 …… 親タスクがオフラインに変更されたためにリッチクライアント以外のタスクタイプが定義された子タスクのタスクタイプに設定が不正になったことを示しています。
- EP0376 …… バッチタスクにローカルデータソースが定義されていることを示しています。

２．選択テーブルプログラム

［選択テーブルプログラム］は、データを一覧表形式で表示しエンドユーザが値を選択できるようにした RIA プログラムです。

［選択テーブルプログラム］を使用するときは、次の二点に留意してください。

- まず［選択テーブルプログラム］を作成します。
- 作成した［選択テーブルプログラム］をホストプログラムから［コール］処理コマンドを使用して呼び出します。

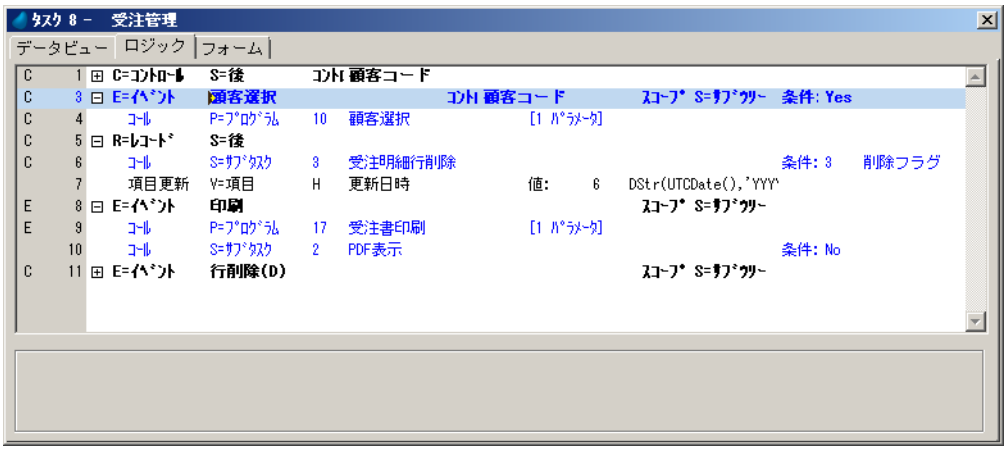
顧客選択テーブルプログラムの修正

このセクションでは、＜顧客選択＞プログラムをオフライン RIA プログラムに修正し、選択テーブルプログラムとして動作するようにします。

1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. ＜顧客選択＞行にパークします。
3. ［オフライン］カラムをチェックします。

これだけです。

この時点で、もう一度＜受注管理＞プログラムの［ロジック］エディタを開いてみてください。3 ～ 4 行目の番号欄の左側の表示は＜ C ＞に変更されました。これは、この行がクライアント側の処理であることを示しています。



この時点では、［印刷］ イベント に対する［イベント ］ロジック ユニット にまだエラーがありますが、次章以降でで対応します。

ここまでのまとめ

この作業で＜顧客選択＞プログラムをオフラインに変更することで、＜受注管理＞プログラムから呼び出せるようになりました。

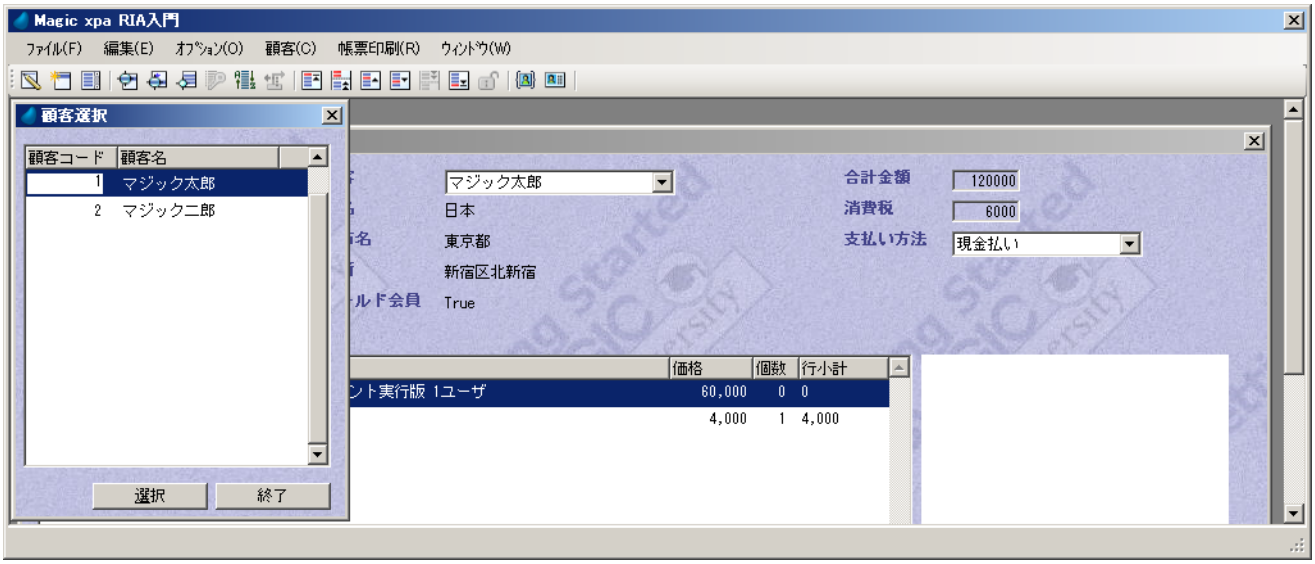
３．［選択テーブルプログラム］の呼び出し

［選択テーブルプログラム］のテスト

ここでは＜受注管理＞プログラムを実行させ、作成した＜顧客選択＞プログラムを起動させてみましょう。

1. 変更内容を保存して、＜受注管理＞プログラムを閉じます。
2. F7 キーを押下して、＜受注管理＞プログラムを実行します。
3. Tab キーを押下して、＜顧客＞コントロールにパークします。

4. ＜顧客＞からズーム（F5）します。＜顧客選択＞ウィンドウが表示され、顧客一覧から選択できるようになります。



４．要約

本章では、オフライン RIA プログラムからオフライン RIA プログラムを呼び出す場合のプログラムについて学習しました。

オフライン RIA プログラムからオンライン RIA プログラムを直接呼び出すことは基本的にはできません。このようなプログラムは、エラーとなります。呼び出すプログラムもオフラインにする必要があります。

オフライン RIA プログラムから間接的にオンライン RIA プログラムを呼び出す方法は、「第 10 章：帳票印刷」（70 ページ）で学びます。

本章でインタラクティブなリッチクライアントタスクに関する説明は終わります。

次章からは非インタラクティブタスク（バッチタスク）について学びます。

第 8 章 非インタラクティブタスクのアプリケーションエンジン

本章では、オフライン RIA における非インタラクティブタスクの扱いについて学習します。

キーワード

- タスクタイプ
- 非インタラクティブなオフラン RIA プログラム
- データの整合性

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- 非インタラクティブタスクでの Magic エンジン動作
- インタラクティブタスクと非インタラクティブタスクの違い

参照

Magic ヘルプの［Magic xpa リファレンス／ Magic エンジン］

１．はじめに

これまでの章では、インタラクティブ（対話型）なプログラムについて学びました。一方、実際の運用を考えたとき、ユーザの介入なしで処理を実行するプログラムも必要になります。

Magic xpa ではこのようなプログラムをバッチタスク（プログラム）または、非インタラクティブタスク（プログラム）と呼んでいます。

これからの章では、オフライン RIA における非インタラクティブタスクの概念や動作について学んで行きます。また実習を通して、この非インタラクティブタスクの作成法を学びます。

非インタラクティブタスクを使用する例としては、帳票印刷、レコード一括更新、一括削除、集計計算などがあります。オフライン RIA ではサーバ側の処理を直接呼び出すことはできません。従って、サーバ側で実行するバッチタスクはサーバに接続された場合だけ実行するように変更するか、バッチタスクを非インタラクティブなリッチクライアントタスクに変更してオフライン状態でも実行できるようにする必要があります。

非インタラクティブタスクでも RIA プログラムにすることでフォームを定義し、実行時に表示させることもできますが、多くの場合エンドユーザに対する表示なしで処理を実行します。

本章では非インタラクティブタスクの実行エンジン動作について学びます。また非インタラクティブタスクをどういう場合に使用するべきかを検討し、簡単な非インタラクティブタスクを作成してみます。

RIA における非インタラクティブタスクには、以下の２種類があります。

- **バッチタスク …… サーバで実行されます。この場合、実行時にフォームを表示させることはできません。**
- **非インタラクティブなリッチクライアントタスク …… クライアントで実行されます。**

本書では、まとめて「非インタラクティブタスク」として説明しています。この章では、非インタラクティブなリッチクライアントタスクについて説明しています。バッチタスクは、オフライン RIA プログラムから直接呼び出すことができないため、別途説明します。

２．非インタラクティブタスクによるレコードの削除

前章で１対多のデータリレーションシップについて＜受注管理＞プログラムの実習を通して学びました。

このプログラムでは、エンドユーザはまず最初のタスクで主データソース＜受注＞に１レコード登録し、次に従データソース＜受注明細行＞へ複数のレコードを登録して行きます。

もし主データソースで受注レコードが削除されると、そのままでは従データソースの対応レコードが迷子のまま残されることになってしまいます。

このような事態を避けるために、主データソースでレコードが削除されるときには、非インタラクティブタスクを利用して従データソースの関連レコードをすべて削除するように構成します。

次ページ以降で、この＜受注管理＞プログラムを改良し、迷子のレコードを処理するための非インタラクティブタスクを作成することにしましょう。

オフライン RIA プログラムでは、この場合、実際にレコードを削除することはなく、各レコードの［削除フラグ］カラムを＜True＞に更新するように変更します。

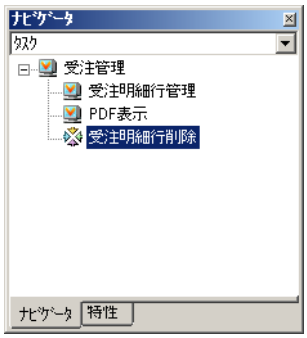
３．削除処理を行う非インタラクティブタスクの例

ここでは＜受注管理＞タスクにサブタスクを追加し、エンドユーザが＜受注管理＞タスクでレコードの削除処理を行うたびに呼び出されるようにします。

そのサブタスクはバッチタスクで、＜受注管理＞タスクで削除した＜受注＞レコードの受注番号と同じ受注番号を持つ＜受注明細行＞レコードの削除処理を行うようにします。

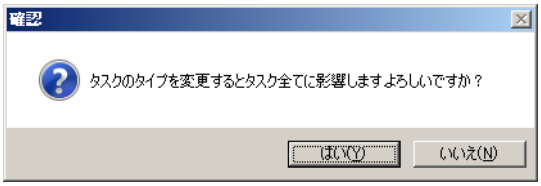
受注明細行レコードを削除するサブタスクの修正

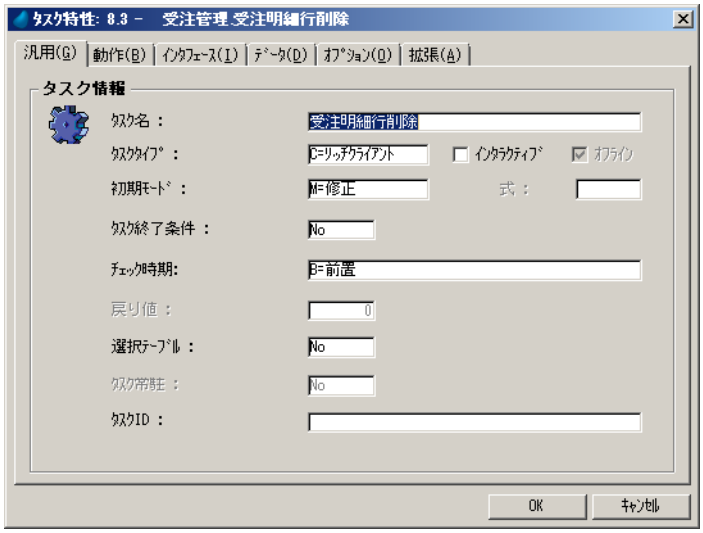
1. ［プログラム］リポジトリ（Shift+F3）を開きます。
2. ＜受注管理＞（#8）プログラムにズームします。
3. ［ナビゲータ］ペインを開き、＜受注明細行削除＞ノードにパークしてタスクを開きます。
4. ［タスク特性］を開いて、次のように設定します。



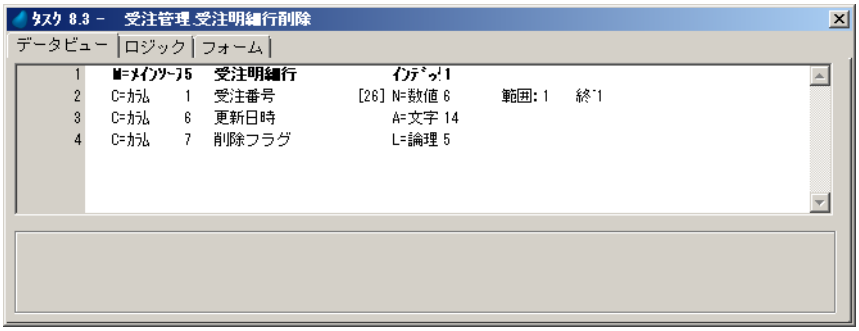
5. ［タスクタイプ］特性を、「C= リッチクライアント」に変更します。タスクタイプを変更することによる確認ダイアログが表示されますが、ここでは「はい」を選択してください。
6. ［初期モード］特性を、「M= 修正」とします。

［インタラクティブ］特性は、自動的にオフになります。





7. ［OK］ボタンをクリックします。
8. ［データビュー］エディタで［更新日時］と［削除フラグ］のデータビューを追加します。



9. ［ロジック］エディタを開き、［レコード後］ロジックユニットを作成し、以下のようにコマンドを定義します。

処理コマンド	項目	式	条件
項目更新	更新日付	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes
項目更新	削除フラグ	'TRUE'LOG	Yes

＜受注明細行削除＞サブタスクの呼び出し

＜受注明細行削除＞サブタスクは、エンドユーザが受注レコードを削除するときに＜受注管理＞タスクから呼び出されなければなりません。ただし、受注レコードは実際には削除されないため、Stat() 関数による削除モードでは判断できません。受注レコード内の［削除フラグ］カラムで判断します。

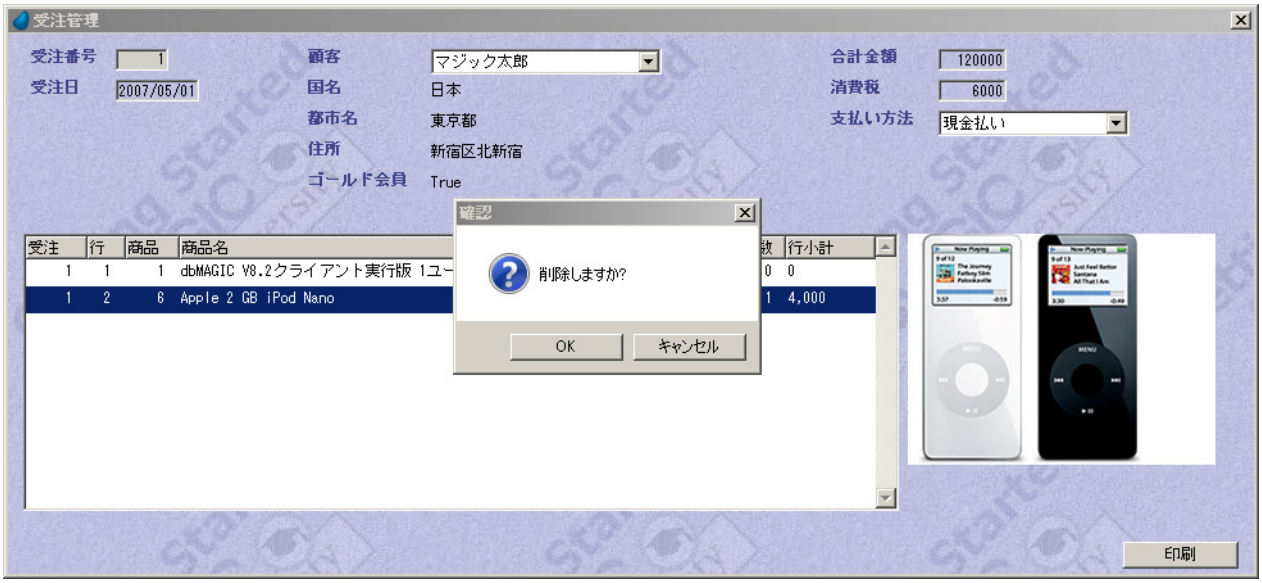
1. ［ナビゲータ］ペインを開き（Alt+F1）、タスクツリーで＜受注管理＞ノードにパークします。
2. ［ロジック］エディタを選択します。
3. ［レコード後］ロジックユニットにパークして、＜受注明細行削除＞を呼び出す［コールタスク］処理コマンドにパークします。
4. ［条件］カラムでズームして、［式］エディタを開きます。式（現在は、Stat(0,'D'MODE) に設定されています）を修正し、＜受注＞データの［削除フラグ］カラムが True の場合のみ実行するようにします。

プログラムの動作確認

変更内容を保存し、プログラムを閉じたら、動作の確認をしましょう。

1. F7 キーを押下して＜受注管理＞プログラムを起動しましょう。
2. ＜受注番号＞項目にパークしてください。
3. モードが［登録］に切り替わるまで PageDown キーを押下してください。（修正モードから登録モードへの一時的な切替え）
4. 下のイメージのような受注データを作成してください。
5. ＜受注番号＞欄にパークし、PageUp キーを押下してください（作成したレコードが保存されます）。
6. PageDown キーを押下して、登録したレコードに戻ります。
7. F3 キーを押下してレコードを削除してみましょう。
8. ［確認］ダイアログで「はい」をクリックします。
9. ＜受注管理＞プログラムを終了します。

画面では非インタラクティブタスクの動いている様子は分かりませんが、実際には非インタラクティブタスクが起動され、削除対象のレコードは表示されなくなります。



APG を使用して＜受注明細＞データソースを表示させることで、実際に［削除フラグ］が＜ True ＞に設定されていることを確認することができます。

４．要約

本章では非インタラクティブなリッチクライアントタスクについて説明し、次のことを学びました。

- オフライン RIA プログラムにおける非インタラクティブタスクの定義方法について

実習として受注明細行レコードを削除する非インタラクティブタスクを作成しました。このプログラムは＜受注管理＞プログラムにおいて、データの参照整合性を維持するためのものでした。

ここまでは、オフライン RIA プログラム内での処理の説明でした。次章は、サーバ側のデータとの同期処理について説明します。

第 9 章 サーバ側とのデータ同期

本章ではサーバ側のデータの定義とクライアント側との同期処理について扱います。

キーワード

- ［データソース］テーブル
- データの同期処理
- DataViewToDataSource() 関数

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- サーバ側の DBMS を使用したデータソースの定義
- ローカルデータソースとサーバデータソースの同期処理の定義方法

参照

ホワイトトペーパの［オフラインアプリケーションの開発／ローカル（オフライン）ストレージ］

同様に、［オフラインアプリケーションの開発／クライアント／サーバ間でのデータ同期］

１．はじめに

今までは、ローカルデータソースを扱うプログラムについて説明してきました。全てのクライアントがオフライン環境で独立した処理を行う場合は、これで完了しますが、通常は、サーバ側のデータを元に各クライアントが連携していく必要があります。

このため、サーバとクライアント間でのデータの同期処理が必要となります。

第 4 章「データソースの内容参照」で説明したように、ローカル側のデータソースの各レコードが更新されると、その時の更新日時が記録され、レコードの削除処理が行われても実際には削除されず、レコード内の削除フラグが True に更新されます。データの同期処理は、この情報をもとに、更新されたレコードでサーバ側のデータソースを更新することで実現します。この処理は、サーバ側の更新レコードをクライアント側に反映される場合にも当てはまります。

本章ではサーバ側のデータソースを定義し、ローカルデータソースとの同期処理を行うためのプログラムの作成方法について学びます。また、更新日付を管理するために、管理用のデータソースを別途定義し、同期処理で使用します。

本セミナーでは、更新日時として UTC（協定世界時）を YYYYMMDDHHMMSS の文字列で格納し、比較するようにしています。これは、サーバや各クライアントの実行時のタイムゾーンが異なる場合でも一元管理できるようするためです。

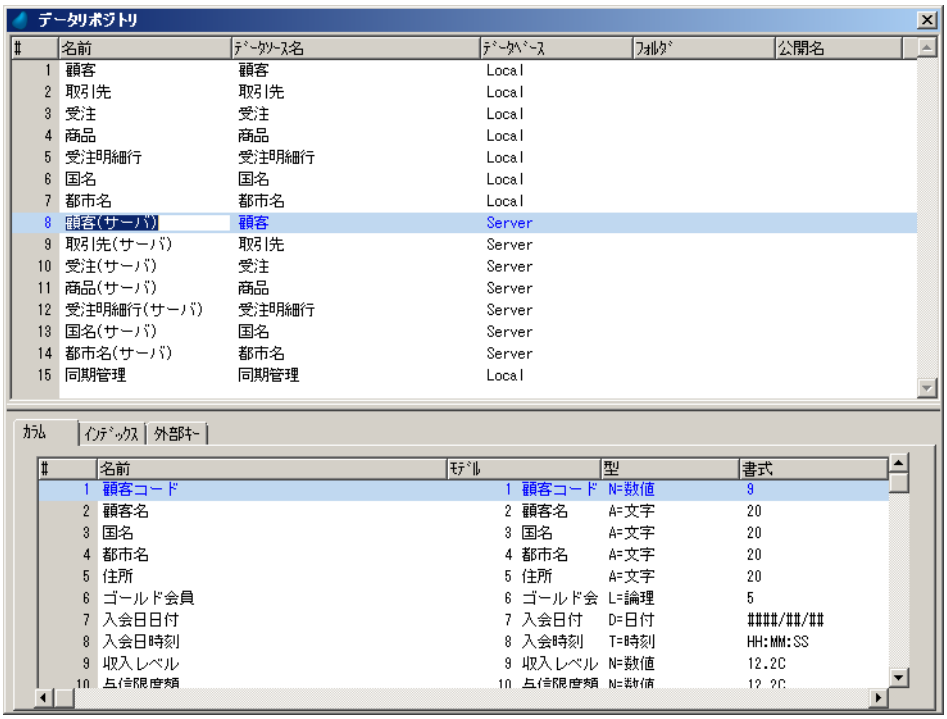
本コースは、最初にクライアント側の処理を定義してからサーバ側の処理に移行する流れのため、クライアント側のデータソースを元にサーバのデータソースを定義していますが、一般的にはこの逆の手順で作成することになります。

２．サーバ側のデータソース定義

サーバ側のデータソースを定義する方法は、今までのオンライン RIA の場合と同じように行いますが、ローカル側のデータソースと同じカラム構成にする必要があります。基本的にローカル側のデータソースを［データソース］リポジトリ上で複写し、データベースのみ変更することで対応できます。

1. ［データソース］リポジトリ（Shift+F2）を開きます。
2. 最終行（#7）にパークして Ctrl+Shift+R キーを押下して［複写登録］ダイアログを表示します。
3. ［開始番号］を＜ **1** ＞、［終了番号を］を＜ **7** ＞に設定して［OK］をクリックします。
4. 複写行（#8 ～ #14）の［データベース］カラムを＜ Local ＞から＜ Server ＞に変更します。

5. また、名前の末尾に＜（サーバ）＞を追加して、区別できるようにします。



＜ Server ＞データベースは、SQLite を使用するように定義されています。

SQLite DBMS は、シングルユーザ用のため、通常のアプリケーショ ンには向いていません。Local DBMS とのファイル互換性が必要な場合のみの利用してく ださい。

３．＜同期管理＞データの作成

［データソース］リポジトリに＜同期管理＞データを作成します。このテーブルは、ローカル側に作成し、同期処理を実行した最終日時を文字列で記録します。

1. ［データソース］リポジトリ（Shift+F2）を開きます。
2. 最終行（#14）にパークして F4 キーを押下して、一行作成します。
3. ［名前］カラムに **＜同期管理＞** と入力します。［データベース］カラムは＜ Local ＞を設定します。

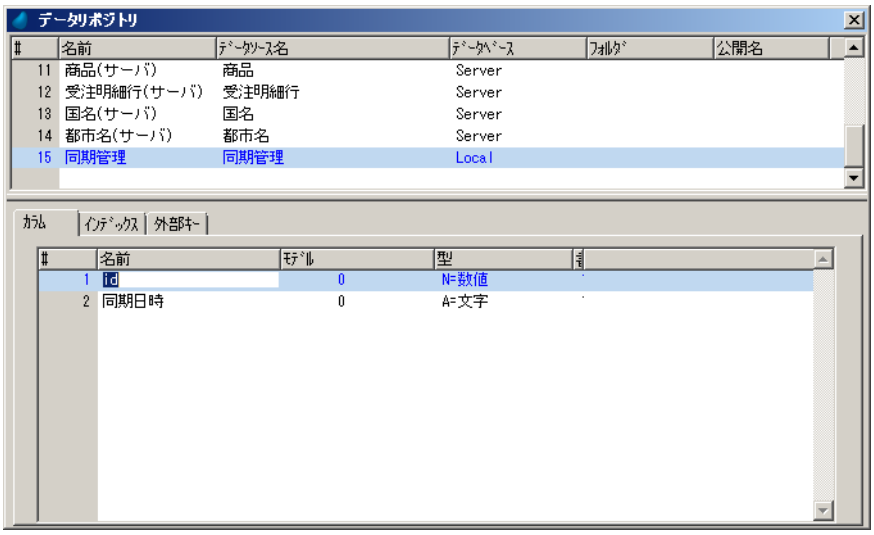
4. 下部ペインの［カラム］タブをクリックします。以下のようにカラムを定義します。

#	名前	モデル	型	書式
1	id	0	N= 数値	1
2	同期日時	0	A= 文字	14

5. 次に、下部ペインの［インデックス］タブをクリックします。
6. 一行作成（F4）します。
7. 次のようにインデックス名とインデックスセグメントを定義します。

#	名前	タイプ	プライマリキー
1	id	U= 重複不可	

#	名前	サイズ	順序
1	id	2	A= 昇順



ここでは省略していますが、通常は、異なるクライアントで同一インデックスのデータが作成されないようにする工夫が必要です。例えば、端末番号毎などをインデックスの一部に追加することで、クライアント毎にユニークなレコードを作成するようにしてください。

４．同期処理プログラムの作成

データソースの同期処理プログラムを作成します。

同期処理には、サーバからクライアントにコピーする処理とクライアントからサーバにコピーする処理の二つがあります。どちらも以下のような流れのプログラムになります。

1. 最終更日を取得
2. 最終更新日以降のレコードをコピー
3. 最終更新日を更新

まず、最終更新日の取得と更新を行うプログラムを作成します。

最終更新日の取得プログラムの作成

1. ［プログラム］リポジトリを開き、最終行に位置付けします。
2. 一行追加して**＜最終更新日取得＞**プログラムを登録します。
3. ［タスク特性］を以下のように設定します。
   - **タスクタイプ …… C= リッチクライアント**
   - **インタラクティブ …… オフ**
   - **オフライン …… オフ**
   - **初期モード …… M= 修正**
   - **タスク終了条件 …… Yes**
   - **チェック時期 …… A= 後置**

4. ［データビュー］エディタを開き、二つのパラメータ項目を追加します。

名前	モデル	型	書式
p_id	0	N= 数値	1
p_ 同期日時	0	A= 文字	14

p_id の値は、以下のように扱うことを前提にしています。

- **p_id=1 …… サーバからクライアントへの同期処理**
- **p_id=2 …… クライアントからサーバへの同期処理**

5. 最終行で Ctrl+H を押下してヘッダ行を追加し、［書込リンク］コマンドを定義します。

処理コマンド	データ	インデックス	方向	条件
W= 書込リンク	15 同期管理	1	D= デフォルト	Yes

また、［同期管理］データのカラムを以下のように追加します。

名前	型	書式	位置付（最小／最大）	代入
id	N= 数値	1	p_id	p_id
同期日時	A= 文字	14		

6. ［ロジック］エディタを開き、Ctrl+H を押下してヘッダ行を追加し［レコード後］ロジックユニットを追加します。
7. 一行作成（F4）し、以下の処理コマンドを追加します。

処理コマンド	項目	式	条件
項目更新	p_ 同期日時	同期日時	Yes

最終更新日の設定プログラムの作成

1. ［プログラム］リポジトリを開き、最終行に位置付けします。
2. 一行追加して＜最終更新日設定＞プログラムを登録します。
3. ［タスク特性］を以下のように設定します。
   - タスクタイプ …… C= リッチクライアント
   - インタラクティブ …… オフ
   - オフライン …… オフ
   - 初期モード …… M= 修正
   - タスク終了条件 …… Yes
   - チェック時期 …… A= 後置

4. ［データビュー］エディタを開き、一つのパラメータ項目を追加します。

名前	モデル	型	書式
p_id	0	N= 数値	1

5. 最終行で Ctrl+H を押下してヘッダ行を追加し、［書込リンク］コマンドを定義します。

処理コマンド	データ	インデックス	方向	条件
W= 書込リンク	15 同期管理	1	D= デフォルト	Yes

また、［同期管理］データのカラムを以下のように追加します。

名前	型	書式	位置付（最小／最大）	代入
id	N= 数値	1	p_id	p_id
同期日時	A= 文字	14		

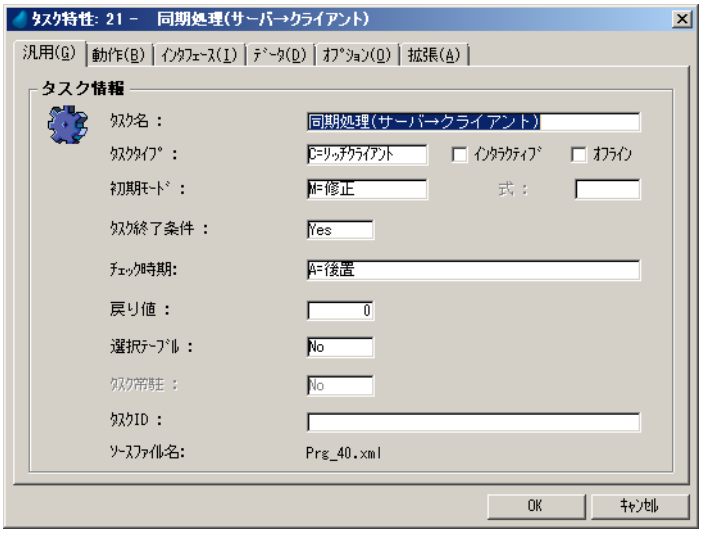
6. ［ロジック］エディタを開き、Ctrl+H を押下してヘッダ行を追加し［レコード後］ロジックユニットを追加します。
7. 一行作成（F4）し、以下の処理コマンドを追加します。

処理コマンド	項目	式	条件
項目更新	同期日時	DStr(UTCDate(),'YYYYMMDD')&TStr(UTCTime(),'HHMMSS')	Yes

これで。最終更新日の設定と取得を行うことができるようになりました。

レコードコピー処理（サーバ→クライアント）プログラムの作成

1. ［プログラム］リポジトリを開き、最終行に位置付けします。
2. 一行追加して＜同期処理 (サーバ→クライアント) ＞プログラムを登録します。
3. ［タスク特性］を以下のように設定します。
   - タスクタイプ …… C= リッチクライアント
   - インタラクティブ …… オフ
   - オフライン …… オフ
   - 初期モード …… M= 修正
   - タスク終了条件 …… Yes
   - チェック時期 …… A= 後置

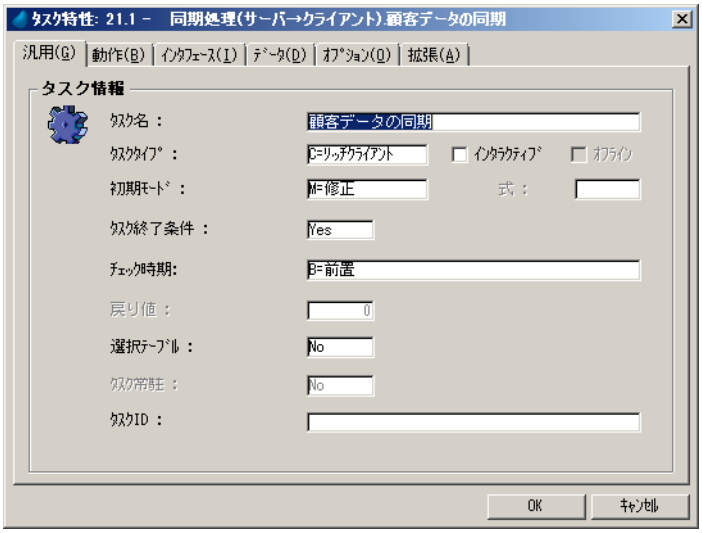


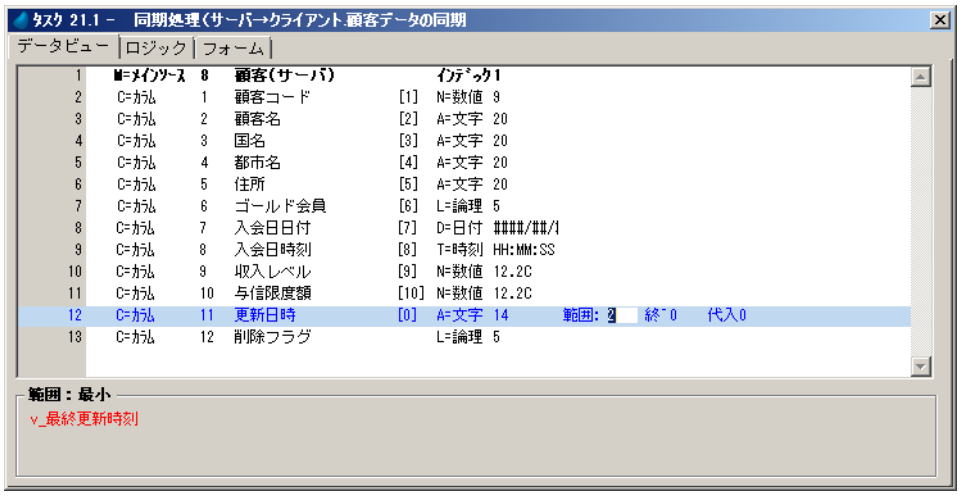
4. ［データビュー］エディタを開き、変数項目を追加します。

名前	モデル	型	書式
v_ 最終更新日時		A= 文字	14

サブタスク（各データソースのコピー処理）の作成

5. ［ナビゲータ］ペインを表示して＜同期処理 (サーバ→クライアント) ＞にパークします。
6. 一行作成（F4）します。新しいエントリ（この場合、サブタスク）がタスクツリーに作成され、その［タスク特性］ダイアログが自動的に開きます。
7. ［タスク名］に＜顧客データの同期＞と入力します。
8. ［タスク特性］を以下のように設定します。
   - タスクタイプ …… C= リッチクライアント
   - インタラクティブ …… オフ
   - 初期モード …… M= 修正
   - タスク終了条件 …… Yes
   - チェック時期 …… B= 前置
9. ［OK］ボタンをクリックします。
10. ［データビュー］エディタを開いて、メインソースに＜顧客（サーバ）＞を設定します。
11. ＜顧客（サーバ）＞データのすべてのカラムをデータビューとして登録します。





12.[更新日時] カラムの [範囲：最小] 特性に、親タスクの< v_ 最終更新日時>を設定します。これによって、指定された日時以降のデータのみがコピーされます。

13.[ロジック] エディタを開いて、[タスク後] ロジックユニットを定義します。

14.一行作成（F4）して、次のように処理コマンドを作成します。

処理コマンド		式	戻り値	条件
アクション	E= 式	DataViewToDataSource(0,'','1'DSOURCE,'','')		Yes

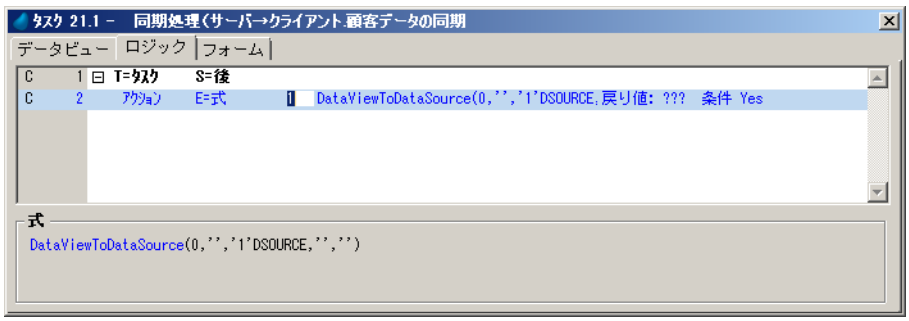
DataViewToDataSource() 関数は、タスクのデータビューの内容を データソースにコピーします。以下のように定義します。

構文：DataViewToDataSource (世代番号 , タスクの項目名 , 出力データソース番号 , 出力データソース名 , 出力カラム名)

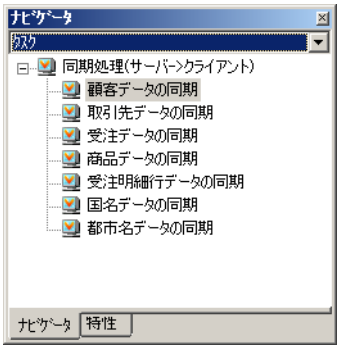
- **世代番号 …… タスクの階層位置を表す番号。カレントのタスクが 0、親タスクが 1、その親タスクが 2 などとなります。**
- **タスクの項目名（文字）…… 出力する項目の名前をカンマ区切りでリストアップした文字列。**
- **出力データソース番号（数値）…… ［データ］リポジトリ上の通番を表す数値（例：'3'DSOURCE）。**
- **出力データソース名（文字）……このパラメータは、ソース番号の代わりに使用することができます。必要ない場合は、空白 ('') で指定してください。**
- **出力カラム名（文字）…… 出力先のデータソースで更新されるカラムのすべての名前をカンマ区切りで指定します。**

例えば、DataViewToDataSource (0, '', '2'DSOURCE, '', '') を実行すると、現在のタスクに定義されたデータビューの内容が［データソース］リポジトリの 2 番目のデータソースにコピーされます。この関数を利用することで比較的速くデータソースのレコードレベルのコピー処理を実行することができます。

この関数を実行することで、タスクの（サーバ上の）データビューを（クライアント側の）データソース（1'DSOURCE）にコピーします。



15.同様のサブタスクを各データソースに対して作成します。



親タスクでの［ロジック］エディタの定義

16.親タスクに戻り、［ロジック］エディタを開き、Ctrl+H を押下して［レコード後］ロジックユニットを追加します。

17.［タスク後］に以下の処理コマンドを定義します。

処理コマンド		式	パラメータ	条件
コール	P= プログラム	19 最終更新日取得	2 ※ 1	Yes
コール	S= サブタスク	1 顧客データの同期		Yes
コール	S= サブタスク	2 取引先データの同期		Yes
コール	S= サブタスク	3 受注データの同期		Yes
コール	S= サブタスク	4 商品データの同期		Yes
コール	S= サブタスク	5 受注明細行の同期		Yes
コール	S= サブタスク	6 国名データの同期		Yes
コール	S= サブタスク	7 都市名データの同期		Yes
コール	P= プログラム	20 最終更新日設定	1 ※ 2	Yes

※ 1 : パラメータテーブルには以下のように定義します。

項目	式	説明
	1	1
A		v_ 最終更新日時

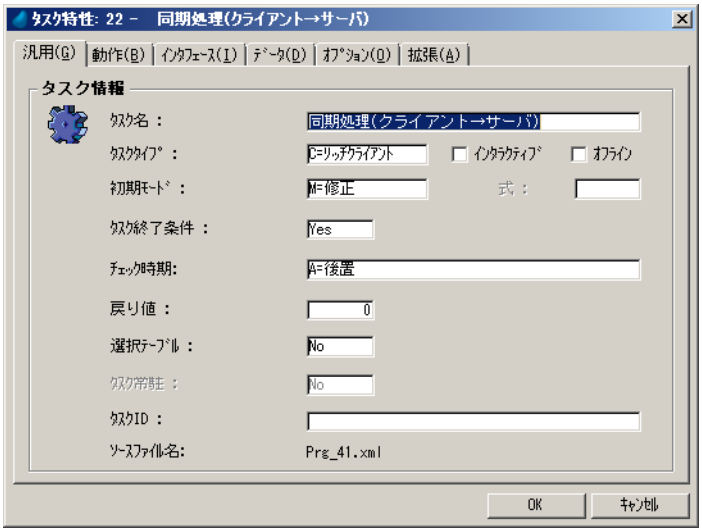
※ 2 : パラメータテーブルには以下のように定義します。

項目	式	説明
	1	1

レコードコピー処理（クライアント→サーバ）プログラムの作成

クライアントからサーバへのコピー処理は、前述のサーバからクライアントにコピーする処理と同じタスク構成で作成します。

1. ［プログラム］リポジトリを開き、最終行に位置付けします。
2. 一行追加して＜同期処理 (クライアント→サーバ) ＞プログラムを登録します。
3. ［タスク特性］を以下のように設定します。
   - タスクタイプ …… C= リッチクライアント
   - インタラクティブ …… オフ
   - オフライン …… オフ
   - 初期モード …… M= 修正
   - タスク終了条件 …… Yes
   - チェック時期 …… A= 後置

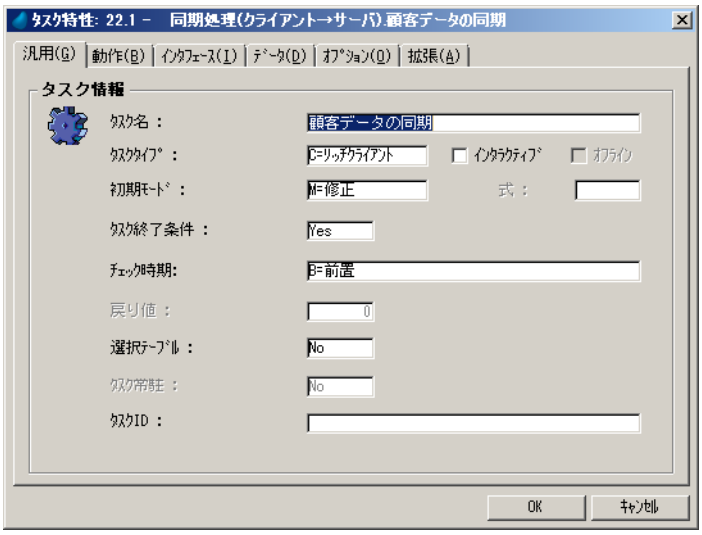


4. ［データビュー］エディタを開き、変数項目を追加します。

名前	モデル	型	書式
v_ 最終更新日時		A= 文字	14

サブタスクの作成

5. ［ナビゲータ］ペインを表示して＜同期処理 (クライアント→サーバ) ＞にパークします。
6. タスクのデータビューをサーバのデータソースにコピーするサブタスクをデータソース毎に作成します。
7. ［タスク特性］を以下のように設定します。
   - タスクタイプ …… C= リッチクライアント
   - インタラクティブ …… オフ
   - 初期モード …… M= 修正
   - タスク終了条件 …… Yes
   - チェック時期 …… B= 前置



8. ［データビュー］エディタを開いて、メインソースに＜顧客＞を設定します。
9. ＜顧客＞データのすべてのカラムをデータビューとして登録します。
10. ［更新日時］カラムの［範囲：最小］特性に、親タスクの＜ v_ 最終更新日時＞を設定します。これによって、指定された日時以降のデータのみがコピーされます。
11. ［ロジック］エディタを開いて、［タスク後］ロジックユニットを定義します。
12. 一行作成（F4）して、次のように処理コマンドを作成します。

処理コマンド		式	戻り値	条件
アクション	E= 式	DataViewToDataSource(0,'','8'DSOURCE,'','')		Yes

この関数を実行することで、タスクの（クライアント上の）データビューを（サーバ側の）データソース（'8'DSOURCE）にコピーします。

［ロジック］エディタの定義

13.親タスクに戻り、［ロジック］エディタを開き、Ctrl+H を押下して［レコード後］ロジックユニットを追加します。

14.［レコード後］に以下の処理コマンドを定義します。

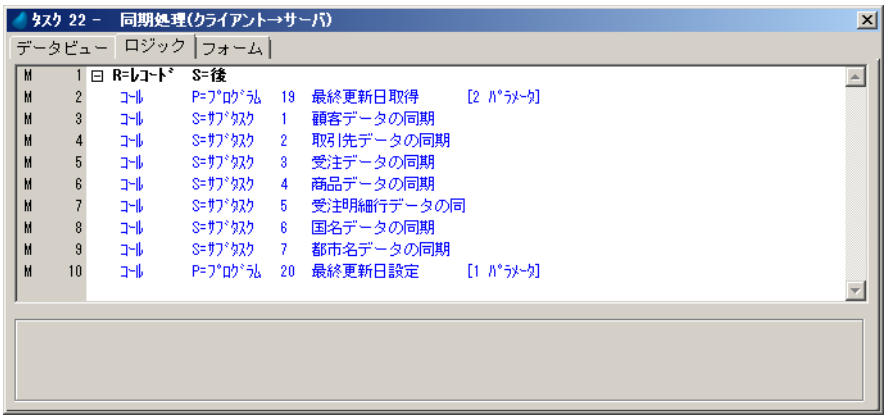
処理コマンド		式	パラメータ	条件
コール	P= プログラム	19 最終更新日取得	2 ※ 1	Yes
コール	S= サブタスク	1 顧客データの同期		Yes
コール	S= サブタスク	2 取引先データの同期		Yes
コール	S= サブタスク	3 受注データの同期		Yes
コール	S= サブタスク	4 商品データの同期		Yes
コール	S= サブタスク	5 受注明細行の同期		Yes
コール	S= サブタスク	6 国名データの同期		Yes
コール	S= サブタスク	7 都市名データの同期		Yes
コール	P= プログラム	20 最終更新日設定	1 ※ 2	Yes

※ 1 : パラメータテーブルには以下のように定義します。

項目	式	説明
	1	2
A		v_ 最終更新日時

※ 2 : パラメータテーブルには以下のように定義します。

項目	式	説明
	1	2



レコードコピー処理プログラムの呼び出し

アプリケーションの起動時に、前述のレコードコピー処理を呼び出すように処理を定義します。定義する場所は以下のロジックが考えられます。

- ［メインプログラム］の［タスク前］。ただし、［タスク前］では、［コール］処理コマンドで RIA プログラムを呼び出すことができないため、イベント経由で呼び出す必要があります。
- アプリケーションの起動時に必ず実行される（例えば、メニュー表示用）プログラム。このプログラムがオフライン RIA プログラムの場合、［コール］処理コマンドで直接起動させることはできないため、前述のようにイベント経由で呼び出す必要があります。

アプリケーションの終了時に、クライアントのレコードをサーバにアップする必要もあります。定義する場所は以下のロジックが考えられます。

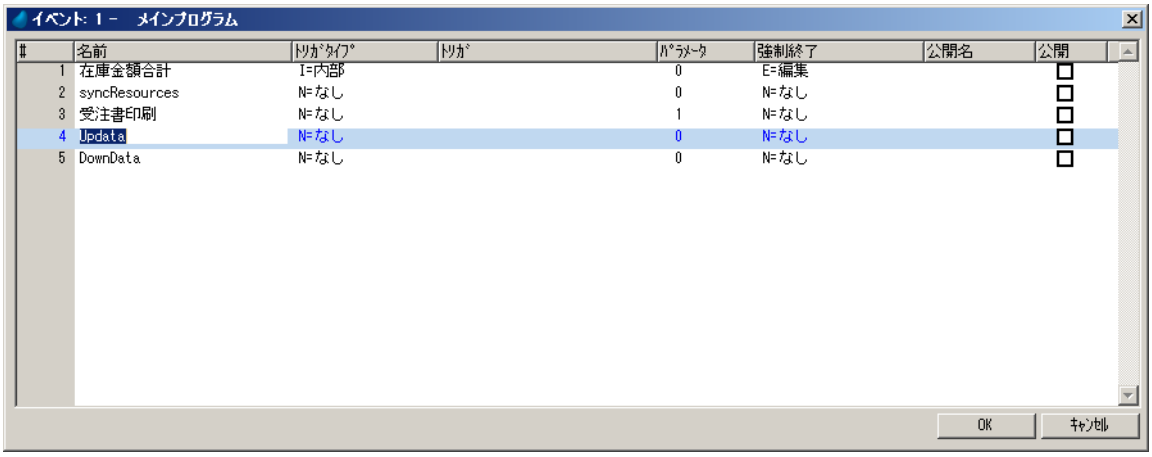
- ［メインプログラム］の［終了］イベントに対する［イベント］ロジックユニット

- アプリケーションの起動時に必ず実行されるプログラム。このプログラムの［タスク後］に［コール］処理コマンドを定義します。

ここでは、［メインプログラム］に処理を定義します。

1. ［メインプログラム］を開きます。
2. Ctrl+U を押下して［イベント］テーブルを開きます。
3. 以下のようにユーザ定義イベントを定義します。

名前	トリガタイプ	強制終了
UpData	N= なし	N= なし
DownData	N= なし	N= なし



4. ［ロジック］エディタを開き、Ctrl+H でロジックヘッダを追加し、以下のように［イベント］ロジックユニットを定義します。

ユニットタイプ	イベント	条件
イベント	UpData	Yes

5. 一行作成（F4）し、以下の処理コマンドを定義します。

処理コマンド			条件
コール	P= プログラム	同期処理 (クライアント→サーバ)	Yes

6. Ctrl+H でロジックヘッダを追加し、以下のように［イベント］ロジックユニットを定義します。

ユニットタイプ	イベント	条件
イベント	DownData	Yes

7. 一行作成（F4）し、以下の処理コマンドを定義します。

処理コマンド			条件
コール	P= プログラム	同期処理 (サーバ→クライアント)	Yes

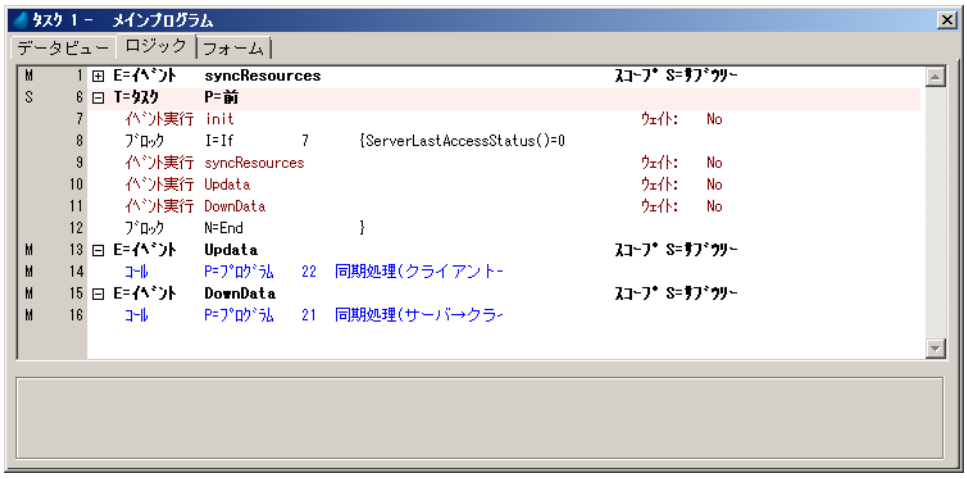
8. ［タスク前］ロジックユニットに以下の処理コマンドを定義します。

処理コマンド	イベント	ウェイト	条件
イベント実行	UpData	No	Yes
イベント実行	DownData	No	Yes

9. Ctrl+H を押下して［終了（x)］イベントに対する［イベント］ロジックユニットを作成します。
   このロジックユニットの［伝播］特性を「Yes」に設定します。
10. 一行作成（F4）し、以下の処理コマンドを定義します。

処理コマンド			条件
コール	P= プログラム	同期処理 (クライアント→サーバ)	Yes

これによって、クライアント側でアプリケーションが起動する場合と、サーバ側のデータソースをクライアント側にコピーする同期処理とクライアント側のデータソースをサーバ側にコピーする同期処理が実行されます。



では試してみましょう。

同期処理の動作確認

今まで作成したプログラムは、オフライン用でローカルのデータソースを更新していました。同期処理を確認するには、サーバ側のデータソースにアクセスするオンライン RIA プログラムが必要になります。事前にこのプログラムを作成した後、同期処理の確認を行ってみましょう。

サーバ上の＜顧客＞データソースにアクセスするプログラム

ローカルデータソースをアクセスするプログラムをコピーして、メインデータソースを変更することで確認用のプログラムを作成します。

1. ［プログラム］リポジトリの最終行にパークします。
2. Ctrl+Shift+R を押下して［複写登録］ダイアログを開きます。
3. ［開始番号］を＜ **3** ＞、［終了番号］にも＜ **3** ＞を入力して、［OK］をクリックします。

4. 最終行に＜照会－顧客＞プログラムがコピーされるので、［名前］を**＜照会－顧客（サーバ）＞**に変更します。
5. ［オフライン］カラムをオフに変更し、ここからズームして［データビュー］エディタを開きます。
6. メインソースを＜ #1 ＞（顧客）から＜ #8 ＞（顧客（サーバ））に変更します。

これで、サーバ上の＜顧客＞データソースを表示するプログラムが作成されました。

＜顧客＞データソースの修正と同期確認（クライアント→サーバ）

1. ［プログラム］リポジトリの＜照会－顧客＞（#3）にパークします。
2. F7 を押下してプログラムを実行します。
3. Ctrl+M を押下して修正モードに切り替え、データを追加したり、修正したりします。

4. ［Esc］キーを押下してプログラムを終了します。
5. 次に、＜照会－顧客（サーバ）＞（#23）にパークして、F7 を押下してプログラムを実行します。
6. 先ほど修正したローカル側のデータソースの内容がサーバ側のデータソースに反映されていることを確認してください。

＜顧客＞データソースの修正と同期確認（サーバ→クライアント）

7. Ctrl+M を押下して修正モードに切り替え、データを追加したり、修正したりします。
8. もう一度＜照会－顧客＞（#3）にパークして、F7 を押下してプログラムを実行します。
9. 修正したサーバ側のデータソースの内容がローカル側のデータソースに反映されていることを確認してください。

データ同期のタイミングは、アプリケーションの仕様に基づいて必ず実行される処理内に定義するようにしてください。

５．要約

本章では、次のことを学びました。

- サーバ側のデータソースを定義しました。
- ローカル側のデータソースとサーバ側のデータソースの同期プログラムを作成しました。
- 同期処理によってサーバ側のデータが反映されることを確認しました。

次章では、サーバ側のプログラムを呼び出す処理について説明します。

印刷プログラムのように［入出力ファイル］テーブルを使用するプログラムは、サーバ側のバッチプログラムとして定義する必要があり、このようなプログラムを呼び出すには、イベント経由で行う必要があります。また、事前にサーバ側のデータとクライアント側のデータを同期させる必要があります。

第 10 章 帳票印刷

本章では、オフライン RIA プログラムから単純な帳票処理を呼び出す処理の作成方法について学習します。リッチクライアントプログラムでは、入出力デバイスアクセスすることができないため、サーバ上で実行するバッチタスクを呼び出す必要があります。

しかし、オフライン RIA プログラムからサーバ上のプログラムを呼び出すには、イベントを経由する必要があります。このため、サーバ上で実行するバッチプログラム自体を修正することはありませんが、このプログラムの呼び出し方法を工夫する必要があります。

キーワード

- **バッチタスク**
- PDF **出力**

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- **サーバ上で実行する非インタラクティブタスクの呼び出し方法**
- **［ブラウザ］コントロールを使用した PDF の表示**

参照

Magic ヘルプの［Magic xpa リファレンス／［フォーム］エディタ／［フォーム］エディタのカラム］

同様に、［Magic xpa リファレンス／［ロジック］エディタ／処理コマンド／フォーム／フォーム出力］

注意

印刷処理は、サーバ側で実行されるバッチタスクでのみ実現可能です。このため、説明では、バッチタスクとして説明しています。RIA で利用する場合、実際の印刷結果はサーバに接続されたプリンタに出力されるか、PDF (Portable Document Format) としてサーバ上に出力し［ブラウザ］コントロール上で表示させる場合が一般的です。

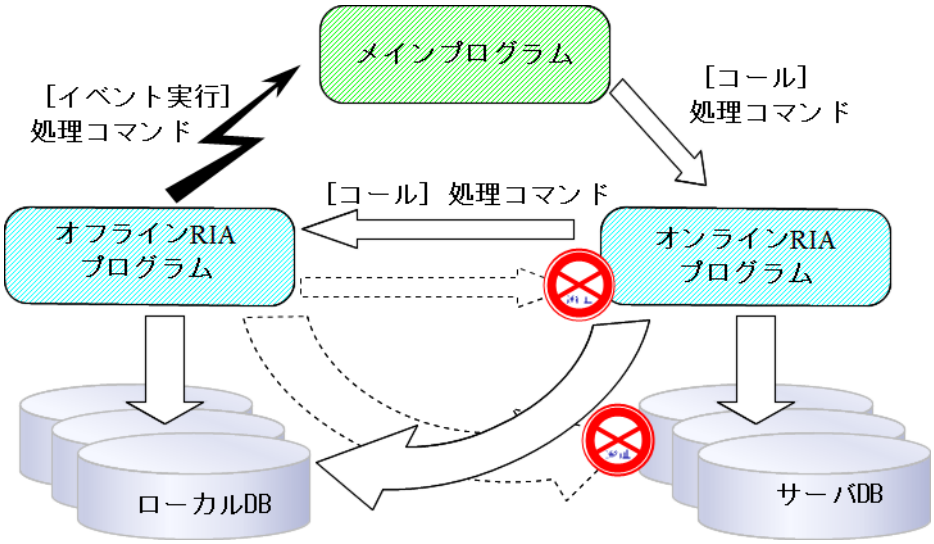
ここでは、PDF を出力する方法を紹介しています。

１．はじめに

帳票作成処理は、サーバ側でのみ実行される非インタラクティブ処理です。オフライン RIA プログラムからサーバで実行されるオンライン RIA プログラムを呼び出す方法として、［コール］処理プログラムは利用できません。

オフライン RIA プログラムからオンライン RIA プログラムの呼び出し

［メインプログラム］に［イベント］ロジックユニットを定義し、ここからオンライン RIA プログラムを［コール］処理コマンドで呼び出すようにします。オフライン RIA プログラムでは、［メインプログラム］の［イベント］ロジックユニットを実行させるためのイベントを発行させることで、間接的に呼び出すことができます。



オフライン RIA プログラムとオンライン RIA プログラムの動作の制約

オフライン RIA プログラムとオンライン RIA プログラム間の連携処理を実現する場合は、以下のような制約事項をあらかじめ理解しておいてください。

プログラムの呼び出し

オフライン→オンライン	×（イベント経由で呼び出す必要があります）
オンライン→オフライン	○

データソースへのアクセス

プログラム	データソース	
オフライン	ローカル	○
オフライン	サーバ	×
オンライン	ローカル	○
オンライン	サーバ	○

本章ではオフライン RIA プログラムから帳票作成プログラムの呼び出し方法について学びます。

最初に、帳票結果を PDF として出力するための環境設定について紹介します。

２．帳票出力の環境設定

帳票結果を PDF として出力するには、専用の仮想ドライバソフトをサーバ側にインストールする必要があります。以下の機能を持つソフトを推奨します。

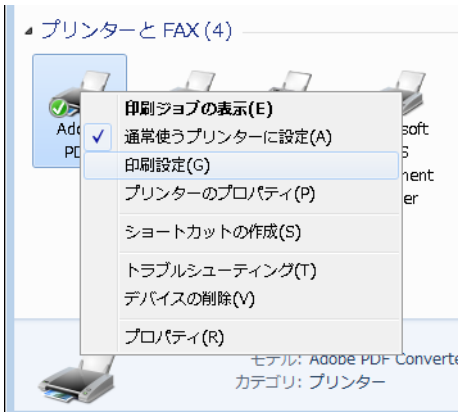
- 通常のプリンタドライバと同じように Windows に登録される

- PDF の出力先が指定できる
- （ファイルの上書きや、書込先指定の）ダイアログを表示させないように設定できる
  ダイアログがサーバ上で表示されると処理が停止してしまうため

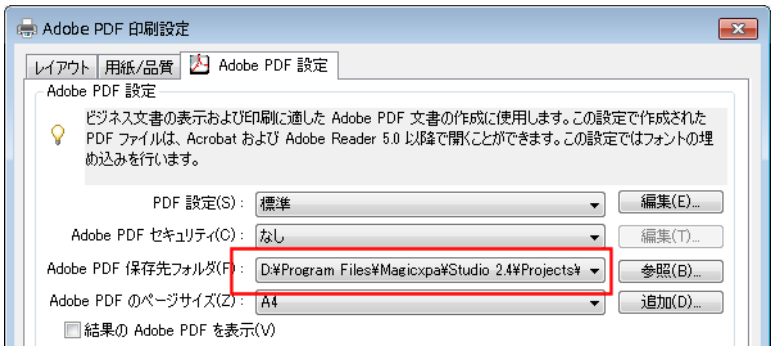
本書では、有償の Adobe Acorbat によってインストールされる Adobe PDF を使用した手順を紹介していますが、フリーソフト（例：Bullzip PDF Printer）を利用することも可能です。

プリンタデバイスの確認

1. Adobe Acorbat をインストールすると Windows の［デバイスとプリンタ］には、「Adobe PDF」というデバイス名が登録されます。
2. このアイコン上で右クリックして「印刷設定」を選択します。



3. ［印刷設定］ダイアログで［Adobe PDF 保存先フォルダ］を指定しておきます。この例では、**＜ D:¥Program Files¥Magicxpa¥Studio 2.4¥Projects¥Magic_xpa_Offline_RIA 入門 ¥PDF ＞**フォルダになっています（設定方法は、ソフトによって異なります）。
4. “結果として出力される PDF の自動表示”や”ファイルの上書き確認のオプション”は無効に設定してください（オプション名は、ソフトによって異なります）。

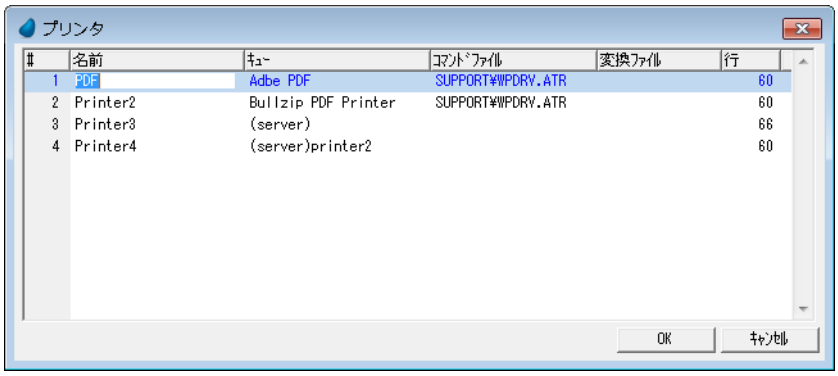


［OK］ボタンをクリックして、設定を保存します。次に、Magic xpa 側の設定を行います。

1. ［オプション］メニューから［設定／プリンタ］を選択して［プリンタ］テーブルを開きます。
2. 一行目を以下のように変更します。

名前	キュー	コマンドファイル	変換ファイル	行
PDF	Adobe PDF	SUPPORT¥WPDRV.ATR		60

キュー名は、Wndow に定義されているプリンタ名です。Adobe PDF 以外を使用する場合は、その名前を指定してください。



3. これで設定は終わりました。［OK］ボタンをクリックしてテーブルを閉じます。

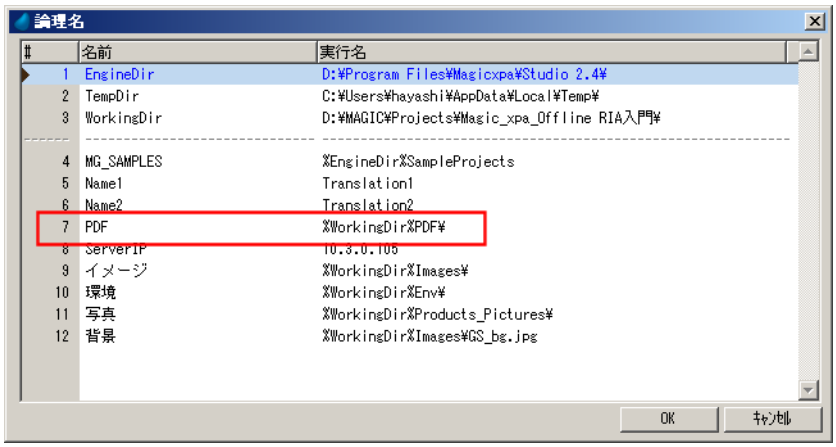
作成済みの＜受注書印刷＞プログラムは、ここで定義されたプリンタに出力されます。このプログラムを呼び出すようにします。

論理名の追加

画像と PDF の格納フォルダ名を論理名で定義しましょう。

1. ［オプション］メニューから、［設定／論理名］を選択し、［論理名］テーブルを開きます。
2. 最終行にパークし、一行作成（F4）します。
3. ［名前］カラムに＜イメージ＞と入力します。

4. ［実行名］カラムに、＜ %WorkingDir%Images¥ ＞と入力します。
5. さらに一行作成（F4）します。
6. ［名前］カラムに＜ PDF ＞と入力します。
7. ［実行名］カラムに、＜ %WorkingDir%PDF¥ ＞と入力します。
8. ［OK］ボタンをクリックします。



Images フォルダには、本コース添付の画像ファイルが入っています。

３．帳票印刷プログラムのデータソースを変更

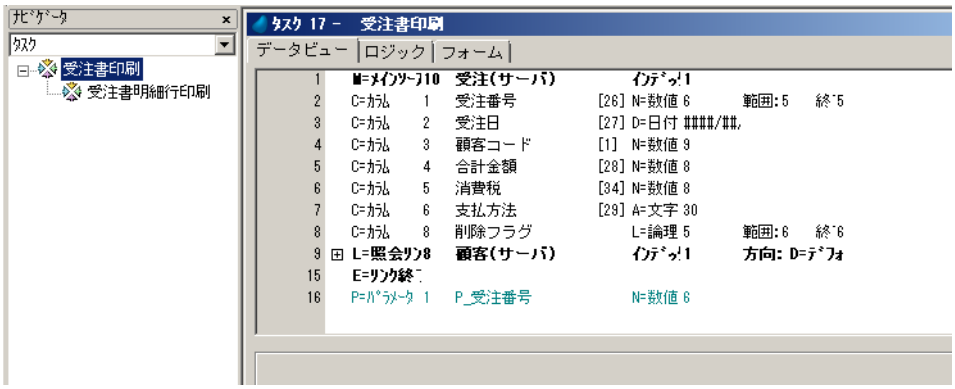
＜受注書印刷＞プログラムは、サーバ側のオンライン RIA プログラムです。このため、アクセスするデータソースもサーバ側のデータを指定する必要があります。プログラムを開いてデータソースの設定を変更します。

＜受注書印刷＞プログラムの修正

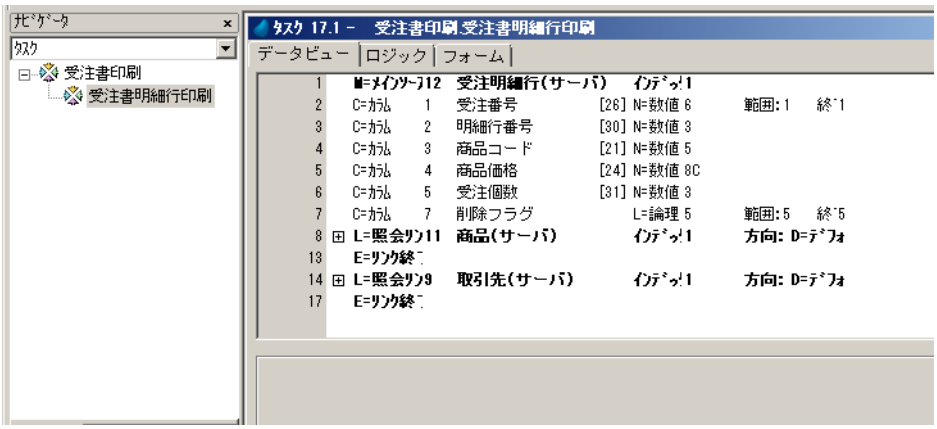
1. ＜受注書印刷＞プログラムをズームして開きます。
2. ［データビュー］エディタを開きます。
3. メインソースを＜受注（サーバ）＞に変更します。

4. メインソースの［削除フラグ］カラムをデータビューに追加し［範囲］特性に＜ 'FALSE'LOG ＞を設定します。

5. ［照会リンク］のデータソースを＜顧客（サーバ）＞に変更します。



6. ［ナビゲータ］ペインを開いて＜受注書明細行印刷＞タスクをクリックして開きます。
7. メインソースを＜受注明細行（サーバ）＞に変更します。
8. ［照会リンク］のデータソースを＜商品（サーバ）＞と＜取引先（サーバ）＞に変更します。
9. メインソースの［削除フラグ］カラムをデータビューに追加し［範囲］特性に＜**'FALSE'LOG**＞を設定します。



これで修正が終了しました。プログラムを閉じて保存します。

４．帳票印刷プログラムの呼び出し

オフライン RIA プログラムからサーバ側のオンライン RIA プログラムを呼び出すには、イベント経由で行う必要があります。この場合、以下の三つの処理を追加する必要があります。

- メインプログラムの［イベント］テーブルに印刷処理用のユーザ定義イベントを作成します。
- メインプログラムの［ロジック］エディタに、このイベントに対する［イベント］ロジックユニットを作成し、ここでオンライン RIA プログラムを呼び出します。
- オフライン RIA プログラムで、メインプログラムに定義されたイベントを発行します。

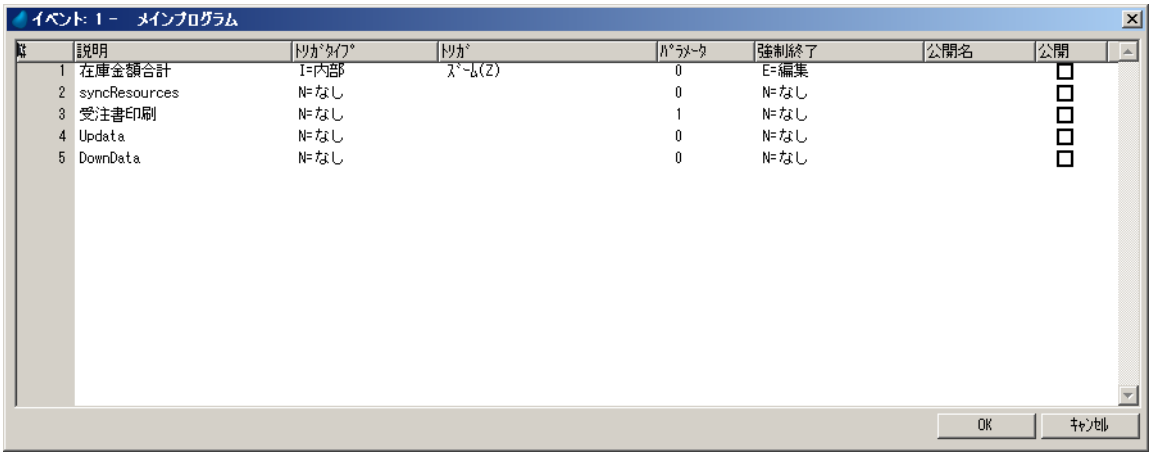
ここでは＜受注管理＞プログラムから＜受注書印刷＞プログラムを呼び出す処理を修正してゆきます。このとき、＜受注番号＞をパラメータとして＜受注書印刷＞プログラムに渡すようにします。

＜受注書印刷＞イベントの追加

1. ［メインプログラム］にズームして開きます。
2. Ctrl+U を押下して、［イベント］テーブルを開きます。
3. 最終行にパークし、一行作成（F4）します。

4. 次のように設定します。

#	名前	トリガタイプ	トリガ	パラメータ	強制終了
3	受注書印刷	N= なし		1 ※	N= なし



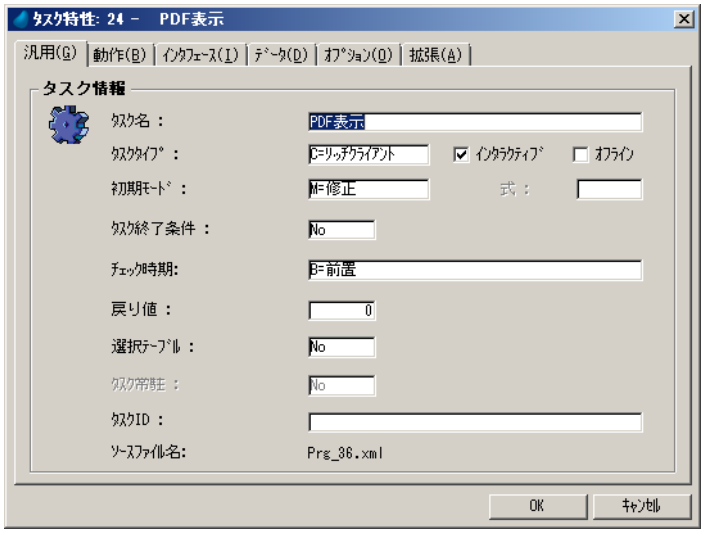
※［パラメータ］カラムからズームして、［イベントパラメータ］テーブルを開きます。一行追加して以下のようにパラメータを定義します。

#	名前	モデル	型	書式
1	P_ 受注書印刷	0	N= 数値	6

＜ PDF 表示＞プログラムの作成

＜受注管理＞プログラムには、＜ PDF 表示＞サブタスクが定義されていますが、＜受注書印刷＞プログラムがイベント経由で起動されるため、サブタスクが起動される前に PDF が作成されない可能性があります。これに対応するため＜ PDF 表示＞サブタスクと同じ構成のオンライン RIA プログラムを作成し、＜受注書印刷＞プログラムの実行後に起動されるようにします。

1. ［プログラム］リポジトリの最終行にパークして、一行追加（F4）します。
2. 名前を**＜ PDF 表示＞**としてズームしてプログラムを開きます。
3. ［タスク特性］やフォームは、＜ PDF 表示＞サブタスクと同じように定義します。
4. プログラムの［オフライン］特性は設定せず、オンライン RIA タスクとしています。



［イベント］ロジックユニットの追加

1. ［メインプログラム］の［ロジック］エディタを開きます。
2. 最終行にパークし Ctrl+H を押下して、［受注書印刷］イベントに対する［イベント］ロジックユニットを作成します。
3. このとき、パラメータ項目を追加するかどうかを確認するダイアログが表示されます。
4. ＜はい＞を選択すると、前述の＜ P_ 受注書印刷＞というパラメータ項目が追加されます。

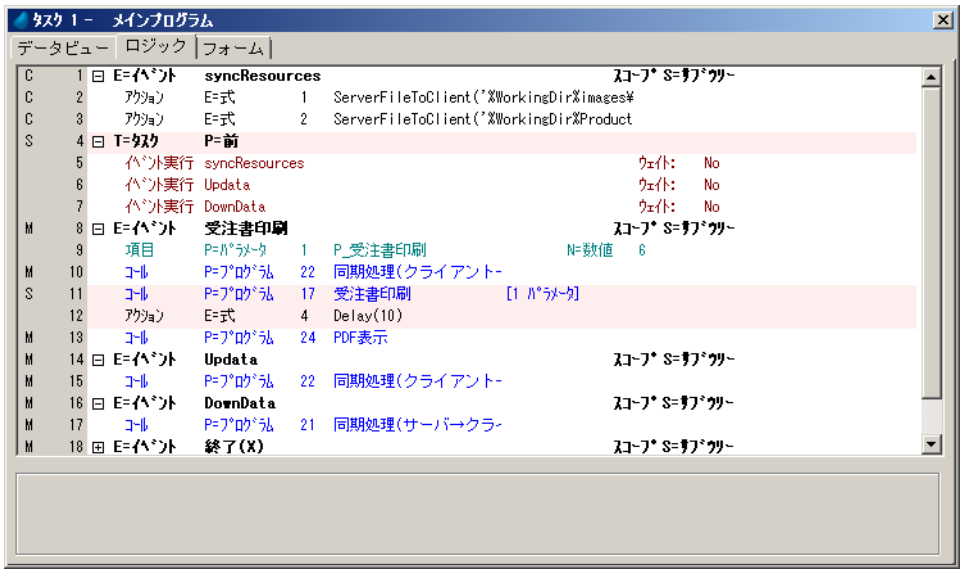
5. 四行追加して以下のように［コールプログラム］処理コマンドを追加します。

処理コマンド	項目	式	パラメータ
コール	P= プログラム	同期処理（クライアント→サーバ）	
コール	P= プログラム	受注書印刷	1 ※
アクション	E= 式	Delay(10)	
コール	P= プログラム	PDF 表示	

※［パラメータ］カラムからズームして、［パラメータ］テーブルを開きます。一行追加して以下のようにパラメータを定義します。

項目	式	説明	スキップ
A	0	P_ 受注番号	

- **印刷処理は、サーバ側で行われるため、事前に＜同期処理（クライアント→サーバ）＞プログラムを実行してサーバ側にデータをアップする必要があります。**
- **PDF の変換処理に時間がかかるため、＜受注書印刷＞プログラムと＜ PDF 表示＞プログラムの間で Delay() 関数を実行させて表示処理を遅らせています。**



イベント発行処理の追加

1. ＜受注管理＞プログラムの［ロジック］エディタを開きます。
2. ［印刷］イベントに対する［イベント］ロジックユニットの［コール］処理コマンドの行を以下のように修正します。

処理コマンド	項目	パラメータ	ウェイト	条件
イベント実行	受注書印刷	1 ※	No	Yes

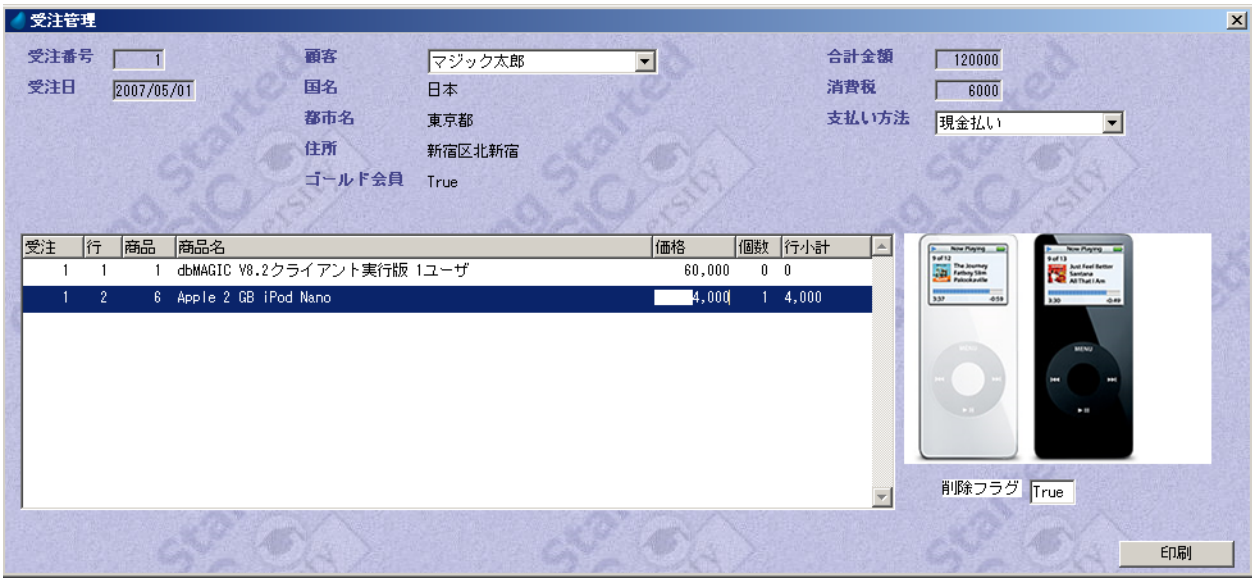
※［パラメータ］カラムからズームして、［パラメータ］テーブルを開きます。一行追加して以下のようにパラメータを定義します。

項目	式	説明	スキップ
B	0	受注番号	

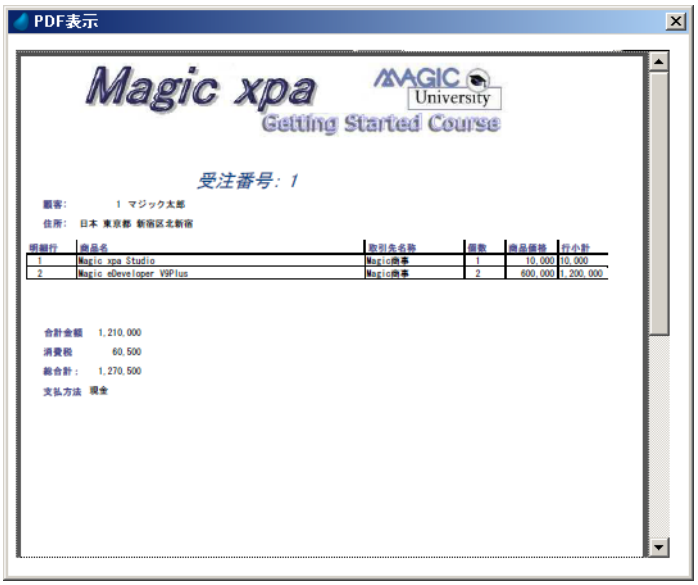
実行確認

作成したプログラムを実行してみましょう。

1. ［プログラム］リポジトリの＜受注管理＞にパークして、F7 で実行します。
2. ［受注管理］プログラムが実行したら必要に応じてデータを追加／修正してみてください。
3. ［印刷］ボタンをクリックしてください。



PDF が表示されます。



５．要約

本章では、オフライン RIA プログラムからサーバ側で実行するプログラムの呼び出し処理の作成について学びました。

具体的には、次のことを実習を通して学びました。

- PDF として出力させるための環境設定
- オフライン RIA プログラムからオンライン RIA プログラムを呼び出す処理の作成

ここまでで、サーバ上のバッチプログラムを実行して帳票を PDF として作成して表示する一連のプログラムが作成されました。この処理では、オンラインの処理を実行させるため、サーバに接続されている必要があり、接続されていない場合は、エラー処理を行う必要があります。

次章では、オフライン RIA プログラムとして必要な設定や、オンライン処理が要求された場合の処理について説明してゆきます。

第 11 章 オフラインの状態確認

本章ではオフライン RIA を実行させる上で必要な設定について説明します。また、オフライン状態でオンライン RIA プログラムが呼び出された場合の処理についても説明します。

キーワード

- ClientSessionSet() 関数
- ［無効サーバ］イベント
- ServerLastAccessStatus() 関数

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- オフライン RIA の事前設定
- オフライン状態でのステータス管理

参照

Magic ヘルプの［Magic xpa リファレンス／式エディタ／関数ディレクトリ］

同様に、［Magic xpa リファレンス／ Magic エンジン／エンジン実行レベル／イベントレベル］

１．はじめに

Magic xpa におけるオンライン RIA は、サーバに接続された状態で実行します。このため、サーバに接続されていない場合は、エラーとして処理されます。

しかし、オフライン RIA は、サーバに接続されていなくても動作することができ、エラーにならないようにしなければなりません。

また、サーバとのデータの同期処理や前述の印刷処理などは、サーバに接続するため、接続状態を判断する必要もあります。

２．接続エラーを発生させない設定

リッチクライアントのクライアント側のアプリケーションは、起動時にサーバ側のアプリケーションへの接続を試みます。通常は、接続できない場合、エラーとしてメッセージボックスを表示しますが、ClientSessionSet() 関数を実行させることでエラーメッセージを表示させないようにすることができます。

ClientSessionSet() 関数は、RIA におけるセッション環境の設定を行います。以下のように定義します。

構文：ClientSessionSet (キー , 値)

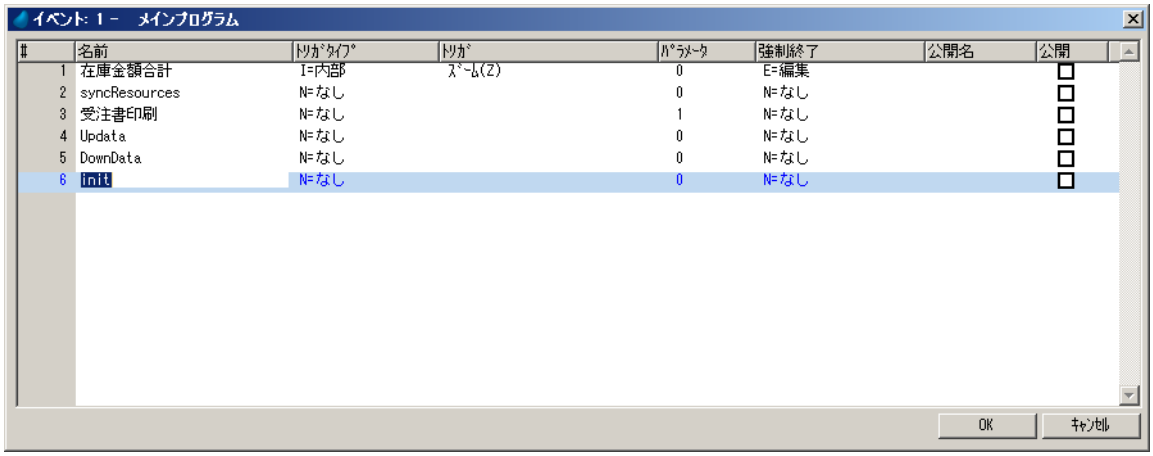
- **キー …… 環境設定用のキーを指定します。Ver2.4c では、＜ EnableCommunicationDialogs ＞の値のみサポートされています。**
- **値 …… キーに設定する値を指定します。**

ClientSessionSet(EnableCommunicationDialogs,False) を実行することで、クライアントがサーバに接続できなくてもエラーダイアログを表示しなくなります。

ClientSessionSet() 関数は、クライアント関数のため、［タスク前］に定義することはできません。メインプログラム内の［イベント］ロジックユニット内で実行させ、［タスク前］でこのイベントを発行するようにします。

1. ［メインプログラム］を開き、［ロジック］エディタを開きます。
2. Ctrl+U を押下して一行作成（F4）します。
3. 次のようにイベントを定義します。

名前	トリガタイプ	強制終了
init	N= なし	N= なし



4. ［ロジック］エディタに戻り、Ctrl+H を押下して、ロジックユニットを追加します。以下のように設定します。

ユニットタイプ	イベント	条件
E= イベント	init	Yes

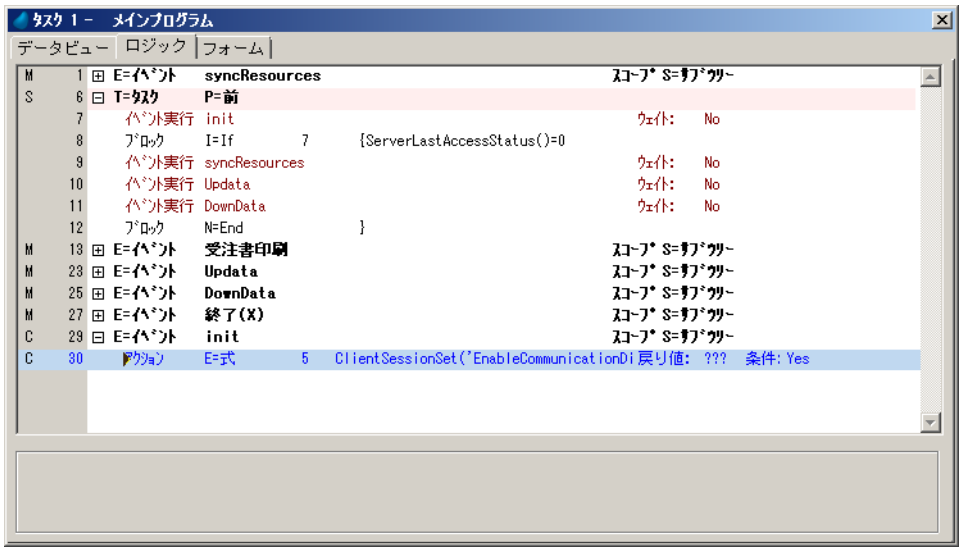
5. 一行作成（F4）して、以下のように処理コマンドを定義します。

処理コマンド		式	条件
アクション	E= 式	ClientSessionSet('EnableCommunicationDialogs','FALSE'LOG)	Yes

この関数を実行することで、クライアントがサーバに接続できなくても［エラー］ダイアログが表示されなくなります。

6. ［タスク前］ロジックユニットに一行作成（F4）して、以下のように処理コマンドを定義します。

処理コマンド	項目	ウェイト	条件
イベント実行	init	No	Yes



３．接続状態の確認

アプリケーションの起動時に前述の処理を実行することで、クライアントがサーバに接続できなくても［エラー］ダイアログが表示されなくなります。しかし、サーバ側のプログラムを呼び出すなどでサーバに接続する必要がある場合には、エンドユーザ向けにメッセージを表示させる必要があります。ここでは、サーバにアクセスできなかった場合の対応と、サーバ接続状態をチェックする処理をプログラムに反映します。

前述の処理で、サーバに接続できない場合に Magic xpa のエンジン側ではエラーにならないように設定されました。この状態でクライアントがサーバに接続できないと、［無効サーバ］という内部イベントが発生します。このイベントに対する［イベント］ロジックユニット内でメッセージを表示させることで、未接続状態であることが確認できます。

また、ServerLastAccessStatus() 関数を実行すること、最後にアクセスした時の状態を確認することができます。

ServerLastAccessStatus() 関数は、クライアント が最後にサーバにアクセスした時の状態を返します。以下のように定義します。

構文：ServerLastAccessStatus()

戻り値：サーバの直近の状態を数値で返します。

- **0 …… 成功**
- **1…… メタデータファイルが同期していない**
- **2…… サーバは、利用できません。**
- **3…… サーバは、リクエストを処理できません。**
- **4…… コンテキストは、利用できません**
- **5…… Magic xpa Studio で、リクエストを処理できません。**
- **6……RIA クライアントの実行特性がサーバへの接続をスキップするように設定されている場合。**

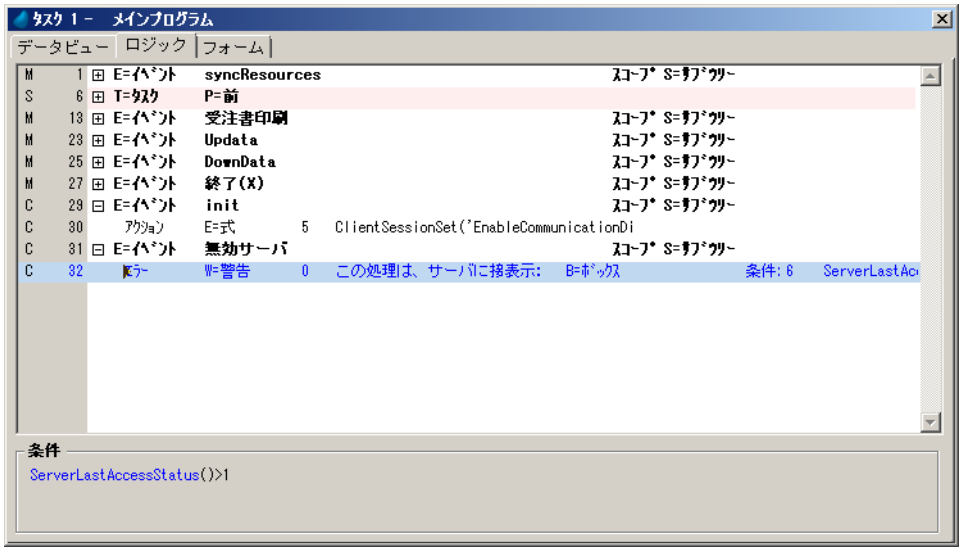
1. ［メインプログラム］の［ロジック］エディタを開きます。
2. Ctrl+H を押下してロジックユニットを追加します。
3. 次のように設定します。

ユニットタイプ	イベント	条件
E= イベント	無効サーバ	Yes

4. 一行作成（F4）して、以下のように処理コマンドを定義します。

処理コマンド	項目		条件
エラー	W= 警告	この処理は、サーバに接続する必要があります。	ServerLastAccessStatus()>1

ServerLastAccessStatus()>1 は、直近のサーバへのアクセスができなかったことを示しています。



次に、データの同期処理や印刷処理を接続状態の時にしかできないように実行条件を設定します。

データの同期処理に実行条件を設定する

データの同期処理は、サーバに接続されている状態で実行させる必要があります。この処理は［メインプログラム］の［タスク前］で起動するようにしていますが、ここで実行条件を指定します。

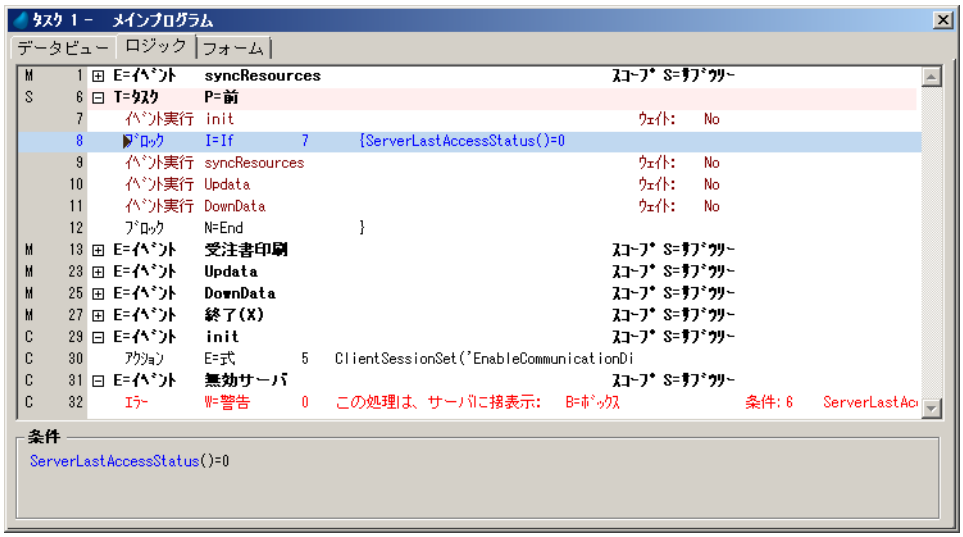
1. ［メインプログラム］の［タスク前］にパークします。
2. ＜ syncResource ＞イベントに対する［イベント実行］処理コマンドの前に一行作成（F4）します。
3. 次のように設定します。

処理コマンド	項目		条件
ブロック	I=If		ServerLastAccessStatus()=0
ブロック	N=End		

［ブロック End］処理コマンドは、自動的に追加されます。

4. 以下に定義されている三つの［イベント実行］処理コマンドを［ブロック］処理コマンド内に移動します。これによって、以下のようになります。

処理コマンド	項目		ウェイト	条件
ブロック	I=If			ServerLastAccessStatus()=0
イベント実行	syncResource		No	Yes
イベント実行	DownData		No	Yes
イベント実行	UpData		No	Yes
ブロック	N=End			



これによって、前回（[init] イベント処理時）のネットワーク接続時にエラーが発生していない場合のみ、データの同期処理を行うようになります。

5. ［終了（x)］イベントの［イベント］ロジックユニットの同期処理にも同様の条件を設定します。

処理コマンド			条件
コール	P= プログラム	同期処理（クライアント→サーバ）	ServerLastAccessStatus()=0

受注書印刷処理に実行条件を設定する

受注書印刷処理は、［メインプログラム］の［受注書印刷］イベントに対する［イベント］ロジックユニット内で起動していますが、ここにも実行条件を指定しましょう。

1. ［メインプログラム］の［受注書印刷］イベントに対する［イベント］ロジックユニットにパークします。
2. ＜同期処理（クライアント→サーバ）＞プログラムの呼び出し処理の前に一行作成（F4）します。
3. 次のように設定します。

処理コマンド	項目		条件
ブロック	I=If		ServerLastAccessStatus()=0
ブロック	N=End		

［ブロック End］処理コマンドは、自動的に追加されます。

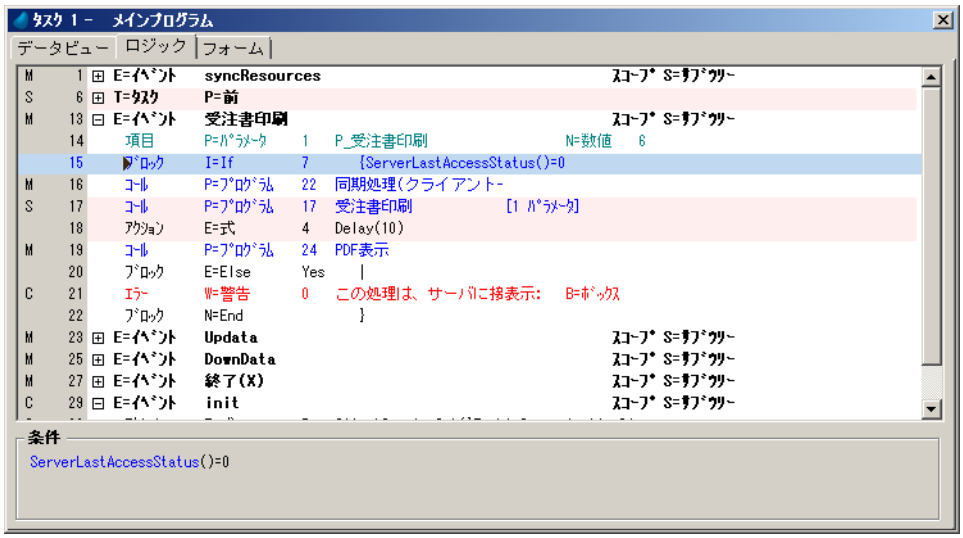
4. 下に定義されている四つの処理コマンドを［ブロック］処理コマンド内に移動します。これによって、以下のようになります。

処理コマンド	項目		パラメータ	条件
ブロック	I=If			ServerLastAccessStatus()=0
コール	P= プログラム	同期処理（クライアント→サーバ）		Yes
コール	P= プログラム	受注書印刷	1	Yes
アクション	E= 式	Delay(10)		Yes
コール	P= プログラム	PDF 表示		Yes
ブロック	N=End			

5. 次に、サーバに接続できなかった場合にメッセージを表示させるようにしましょう。＜ PDF 表示＞プログラムの呼び出し処理の行にパークして一行作成（F4）します。
6. ［ブロック Else］処理コマンドを定義して、一行作成（F4）します。

7. 以下のように警告メッセージを表示させます。

処理コマンド	項目		条件
ブロック	E=Else		Yes
エラー	W= 警告	この処理は、サーバに接続する必要があります。	Yes



これによって、前回のネットワーク接続時にエラーが発生していない場合のみ、クライアントのデータをサーバにアップした後、印刷処理を行うようになります。

４．要約

本章では、クライアントがサーバに接続されているかどうかの確認処理をアプリケーションに反映する方法について学びました。

具体的には、次のことを実習を通して学びました。

- **［無効サーバ］イベントによるメッセージの表示**
- **サーバ接続時のみオンライン RIA プログラムを呼び出すような条件設定**

これで簡単なオフライン RIA の作成が終了しました。

次章では、実際にオフラインとして動作することを確認していきます。

第 12 章 アプリケーションの公開

本章では、作成した RIA を別の PC からアクセスすることで、オフラインとしての動作を確認していきます。

キーワード

- リッチクライアントインタフェースビルダ
- 公開用 HTML ファイル

学習目標

本章を学ぶことで、次の内容に対する理解を深めましょう。

- 公開に必要なファイルの作成方法
- 他の PC からアクセスする方法
- オフラインアプリケーションとしての動作確認

参照

Magic ヘルプの［Magic xpa リファレンス／プロジェクトとアプリケーション］

同様に、［Magic xpa リファレンス／ Web 開発／リッチクライアントアプリケーション］

PC は既に RIA が実行可能な環境になっていることを前提として説明しています。環境設定について確認したい場合は『Getting Started for RIA』を参照してください。

１．アプリケーションの公開準備

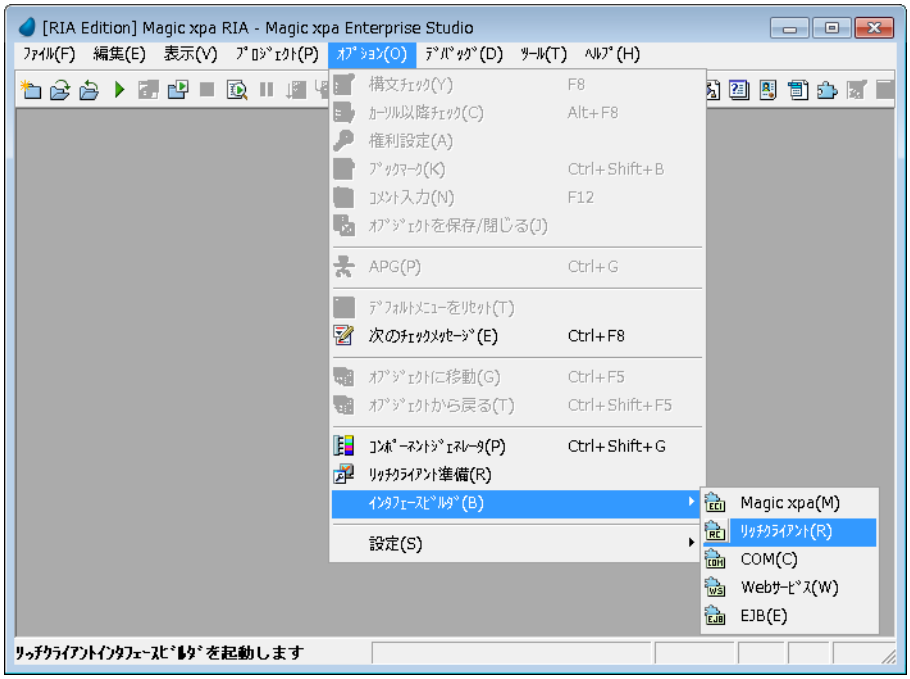
ここでは、RIA を公開するための作業を行います。作業内容は、通常のオンライン RIA の場合と同じです。他の PC から実行するためには、以下の二つのファイルを作成する必要があります。

- 公開用 HTML ファイル …… クライアントでの Click Once の有効性をチェックし、必要であればインストールします。ここには、起動プログラムなどのいくつかのアプリケーション実行環境などの XML 定義が含まれています。
- マニフェストファイル …… サーバパスなどのアプリケーション環境が含まれています。

これらのファイルを作成するには、「リッチクライアントインタフェースビルダ」と呼ばれるウィザード形式のユーティリティを使用します。開発環境でリッチクライアントインタフェースビルダを起動して公開用のファイルを作成していきましょう。

リッチクライアントインタフェースビルダの実行

1. ＜ Magic xpa RIA ＞プロジェクトを開いている状態でプルダウンメニューから［オプション／インタフェース ビルダー／リッチクライアント］を選択します。［ようこそ］ダイアログが表示されます。



2. 以降、ウイザードの指示に従ってクライアント環境設定用のファイルを作成してください。

MDI から各プログラムを実行させるため、［開始プログラム名］は指定しないでください。

２．クライアント側でのインストール処理

今までの作業で、アプリケーション公開用のファイルが作成されました。次は、このファイルを使用してクライアントからアプリケーションを実行させてみましょう。

Magic xpa が実行している PC でも構いませんし、この PC に TCP/IP で接続できる別の Windows PC があればそこからでも構いません。

同じ PC 上で確認を行う場合は、プロジェクト実行時に作成されるオフライン RIA のキャッシュファイルとの競合が発生するため、確認方法に注意が必要です。

まず、プロジェクトを実行し、その後、クライアント側で公開用 HTML ファイルを開きます。

プロジェクトの実行

1. Magic Sutdio を起動し、Magic xpa RIA プロジェクトを開きます。
2. ［デバッグ］メニューから、［プロジェクトの実行］（Ctrl+F7）を選択します。プロジェクトは実行状態になり、Magic の MDI が表示されます。

クライアントから実行

1. クライアント PC で IE（Internet Explore：※）を起動し、リッチクライアントインタフェースビルダの［概要］ダイアログに表示された URL を開きます。前述の例では、以下のようになります。

   http:// ＜ホスト名＞ /Magic24RIAApplications/Magic%20xpa%20RIA/WinDesktop/Magic%20xpa%20RIA.publish.html

2. 以下のようなポータル画面が表示されます。

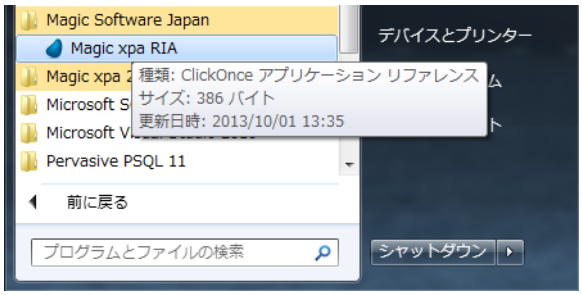
3. ［起動］をクリックすると、証明書の確認ダイアログが表示されれ、［OK］をクリックするとインストール処理が実行されます。

クライアント側のインストール処理を行ったことで、クライアント側の［スタート］メニューに起動用のメニューが登録されます。今後は、これを選択することで実行することができるようになります。

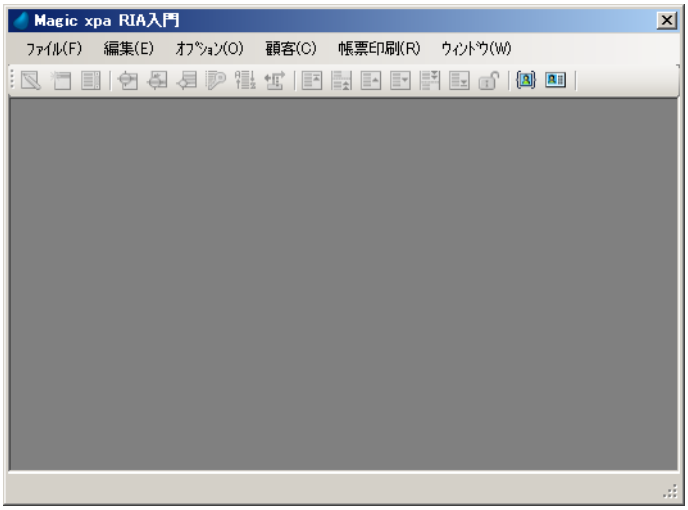
※：IE の他に、Firefox 上でも （.NET Framework 3.5 SP1 が必要）実行できます。

クライアントから実行

1. Windows のスタートメニューを開きます。［すべてのプログラム］を選択します。
2. 「Magic Software Japan」というメニューをクリックすると「Magic xpa RIA」が表示されます。このメニューをクリックするとアプリケーションが起動されます。



「Magic Software Japan」は、リッチクライアント インタフェースビルダで［発行元の名前］に設定した名前です。
「Magic xpa RIA」は、［アプリケーション タイトル］に設定した名前です。



3. 起動されると、上のような Magic xpa の MDI ウィンドウが表示されます。
4. Magic xpa Studio が実行している同じ PC 上で確認している場合は、同じ MDI が二つ表示されているはずなので、プロジェクト実行によって開いた MDI を閉じてください。
5. MDI 上に表示されているプルダウンメニューからプログラムを選択して動作を確認してみてください。

オフライン RIA のクライアント実行を終了してください。次に、オフラインとしての動作を確認してみましょう。

３．オフラインの確認

では、オフラインでの動作を確認してみましょう。サーバ側でアプリケーションの実行を止めることで、クライアント側で実行することが確認できます。

サーバ側を停止して確認

1. サーバ側（Magic xpa Studio）でプロジェクトが実行状態になっている場合は、［デバッグ］メニューの［停止］をクリックして停止します。
2. クライアント側の「Magic Software Japan」／「Magic xpa RIA」メニューをクリックしてアプリケーションを起動します。

先ほどと同じ画面が表示されるはずです。このとき、サーバ側のアプリケーションは停止状態です。

3. プルダウンメニューから［顧客（C)］／［照会 - 顧客］を選択してください。［顧客］データの内容が表示されます。これはクライアント側の［顧客］データを表示しています。

4. ［帳票印刷（R)］／［受注管理］を選択してください。［受注管理］プログラムが表示されます。壁紙や製品イメージも表示されます。
5. ［印刷］ボタンをクリックしてみてください。＜この処理は、サーバに接続する必要があります。＞というメッセージが表示され、何も行われません。
6. クライアント側でアプリケーションを終了してください。

サーバ側を再度起動して確認

1. サーバ側で Ctrl+F7 を押下してアプリケーションを実行させます。
2. クライアント側の「Magic Software Japan」／「Magic xpa RIA」メニューをクリックしてアプリケーションを起動します。

3. プロジェクト実行によって開いた MDI を閉じてください。
4. ［帳票印刷（R)］／［受注管理］を選択してください。［受注管理］プログラムが表示されます。
5. ［印刷］ボタンをクリックしてみてください。今度は、印刷処理が実行されるはずです。

４．要約

本章では、オフライン RIA をオフライン環境で実行することを確認しました。このようにしてサーバへの接続が無くても実行することができるアプリケーションを開発することができます。

- **オフライン環境での動作**
- **オフライン環境でサーバプログラムを呼び出した場合の動作**

今まで作成してきたオフライン RIA は、非常に簡単なものです。実際の運用するアプリケーションを開発する場合、どのような機能をオフライン化するべきか、またデータ同期のタイミングなどを色々検討する必要があるはずです。

オフライン RIA の開発作業に本格的に取り組むことになった場合は、そのようなアプリケーション仕様に基づいた設定について検討していただき、ぜひ、チャレンジしてみてください。

第 13 章 まとめ

いままで、Magic xpa でオフライン RIA を開発する上での基本的なプログラム方法について説明していきました。最後に内容の整理を行い、また追加情報を説明してゆきます。

１．オフライン RIA の開発の流れ

Magic xpa のオフライン RIA の開発の手順をまとめると以下のようになります。

データベースの準備

オフライン用に Local DBMS を使用したデータベースを定義します。その際、［データベース］テーブルの［位置］カラムにデータベースファイルを指定します。

第 4 章「＜ Local/Server ＞データベースの定義」（ページ 15）を参照してください。

データソースの定義

オフライン用とオンライン用に同じデータ構成のデータソースを定義します。その際、以下の二つのカラムを追加しておく必要があります。

- **更新日時 …… 文字型 14**
- **削除フラグ …… 論理型**

第 4 章「データソースの変更」（ページ 17）を参照してください。

オフライン環境の設定処理の作成

メインプログラムに［イベント］ロジックユニットを定義し（本コースでは、［init］というユーザイベント）、この中で次の関数を実行します。

```
ClientSessionSet('EnableCommunicationDialogs','FALSE'LOG)
```

メインプログラムの［タスク前］で、このイベントを発行します。

これによって、クライアントがサーバに接続できなくてもエラーダイアログが表示されなくなります。

第 11 章「接続エラーを発生させない設定」（ページ 80）を参照してください。

リソースのダウンロード処理の作成

メインプログラムに［イベント］ロジックユニットを定義し（本コースでは、［syncResources］というユーザイベント）、この中で次の関数を実行します。

```
ServerFileToClient（＜サーバ側のフォルダ名＞）
```

メインプログラムの［タスク前］で、このイベントを発行します。

これによって、サーバ側の指定されたフォルダ内のファイルがクライアントにコピーされます。

第 3 章「画像データの同期処理」（ページ 10）を参照してください。

データソースの同期処理の作成

次の関数をタスク内で実行すると、そのタスクのメインソースによるデータビューの内容を別のデータソースにコピーできます。

```
DataViewToDataSource(0,''，＜データソース番号＞,'','')
```

これを利用して、サーバとローカル間でのデータソースの同期プログラムを作成し、実行するようにします。

第 9 章「サーバ側とのデータ同期」（ページ 56）を参照してください。

実行タイミングは以下のような場所が考えられます。

自動

- メインプログラムの［タスク前］…… サーバ → クライアント
- メインプログラムの［タスク後］…… クライアント → サーバ
- サーバ側の処理を実行する前 …… クライアント → サーバ

手動

プッシュボタンやメニューから起動できるようにする。

オフライン RIA プログラムの作成／修正

オフライン RIA プログラムを作成する場合に必要な設定事項は以下の通りです。

- ［プログラム］リポジトリの［オフライン］カラムをオンにします。
- 利用するデータソースは、全てローカルデータソースにします。
- ［削除フラグ］カラムが< False >に設定されているレコードのみ表示します。
- ［コール OS コマンド］処理コマンドの［実行］特性を、< C= クライアント>に設定します。
- サーバ側のリソース（イメージファイルなど）は、事前にクライアントに転送しておきます。
- レコードが更新された場合、更新日時をレコード毎に記録します。
- レコードの削除処理を行う場合、実際にはレコードを削除せず、［削除フラグ］カラムを< True >に設定します。
- クライアントで実行させるバッチタスクは、非インタラクティブなリッチクライアントタスクに変更します。

第 2 章「オフライン RIA プログラムの作成」（ページ 2）を参照してください。

オフライン RIA プログラムからオンライン RIA プログラムを呼び出す処理の作成

オフライン RIA プログラム内から［コール］処理コマンドでオンライン RIA プログラムを呼び出すことはできません。呼び出すには、以下のような処理を組み込んでイベントを経由する必要があります。

1. ［メインプログラム］にユーザ定義関数を定義します。
2. このユーザ定義関数をトリガとする［イベント］ロジックユニットを作成し、ここから［コール］処理コマンドでオンライン RIA プログラムを呼び出します。
3. オフライン RIA プログラム内から［イベント実行］処理コマンドでユーザ定義関数を発行します。

第 10 章「帳票印刷プログラムの呼び出し」（ページ 74）を参照してください。

サーバとの接続状況の把握

オフライン内でオンライン RIA プログラムを呼び出す場合は、サーバに接続しているかどうかを確認する必要があります。次の二つの処理をメインプログラムに反映します。

- ［無効サーバ］イベントによる［イベント］ロジックユニットを定義する。
- オンライン RIA プログラムを呼び出す実行条件に以下の式を指定します。

```
ServerLastAccessStatus()=0
```

第 11 章「接続状態の確認」（ページ 81）を参照してください。

２．パフォーマンスの改善

オフライン RIA は、サーバとのデータやリソースの転送処理が処理のパフォーマンスに影響します。この処理の負荷を低減する方法として、データファイルの転送処理について説明します。

Local DBMS によって作成されるデータファイルは、SQLite DBMS と互換性があります。このため、SQLite で作成されたデータファイルをローカルにそのままコピーすることでローカル用データとして利用することができます。

確認してみましょう。

1. 本コースの DB フォルダ内に ＜ Server.sqlite ＞ファイルがあることを確認してください。このファイルは、サーバ側の SQLite のデータファイルです。
2. このファイルを**＜ Local.sqlite ＞**という名前でコピーしてください。
3. ＜ Local.sqlite ＞ファイルをクライアントのキャッシュフォルダに移します。フォルダ名は、第 3 章「プログラムの実行と確認」（ページ 12）を参照してください。以下のようになります。

   C:¥Users¥<User id>¥AppData¥Local¥Temp¥MgxpaRIACache¥< クライアントのホスト名 >
4. 作成したオフライン RIA プログラムを実行してデータの内容を確認してみてください。

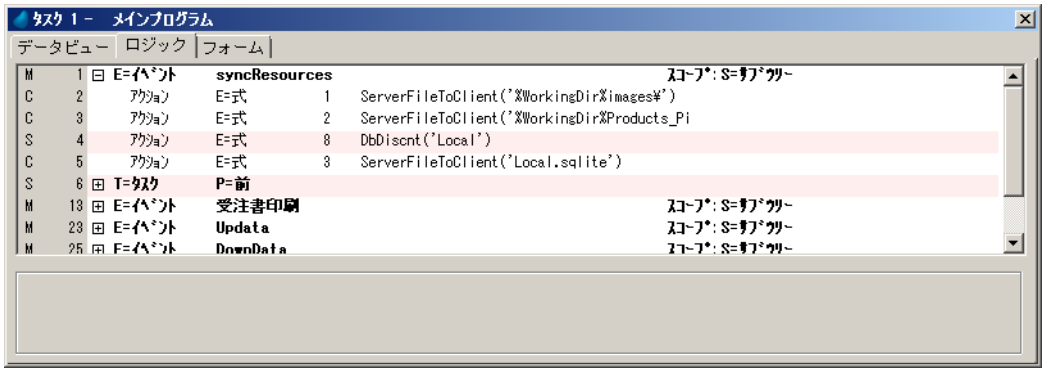
この処理をアプリケーション内で行う場合は、プログラムを以下のように修正します。

1. 本コースの DB フォルダ内の ＜ Server.sqlite ＞ファイルを**＜ Local.sqlite ＞**という名前でプロジェクトの作業フォルダにコピーします。
2. ［メインプログラム］を開きます。
3. ［syncResource］イベントの［イベント］ロジックユニットの最終行にパークします。
4. 二行追加（F4）して次のように［アクション］処理コマンドを追加します。

コマンド	項目	式	条件
アクション	E= 式	DbDiscnt('Local')	Yes
アクション	E= 式	ServerFileToClient('Local.sqlite')	Yes

DbDiscnt() 関数を実行することで、事前に Magic エンジンが 'Local' データベース切断し、上書きできるようにします。

DB ファイルの指定には、パスが設定されていないことに注意してください。［データベース］テーブルの＜ Local ＞データベースの［位置］には、＜ Local.sqlite ＞と設定されているため（第 4 章「＜ Local/Server ＞データベースの定義」（ページ 15）を参照してください）、同じように指定するだけでアクセスできるようになります。［位置］カラムにパスが設定されている場合は、ServerFileToClient() 関数のパラメータにもパスを設定する必要があります。



5. 動作を確認する前に、あらかじめクライアントのキャッシュを削除しておいてください。各オフライン RIA プログラムを実行すると、既にデータが入力されていることが確認できます。

この処理は、レコード数が多く、クライアント側で更新することないマスタ系のデータソースに対して利用すると効果的です。このような場合は、サーバ／クライアントの両方のデータベースをマスタデータ用とトランザクションデータ用に分け、マスタ用のデータファイルのみをクライアントコピーするようにしてください。

３．サンプルアプリケーションについて

制限事項

本コースの作成されたアプリケーションは、マルチユーザはサポートされません。特定のクライアント（またはユーザ）とサーバ間のみに有効です。

実際のマルチユーザでのシステム設計時には、各クライアントのローカルデータは、特定のクライアントのデータやユーザデータのみを確保する等の対応が必要です。

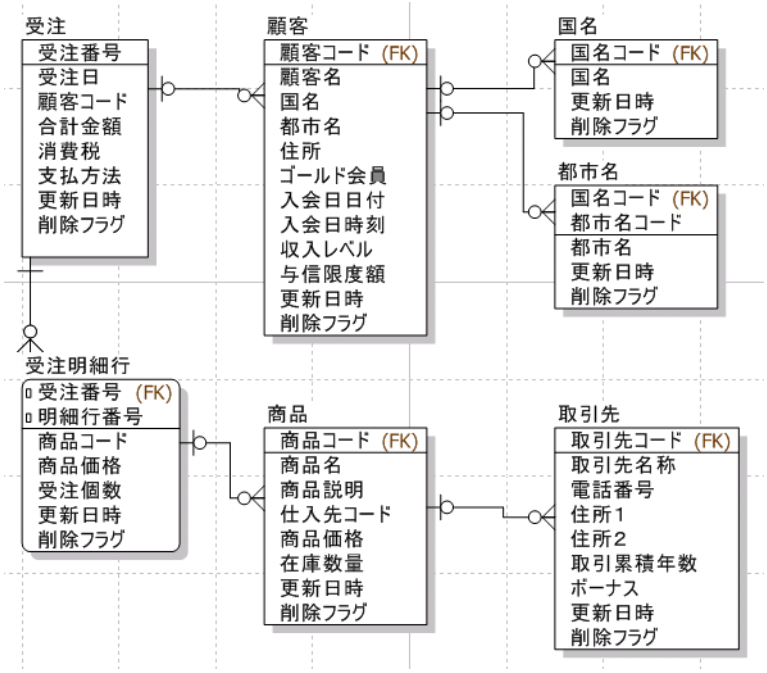
その他の修正

本セミナー上には説明がありませんが、以下のプログラムは修正が必要になります。

- （#14 ）＜印刷 - 顧客＞ …… メインソースを（#1）＜顧客＞から（#8）＜顧客（サーバ）＞に変更
- （#15 ）＜印刷 - 顧客２＞ …… メインソースを（#1）＜顧客＞から（#8）＜顧客（サーバ）＞に変更
- （#16 ）＜印刷－仕入先一覧＞ …… メインソースを（#2）＜取引先＞から（#8）＜取引先（サーバ）＞に変更
- （#18 ）＜仕入先別製品印刷＞ …… メインソースを（#2）＜取引先＞から（#8）＜取引先（サーバ）＞に変更
- （#18.1 ）＜製品印刷＞ …… メインソースを（#4）＜商品＞から（#11）＜商品（サーバ）＞に変更

４．補足資料

本セミナー用サンプルアプリケーションのデータ構成は以下のようになっています。



Getting Started for Offline RIA Magic xpa

第二版　2014 年 9 月 26 日
発行　〒 169-0074　東京都新宿区北新宿２丁目２１番地１号
新宿フロントタワー 24 階

Magic Software Japan K.K.