.NET チュートリアル

Magic xpa

目次

はじめに

このドキュメントは、Magic xpa で追加された.NET 機能を使用したサンプルについて説明したものです。

このドキュメントは、追加された各機能について全て説明しているわけではありません。各機能の詳細は、Magic xpa のリファレンスヘルプを参照してください。

このパッケージには、.NET 機能を使用したサンプルプロジェクトが含まれています。このガイドに従うことによって、最初からプロジェクトを作成することができます。

必要条件

サンプルアプリケーションを使用するためには、PC 上に以下のソフトがインストールされている必要があります。

- Magic xpa Ver2.2
- .NET Framework Ver2.0 SP1以上。

.NET アセンブリの定義

.NET を利用したアプリケーションを開発するには、使用するアセンブリを定義する必要があります。アセンブリ定義は、[コンポーネント]リポジトリで行います。

1. ［コンポーネント］リポジトリを開き（Shift+F7）、１行追加します。
2. ［タイプ］カラムで「.NET」を選択します。
3. ［名前］カラムからズーム（F5）し、［.NET アセンブリ選択］一覧を開きます。
4. ［.NET アセンブリ選択］一覧には、現在インストールされているアセンブリのリストが表示されます。また、他の場所にあるアセンブリを選択することもできます。ここで、System アセンブルを選択します。同じ名前で複数のバージョンのアセンブリが表示される場合があります。このような場合は、最新のバージョンを選択してください。
5. 一行追加して、同様に System.Drawing アセンブリを選択します。
6. さらに一行追加して、System.Windows.Forms アセンブリを選択します。

これでアセンブリが使用できるようになりました。

.NET コントロールの使用

.NET を利用する場合は、NET コントロールのオブジェクトタイプを定義する必要があります。

オブジェクトタイプは、.NET コントロールのモデルか.NET 型のデータ項目で定義されます。

.NET コントロールモデルの定義

次に、フォーム上でコントロールを使用するために、[.NET]コントロールのモデルを定義します。

後述しますが、モデルを定義しなくても[.NET]コントロールを使用することはできます。

1. ［クラス］を「GUI 表示」または「リッチクライアント表示」に設定したモデルを作成して、その［型］を「.NET」に設定してください。
2. モデルの［特性］シートを開いて、［オブジェクトタイプ］を定義してください（例えば System.Windows.Forms.MonthCalendar）。
   Ctrl+Space を押下することでオートコンプリート機能を使用することができます。ドット（.）を入力すると、（Magic xpa の内部関数を選択する場合と同じように）利用可能なオプションの選択リストが開きます。
3. 新しいプログラムを作成してください。
4. プログラムの［フォーム］エディタを開き、［.NET］コントロールをフォームに配置してください。
5. ［.NET］コントロールにモデルを割り当ててください。

"System.Windows.Forms.Control"を継承するオブジェクトタイプが定義されたモデルのみ、フォーム上の［.NET］コントロールに割り当てることができます。

これで、［.NET］コントロールの定義は終了しました。このプログラムを実行すると、プログラム上には、.NET のカレンダーコントロールが表示されます。

注：リッチクライアント・プログラムでは、別のタイプのデータ項目を定義しておく必要があります。

サンプルプロジェクトでは、オンライン・プログラム（#3）とリッチクライアント・プログラム（#33）は、上記で作成されたものです。

コントロール特性の変更

[.NET]コントロールの[.NET]特性を変更します。

1. 既に作成されたプログラムを開き、［フォーム］エディタを開き、［.NET］コントロールを選択します。
2. コントロールの［特性］シートを開き、［.NET］セクションに移動します。
3. ［オブジェクトプロパティ］特性でズームします。
4. Visual Studio で表示させる場合と同じように、現在の［.NET］コントロールのプロパティを変更することができます（オブジェクトの内容を元に表示しているため、英語表記になります）。
   例えば、［FirstDayOfWeek］プロパティを「Tuesday」に変更します。

これで終了です。このプログラムを実行すると、カレンダーの左側が火曜日から始まっていることが確認できます。

サンプルプロジェクトでは、オンライン・プログラム（#4）とリッチクライアント・プログラム（#34）は、上記で作成されたものです。

注意：

- 簡単なプロパティであれば、値を直接入力したり、コンボボックスから入力したりすることで変更できます。
- .NETオブジェクトの複雑なプロパティを設定する場合は、式でのみ値を設定できます（後述します）。
- コントロール用のモデルを定義して、そこで簡単なプロパティを定義することができます。

データを.NET コントロールにリンクする

コントロールの[データ]特性を使用することで Magic の組み込みコントロールと同じように.NET コントロールにデータを簡単にリンクさせることができます。

.NET コントロールは、多くのプロパティとイベントを持っているため、データ項目をデータ特性に割り当てる前に、どの.NET プロパティがデータ値を取得し、そして、どの.NET イベントがデータ項目に.NET のプロパティ値を設定するためのトリガであるか定義する必要があります。

この定義は、[Value プロパティ名]と[Value Changed イベント名]特性というモデルレベルの特性で定義されます。

これで、[.NET]コントロールのモデルに.NET バインディング宣言を追加されます。

この定義は、データをリンクさせたいコントロールのタイプ毎に一度だけ行うことができます。

1. 作成した［.NET］コントロールのモデルの［特性］シートを開きます。
2. ［Value プロパティ名］特性にデータ項目にリンクしたり、.NET プロパティ（例えば SelectionStart）を設定したりしてください。
3. ［Value Changed イベント名］特性にデータ項目を更新させるトリガとなるイベント（例えば DateChanged）を設定してください。

モデルは、次に、データ項目データを.NET コントロールにリンクさせるために使用することができます。

次にデータ項目を.NET コントロールに結びつけます。

4. 作成したプログラムを開いてください。
5. このプログラムでデータ項目を作成して、現在までその属性を設定してください。
6. フォームにズームして、.NET コントロールを選択します。
7. コントロールの特性シートを開いてください。
8. データ特性に作成したデータ項目を割り当ててください。

日付データ項目を Month Calendar の.NET コントロールにリンクすることができました。リンクされたデータを表示させるには、日付データ項目を更新して、フォームにコントロールを配置します。

9. 作成された日付データ項目へ行って、代入式（例えば Date()）を設定してください。
10. フォームにズームして、データ特性に式を追加します。.NET コントロールにすでに割り当てられているため、データ項目を直接フォームに配置することはできません。

これで、新しいプログラムを実行すると、.NET カレンダーコントロールが日付データ項目によって更新されます。そして、.NET カレンダーのコントロール値を変更すると日付データ項目も更新されます。

サンプルプロジェクトでは、オンライン・プログラム（#5）とリッチクライアント・プログラム（#35）は、上記で作成されたものです。

.NET 項目の定義

次に、フォーム上のコントロールとして使用する.NET 項目を定義します。

後で項目を定義して.NET メソッドを実行させる事もできます（後述します）。

1. 新しいプログラムを作成します。［タスクタイプ］を「リッチクライアント（または、オンライン）」に設定します。
2. このプログラムに変数項目を作成し、型を「.NET」に設定します。
3. 項目の［特性］シートを開き、［オブジェクトタイプ］特性を定義します（例：System.Windows.Forms.MonthCalendar）。
   Ctrl+スペースを押下することでオートコンプリート機能を使用することができます。ドット（.）を入力すると、（Magic xpa の内部関数を選択するように）入力可能なオプションを持つ選択リストが開きます。
4. プログラムの［フォーム］エディタを開き、この項目をフォームに配置します。
   オブジェクトタイプが「System.Windows.Forms.Control」から継承される項目だけがフォーム上のコントロールとして使用できます。

これで、.NET 項目の定義が終わりました。

作成したプログラムを実行すると、リッチクライアント・プログラム上に.NET カレンダーコントロールが表示されます。

サンプルプロジェクトでは、オンライン・プログラム（#6）とリッチクライアント・プログラム（#36）は、上記で作成されたものです。

.NET イベントの処理

.NET オブジェクトによって発行されたイベントに対応した処理を定義します。

1. プログラムを開き、［ロジック］エディタを開きます。
2. 新しい［イベント］ロジックユニットを作成し、［イベントタイプ］を「.NET」に設定します。
3. ［イベント］欄でズームして、［.NET イベント］ダイアログを表示します。
4. ［項目］欄には、「My .NET Calander」を選択します。
5. ［イベント］欄には、「DateSelected」を選択します。
6. ［エラー］処理コマンドを追加して、メッセージテキストとして「Hellow World」を設定します。

これで、このプログラムを実行し、日付をクリックと、メッセージが表示されることが確認できます。

注意：

フォームに配置されていない.NET オブジェクトによって発行されたイベントは、Magic イベントとは異なる処理になります。詳細は、Magic xpa のリファレンスヘルプを参照してください。

特性プロパティの値を取得

.NET オブジェクトの現在の値を取得します。

.NET オブジェクトの値を取得するには、単純に.NET 項目を記述し、ドット(.)に続くプロパティを設定するだけです。ドットを押下すると、オートコンプリート機能が動作します。

(Magic xpa の内部関数を選択するように)Ctrl+スペースを押下することで選択リストが開きます。選択された日付を取得するには、項目の[SelectionStart]プロパティを使用します。

[SelectionStart]プロパティは、.NET の DateTime 型のプロパティです。[エラー]処理コマンドでこの内容を表示させるには、文字列に変換する必要があります。Magic xpa の DStr()関数を使用することで変換できます。

[イベント]ロジックユニットで、「Hellow World」のメッセージを以下の式に置き換えてください。

DStr(A.SelectionStart,'YYYY/MM/DD')

Magic xpa は、自動的にデータの型を.NET タイプから Magic タイプに変換するため、直接 Magic 関数で使用することができます。

これで終了です。

このプログラムを実行し日付をクリックすると、クリックした日付が表示されることが確認できます。

.NET メソッドを呼び出す

DateTime 値を文字列に変換する場合、Magic xpa の Dstr()関数を使用する代わりに、ToShortDateString()という.NET のメソッドを使用することもできます。

[SelectionStart]プロパティで取得した値は、このメソッドを実行する事で直接変換することができます。.NET のメソッドを実行する場合は、プロパティの値を変数項目に一時的に保存する必要がありません。

[イベント]ロジックユニットで定義するには、以下の式に差し替えてください。

DStr(A.SelectionStart,'YYYY/MM/DD') → A.SelectionStart.ToShortDateString()

これで終了です。このプログラムを実行し、日付をクリックすると、クリックされた日付が表示されることが確認できます。

サンプルプロジェクトでは、オンライン・プログラム(#7)とリッチクライアント・プログラム(#37)は、上記で作成されたものです。

.NET プロパティの値を更新

.NET プロパティの値を動的に変更する場合、DNSet()関数を使用することができます。

この関数は、適切な値を持つどのような.NET プロパティに対しても更新することができます。

例：

1. 新しい［イベント］ロジックユニットを作成し、システムイベントの Ctrl+1 によって実行されるように設定します。［アクション］処理コマンドを定義し、以下の式を設定します。
   DNSet(A.ShowWeekNumbers,'TRUE'LOG)
   これにより、各行の最初に週番号が表示されます。
2. プログラムを実行し、Ctrl+1 を押下してください。カレンダー表示がそれに応じて変更されます。

プロパティを更新するために、簡単な値を使用しているため、簡単に更新できます。列挙値(次のサンプルで確認できます)や他のメソッドからの戻り値(後述します)でプロパティを更新することもできます。

サンプルプロジェクトでは、オンライン・プログラム(#8)とリッチクライアント・プログラム(#38)は、上記で作成されたものです。

プロパティについての内容は、各オブジェクトの仕様を参照してください。.NET Framework によって提供されているメンバのプロパティについては、Microsoft 社のサイトを参照してください。

DotNet 内部コード

クラスや値を選択するためには、.NET にアクセスする必要があります。例えば、カレンダーコントロールの背景色を青に設定する場合、System.Drawing.Color.Blue の列挙値を使用することができます。

.NET オブジェクトに直接アクセスするには、必要な.NET オブジェクトの接頭辞として"DotNet"と呼ばれる内部コードを使用することができます。例えば、DotNet.System.Drawing.Color.Blue では必要な値を戻します。

従って、カレンダーコントロールの背景色を変更したい場合、以下のように設定してください。

1. 新しい［イベント］ロジックユニットを作成し、システムイベントの Ctrl+2 で実行するようにします。［アクション］処理コマンドを追加して、以下の式を設定します。
   DNSet(A.BackColor,DotNet.System.Drawing.Color.Blue)
2. プログラムを実行し、Ctrl+2 を押下します。それに応じてカレンダーの表示が切り替わることが確認できます。

.NET 変数を使用することで、式エディタ内で.NET メソッドを直接使用することができます。この場合も、DotNet キーワードが必要です。

カレンダーコントロールの背景色を特定の色に変更する場合に、System.Drawing.Color.FromArgb()メソッドを呼出し、RGB 値(250、30、20)で変更するには、以下のように設定します。

3. ［アクション］処理コマンドの式を以下のように変更します。
   DNSet(A.BackColor,DotNet.System.Drawing.Color.FromArgb(250,30,20))
4. プログラムを実行し、Ctrl+2 を押下します。それに応じて、カレンダーの表示が切り替わることが確認できます。

同じ方法で、カレンダーコントロールのフォントを変更する場合、System.Drawing.Font()メソッドを使用することができます。

5. ［アクション］処理コマンドを追加し、以下の式を設定します。
   DNSet(A.Font,DotNet.System.Drawing.Font('Times New Roman',10))

サンプルプロジェクトでは、オンライン・プログラム（#8）とリッチクライアント・プログラム（#38）は、上記で作成されたものです。

注意：

Windows の視覚効果でテーマを使用している場合は、色やフォントが変更できません。Windows でテーマを無効にするか、Magic xpa の[動作環境]ダイアログの[Windows XP テーマを使用]を「No」に設定して確認してください。

参考：

太字フォントを指定する場合は、FontStyle 列挙体で以下のように指定します。

```
DNSet(A.Font,DotNet.System.Drawing.Font('Times New Roman',10,
DotNet.System.Drawing.FontStyle.Bold))
```

例外の処理

Magic xpa は、.NET オブジェクトを使用する際に発生した.NET の例外処理を支援するために 2 つの関数を提供しています。

- DNExceptionOccurred……直近の.NETの処理でエラーが発生したことを通知します。
- DNException…….NET処理によって発生した直近の例外の参照を返します。実際に使用する場合は、取得したいプロパティを指定します（例:エラーメッセージの場合、DNException.Message）。

2 月 30 日という日付を設定する不正処理を実行したとします。

1. 新しい［イベント］ロジックユニットを作成し、システムイベントの Ctrl+3 を定義します。［アクション］処理コマンドを追加してく、以下の式を設定します。
   A. SetDate(DotNet.System.DateTime(2001,2,30))
2. ［エラー］処理コマンドを追加し、以下の式 DNException().Message をメッセージとします。実行条件には、以下の式を定義します。
   DNExceptionOccurred()
3. プログラムを実行し、Ctrl+3 を押下します。

エラーメッセージが表示されることを確認します。日付が適切であれば、メッセージは表示されません。

サンプルプロジェクトでは、オンライン・プログラム（#9）とリッチクライアント・プログラム（#39）は、上記で作成されたものです。

参考：

DNException で指定できるプロパティの内容は、.NET Framework の Exception クラスを参照してください。

エイリアス

ここまで、何度も、'System.Windows.Forms'と'System.Drawing'という接頭辞を使用しました。

この処理を簡単にするため、Magic xpa は、.NET エイリアスを定義することによって.NET オブジェクトに対するショートカットを作成することができます。

1. ［コンポーネント］リポジトリを開き（Shift+F7）を開き、［.NET エイリアス］ボタンをクリックします。
2. １行追加し、必要なショートカットを定義します。

例えば、Forms という名前で System.Windows.Forms と定義します。

以上で定義が終了します。

今後、Forms という文字列を使用した場合、System.Windows.Forms を記述した場合と同じように動作します。

どのようなエイリアスも登録することができます。例えば、System.Windows.Forms.MonthCalendar に対してエイリアスを定義することで、.NET 変数のオブジェクトタイプとして使用することができます。

.NET 項目の初期化

今まで、コントロールに対応した.NET 項目を使用していました。この項目は、フォームの作成時に自動的に初期化されます。しかし、コントロールに関連しない.NET 項目を使用する場合もあります。このような項目は、コンストラクタによって明示的に初期設定する必要があります。

以下の例では、Windows タスクバーのシステムトレイにアイコンを追加します。

新しいプログラムを作成します。

1. ［データビュー］エディタで変数項目を作成し、型を「.NET」に設定します。
2. 項目の［特性］シートを開き、［オブジェクトタイプ］特性を「Forms.NotifyIcon」に設定します。「Forms」は、前述で定義されたエイリアスです。フルネームでオブジェクトタイプを System.Windows.Forms.NotifyIcon として定義することもできます。

この項目は、フォーム上のコントロールとして配置されていないため、手動で初期設定する必要があります。既存の.NET オブジェクトで更新するか、または.NET オブジェクトを作成する.NET メソッドを実行することで実現できます。

3. ［ロジック］エディタに切り替え、［レコード前］ロジックユニットに１行追加し、［項目更新］処理コマンドを追加します。変数項目を DotNet.Forms.NotifyIcon()で更新します。

これで変数項目は、初期設定された.NET オブジェクトとして扱うことができます。

実行エンジンが終了すると、このオブジェクトは破棄されます。Magic xpa Studio から数回このサンプルを実行すると、実行エンジンは終了されないため、手動で.NET オブジェクトを破棄する必要があります。

4. ［タスク後］のロジックユニットを作成して、［アクション］処理コマンドを追加してください。式として以下を設定してください：A.Dispose（）。そして実行条件として　Not IsNull(A) を設定してください。

プログラムが閉じると、NotifyIcon の .NET オブジェクトは即時に廃棄されます。

次に、アイコンとテキストによって.NET オブジェクトを更新します。

1. ［レコード前］ロジックユニットに１行追加し、［アクション］処理コマンドで以下の式を設定します（アイコンイメージを設定します）。
   DNSet(A.Icon,DotNet.System.Drawing.Icon(ServerFileToClient(Translate('%WorkingDir%Images¥Mgxpa.ico'))))
   ServerFileToClient(Translate('%WorkingDir%Images¥Mgxpa.ico')) は、サーバの%WorkingDir%Images フォルダからクライアントへ Mgxpa.ico ファイルをコピーします。これにより、このイメージを.NET オブジェクトで使用することができます。
2. ［レコード前］ロジックユニットに１行追加し、［アクション］処理コマンドで以下の式を設定します（アイコンをクリックしたときに表示させるテキストを指定しています）。
   DNSet(A.Text,'Click me')
3. ［レコード前］ロジックユニットに１行追加し、［アクション］処理コマンドで以下の式を設定します（アイコンを表示可能に設定します）。
   DNSet(A.Visible,'TRUE'LOG)

このプログラムを実行すると、新しいアイコンがシステムトレイに追加されます。

サンプルプロジェクトでは、オンライン・プログラム(#11)とリッチクライアント・プログラム(#41)は、上記で作成されたものです。

次に、アイコン上でマウスクリックした場合のロジックをプログラムに追加します。これは、以前のサンプルと同じように定義します。

1. プログラムを開き、［ロジック］エディタを開きます。
2. ［イベント］ロジックユニットを作成し、［イベントタイプ］を「.NET」に設定します。
3. ［項目］欄に.NET 項目を選択します。
4. ［イベント］欄で「DoubleClick」を選択します。
5. ［エラー］処理コマンドを追加し、メッセージテキストとして'HellowWorld'を設定します。

以上です。このプログラムを実行し、アイコンをダブルクリックすると、メッセージが表示されます。

サンプルプロジェクトでは、オンライン・プログラム(#12)とリッチクライアント・プログラム(#42)は、上記で作成されたものです。

サンプルの拡張

次に、受信した警告メッセージをエミュレートするように拡張します。これは、.NET の ShowBalloonTip メソッドを使用することで実現できます。

1. プログラムを開き、［ロジック］エディタを開きます。
2. 新しい［イベント］ロジックユニットを作成し、［イベントタイプ］を「タイマ」に設定し、［時間］を５秒として定義します。アプリケーションで変更をチェックするために、このタイマーイベントによる処理を定義します。
3. ［イベント］ロジックユニット内で、［アクション］処理コマンドを追加し、以下の式を設定します。
   DNSet(A.BalloonTipText,'Message arrived')

4. 別の［アクション］処理コマンドを追加して、以下の式を設定します。
   A.ShowBalloonTip(30)

プログラムを実行すると、アイコン上でバルーンが 5 秒ごとに表示されます。

ShowBalloonTip メソッドを選択すると、このメソッドには 2 つの構文を指定できることが分かります。このような場合、ツールチップの矢印表示を使用することで、どちらを使用するかを参照することができます。

ツールチップをもう一度開くと、マッチしている構文を検索しようとします。メソッドのパラメータと整合している重複した構文がある場合、最初のメソッドの構文が表示されます。

次に、異なるオーバーロードのメソッドを使用します。

1. ［イベント］ロジックユニットの中で、以下の式が定義された［アクション］処理コマンドを削除します。
   DNSet(A.BalloonTipText,'Message arrived')
2. 2 番目の［アクション］処理コマンドの式を変更します。
   旧：A.ShowBalloonTip(30)
   新：A.ShowBalloonTip(30,'Information','Message arrived',DotNet.Forms.ToolTipIcon.Info)

以上です。

プログラムを実行すると、アイコン上でバルーンが 5 秒ごとに表示され、またアイコンとヘッダテキストのような機能が追加されています。

サンプルプロジェクトでは、オンライン・プログラム（#13）とリッチクライアント・プログラム（#33）は、上記で作成されたものです。

サンプルをさらに拡張

サンプルをさらに拡張し、コンテキストメニューを通知アイコンに追加します。

これは、通知アイコンに ContextMenuStrip クラスのオブジェクトを関連付けることで実現できます。

1. プログラムを開き、［データビュー］エディタを開きます。
2. プログラムの変数項目を作成し、型を「.NET」に設定します。
3. 項目の［特性］シートを開き、［オブジェクトタイプ］特性を「Forms.ContextMenuStrip」に設定します。

この項目はフォーム上のコントロールとして配置されていないため、NotifyIcon 項目と同じように、明示的に初期設定する必要があります。

4. ［ロジック］エディタを開き、［レコード前］ロジックユニットに［項目更新］処理コマンドを追加します。ここで、DotNet.Forms.ContextMenuStrip()を使用して変数項目を更新します。

次に、コンテキストメニューに項目を追加します。このために、ContextMenuStrip 項目の Items.Add()メソッドを使用します。

5. ［レコード前］ロジックユニットに、2 つの［アクション］処理コマンドを追加し、以下の式を設定します。
   - B.Items.Add('Open Message')
   - B.Items.Add('Exit')

次に、NotifyIcon 項目に ContextMenuStrip オブジェクトを割り当てます。NotifyIcon 項目の［ContextMenuStrip］プロパティの値を、作成された ContextMenuStrip 項目で更新することによって実現します。

6. ［レコード前］ロジックユニットに、［アクション］処理コマンドを追加し、以下の式を設定します。
   DNSet(A.ContextMenuStrip,B)

プログラムを実行し、アイコン上で右クリックを行うと、コンテキストメニューが表示されます。

次に、コンテキストメニュー上で［クリック］イベントを処理するためのロジックユニットを追加します。

1. プログラムを開き、［ロジック］エディタを開きます。
2. ［イベント］ロジックユニットを作成し、［イベントタイプ］を「.NET」に設定します。
3. ［項目］欄で ContextMenuStrip 項目を選択します。
4. ［イベント］欄で「ItemClicked」を選択します。
5. ［エラー］処理コマンドを追加して、メッセージを定義し、実行条件式として F.ClickedItem.Text='Open Message'を設定します。
6. ［イベント実行］処理コマンドを追加して、［終了］イベントを割り当てます。実行条件を F.ClickedItem.Text='Exit'とします。

以上です。プログラムを実行すると、フォームは表示されず、アイコンのコンテキストメニューが表示され、［Exit］をクリックするとプログラムが終了します。

サンプルプロジェクトでは、オンライン・プログラム（#14）とリッチクライアント・プログラム（#44）は、上記で作成されたものです。

1 つの共通イベントを使用して実行条件を設定する代わりに、各メニュー項目に対して、ロジックユニットを割り当て、処理を定義することもできます。この場合は、System.Windows.Forms.ToolStripMenuItem タイプの.NET 項目を定義します。

1. 新規プログラムを作成し、［データビュー］エディタを開きます。
2. 2 つの変数項目（ToolStripMenuItem1、ToolStripMenuItem2）を作成し、型を「.NET」に設定します。
3. 2 つの変数項目の［オブジェクトタイプ］特性を「Forms.ToolStripMenuItem」に定義します。
4. ［ロジック］エディタを開き、［レコード前］ロジックユニットで［項目更新］処理コマンドを追加し、変数項目：ToolStripMenuItem1 を DotNet.Forms.ToolStripItem()で更新します。
5. ［アクション］処理コマンドを追加し、DNSet(C.Text,'Open Message')を設定します。
6. ［アクション］処理コマンドを追加し、B.Items.Add(C) を設定します。
7. 変数項目：ToolStripMenuItem2 に対して同様のロジックを定義します。
8. ［イベント］ロジックユニットを作成し、［イベントタイプ］を「.NET」に設定します。

9. ［項目］欄で ToolStripMenuItem1 項目を選択します。
10. ［イベント］欄で「Click」を選択します。
11. イベントロジック内に処理を定義します。
12. 変数項目：ToolStripMenuItem2 に対する［イベント］ロジックユニットを作成し、同様のロジックを定義します。

プログラムを実行すると、前述と同じように動作します。サンプルプロジェクトでは、オンライン・プログラム(#25)とリッチクライアント・プログラム(#53)は、上記で作成されたものです。

キャストと参照

ほとんどの場合、(.NET オブジェクトのプロパティを Magic の値で更新したり、.NET メソッドに Magic 値を送ったりして).NET を含んだ Magic xpa のデータを使用する場合、Magic のデータ型と.NET オブジェクトタイプの間で自動的なデータ変換を行うことができます。

しかし、変換が、ユニークでない(例えば、メソッドがオーバーロードされていて、Int 型と Long 型の両方を受け取ることができる)場合や、特別に値を.NET タイプに変換する必要がある場合などがあります。

このような場合の Magic のデータ型を希望する.NET タイプに変換するため、DNCast()関数を使用することができます。

例えば、System.Text.StringBuilder クラスで説明します。このクラスは、Append と呼ばれるメソッドを持っています。このメソッドは、文字列をオブジェクトのインスタンスの内容に付加します。しかし、Append メソッドはオーバーロードされており、String や Char または、Char[]を含めた、異なるクラスを受け取ることができます。メソッドをそのまま使用して、A.Append('abc')の式を評価する場合、Magic xpa は、文字列'abc'がどのクラスに対応して変換すべきかが分かりません。このような場合、構文チェックユーティリティは、この式に対してエラーを返します。

この不一致を解消するためには、DNCast('abc',DotNet.System.String)を実行することで、文字列'abc'を System.String クラスにキャストできます。これにより式は、A.Append(DNCast('abc',DotNet.System.String))となります。

サンプルプロジェクトでは、オンライン・プログラム(#16)とリッチクライアント・プログラム(#46)は、上記で作成されたものです。Ctrl+1 を押下するとオブジェクトに定義された文字列が表示されます。

.NET 項目を参照渡しで.NET メソッドに渡す必要がある場合、DNRef()関数を使用することで可能になります。値を参照渡しで渡す方法については、後述します。

.NET コードの実行

Magic xpa は、プログラム内で直接.NET コードを実行することができます。

これは、[外部コール]処理コマンドに追加された新しいオプションを利用することで可能になります。

ここでは、2 つの数値を合計する簡単なプログラムを紹介します。

1. 新しいプログラムを作成し、［タスクタイプ］を「リッチクライアント」に設定し、プログラムに公開名を設定します。

2. このプログラムに 3 つの数値型変数（2 つの数値と 1 つの戻り値）を作成します。
3. ［ユーザイベント］テーブルに「Caluculate」という名前のユーザイベントを作成し、［強制終了］カラムを「編集」に設定します。
4. フォームに［プッシュボタン］コントロールを配置し、このユーザイベントを割り当てます。
5. 作成されたユーザイベントに対応する、ロジックユニットを作成します。
6. ロジックユニット内に、［外部コール］処理コマンドを追加します。［.NET コード］特性でズームして［.NET コード］ダイアログを開きます。ここで以下の処理を行います。

- a.［メソッド名］欄に「Sum」を設定します。これは、作成される.NET のメソッドの名前です。
- b.［パラメータ］欄でズームします。
  - 一行追加して最初の数値型変数を選択します。
  - ［.NET タイプ］カラムには、［項目］カラムに指定したデータ項目に対応する.NET タイプが推奨値として設定されます。必要があれば、この値を変更することができます。
  - ［.NET 項目］カラムに移動して、項目の名前を（num1 など）入力します。この名前は.NET コードで使用されます。
  - 2 番目の項目に対しても同じ処理を行います。
  - ［パラメータ］テーブルを閉じます。
- c.［戻り値］欄でズームして、3 番目の項目を設定します。［戻り値のタイプ］欄には、設定した項目のデータ型に対応する.NET タイプが推奨値として設定されます。必要があれば、この値を変更することができます。
- d.［.NET コード］欄は、設定内容に応じて変更されます。
- e.任意にコードを記述することができます。内部メソッドの本体部分に以下の行を入力します。
  return num1 + num2
- f.［OK］をクリックして［.NET コード］ダイアログを閉じます。ダイアログが閉じる時点でコンパイル処理が行われます。.NET コードに問題がある場合は、エラーメッセージが表示されます。

プログラムを実行し、項目に値を入力し、ボタンをクリックします。入力項目の加算値によって結果項目が更新されます。

サンプルプロジェクトでは、オンライン・プログラム（#18）とリッチクライアント・プログラム（#38）は、上記で作成されたものです。

複雑な.NET コードの実行

前述では、特定の処理を実行し、値を戻すための.NET コードの簡単なサンプルを紹介しました。この機能を使用することで、クラスを作成しコードの中でそれらを使用するというような複雑な.NET コードを実行することもできます。

.NET コードのエディタでは、コード全体を選択したり、コピーしたりすることができます。従って、外部ツール（Visual Studio など）を使用してコードの編集や確認を行うことができます。

次の例では、［コール.NET］処理コマンドを使用しています。

1. 新しい Form クラスを定義します。
2. 新しいクラスに基づいた新しいフォームを開きます。

3. .NET フォームからイベントを発行し、Magic xpa で処理するようにします。
4. .NET フィールドで変更があった場合、Magic xpa の項目の値を更新します。
5. Magic xpa から.NET フォームを処理します。
6. .NET フォームを閉じます。

サンプルプロジェクトでは、オンライン・プログラム(#19)とリッチクライアント・プログラム(#49)は、上記で作成されたものです。

1. Open form ボタンをクリックします。半透明で、丸い.NET フォームが表示されます。このフォームはマウスでドラッグして任意に移動させることができます。
2. 丸いフォーム内のボタンをクリックします。このクリックは、リッチクライアント・プログラムで処理され、結果として、フォーム上のテキストが交互に見え隠れします。
3. 丸い.NET フォーム上のテキストボックスにテキストを入力します。このテキストがリッチクライアントフォーム上の［エディット］コントロールに即反映されます。
4. リッチクライアントフォームで、［エディット］コントロールにテキストを入力します。このテキストが丸い.NET フォーム上のテキストボックスに即反映されます。
5. リッチクライアントフォームの［スライダ］コントロールを動かします。.NET フォームの不透明表示がそれに応じて変化します。
6. リッチクライアントフォームの Close form ボタンをクリックすると、丸い.NET フォームが閉じます。

プログラムの背景にある基本的な考え方は、.NET コードを使用して.NET の丸いフォーム上の全ての GUI を処理し、ロジックを処理するために Magic xpa 側のイベントを使用することです。

もちろん、[コール.NET]処理コマンドを使用する代わりに外部アセンブリで行うこともできます。

プログラムがどのように動作するかを説明します。

- 丸い.NETフォーム上の各項目に対して、Magicプログラム側で.NET項目を定義しています。この項目への参照情報のみを示す値がパラメータとして.NETコードに渡されます。
- （このパラメータのために作成された）対応する.NET項目は、.NETフォーム上で作成されたオブジェクトによって更新されるため、Magic項目は、.NETフォーム上に存在している.NETオブジェクトを参照することができます。
- このオブジェクトに対して様々な処理を行うことができます。例えば、Clickイベントを処理することで.NETボタンのクリックの処理を受け取ることができます。また、TextChangeイベントを処理することで、NET項目が変更されたらMagic項目を変更したり、Magic項目が変更されたら.NET項目を更新したりするようなことができます。
- .NETコードは、Magic xpaの.NET項目に.NETフォームの参照情報を返します。これによって、Magic xpaからフォームをコントロールし、Opacityなどのプロパティを変更したり、Closeなどのメソッドを呼び出したりすることができます。コントロールに対して行うように、パラメータとしてのフォーム参照を戻すこともできます。

.NET のチョイスコントロールを使用する

Magic xpa に組み込まれているコントロールを使用するように、コントロールのデータ特性を使用してデータをバインドすることができます。（「データを.NET コントロールにリンクする」を参照)

しかし、Magic xpa のチョイスコントロールと同じように、.NET のチョイスコントロールも、（データソース・オブジェクトとして知られている）項目リストを取得することができます。

.NET オブジェクト参照項目を通して.NET コマンドを使用したり、内部のチョイスコントロールに対応できるように Magic xpa の特性を設定することで実現することができます。

ここでは、.NET コントロールのデータソースを定義するために Magic xpa の特性を使用することについて説明しています。

.NET コントロールには多くの特性があるため、どの特性がデータソースを取得し、どの特性が値と表示のメンバ名を保持するかを定義する必要があります。

この定義は、モデルの DataSource プロパティ名、DisplayMember プロパティ名、Value Changed プロパティ名、ValueMember プロパティ名の各特性で行われます。

次に.NET のデータソース宣言を.NET コントロールモデルに追加します。

この定義は、設定したいデータのタイプに対応するコントロールタイプ毎に、一度だけ実行されます。

1. GUI 表示またはリッチクライアント表示の各クラスのモデルを作成し、型を「.NET」に設定します。
2. モデルの特性シートを開き、［オブジェクトタイプ］特性を定義（例：System.Windows.Forms.ComboBox）します。
   Ctrl+Space を押下するオートコンプリート機能が実行されます。この後で、"."を入力すると、利用できるオプションが（Magic xpa の関数一覧と同じように)リスト表示されます。
3. ［Value プロパティ名］特性にデータ項目にバウンドさせるプロパティ名（例：SelectedValue）を設定します。
4. ［Value Changed イベント名］にデータ項目に値を返すためのトリガとなるイベント（例：SelectedValueChanged）を設定します。
5. ［Datasource プロパティ名］特性に項目リストを受け取るプロパティ名を（例：Datasource）を設定します。
6. ［DisplayMember プロパティ名］特性には、表示されルカラムの名前を定義するプロパティ名（例：DisplayMember）を設定します。
7. ［ValueMember プロパティ名］特性には、データが格納されているカラムの名前を定義するプロパティ（例：ValueMember）を設定します。

これだけです。これでモデルをプログラムで使用することができます。

次に、プログラムで.NET コントロールモデルを使用します。

8. 新しいプログラムを作成します。
9. このプログラムに変数項目を作成します。この変数は、コントロールのデータの取得/設定を行うために使用されます。
10. フォームを開き、［.NET］コントロールをフォームに配置します。
11. ［.NET］コントロールに上記で定義したモデルを割り当てます。

注意：オブジェクトタイプが"System.Windows.Forms.Control"に設定されているモデルを継承している場合のみ、［.NET］コントロールとしてフォームに配置できます。

12. ［データ］特性に変数項目を割り当てます。
13. コントロールの選択用の特性（例：選択項目一覧と選択表示一覧)に通常のチョイスコントロールと同じように設定します。

これだけです。

プログラムを実行すると、［.NET］コントロール上に項目リストが表示され、データ項目の変更内容がコントロール上に反映されます。また、コントロールで値を変更するとデータ項目に反映されます。

サンプルプロジェクトでは、オンライン・プログラム(#21)とリッチクライアント・プログラム(#51)は、上記で作成されたものです。

データビューを.NET コントロールにバインドする（オンラインタスクのみ）

タスクのデータビューを［.NET］コントロール（例えば Grid コントロール）にバインドすることで、通常の［テーブル］コントロールと同じように使用することができます。

複数の［.NET］コントロールをフォーム上に配置することができるため、どの［.NET］コントロールがタスクのデータビューを表示しているかを把握しておく必要があります。

この定義は、［データビューコントロール］特性（フォーム上のコントロールやモデルレベルで)行うことができます。

注意：［Datasource プロパティ名］特性に値が設定されており、［DisplayMember プロパティ名］や［ValueMember プロパティ名］の各特性に値が設定されていない場合、この特性が有効になります。

この特性の値を「Yes」に設定すると、［データビューコントロールのフィールド］特性にズームすることで.NET コントロールに表示させるデータ項目を定義することができます。

実行時、.NET System.Data.DataTable オブジェクトがタスクのデータビューから自動的に作成されます。そして、このオブジェクトはモデルレベルの特性として［DataSource property 名］特性内に定義された［.NET］コントロールの特性に割り付けられます。

Magic xpa のデータ項目の変更内容が、.NET コントロール上に反映され、その逆も実行されます。

注：いくつかの Grid コントロールは、Grid の終わり空の列を追加します。通常、これはコントロールの "AllowUserToAddRows"という Grid コントロールのプロパティによって管理されます。この動作は、ユーザがこの列を更新する際、Grid と Magic xpa タスクの間で矛盾を発生させます。Magic xpa は新しいレコードが作成されていることを知らないため、タスクは登録モードに変更されません。この動作を回避するには、この特性を"False"に設定することを推奨します。レコードの新規作成は、Magic xpa の［行作成］イベント（F4)を使用して行われます。

次に、データソースの内容を表示するために、.NET データビューコントロールを使用するプログラムを作成します。

1. GUI 表示クラスのモデルを作成して、その型を「.NET」に設定します。
2. .NET コントロール・モデルの特性シートを開きます。
3. ［オブジェクトタイプ］特性でデータソース・オブジェクトをサポートする.NET コントロール（例：System.Windows.Forms.DataGrid）を設定します。
4. ［DataSource プロパティ名］特性にデータを取得するプロパティ（例：DataSource）を設定します。
5. ［データビューコントロール］特性を「Yes」に設定します（これによって、フォーム上のコントロール特性の代わりとして使用できます）。
6. 新しいプログラムを作成します。
7. プログラムの［タスク特性］を開き、［ビュー事前読込］特性「Yes」に設定します。
8. プログラムのデータビューを定義します（データソースとカラムを追加します)。
9. ォームを開き、［.NET］コントロールをフォームに配置します。
10. コントロールのモデルとして新規作成したモデルを割り当てるか、.NET 項目をコントロールの［.NET オブジェクト］特性に割り当てます。
11. コントロールの特性シートを開きます。

12. ［データビューコントロールのフィールド］特性でズームして、.NET コントロールに割り当てる DataTable オブジェクトに追加するデータ項目の順番を定義します。「0」が設定された場合、データ項目は.NET コントロールで利用できないことを意味します。

これだけです。

プログラムを実行すると、データビューが DataGrid の.NET コントロールにバインドされて表示されます。

行間の操作や.NET コントロールのデータ更新処理が Magic xpa のデータ項目に自動的に反映されます。逆もまた同じです。

サンプルプロジェクトでは、オンライン・プログラム(#23)は、上記で作成されたものです。

配備

アセンブリがクライアントにインストールされていない場合、クライアント PC からアプリケーションを実行すると、サーバからクライアントマシンにアセンブリがコピーされます。

サーバ側の位置は、[コンポーネント]リポジトリの[アセンブリ特性]に定義された値を参照します。

この位置を変更することができます。

1. ［コンポーネント］リポジトリを開き（Shift+F7）、アセンブリ上でパークします。
2. ［名前］カラムでパークし、ズームして、［.NET アセンブリ特性］ダイアログボックスを開きます。
3. ここで現在のアセンブリパスを変更することができます。

開発環境と実行環境で異なる位置になる場合は、アセンブリパス用に論理名を使用するよう推奨します。