.NET の基本
Magic xpa 3.x

Magic Software Enterprises Ltd., 5 Haplada Street, Or-Yehuda 60218, Israel
Tel: +972 (0)3 538 9292 | Fax: +972 (0)3 538 9333, +972 (0)3 538 9393 | info@magicsoftware.com | www.magicsoftware.com

Chapter 1

.NET とは?

.NET について良く耳にすることと思います。あなたの PC にもたぶん.NET アプリケーションが動いていると思います。それは、一体何なのでしょうか?

Microsoft.NET は「ドットネット」と発音され、.NET または DotNET などと記述されることがあります。これは、Windows OS 上で動作するコンポーネントです。. NET は、開発者に対してより速く、より簡単に Windows ソフトウェアを作成することを可能にするツールとライブラリを提供しています。.NET アプリケーションを実行するためには、.NET Framework を PC 上にインストールする必要があります。

しかし、Microsoft は何と言っていますか?

Microsoft は、.NET を以下のように説明しています。「Microsoft.NET は、ソフトウェアを通して情報、人々、システム、および機器を接続する Microsoft の Web サービス戦略です。Microsoft のプラットフォームの垣根を越えて統合され、.NET テクノロジーは、Web サービスを使用したセキュリティの向上したソリューションの構築や配備、運用、接続を迅速に行う機能を提供します。.NET に接続されたソリューションによって、ビジネスは、より速くそれらのシステムを統合し、より迅速に、さまざまな機器で何時でも何処でも情報を取得できるようになります。」.NET について説明する場合、必ずしも Web サービスが、最初に挙げられるわけではありません。

しかし Microsoft は、さらに説明しています。「.NET Framework は、見栄えが良く、シームレスで、機密性の高い通信機能を持ち、ビジネスプロセスの範囲を形作ることのできるアプリケーションを作成するための Microsoft のプラットフォームです。包括的で、一貫したプログラミングモデルと API の共通のセットを提供することによって、.NET Framework は、ソフトウェアやサービス、および機器に渡って、使用したいプログラミング言語で、望んでいる方法で動作するアプリケーションを開発することを支援します。」アプリケーションやユーザの経験などを開発すること、これは、私達が望んでいるものです。

Magic プログラマとして、私達は、.NET Framework によって提供されるすばらしいオプションを利用することができればよろこばしいことです。

しかし、私達が開発作業を行う前に、.NET の基本的な概念を理解する必要があります。これらの基本的概念についての理解することで、Magic xpa と.NET を利用する上での手助けとなるでしょう。

Chapter 2

.NET のアーキテクチャ

古い歌ですが、 "Let's start at the very beginning, a very good place to start"（ドレミの歌から）の言葉において、私達は、基本となる.NET アーキテクチャについて説明します。

.NET アーキテクチャは、これから説明するように様々なレベルから構成されています。

共通語ランタイム

共通語ランタイム(CLR)とは、.NET アーキテクチャの最も重要なコンポーネントです。これは.NET の実行環境で、.NET コードの実行に対して責任を負うものです。CLR の主な役割は、.NET タイプを検索し、読み込み、管理することです。CLR は、.NET のコンパイルコードをプラットフォーム用のネイティブなコードに変換します。

CLR は、.NET プログラマが OS 上の低レベルな処理コマンドを無視し、タスクの処理内容に専念することを可能にします。CLR はまた、メモリ管理やスレッド管理、例外処理、ガベージコレクションやセキュリティなどのサービスを提供しており、プログラマはコード内で利用することができます。これらについてよく知っていますか？

. NET プログラミング言語

.NET アプリケーションを開発するには、プログラマは、利用する言語を選択する必要があります。C#や VB .NET、J#、など、多くの.NET 対応言語があります。

共通語指定

共通型システム(CTS)は、CLR によってサポートされたすべての.NET タイプとプログラミング規約を記述する上での仕様です。CTS はマイクロソフトによって作成され、Microsoft .NET Framework は、CTS を基準に実行します。

.NET 言語は、CTS 仕様によって定義されるすべての機能をサポートしていないかもしれません。そのような場合は、共通語指定(CLS)で対応します。CLS は、CTS 仕様の共通タイプとすべての.NET プログラミング言語が合意できるプログラミング構成のサブセットを定義する仕様です。

Magic xpa の.NET 統合機能は、.NET CLS に忠実に動作します。これは、Magic xpa が、CLS 準拠の機能を公開する.NET タイプを作成できることを意味しています。

管理されたコード

.NET 環境では、VB .NET や C#などの多くのプログラミング言語を使用して、.NET コードを開発することが可能です。このコードをコンパイルすると、コンパイラはバイナリコードを作成します。このコンパイルコードは、プログラムコードのための実行環境を定義する共通言語基盤(CLI)と呼ばれる標準に基づいています。コンパイルコード

は、管理されたコードまたはマイクロソフト中間言語(MLI)としても知られています。管理されたコードにコンパイルできるプログラミング言語であれば、.NET 言語と呼ぶことができます。

コンパイルされた.NET プログラムを実行する場合、CLR はジャストインタイムコンパイラを内蔵しており、それは中間言語を取得して、プラットフォームに対してネイティブなコードに変換し、実行します。ジャストインタイムのテクニックは、実行環境で性能を改善するためによく知られている方法です。

基本クラスライブラリ

基本クラスライブラリ(BCL)は、.NET プラットフォームの一部であり、すべての.NET 言語で利用可能です。BCL はこれらの言語によって使用されるライブラリやサービスを含んでいます。例えば、文字列操作や入出力のアクセス、コレクション、配列やリストを含める処理、データベースアクセスや XML ドキュメントの操作などさまざまな内容があります。

アセンブリ

既に述べたように、CLR は部分的にコンパイルされたコードを取得し、実行時にコードをコンパイルし、それをコンパイルするために、内蔵するジャストインタイムコンパイラを使用します。このような部分的にコンパイルされたコードは、アセンブリと呼ばれ、CLI コード（これは、プラットフォームが指定されてコンパイルされたものではないバイナリコードです）を含んでいます。

アセンブリは、1 つ以上のファイルから構成されています。アセンブリが 1 つの*.dll または*.exe モジュールから構成されている場合、シングルファイルアセンブリを持っていることになります。これらのアセンブリには、1 つのパッケージの中に必要な CIL（共通中間言語）、メタデータ、および関連するマニフェストが全て含まれています。マルチファイルアセンブリは、複数の.NET バイナリやモジュールによって構成されています。マルチファイルアセンブリを作成する場合、モジュールの 1 つとして様々なタイプのための CIL 手順とメタデータを持つアセンブリマニフェストが含まれます。

Magic xpa は、プログラム内で.NET コードを記述する機能があります。この機能は、［外部コール］処理コマンドを定義する際に利用できます。図 1 は、Magic xpa 内で.NET コードを定義する例を示しています。Magic xpa のキャビネットファイル(ECF)を作成する際に、.NET コンパイラ（これは.NET Framework によって提供されたサービスの 1 つです）を呼び出すことによって、コードをコンパイルします。このため、キャビネットファイルには.NET アセンブリが含まれています。

Magic xpa の実行中は、.NET コードを実行するために、クライアント PC 上の Magic エンジンが CLR を呼び出します。

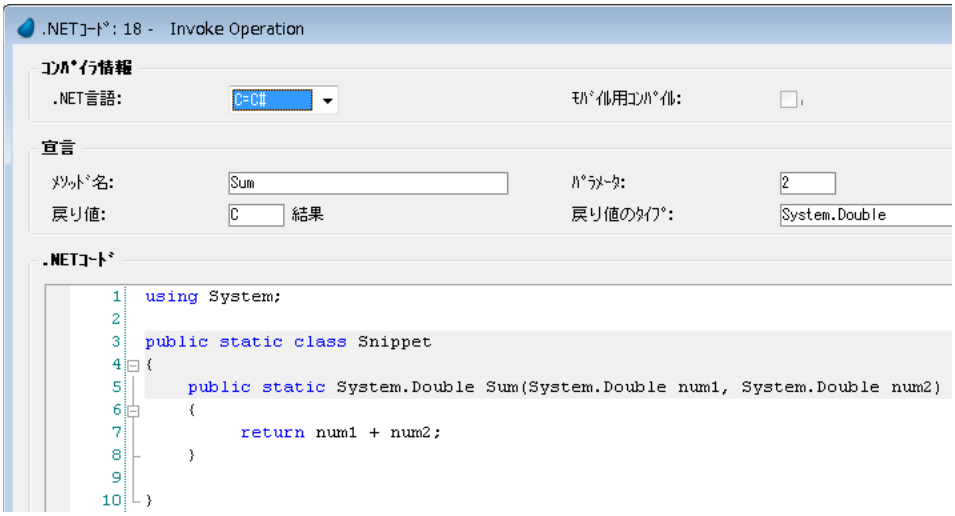

図 1 Magic xpa で.NET コードを作成する

ガベージコレクション

ガーベージコレクタは、アプリケーションで使用されていないオブジェクトによって使用されたメモリの開放処理を行う.NET Framework の機能です。

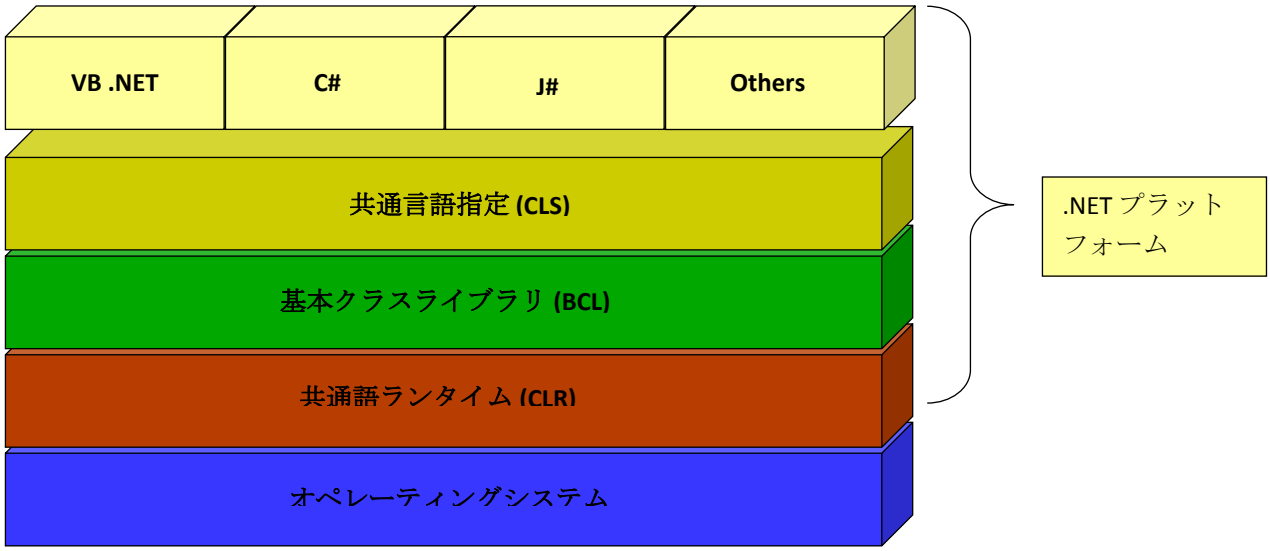

図 2 .NET プラットフォーム構成

Chapter 3

.NET 言語コンポーネント

.NET Framework についての理解をしていただいた次は、.NET 言語で使用された専門用語について理解していただく必要があります。これは、Magic xpa で.NET コードを実装する上で手助けとなるはずです。

ネームスペース

ネームスペースは、タイプ名をグループ化する方法です。これによって重複した名前を持つ可能性を減らします。名前を持つ異なるタイプが区別されるように、コンテナの中にタイプを定義する方法です。ネームスペースは、他のネームスペースとタイプを含めることができます。

例えば、System ネームスペースは、Windows ネームスペースを含んでいます（さらに、Forms ネームスペースを含んでいます）。.NET では、これを System.Windows.Forms として記述して定義します。

クラス

クラスはプロパティやメソッド、およびイベントを含む定義またはテンプレートです。クラスは、オブジェクト指向プログラミングの定義に関する考え方の 1 つです。クラスは、クラスの特徴のすべてまたは一部を継承するサブクラスを持つことができます。サブクラスは、その親クラスの一部ではないそれ自身のメソッドと変数を定義することもできます。クラスとそのサブクラスの構造は、クラス階層と呼ばれます。.NET クラスの例として、Button クラスがあります。.NET の省略語を使用してこのクラスにアクセスする場合、System.Windows.Forms.Button と指定します。

オブジェクトとインスタンス

クラスを使用する場合は、クラスのインスタンスを作成します。このインスタンスは、オブジェクトと呼ばれます。このオブジェクトは、クラスによって定義されたすべてのプロパティやメソッド、およびイベントを継承します。同じクラスの複数のオブジェクトを定義することができます。クラスからインスタンスを新規に生成するたびに、そのクラスの新しいインスタンスを取得します。

Magic xpa で、.NET 項目を定義することができます。.NET 型のデータ項目を定義し、［オブジェクトタイプ］特性を設定することで、.NET クラスを選択したことになります。図 3 では、System.Windows.Forms ネームスペースから MonthCalendar クラスを選択する例を示しています。.NET 型項目がフォーム上のコントロールとして使用される場合、Magic xpa は実行時に自動的にこのクラスのインスタンスを作成します。.NET 型項目がフォーム上で使用されていない場合は、自動的にインスタンス化されません。

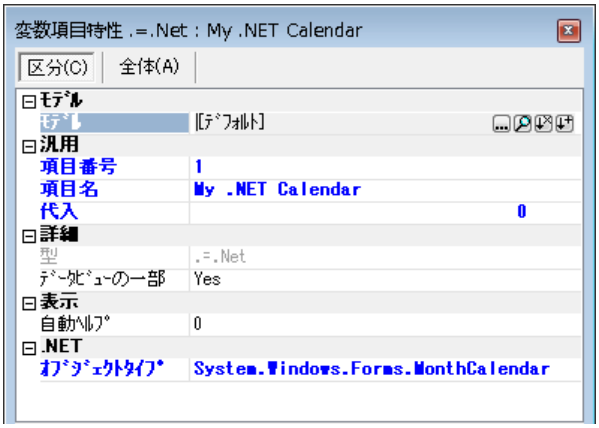

図 3 Magic xpa で .NET 項目を定義する

メソッド

メソッドは、クラスの一部として定義される手続です。メソッドは、クラスのオブジェクトが実行することができる動作を定義しています。これらは、一般的に、オブジェクト上の処理タスクを実行するために使用されます。クラスは複数のメソッドを持つことができます。これらのメソッドは、そのクラスのために作成されたどのようなオブジェクトの中にも含められています。これらはパラメータを受け取ったり、値を戻したりすることができます。

パラメータを受け取る Magic プログラムは、メソッドの一種と考えることができます。また、Magic 関数も、メソッドの一種です。

Void

Void 型は、値を戻さない関数またはプロシージャの結果です。これは.NET で新しく追加されたわけではありません。Windows の関数を使用する場合、戻り値が Void になっているものを見かけます。関数はパラメータを受け取り、渡されたパラメータの値を更新することはできますが、戻り値は返しません。

メソッドのオーバロード

メソッドのオーバロードは、同じ名前でパラメータや戻り値が異なるメソッドを定義する機能です。

コンストラクタ

コンストラクタは、クラスのインスタンスを作成するメソッドです。これは、オブジェクトを初期設定する役割も果たします。これは、Void ではなく戻り値のタイプもないという点で、他のメソッドと異なります。コンストラクタは、それを定義するクラスと同じ名前を持っています。

コンストラクタは、他のメソッドのようにパラメータを受け取ることができます。パラメータを受け取らないコンストラクタは、デフォルトコンストラクタと呼ばれます。

.NET 項目がフォーム上に定義されていない場合、前述したように、自動的にインスタンス化されません。プログラムによってオブジェクトをインスタンス化する処理が必要です。図 4 では、コンストラクタを使用して、項目をインスタンス化する式の例を示しています。

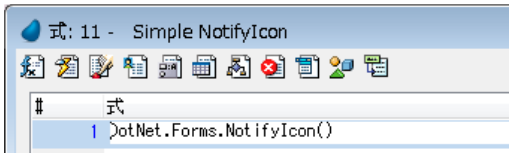

図 4 コンストラクタを示す例

シグネチャ

メソッドのシグネチャは、メソッドのために入出力を定義しています。メソッドの戻り値のタイプとパラメータのタイプを指定します。Int32 は、戻り値タイプの例です。

イベント

Magic xpa と同じように、何らかの事象がオブジェクトで発生した場合、イベントが発行されます。プログラムでそのイベントを処理するように定義することができます。

値渡し／参照渡し

Magic xpa と同じように、プログラムを作成する際に情報を渡す必要があります。この情報は、パラメータを使用して呼び出されたプログラムに渡すことができます。

.NET では、パラメータを値渡しや参照渡しで渡すことができます。どちらの方法でパラメータを受け取るかを定義する必要があります。参照渡しで渡されるパラメータを定義する場合は、メソッド内のパラメータに対してどのような変更が発生しても、呼び出すメソッド内の項目に反映するように定義します。

文字列などのオブジェクトのほとんどは参照タイプであり、自動的に参照渡しが行われます。Int32 などの他のオブジェクトは、値渡しとなります。

プロパティ

Magic xpa では、すべてのコントロールは、その特性を定義する特性リストを持っています。それらの値が式で定義されている場合、実行時に特性値を設定することができます。しかし、特性の現在の値は取得できません。

.NET の場合は、プロパティはコントロール上で定義されるだけでなく、クラスメンバとしても定義することができます。これらはクラスの名前付けメンバです。プロパティの値を取得したり、変更したりすることができます。図 5 は、MonthCalendar クラスの ShowWeekNumbers プロパティを更新するために、Magic xpa の DNSet 関数を使用する例を示しています。

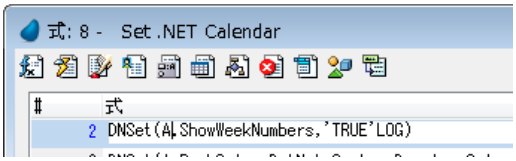

図 5 プロパティの更新で DNSet 関数を使用する例

フィールド

.NET フィールドは、クラス内に定義された項目です。クラスの新しいインスタンスが作成されると、これらのフィールドは初期設定されます。これらのフィールドの値を初期設定するのは、通常そのコンストラクタです。

アクセス修飾子

クラスまたはクラスのメンバを宣言する際に、それらのアクセス可能なレベルを定義することができます。それらが自分のアセンブリ内のみなのか、または他のアセンブリからでもアクセス可能かを制御します。

- public ……メンバがどこかかでもアクセスできることを意味しています。
- private …… メンバがそのクラス内からでのみアクセスできることを意味しています。
- protected …… メンバがそのクラス内からまたは、派生クラスからのみアクセスできることを意味しています。
- internal …… メンバが同じアセンブリからのみアクセスできることを意味しています。
- protected internal …… メンバが同じアセンブリから、または異なるアセンブリの派生クラスからののみアクセスできることを意味しています。

スタティック

スタティックなメンバは、クラスのインスタンスを作成せずにアクセスすることができるクラスメンバです。例えば、StringCompare()メソッドは、クラスのインスタンスにの代わりにクラス自身に呼び出されることができる String クラスのスタティクメソッドです。スタティックなフィールドメンバは、クラスのすべてのインスタンスのために同じ値を持っています。

スタティックなクラスは、スタティックなメンバのみを含むことができ、インスタンス化されず、継承もされず、コンストラクタを含むことができません。

Enum または列挙

列挙とは、実際の値が数値である名前付き定数リストです。例として、Windows 関数の MessageBox が挙げられます。ダイアログボックスが表示されると、ボタンの数と MB_YESNOCANCEL（これは 3 つのボタンを表示します）など表示されるテキストを制御することができます。この場合、実際の値は、定数の 3 になります。

Magic xpa では、DNSet 関数を使用する際に.列挙値を使用することができます。図 6 の例は、MonthCalendar クラスの BackColor の値を設定する際に、Blue の列挙値を設定しています。

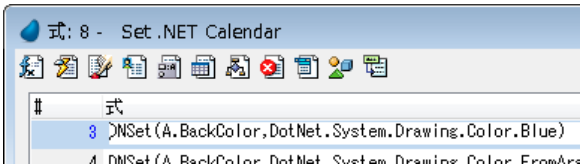
式: 8 - Set .NET Calendar
#
式
3
DNSet(A.BackColor,DotNet.System.Drawing.Color.Blue)

図 6 列挙値の使用例

Chapter 4

おわりに

この文書によって、.NET Framework についての基本的な内容や、Magic xpa とどのように.NET とアクセスするかについて理解していただけたら幸いです。

Magic xpa は、.NET が提供する優位性（特に CLS において定義されるもの）を利用することができる.NET と同様な環境です。

.NET と同様に Magic xpa の優位性によって、短期間に.NET アプリケーションを作成し、早期の投資回収効果（RROI：Rapid Return on Investment）を提供することが可能となるでしょう。

より深く理解するために、.NET サンプルガイドを読み、Magic xpa でどのように.NET を使用するかを確認してください。