

Micro Focus Net Expressを Microsoft .NET環境で使用する

はじめに

マイクロソフトの .NET フレームワークは、インターネットを軸としたサービス指向のアプリケーション実行環境を提供する革新的なプラットフォームです。 .NET のアプリケーション開発について語るときしばしば言及されるのが、共通言語ランタイム (Common Language Runtime) と IL (Microsoft Intermediate Language) で、これらに対応したアプリケーション開発ツールが注目されています。しかし、一方で .NET フレームワークは、既存のアプリケーションとの協調を可能にするサポートにも優れています。

ここでは、Micro Focus Net Express を使用して .NET フレームワークで稼動するアプリケーションを作成する方法について説明します。新たな言語を習得したり、新たに登場する言語製品のリリースを待つまでもなく、今すぐに既存の COBOL 資産を、Web サービスとして利用可能にすることができます。

目次

1. NET EXPRESS の .NET 対応の概要	1
-WEB サービス	
-.NET フレームワーク	
-MICRO FOCUS の立場	
-現状のままですること	
2. COBOL コンポーネントをマネージドコードから利用する	4
-COBOL COM コンポーネントの作成	
-COM コンポーネントをマネージドコードから利用可能にする	
-例題プログラム	
クラスの作成	
COM 情報の設定	
メソッドの追加	
ユーザーロジックの追加	
メタデータの作成	
マネージドコードのコンパイル	
-バインド方法の選択肢	
-COBOL クラスの呼び出し	
3. COBOL からマネージドコードを利用する	11
4. WEB サービスを COBOL で作成する	12
-WSDL: WEB サービス記述言語	
-WSML: WEB サービスメタ言語	
-WSDL と WSML の生成	
5. まとめ	15

マイクロソフトの .NET フレームワークは、インターネットを軸としたサービス指向のアプリケーション実行環境を提供する革新的なプラットフォームです。 .NET のアプリケーション開発について語るときしばしば言及されるのが、共通言語ランタイム (Common Language Runtime) と IL (Microsoft Intermediate Language) で、これらに対応したアプリケーション開発ツールが注目されています。しかし、一方で .NET フレームワークは、既存のアプリケーションとの協調を可能にするサポートにも優れています。

ここでは、Micro Focus Net Express を使用して .NET フレームワークで稼動するアプリケーションを作成する方法について説明します。新たな言語を習得したり、新たに登場する言語製品のリリースを待つまでもなく、今すぐに既存の COBOL 資産を、Web サービスとして利用可能にすることができます。

1. Net Express の .NET 対応の概要

.NET については多くのことが語られています。 .NET は非常に多くの異なる分野を包含しており、これを一言で説明することは不可能です。また、マイクロソフトの方向性のコアとなるテクノロジーとなっています。

マイクロソフトは、 .NET フレームワークを、「オープンなインターネットプロトコルと業界標準に立脚した革新的なプラットフォーム」であり、「新たなコンピューティングとコミュニケーションの方法を実現するツールとサービスを提供する」ものであるとしています。

しかし、 .NET フレームワークは、単に Web アプリケーション開発のためのテクノロジーではなく、Windows プラットフォームでのアプリケーション開発の根幹に影響を及ぼすものであることに注意しなければなりません。

Web サービス

.NET の設計思想は、分散したシステム間での通信とデータ共有を切れ目なく可能にすることです。 Web サービスはこれを実現するものです。 Web サービスとは、標準的なインターネットプロトコルを経由して他からアクセス可能な、アプリケーションロジックの実行単位です。ここでは、異なるコンピュータ間でアプリケーション同士が透過的な通信を行うために、HTTP、XML、SOAP などの標準プロトコルが使用されます。

.NET が対象とするのは、個々の Web サイトとその物理的資源ではなく、インターネットで接続され、相互に協調しあう一群のコンピュータ、デバイス、サービスであり、これによってより豊かで幅広いソリューションを提供しようとしています。消費者は自分の意志で、必要な情報を必要な形式で必要なときに入手することができますので、インターネット上で情報・製品・サービスを提供する新たなビジネスが可能になります。このビジネスでは、携帯電話や PDA なども対象になるでしょう。

先日、SOAP Toolkit V2.0 がマイクロソフトからリリースされました。 SOAP(Simple Object Access Protocol) は、Web 上で構造を持った情報を交換するための軽量で簡易なプロトコルであり、XML をベースにしています。 SOAP の目的は、強力な自動的な Web サービスを実現することです。 SOAP Toolkit のもっとも強力な機能として、Web サービス要求をサーバ

一側で COM コンポーネントのメソッド呼び出しに変換することができます。これを利用すれば、COM コンポーネント内に存在する COBOL で書かれたロジックが、容易にインターネットで利用可能にすることができます。

.NET フレームワーク

.NET フレームワークは、.NET プラットフォーム全体のためのインフラとなるものです。アプリケーションを作成して実行するための新しい環境を提供するものであり、Web サービス開発を簡易化し、多種の言語で記述されたコンポーネントで共通に利用可能な実行時サービスを提供しています。また、異言語間、異機種間の相互接続性も実現します。.NET フレームワークでは、現在の Windows アプリケーション開発とは大幅に異なる方法が必要になります。それは、過去 DOS での開発から Windows での開発へ変わっていった経緯にも匹敵するものといえます。

.NET フレームワークには、共通言語ランタイム (CLR) とマイクロソフト中間言語 (MSIL) があります。CLR は、異言語間での相互呼び出し、共通の例外処理、バージョン管理、アプリケーション配布サポートを提供するものです。CLR 用の開発され、MSIL にコンパイルされたプログラムのことをマネージドコードと呼びます。現在流通しているすべてのコンパイラでコンパイルされたコードはマネージドコードではありません。Net Express でコンパイルされた COBOL プログラムもマネージドコードではありません。

したがってマイクロソフトは、マネージドコードとそうでないものとを相互に利用可能にするためのメカニズムを .NET フレームワークの中で用意しています。この相互運用性レイヤーによって、既存の COM コンポーネントをマネージドコードから利用したり、マネージドコードを既存アプリケーションから COM コンポーネントとして利用することができますようになります。

Micro Focus の立場

.NET フレームワークが提供する多くの利点を享受するためには、MSIL を生成する COBOL コンパイラがなければならないのでしょうか。そんなことはありません。.NET が提供する完全な相互運用性によって、Net Express は既に .NET アプリケーションの開発に対応済みです。特に COBOL プログラマにとってもっとも大切な、サーバー上のビジネスロジックを記述する分野では、ほぼ何の制限もありません。

Micro Focus の COBOL 製品のもっとも強力な特徴は、特定のプラットフォームに依存しないことです。Micro Focus は、.NET という特定のプラットフォームに対応することの利点よりも、COBOL で書かれたプログラムが .NET に限らずどんなプラットフォームでも稼動することのほうが重要であると考えています。さらに Micro Focus は、提供するソリューションが常に、新しい分野のプログラマのみならず、既存のプログラマにも適用可能なものでなければならないと考えます。Micro Focus の COBOL

製品に対しては、過去 25 年に渡って膨大な量の COBOL プログラムが書かれてきており、決して無視することのできない資産です。

COBOL 開発環境としての Net Express の利点は、プログラマが記述したひとつの共通なビジネスロジックを、幅広いテクノロジー環境下で利用可能にできることです。バッチ、オンライントランザクション、CGI、CORBA、COM など、これまでに対応してきたものに加えて、.NET フレームワークも既にターゲットになっています。

Micro Focus Net Express は、現在は COM 相互運用性を利用して、.NET フレームワークでの使用を可能としており、.NET ネイティブの CLR 対応にはなっていません。

これは主に以下の 2 つの理由からでした：

- 1 CLR と MSIL によって、COBOL 言語と Micro Focus ランタイムシステムが提供する膨大な言語機能のすべてをサポートし、同時に過去の COBOL 製品との互換性を保つことができるかどうかの検証ができていません。
- 1 COBOL プログラムを MSIL にコンパイルしたときの性能、特に、パック十進、数字編集、ビッグエンディアン整数などの CPU 命令に依存するデータ型を十分な性能で処理できるかの検証ができていません。

このほど Micro Focus はこれらの問題を解決する方向を確立し、既に Net Express の CLR 対応版の開発を表明しています。

現状のままですること

Micro Focus Net Express によって、COBOL で書かれたコアビジネスロジックを、インターネットなどの多様な分散環境で利用することができます。プログラマは、既存のビジネスロジックをエンタープライズコンポーネントにラップし、新たに開発する Web 環境のアプリケーションから利用可能にすることができます。既存のコード資産と、開発者の現在のスキルを活用することができますので、システム開発の工期を短縮し、企業が新たなビジネスに参入する機会を的確に捉えることができます。

Micro Focus が、その ObjectCOBOL テクノロジーによって、世界で最初に COM サポートを提供したのは、5 年前のことでした。以来、たゆまずサポートの水準を上げてきており、最新の Net Express では、COBOL から COM コンポーネントを利用する面でも、COBOL で COM コンポーネントを作成する面でも、他の COBOL 製品を凌駕する機能を提供しています。このホワイトペーパーでは、Net Express の COM サポートを使用した以下の手法を紹介します：

- 1 COBOL で COM コンポーネントを作成し、それをマネージドコードから利用する。
- 1 マネージドコードコンポーネントを COBOL から利用する。
- 1 Microsoft SOAP Toolkit を使用して、COBOL コンポーネントを Web サービスに変換する。

2. COBOL コンポーネントをマネージドコードから利用する

.NET 環境下でマネージドコードからアクセス可能になるために、コンポーネントは「メタデータ」を公開する必要があります。メタデータとは、コンポーネントがサポートするメソッドのパラメタや返却値の型に関する情報です。COM の世界ではタイプライブラリがこれにあたります。.NET フレームワーク SDK の中に、TLBIMP というツールが提供されており、COM のタイプライブラリに記述されたインタフェース情報を、.NET のメタデータ形式に変換することができます。これが、既存の COM コンポーネントをマネージドコードで使用可能にするための最も簡単な方法です。

.NET フレームワークは、タイプライブラリを持たない COM コンポーネントを利用する方法も提供していますが、これにはずっと複雑な手続きを必要とします。幸いなことに、Net Express のクラスウィザードで COM コンポーネントを作成するのであれば、タイプライブラリはウィザードが自動生成してくれます。

COBOL COM コンポーネントの作成

既に述べましたように、マネージドコードから利用可能な COBOL プログラムを作成するためには、まず COBOL コードを COM コンポーネントにしておく必要があります。Net Express のクラスウィザードとメソッドウィザードを使用すると簡単に COM コンポーネントを作成できます。

クラスやメソッドという言葉になじみがなくても心配ありません。Micro Focus は、COBOL による COM 対応を開発する際に COBOL のオブジェクト指向拡張構文を使用することを決めました。COM の世界では、コンポーネントを作成するときも使用するときも、すべてはオブジェクト指向の用語で記述されますので、言語としてオブジェクト指向をサポートすることは必須だったからです。実際に、これと同様のテクニックによって、COBOL を Java クラスとしてラッピングする方法も提供されており、同じ COBOL ソースプログラムを、同時に COM と Enterprise Java Bean として利用可能にすることもできます。

しかし、Micro Focus がオブジェクト指向 COBOL 構文を採用したからといって、プログラマが COM コンポーネントを作成するためにオブジェクト指向の文法を勉強しなければならないわけではありません。Net Express のウィザードが、オブジェクト指向 COBOL の骨組みを自動生成してくれますので、そこへロジックとして必要な COBOL のコードを挿入するだけで完成することができます。したがって、ほとんどの場合、これまで使い慣れた COBOL の世界の知識だけで作業することができます。

COM コンポーネントをマネージドコードから利用可能にする

COM コンポーネントを作成したら、次にコンポーネントに関するインタフェース情報をメタデータのファイルに変換します。これは、.NET フレームワーク SDK に含まれる TLBIMP というユーティリティで行います。以下に、TLBIMP のコマンド形式を示します：

TLBIMP <入力ファイル> /out:<出力ファイル>

ここで、<入力ファイル> は、タイプライブラリ情報を含むファイルのパス名です。たいていは、タイプライブラリ情報はコンポーネントの DLL に埋め込まれていますので、ここでは DLL の名前を指定します。

<出力ファイル> は、メタデータを含む出力ファイル名です。結果はやはり DLL になり、COM コンポーネントといっしょに実行環境に置かれることとなります。

例題プログラム

マネージドコードから COBOL を利用する方法に関する各論を、簡単な例題を通して見てみます。

ポイントは既存の COBOL ロジックを再利用することですから、ここでは話をもっとも簡略化し、以下のような COBOL ロジックを考えてみます：

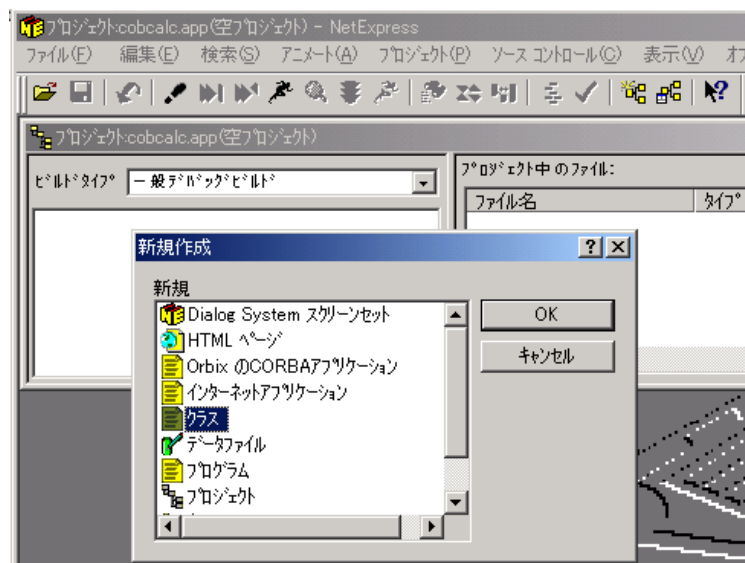
```
COMPUTE RETURN-VAL = INPUT-VAL * 2.
```

すなわち、入力パラメタとして渡された数値を 2 倍して、結果を返却値として返すというものです。

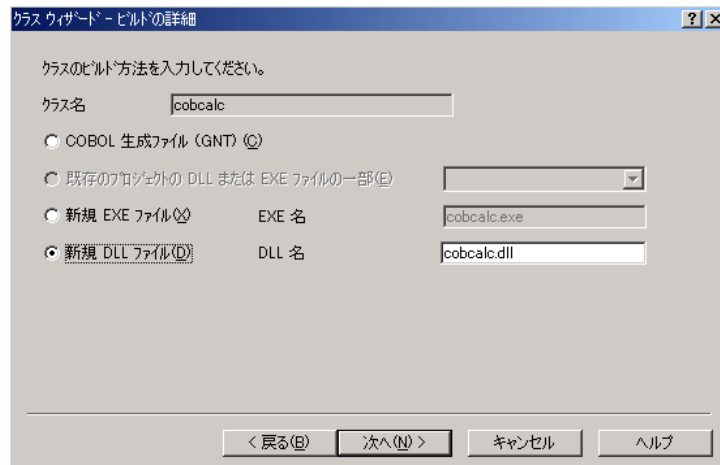
クラスの作成

この簡単なロジックを COM 化するために、Net Express のウィザードがどのように支援するかを見てみます。

まず、Net Express のプロジェクトを作成し、[新規作成] メニューから [クラス] を選択します。

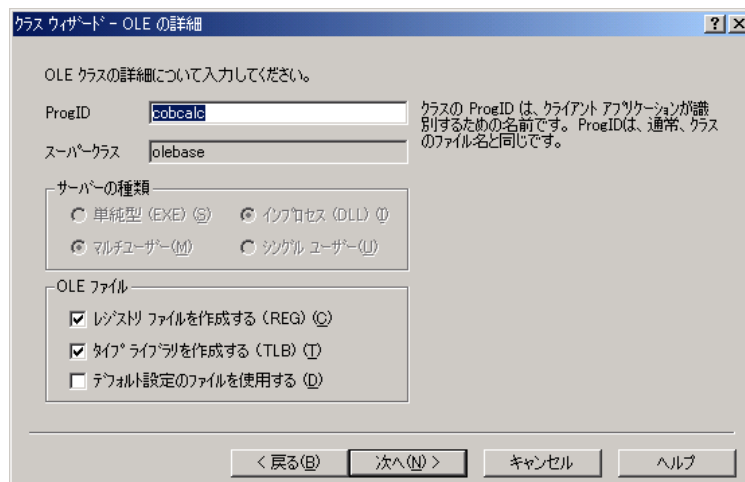


クラスウィザードがスタートし、COBOL クラスを作成する上でのさまざまなオプションを問い合わせます。以下は、クラスのビルド方法を指定するダイアログです。この例題では、クラスの名称を **cobcalc** とします。COM のインプロセスサーバーを作成する場合は、DLL へのビルドを指定します。

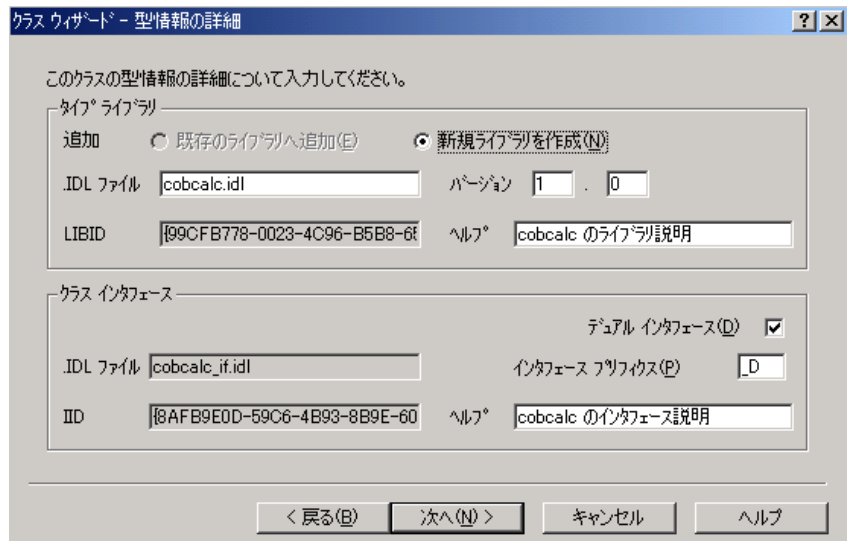


COM 情報の設定

COM コンポーネント作成の場合は、以下のように関連するファイルの生成についても問い合わせます。マネージドコードから利用する COM を作成する場合、ここで注意が必要です。必ず **[タイプライブラリを作成する]** をチェックし、**[レジストリファイルを生成する]** をチェックし、**[デフォルト設定のファイルを使用する]** のチェックをはずしてください。



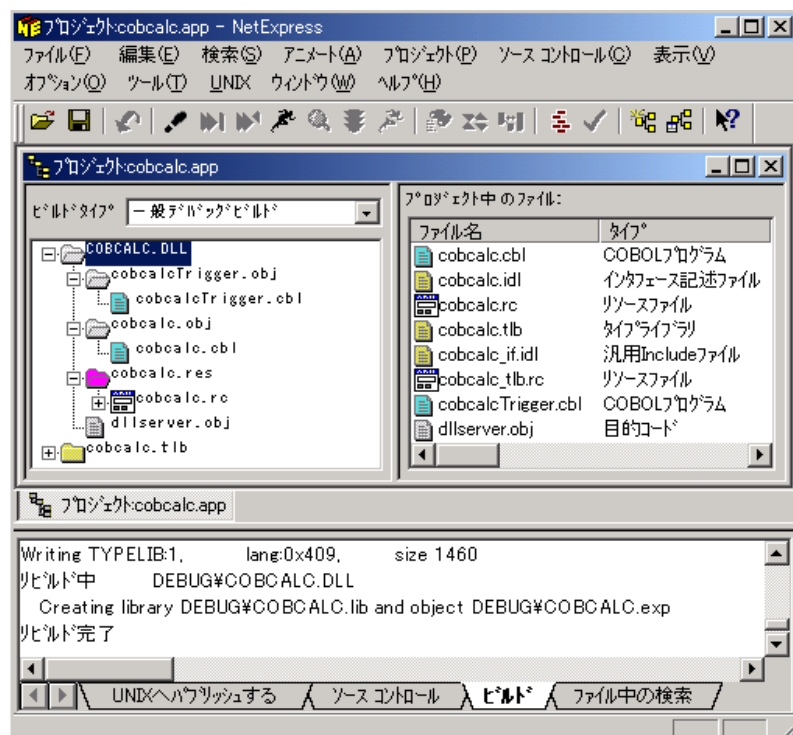
さらに先へ進むと、以下のように生成するタイプライブラリの内容を問い合わせるダイアログが現れます。



ここでのポイントは2つあります。[インタフェースプリフィックス]と[デュアルインタフェース]です。「インタフェース」とは、実際にコンポーネント中のメソッドが呼び出されるときのメカニズムを識別する名称であり、後でマネージドコードからこれ呼び出すときに使用する名称になります。[インタフェースプリフィックス]のデフォルト設定は、_D となっています。この場合、クラス名称 cobcalc の先頭に _D をつけて _Dcobcalc がインタフェース名称になります。この例題ではプリフィックス I を指定し、インタフェース名称 Icobcalc を使用することにします。

もうひとつ、[デュアルインタフェース]ボックスを必ずチェックしてください。これについては後で説明します。

ウィザードですべての指定を完了すると、以下のようにクラスを含むプロジェクトが完成します。



8 個のファイルが生成されているのがわかります。これから編集が必要となるのは、クラス定義の主ソースファイルである `cobcalc.cbl` だけです。

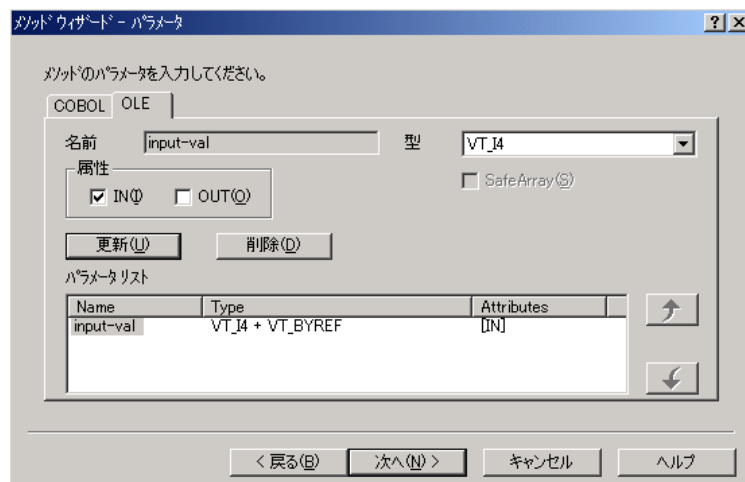
これで、実際にコンポーネント登録可能な DLL が作成されていますが、まだこれは空のコンポーネントであって、内容を何も含んでいません。

メソッドの追加

そこでこのクラスにメソッドを追加します。Net Express の [メソッドを追加] ボタンをクリックすると、メソッドウィザードがスタートします。以下のようにパラメータや返却値の型を対話式で生成するとメソッドを定義するコードが自動的に `cobcalc.cbl` 中に挿入されて行きます。



ここでは整数型の入力パラメータ `input-val` を一つだけ持つメソッド `DoubleIt` を定義します。また、ここでタイプライブラリに記録されるインタフェース情報も追加で定義することができます。



以上のように、COBOL 言語では定義されないパラメータの IN/OUT 区別なども指定することができます。

ユーザーロジックの追加

この結果ウィザードによって以下のようなコードが自動作成されます。

```
method-id. "DoubleIt".
local-storage Section.
*>---USER-CODE. 以下に必要な項目を追加
linkage Section.
01 input-val           pic x(4) comp-5.
01 return-val          pic x(4) comp-5.
procedure division using by reference input-val
                        returning return-val.
*>---USER-CODE. 以下に'リット'を追加
exit method.
end method "DoubleIt".
```

最後に、この手続き部の中に

```
COMPUTE RETURN-VAL = INPUT-VAL * 2.
```

という一文を加筆するだけでクラスは完成します。オブジェクト指向 COBOL 構文についての知識は一切必要のないことがわかります。

ここでプロジェクトをビルドすると **cobcalc.dll** という自己登録型の COM コンポーネントの DLL が作成されます。この DLL はタイプライブラリ情報を内包していますので、Windows の **regsvr32** ユーティリティを使用してレジストリに登録することができます。

(例) > regsvr32 cobcalc.dll

メタデータの作成

続いて、この DLL をマネージドコードから利用可能とするために必要な .NET のメタデータファイルを生成します。Windows コマンドプロンプトから以下のように打鍵します：

```
> tlbimp cobcalc.dll /out:cob_cobcalc.dll
```

この結果、メタデータを含む **cob_cobcalc.dll** が作成されます。この DLL は、**cobcalc.dll** を使用したいマネージドコードが置かれているのと同じディレクトリにコピーしておきます。

マネージドコードのコンパイル

作成した `cobcalc.dll` を使用するマネージドコードを、Visual Basic.NET または C# で作成します。このとき、コンパイラに対して `cobcalc.dll` のメタデータの所在を示してやる必要があります。これには、コンパイラのコマンド行で `/reference (/r と省略) オプションに cob_cobcalc.dll を指定します。Visual Basic.NET の場合、[参照追加] でこの DLL を指定することもできます。`

これに加えて、タイプライブラリで定義されたインタフェース情報をマネージドコードから参照可能にしなければなりません。このためには C# の `using` 文、または Visual Basic.NET の `imports` 文を使用します。これらの文では、タイプライブラリ中に定義されたライブラリ名を指定します。この例題の場合は `cobcalc` になります。

C# の場合：

```
using cobcalc;
```

VB.NET の場合：

```
imports cobcalc
```

バインド方法の選択肢

マネージドコードから COBOL コンポーネントを呼び出すのに 2 つの方法があります。簡単な方法はアーリーバインディングと呼ばれるものです。この他にレイトバインディングという方法があり、これは Visual Basic.NET では簡単にできますが、C# ではやや複雑な手順を必要とします。バインディングとは、別のアプリケーション中のコードを参照するメカニズムのことです。

レイトバインディングは Visual Basic から COM コンポーネントを使用するために以前から使用されていた方法です。呼び出されるコードの解決は実行時になされるため、アーリーバインディングより性能が劣化します。この方法によると、コンポーネントが呼び出されるごとに、暗黙的な探索プログラムが実行されて、登録されたオブジェクトとメソッドをサーチします。アーリーバインディングでは、コンパイル時に呼び出されるアプリケーションに関するすべての情報を必要とします。これによって、型情報の整合性をコンパイル時に検証することができ、実行時には高速に呼び出すことができます。

COM コンポーネントの作成時にデュアルインタフェースの指定をしておくことが重要であったのは、ここで意味があります。この指定がないとアーリーバインディングはできません。

COBOL クラスの呼び出し

Visual Basic.NET や C# から COBOL コンポーネントのインスタンスを作成するには、使用したいインターフェースの型を持つオブジェクトを宣言し、このオブジェクトのインスタンスを新規作成します。以下に例を示します：

C# の場合

```
lcobcalc objCobcalc;  
objCobcalc = (lcobcalc) new  
cobcalc.cobcalc();
```

VB.NET の場合

```
dim objCobcalc as lcobcalc  
objCobcalc = new cobcalc.cobcalc()
```

ここからは通常の方法で、生成されたインスタンスに対して以下のようにメソッドを発行できます：

C# の場合

```
Result = objCobcalc.DoubleIt(123);
```

VB.NET の場合

```
strResult = objCobcalc.DoubleIt(123)
```

3. COBOL からマネージドコードを利用する

前章ではマネージドコードから COBOL コンポーネントを利用する方法を見ました。次に、その反対に COBOL からマネージドコードを利用する方法について見ます。

マネージドコードを COBOL から利用可能にするために、まずそのメタデータをもとにして、これに対応する Windows レジストリエントリを作成しなければなりません。これが完了すると、マネージドコードは COM コンポーネントとして従来のアプリケーションから利用可能になります。これには REGASM コーティリティが使用できます。REGASM.EXE は Microsoft .NET Framework SDK に含まれているコマンドです。マネージドコードを含む DLL に対して、

```
regasm managedcode.dll
```

とコマンド実行します。

この他に、Microsoft .NET Framework SDK には REGSVCS というユーティリティも含まれています。こちらは、レジストリ登録だけでなく、COM+ 1.0 のコンポーネントとしての登録までを行います。

4. Web サービスを COBOL で作成する

Web サービスとは、インターネットを經由して外部のアプリケーションに対して提供される、ロジックとデータのユニットです。Web サービスを利用するアプリケーションは、HTTP、XML、SOAP といった標準に準拠した形式でサービス要求を発行します。このとき相手方の Web サービスが、どんなハードウェアやオペレーティングシステムで、どのように実装されているかについて知る必要がありません。

COBOL で Web サービスを作成するためのキーポイントが Microsoft SOAP Toolkit です。これは、Web サービスを作成するための開発環境であると同時に、マイクロソフトによる SOAP の実装でもあります。SOAP (Simple Object Access Protocol) は、XML をベースとした異機種間メッセージ交換のための標準です。SOAP の規格仕様は www.w3.org/tr/soap で見るすることができます。

Microsoft SOAP Toolkit は、既存の COM コンポーネントを Microsoft IIS 下で利用可能な SOAP サービスに変換するために、以下の 2 つのものを提供しています：

- 1 外部からの SOAP 要求に対して、Web サービス記述言語(WSDL) と Web サービスメタ言語(WSML) によって事前に定義された方式で COM オブジェクトを起動して、サービスを実行するサーバーサイドコンポーネント
- 1 WSDL/WSML Generator _ 既存の COM コンポーネントから、WSDL と WSML を自動生成する

WSDL: Web サービス記述言語

Web サービス記述言語(WSDL) は、Web サービスが提供するインタフェースの仕様を記述する XML 文書です。クライアントから送信されるサービス要求は、ここに定義される形式に準拠していなければなりません。すなわちこれは、サーバーとクライアントとの間で交わされる契約書のようなものです。サービスは、WSDL の記述に準拠していることを前提に提供されます。

たとえば銀行口座の SOAP サービスを提供するのであれば、「残高照会」や「口座振替」のような操作がサービスとして提供されます。サービス提供者は、これらのサービスのパラメタ形式や返信される結果の形式を定義した WSDL ファイルを、サーバー上に置きます。サービスを利用したいクライアントは、まず最初にこの WSDL をダウンロードし、その

内容に準拠した SOAP 要求を XML で作成し、サーバーに送信します。サーバーは、SOAP 要求に基づいてサービスを実行し、クライアントへ口座残高を SOAP 応答として返信します。

WSML: Web サービスメタ言語

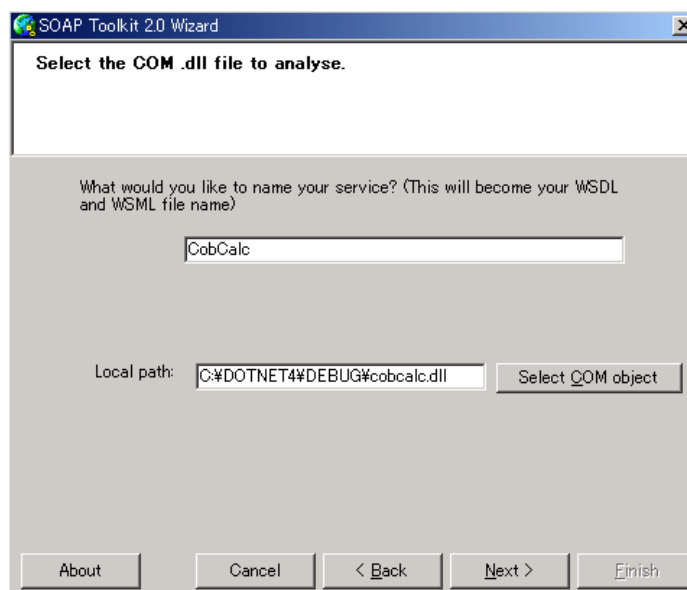
Web サービスメタ言語(Wsdl)は、WSDL 中の定義に準拠する SOAP サービス要求を、どのように COM のメソッドにマップするかを記述する定義ファイルです。

WSDL と WSML の生成

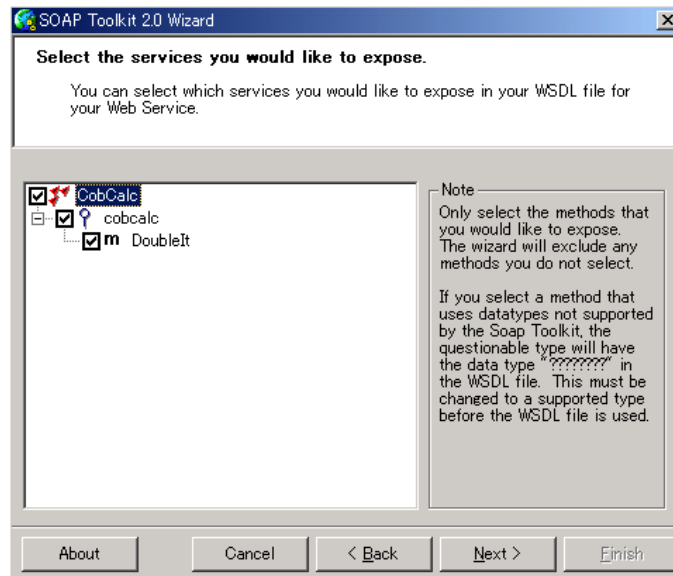
WSDL も WSML も XML ベースですので、エディタで作成することもできますが、手作業による作成は困難です。Microsoft SOAP Toolkit は、これを自動生成するツールを提供しています。既存の COM コンポーネントを入力として、その中で Web サービスとして開示したいクラスやメソッドを指定すると、対応する WSDL/WSML ファイルを自動生成してくれます。

SOAP Toolkit をインストールし、スタートメニューから[WSDL Generator]を選択すると、以下のように 2 章で作成した COBOL の COM コンポーネントに対し、WSDL と WSML を作成します。

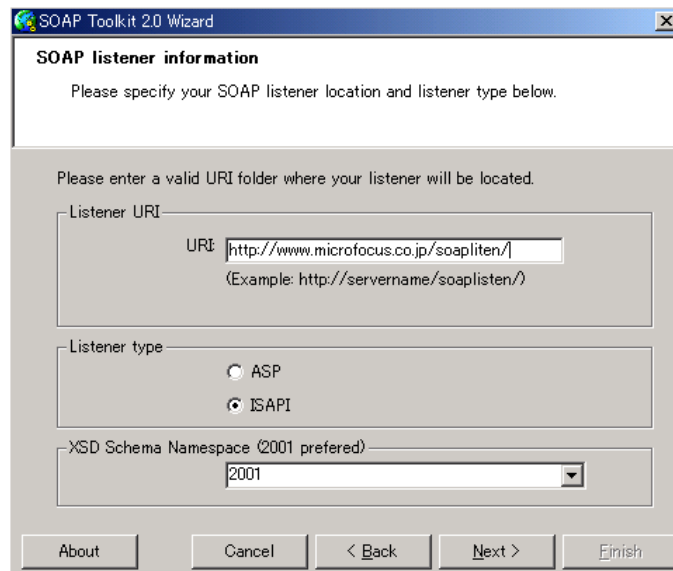
まず、入力となる COM コンポーネントの DLL を指定します:



指定された COM コンポーネント中に定義されたメソッドの一覧が表示されますので、Web サービスとして開示したい部分を選択します:



最後に、サービスのリスナーの URL を指定します:



以下は、生成された WSDL ファイルの一部です:

```
<message name='cobcalc.DoubleIt'>
  <part name='p0' type='xsd:int'/>
</message>
<message name='cobcalc.DoubleItResponse'>
  <part name='Result' type='xsd:int'/>
</message>
<portType name='cobcalcSoapPort'>
```

```
<operation name='DoubleIt' parameterOrder='p0'>
  <input message='wsdlIns:cobcalc.DoubleIt' />
  <output message='wsdlIns:cobcalc.DoubleItResponse' />
</operation>
</portType>
```

この SOAP サービスを利用するには、SOAP 要求を送信することができる任意の言語が使えます。以下に、WSH の VBScript で記述したサービス要求の例を示します:

```
set soapclient = CreateObject("MSSOAP.SoapClient")
Call soapclient.mssoapinit
("http://www.microfocus.co.jp/Soaplisten/cobcalc.wsdl",
 "cobcalc", "cobcalcSoapPort")
wscript.echo soapclient.DoubleIt(1234)
```

このスクリプトは、サーバー上で 2 章で作成した COBOL COM を起動し、結果として 2468 を表示します。

5. まとめ

企業情報システムにおける COBOL への過去の投資を最大限に活用し、.NET プラットフォーム上でのアプリケーションを開発してゆくために、マイクロフォーカスによって既に提供され、大企業ユーザにて実績もあるソリューションが有効であることを見てきました。