COBOL

プログラマのための.NET入門

NetCOBOL for .NET 応用編

NetCOBOL

はじめに

◆本書の目的

本書は、NetCOBOL for .NET を使用したプログラミングの教材です。NetCOBOL for .NET の開発環境を設定することにより、実際に操作しながら学習できます。その際、以下のいずれかの製品が必要となります。

・ NetCOBOL Base Edition 開発パッケージ for .NET V3.0
・ NetCOBOL Standard Edition 開発パッケージ for .NET V3.0

また、本書は以下の読者を対象として記述されています。

・ Windows で NetCOBOL for Windows の開発経験がある。
・ オフコンや汎用機で COBOL の開発経験がある。
・ NetCOBOL for .NET を使用する、または今後の使用を考えている。
・ COBOL 言語の基礎的知識を持っている。
・ Windows の基本的操作ができる。

なお、本書で掲載している画面は、WindowsXP 上でのイメージです。また、ご使用される Visual Studio のオプションによって若干異なることがありますのでご了承ください。

◆本書の構成

本書の構成と内容は以下のとおりです。

第1章　NetCOBOL for .NET とは
NetCOBOL for .NET と.NET Framework の概要、およびそれぞれの関係を学習します。

第2章　NetCOBOL for .NET を使う
プログラム開発の方法を、プログラミングしながら学習します。

第3章　NetCOBOL for .NET の特徴
クラスやオブジェクトなど、手続き型の COBOL とは異なる特徴について学習します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第4章　NetCOBOL for .NET による画面プログラムの作成
Windows フォームアプリケーションの作成の演習を交え学習します。

第5章　データアクセス
データベースやファイルなど、データ資産と連携する方法について学習します。

第6章　印刷処理
定義体を使用しない印刷処理について学習します。

第7章　Web アプリケーションの作成
ASP.NET を利用した Web フォームアプリケーションの開発方法について演習を交え学習します。

付録 A　定義体を使用した印刷
帳票定義体を使用した印刷の方法について解説しています。

付録 B　アプリケーションの配置
作成したアプリケーションを配置(配布)する方法について解説しています。

付録 C　サンプルプログラム概要
本書の演習で利用するサンプルプログラムの概要について解説しています。

付録 D　ADO.NET を使用したサンプル
第 5 章のデータアクセスプログラム(埋め込み SQL)を、ADO.NET を利用したサンプルに変更した例を紹介しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

◆略語

本書の記述	正式名称
Windows2000	Microsoft® Windows® 2000 Server Microsoft® Windows® 2000 Advanced Server Microsoft® Windows® 2000 Professional
WindowsXP	Microsoft® Windows® XP Professional Microsoft® Windows® XP Home Edition
Windows Server 2003	Microsoft® Windows® Server 2003 R2, Standard Edition Microsoft® Windows® Server 2003 R2, Enterprise Edition Microsoft® Windows® Server 2003, Standard Edition Microsoft® Windows® Server 2003, Enterprise Edition
SQL Server	Microsoft® SQL Server 2005

◆商標

※Microsoft、Windows は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。

※Sun、Sun Microsystems、Sun ロゴ、Solaris およびすべての Solaris に関連する商標およびロゴは、米国およびその他の国における米国 Sun Microsystems,Inc.の商標または登録商標であり、同社のライセンスを受けて使用しています。

※Solaris（TM） Operating System は、本資料では「Solaris」と記述しています。

※その他、製品名などの固有名詞は各社の商標または登録商標です。

※その他、本資料に記載されているシステム名、製品名などは必ずしも商標表示（TM・®）を付記していません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第1章 NetCOBOL for .NETとは

1.1 .NET Frameworkの概要
1.2 NetCOBOL for .NETの概要
1.3 NetCOBOL for .NETで開発できるアプリケーション
1.4 NetCOBOL for .NETのシステム構築パターン
1.5 従来のCOBOLとNetCOBOL for.NETの共通点と相違点

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.1 .NET Frameworkの概要

1.1.1 プログラムを作る前に

NetCOBOL for .NETの概要を把握する前に、Microsoft .NET Frameworkの概要を把握する必要があります。

.NET Frameworkは、様々な形式のアプリケーション(Windowsアプリケーション、XML Webサービス、Webアプリケーションなど)に対するサービス・インタフェースを提供しています。“Framework”とは「枠組み、骨組み」という意味で、具体的に言えば、XML Webサービスなどから利用できるコンポーネントやクラスの集合体です。

.NET Frameworkを使用して開発したアプリケーションは、.NET Frameworkが実装された環境であれば、プラットフォーム・ハードウェアやOSの違いを超えて、どこでも実行することができます。

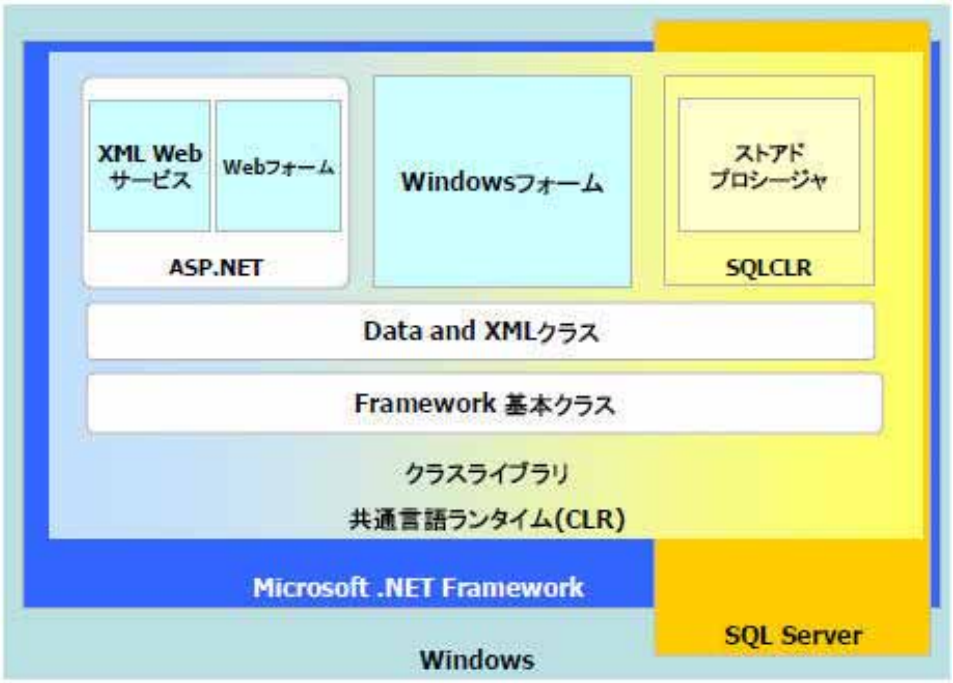


図1.1.1 .NET Frameworkの構成

.NET Frameworkでは、様々なプログラミング言語でアプリケーションの開発を行なうことができるように、共通のプログラミング・インタフェースを提供しています。また、異なる言語のプログラムから作成されたオブジェクトも、共通のランタイムシステムによって協調して実行

All Rights Reserved, Copyright FUJITSU LIMITED 2006

できます。.NET Frameworkを利用することで、アプリケーションの生産性やプログラム・コードの再利用性、システムの信頼性を高めることができます。

.NET Frameworkは次の要素から構成されています。

✓ 共通言語ランタイム(CLR)

ランタイム・サービスを受け持ち、言語(プログラミング言語)の統合、セキュリティ強化、メモリ管理、プロセス管理、スレッド管理などを行います。

✓ クラスライブラリ

クラスライブラリは、複数のクラスで構成されており、.NET Frameworkでサポートされる複数のプログラミング言語に、共通で一貫した開発インタフェースを提供します。

.NET Framework基本クラスは、入出力、文字列操作、セキュリティ管理、ユーザインタフェースデザインなど.NET Frameworkの標準的な機能を提供します。

Data and XMLクラス(ADO.NETクラスとXMLクラス)は、データベースやXMLの操作を行うための機能を提供します。

ASP.NETクラスはWebアプリケーションとXML Webサービスを開発するための機能を提供します。

Windowsフォームクラスは、デスクトップアプリケーション用の開発を行うための機能を提供します。

1.1.2 言語に依存しない仕様

従来のアプリケーションの開発は、言語ごとに使用する開発製品、コンパイラ、コンパイル後に生成された実行モジュールの全てが異なり、言語固有の形式で実行モジュールが作成されていました。従って、配布の際も開発時に使用した言語専用のランタイムライブラリがなければ動作させることはできません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

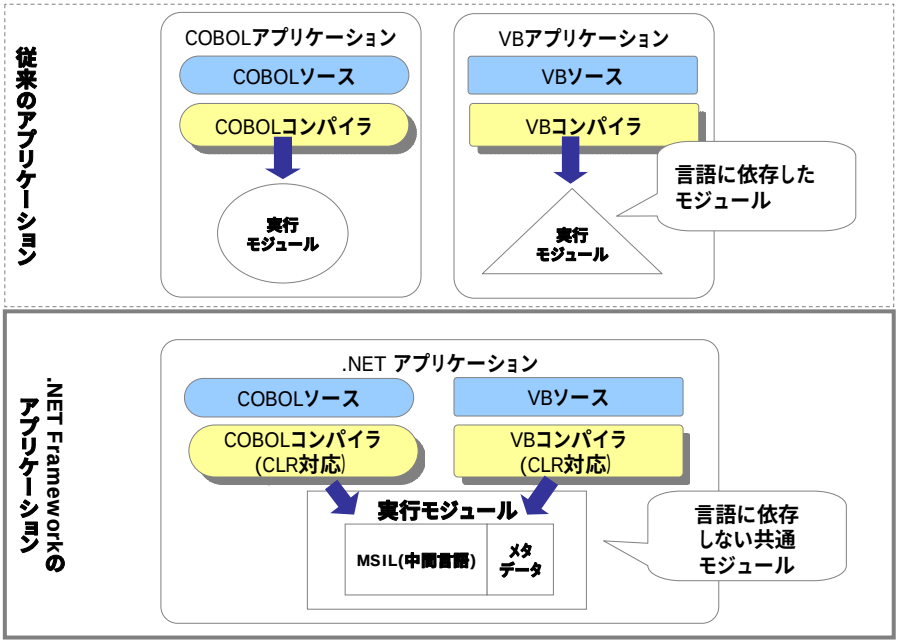


図 1.1.2 言語に依存しない仕様

.NET Frameworkを利用して開発を行う場合、 COBOL、VB、C#、J#、C++などの共通言語ランタイムに準拠している多くの言語を使ってプログラムを記述することが可能であり、各言語のコンパイラは各言語のソースから言語を意識しない実行環境 (CLR：Common Language Runtime)用の実行モジュールを生成します。

1.1.3 OSに依存しない仕様

従来のアプリケーションは、開発時に使用したプラットフォーム上でのみ実行することが可能でした。例えば、NetCOBOL for Windowsは、Windows上で動作するアプリケーションを開発する製品であり、ソースレベルでは他のプラットフォームと互換性があるものの、実行モジュール(EXEやDLL)を他のOS上でそのまま動作させることはできません。各プラットフォーム専用の開発製品でコンパイルし、実行モジュールを作成する必要があります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

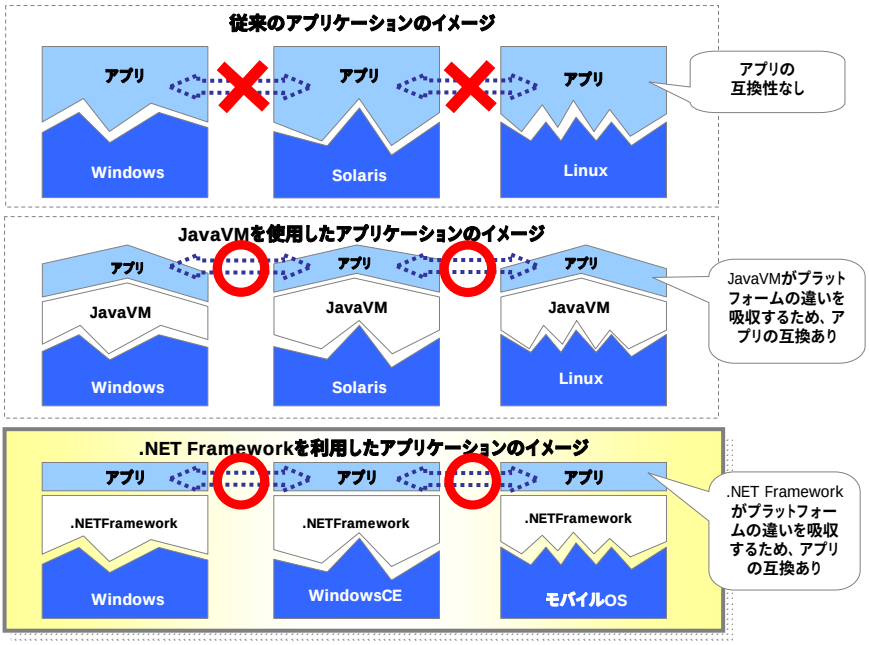


図 1.1.3 OSに依存しない仕様

しかし、.NET Frameworkを利用して開発したアプリケーションは、実行時、OSとアプリケーションの間に、.NET Frameworkが介在することにより、プラットフォームの違いを吸収するため、JavaVMと同じように、あらゆるOS上で同じアプリケーションを実行することが可能になります。

1.1.4 ハードウェアに依存しない仕様

.NET Framework上で開発された.NETアプリケーションは、.NET Frameworkがインストールされた環境であれば、PC、PDA、携帯電話などあらゆるハードウェア上で動作させることが可能になります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

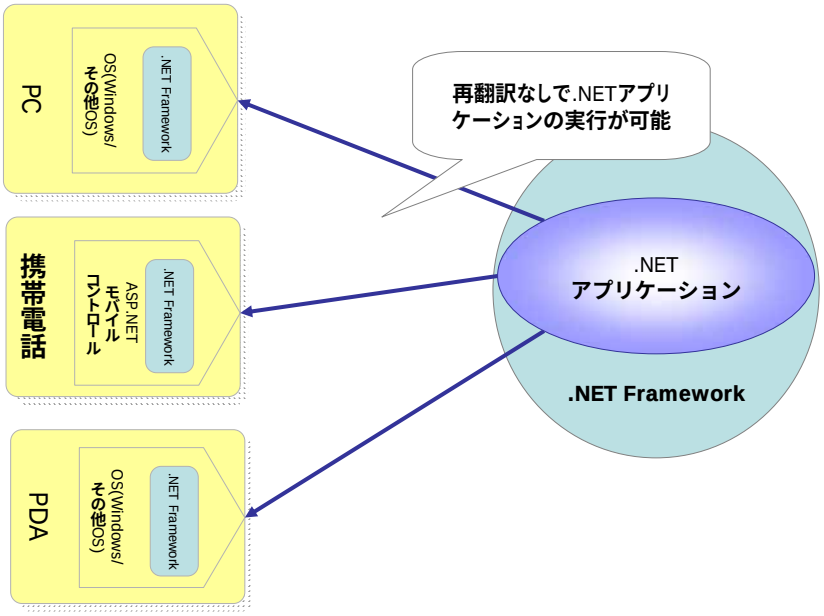


図 1.1.4 ハードウェアに依存しない仕様

また、.NETアプリケーションは再翻訳を行うことなく.NET Framework上で実行することができます。これは、.NET Frameworkが下位互換の動作を保証しているためです。具体的には.NET Framework 1.0／1.1を使用して作成したアプリケーションは.NET Framework 2.0で実行することが可能です。

.NET Frameworkは、NetCOBOL for .NETやVisual StudioのCD-ROM、またMicrosoftのウェブページからダウンロードしてインストールすることが可能です。また、Windows Server 2003には標準で.NET Framework 2.0が提供されています。

1.1.5 言語の特性を活かした開発

既に述べたように.NET Frameworkは、プラットフォーム、OS、言語に依存にしないアプリケーションの開発を可能にします。これは、各言語が得意とする処理を各言語の開発に携わってきた技術者が分散開発することにより、保守性に優れた、より柔軟なアプリケーションの開発が可能になるというメリットがあります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

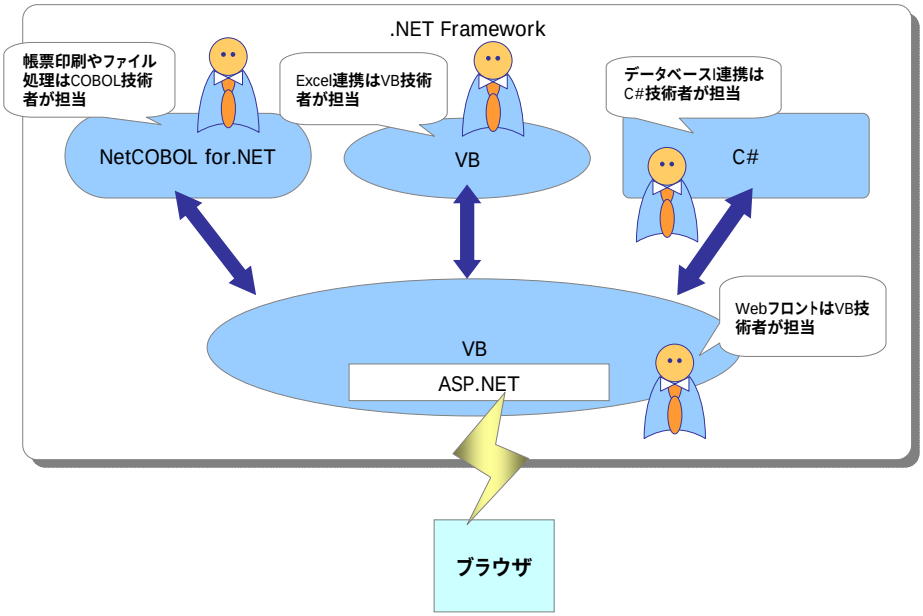


図 1.1.5 言語の特性を活かした開発

例えば、図のようにCOBOL技術者が帳票印刷やファイル処理、VB技術者がExcel連携やWebフロント、C#の技術者がデータベース連携のロジックを担当するといったような開発が可能となります。

1.1.6 .NET Frameworkの活用イメージ

下図は、.NET Frameworkで開発したWebアプリケーションやXML Webサービスを活用したイメージです。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

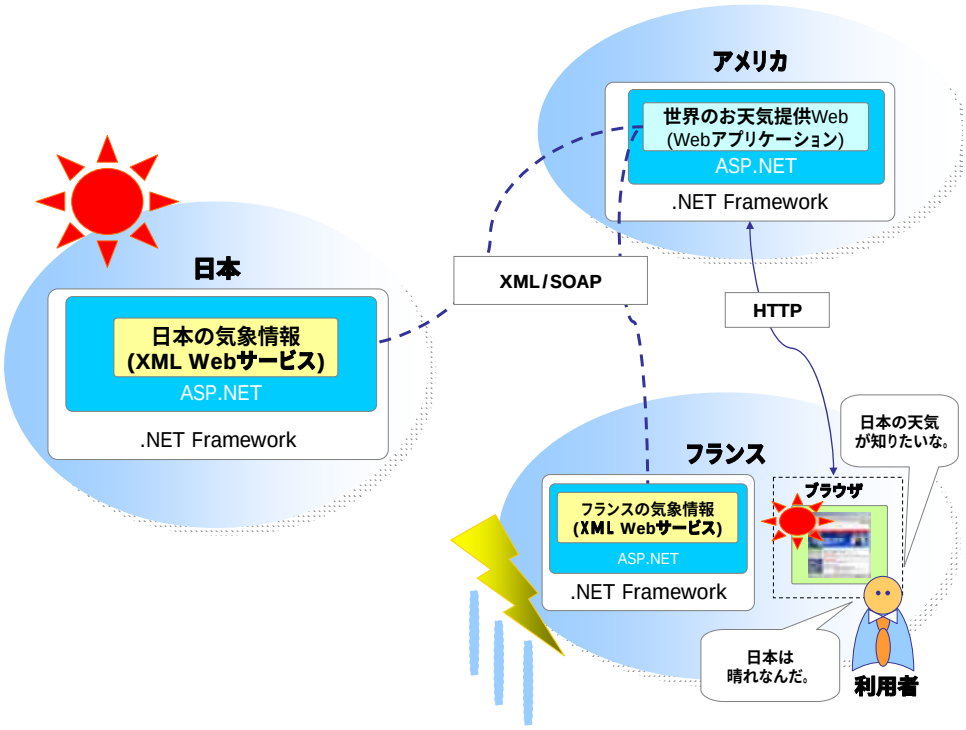


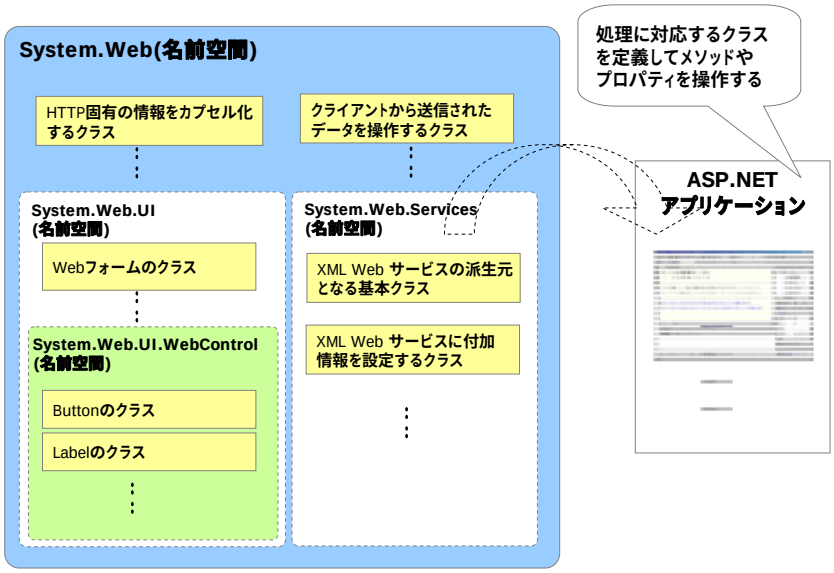
図 1.1.6 .NET アプリケーションの活用イメージ

利用者が日本のお天気情報を知るために、世界のお天気情報を提供しているWebアプリケーションにアクセスしています。世界のお天気提供Webアプリケーションは、利用者の依頼に応じて、日本のお天気情報を提供しているXML Webサービスにアクセスし、情報を取得して利用者にページを返しています。このようにXML Webサービスを利用することで情報を一元管理して提供することができます。

1.1.7 クラスの概要（ASP.NET）

.NET Frameworkを使用してアプリケーションを作成する場合、アプリケーションの形態に応じて使用するクラスが異なります。例えば、WebアプリケーションやWebサービスを開発する場合、ASP.NETの技術を利用して作成しますが、その機能はSystem.Web名前空間に含まれるクラスが提供しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



※System.Web名前空間には上図以外にも多くのクラスと名前空間が含まれています。

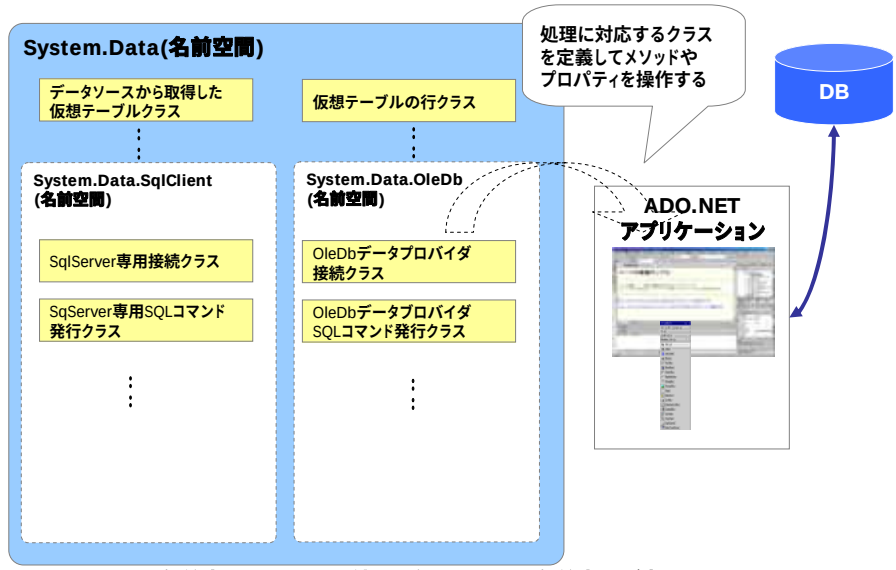
図 1.1.7 System.Web名前空間(ASP.NET)

利用者は、処理に対応するクラスをSystem.Web名前空間から探して定義し、そのクラスから提供されているメソッドやプロパティを操作してアプリケーションを作成します。

1.1.8 クラスの概要 (ADO.NET)

.NET Frameworkを利用してデータベース連携のアプリケーションを作成する場合、ADO.NETの技術を利用して開発しますが、その機能は、System.Data名前空間に含まれるクラスが提供しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



※System.Data名前空間には上図以外にも多くのクラスと名前空間が含まれています。

図1.1.8 System.Data名前空間(ADO.NET)

利用者は、処理に対応するクラスをSystem.Data名前空間から探して定義し、そのクラスから提供されているメソッドやプロパティを操作してアプリケーションを作成します。

1.1.9 .NET Framework基本クラスの概要

.NET Frameworkで作成するアプリケーション共通の処理を行うクラスはSystem名前空間に含まれています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

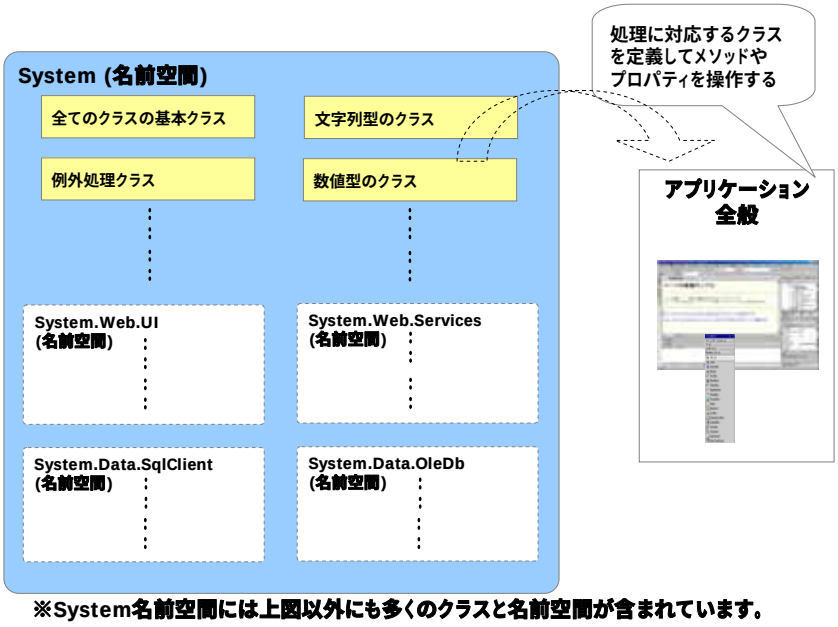


図1.1.9 System名前空間(.NET Framework基本)

System名前空間には.NET Frameworkで使用するデータ型(値型と参照型)やイベント、例外処理を行うクラスが定義されており、前述したASP.NETやADO.NETで使用する名前空間も全てSystem名前空間に含まれています。利用者は、処理に対応するクラスをSystem名前空間から探して定義し、そのクラスから提供されているメソッドやプロパティを操作してアプリケーションを作成します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.2 NetCOBOL for .NETの概要

1.2.1 NetCOBOL for .NETの構成

NetCOBOL for .NETは、.NET Frameworkに対応した富士通製のCOBOL製品であり、.NET Frameworkプラットフォーム上で動作するアプリケーションを開発することができます。

NetCOBOL for .NETは名前の通り、.NET Frameworkに対応しているため、.NET Frameworkが提供する多くのクラスライブラリを利用して、様々な機能を実現できます。例えば、COBOLでWebアプリケーションやWebサービスを構築することができます。

また、従来COBOLのビジネスロジックで作成されているアプリケーションや画面帳票定義体(MeFt)と連携を行うことも可能なため、既存資産と最新技術を融合したアプリケーションの開発ができます。

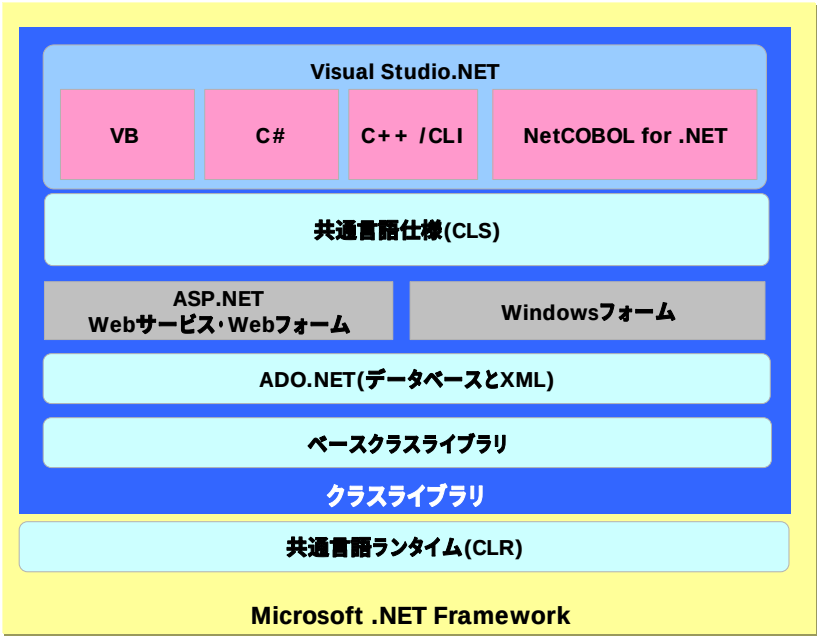


図1.2.1 NetCOBOL for .NETと.NET Frameworkの構成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.2.2 NetCOBOL for .NETと.NET Frameworkの関係

NetCOBOL for .NETは、.NET Frameworkの機能を利用することで、いままでのCOBOLになかった様々な機能を利用できます。特に、画面処理やWebシステムに関する機能が、従来のCOBOLと比較して飛躍的に向上します。

また、従来の手続き型COBOLの主要な機能についてもソースレベルで完全互換しています。

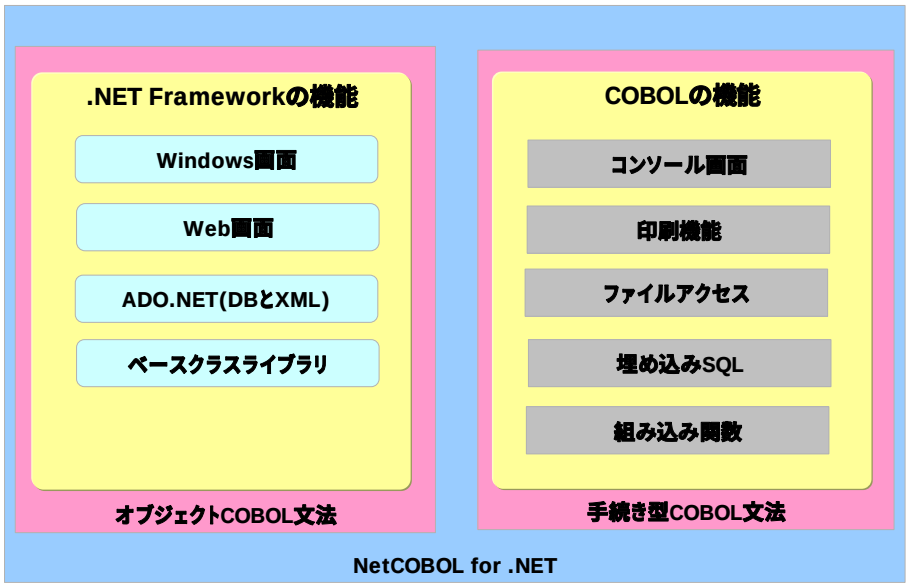


図1.2.2 .NET Frameworkの機能とCOBOLの機能

.NET Frameworkの機能の利用

.NET Frameworkの機能を利用するためには、COBOLからクラスライブラリを呼び出す必要があります。クラスライブラリはCOBOLオブジェクト指向プログラミング機能を用いて利用します。オブジェクト指向プログラミング機能は最新のCOBOLで規格化されています。NetCOBOL for .NETのオブジェクト指向プログラミング機能は規格をベースに多少の.NET固有機能を追加したものです。

.NET Frameworkが提供する機能としては、WindowsやWebのGUI画面、ADO.NETを利用したデータベースアクセス、多様な機能を提供するベースクラスライブラリなどがあります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

COBOL独自の機能の利用

.NET Frameworkに依存しない処理については、従来どおりの手続き型のCOBOL文法を利用することができます。コンソール画面の利用、印刷機能、ファイルアクセス機能など、COBOLの主要機能を利用することができます。

1.2.3 NetCOBOL for .NETの製品構成

NetCOBOL for .NETシリーズは開発環境製品および、サーバ運用環境製品を提供して提供しており、ご利用になる製品は、

「NetCOBOL “エディション名” “パッケージ名”」で決まります。次にそれぞれの製品について説明します。

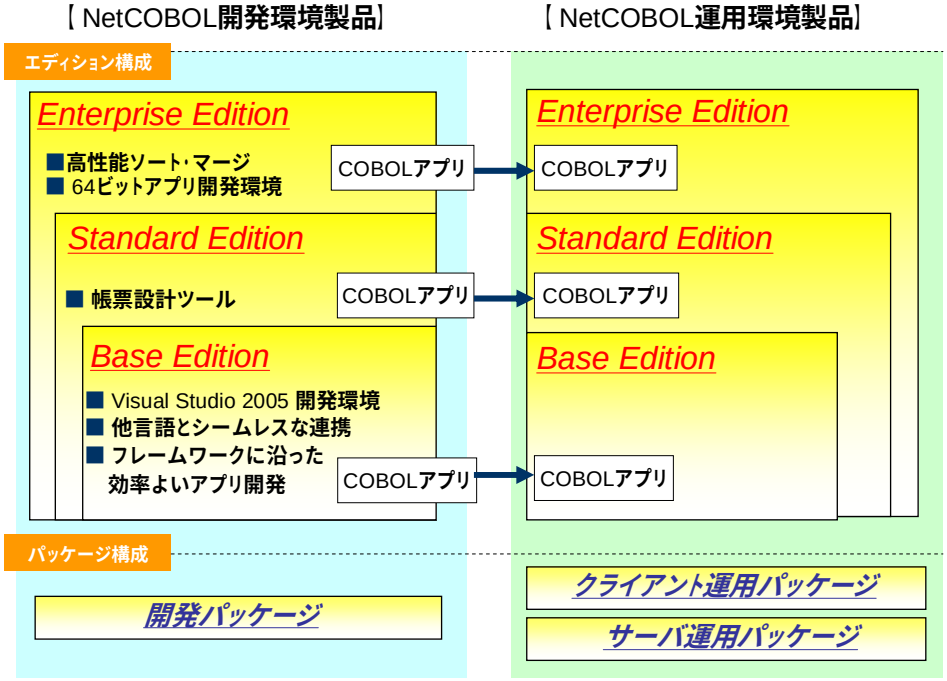


図 1.2.3 NetCOBOL for .NET 製品構成

開発環境製品

NetCOBOL for .NETはCOBOLコンパイラと関連ツールから構成されており、.NETプラットフォーム向けアプリケーションの作成が可能です。また、他の言語で作成されたクラスを継承したり、他の言語で記述されたメソッドを呼び出したりすることができます。また、WebサービスとASP.NETページのビジネスロジックにCOBOLを使用することができ、.NET版でサポートしている言語仕様範囲内であれば、従来のCOBOLソースを流用することも可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

開発環境製品には次の3つのエディションが用意されています。

✓ NetCOBOL Base Edition 開発パッケージ for .NET
COBOLコンパイラを含むVisual Studio 2005 開発環境 を提供
✓ NetCOBOL Standard Edition開発パッケージ for .NET
COBOLコンパイラを含むVisual Studio 2005 開発環境 に 帳票設計ツールを追加
✓ NetCOBOL Enterprise Edition開発パッケージ for .NET
NetCOBOL Standard Edition開発パッケージに、サーバアプリケーション開発を支援する高性能ソート・マージと64ビットアプリ開発環境を追加

運用環境製品

運用環境製品には、動作するOSの違いで「クライアント運用パッケージ」と「サーバ運用パッケージ」の2種類の製品を用意しています。
「クライアント運用パッケージ」は、作成したアプリケーションをクライアントOS側で実行する場合に必要なランタイム製品です。運用アプリケーションが必要とする機能範囲に応じてエディションを選択する必要があります。
なお、NetCOBOL Enterprise Editionは、サーバアプリーケションのみの開発であるため、クライアント運用パッケージは、ありません。

✓ NetCOBOL Base Edition クライアント運用パッケージ for .NET
NetCOBOL Base Edition 開発パッケージで作成したアプリケーションのランタイム製品
✓ NetCOBOL Standard Edition クライアント運用パッケージ for .NET
NetCOBOL Standard Edition 開発パッケージで作成したアプリケーションのランタイム製品

「サーバ運用パッケージ」は、Webアプリケーション、Webサービスアプリケーションの実行や、アプリケーションをサーバOS上で実行する場合に必要なランタイム製品です。運用アプリケーションが必要とする機能範囲に応じてエディションを選択する必要があります。

✓ NetCOBOL Base Edition サーバ運用パッケージ for .NET
NetCOBOL Base Edition 開発パッケージで作成したアプリケーションのランタイム製品

All Rights Reserved, Copyright FUJITSU LIMITED 2006

✓ NetCOBOL Standard Edition サーバ運用パッケージ for .NET
NetCOBOL Standard Edition 開発パッケージで作成したアプリケーションのランタイム製品

✓ NetCOBOL Enterprise Edition サーバ運用パッケージ for .NET
NetCOBOL Enterprise Edition 開発パッケージで作成したアプリケーションのランタイム製品

1.2.4 NetCOBOL関連製品

NetCOBOLシリーズには、開発の支援や、様々な運用を実現するために多くの関連製品が提供されています。

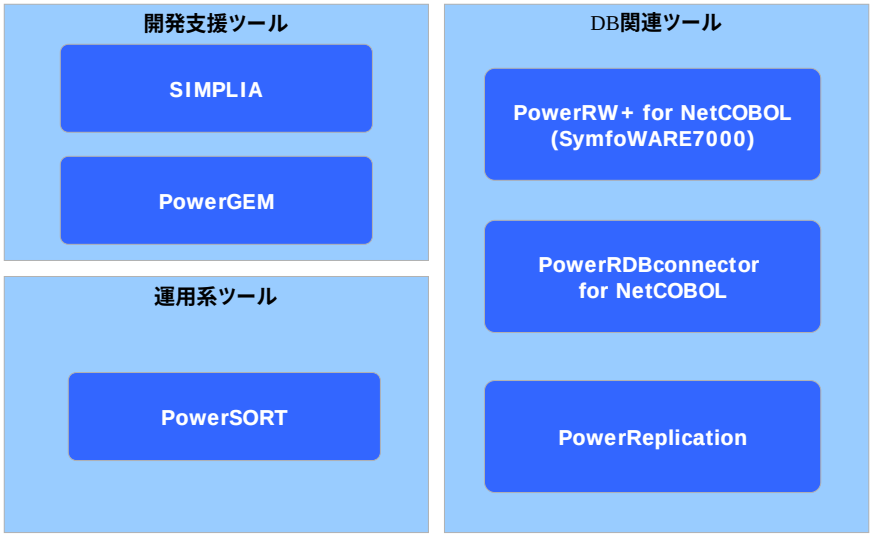


図1.2.4 NetCOBOL関連製品

SIMPLIA

テストデータや各種仕様書の作成などソフトウェアの開発・保守作業の効率化に役立つツール群です。NetCOBOL Professional Edition for WindowsにはSIMPLIAが添付されており、プログラム品質や生産性、保守性の大幅な向上が可能です(NetCOBOL for .NETには添付されていません)。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

PowerGEM

クライアント／サーバ環境で、ソース、定義体、文書など開発資産の修正履歴管理、版数管理を一括して簡単に行うことができます。また、修正中の開発資産に対し、排他制御を行うため、チームでの開発において、品質と開発／保守効率の向上を図ることができます。
NetCOBOL Professional Edition for Windowsには、PowerGEM Plusが添付されています(NetCOBOL for .NETには添付されていません)。

PowerSORT

NetCOBOL単体の機能で、ファイルのソート・マージ処理が行えますが、PowerSORTを利用することで、より大量のデータを、高度なソート技法により短時間で処理できるようになります。コマンド及びGUIで処理を行うことができます。

PowerRW+ for NetCOBOL

PowerRW+ for NetCOBOLは、NetCOBOLの入出力文(READ/WRITE)でアクセスできる信頼性の高いファイルアクセス機能とプログラム間非同期通信機能を提供します。(マルチスレッドに対応していないため、ASP.NETでは利用できません)

PowerRDBconnector for NetCOBOL

NetCOBOLの入出力文(READ/WRITE)を使用したアプリケーションからSQL Serverにファイルアクセスするための製品です。SQL文に書き直すことなく既存のCOBOLアプリケーションを有効活用します。

PowerReplication

PowerRW+の更新データを、異なるサーバ上にあるデータベースに反映することにより、マスタファイルと複製ファイル間のデータを同期する機能を提供します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.3 NetCOBOL for .NETで開発できるアプリケーション

NetCOBOL for .NETでは5つのタイプのアプリケーションを開発できます。それぞれのアプリケーションの形態について説明します。

1.3.1 コンソールアプリケーション

NetCOBOL for .NETのコンソールアプリケーションはNetCOBOL(従来のCOBOL)アプリケーションに最も近い形式 です。

図1.3.1 コンソールアプリケーションの開発イメージ

COBOLの小入出力機能(DISPLAY文やACCEPT文)でコンソールウィンドウから文字を受け取ったり、書き込んだりすることができます。バッチ処理のプログラムであれば、既存のCOBOL資産をそのまま移植することも可能です。上図は、行順ファイルから読み込んだレコードを表示するサンプルプログラムの編集画面と実行画面のイメージです。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.3.2 Windowsアプリケーション

NetCOBOL for .NETのWindowsアプリケーションは、従来のPowerCOBOLアプリケーションに最も近い形式です。

図1.3.2 Windowsアプリケーションの開発イメージ

Windowsフォームと呼ばれるフレームワークを使用して画面のレイアウトを設計します。プログラミングや画面処理において、コンソールアプリケーションやWebアプリケーションと比較して、自由度が高いアプリケーションを開発することができます。図1.3.2は、スケジュールを登録し、一覧を表示するサンプルアプリケーションの編集画面と実行画面のイメージです。

1.3.3 Webアプリケーション

NetCOBOL for .NETのWebアプリケーションとは、従来のCOBOLやNetCOBOLシリーズにはないアプリケーションの形式で、.NET Frameworkの主要な機能であるASP.NETの技術を利用して開発します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

図 1.3.3 Webアプリケーションの開発イメージ

HTMLなどの知識がなくても、GUIでWeb画面の作成が可能で、プログラムから画面へのアクセスにはオブジェクトCOBOLの文法を利用します。Web アプリケーションの実行にはWWWサーバとして、Microsoft Internet Information Service(IIS)が必要です。

1.3.4 Webサービス

NetCOBOL for .NETのXML Webサービスも、Webアプリケーションと同様に従来のCOBOLやNetCOBOLシリーズにはない新しい形式のアプリケーションでASP.NETの技術を利用しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

図1.3.4 Webサービスの開発イメージ

XML Webサービスでは、XML形式でデータ交換を行いプロトコルとしてSOAP(Simple Object Access Protocol）を使用しています。XML Webサービスは、Webアプリケーションなどのバックグラウンドとして、ビジネスロジック層を受けもち、画面を持ちません。

1.3.5 クラスライブラリ

NetCOBOL for .NETのクラスライブラリは、.NET Framework上で共通に使用できる形式(マネージコード)でライブラリ化したものです。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

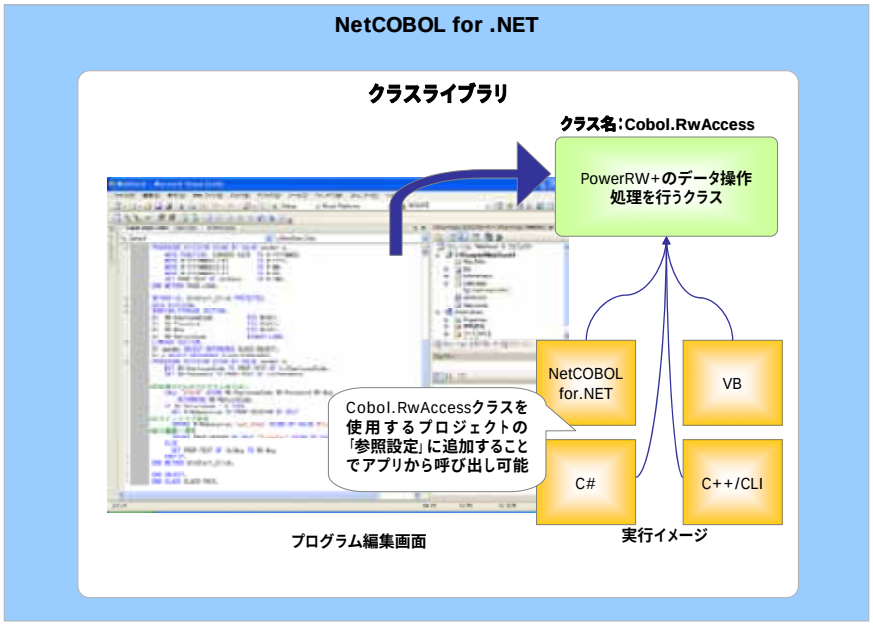


図1.3.5 クラスライブラリの開発イメージ

従来のCOBOLではDLLに相当します。作成したクラスライブラリは.NET Frameworkで動作するアプリケーションであれば、言語を問わずに同じ方法で利用できます。なお、汎用的な処理をあらかじめクラスライブラリとして用意しておくことで、プログラムの再利用性が高まり、プログラミングの効率化を図ることができます。

1.3.6 SQL CLRストアドプロシージャ

これまでストアドプロシージャはデータベース毎に専用の言語で記述する必要がありました。

SQL CLRでは、.NET Frameworkの共通言語ランタイム(CLR)のコンポーネントがSQL Server 2005に統合されたことにより、マネージコードによるデータベースオブジェクト(ストアドプロシージャやユーザ定義関数など)が利用可能になりました。

これによりCOBOLなど.NET Frameworkで利用可能な言語でストアドプロシージャの記述が可能となりました。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

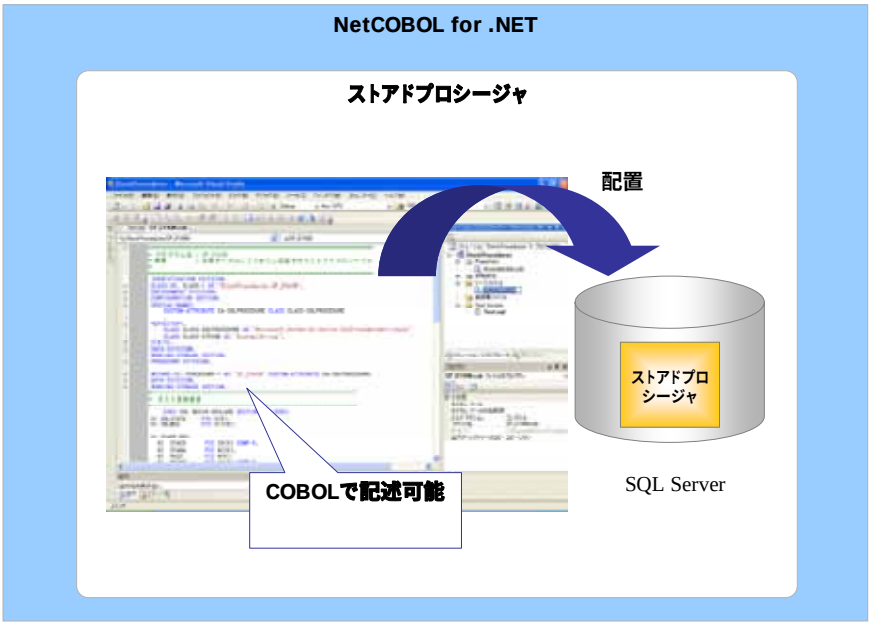


図1.3.6 ストアドプロシージャの開発イメージ

ストアドプロシージャを利用することでより高速なデータベースアクセスが可能となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.4 NetCOBOL for .NETの システム構築パターン

1.4.1 クライアントサーバ

一般的にクライアントサーバとは、サーバ上にデータを配置し、複数のクライアントプログラムからサーバ上のデータを共通に利用する形式を指します。

NetCOBOL for .NETでは、.NET Frameworkのデータベース連携機能であるADO.NETクラスライブラリやCOBOL自身の機能である埋込みSQLを使用して、クラサバシステムを実現することができます。

また、画面には.NET FrameworkのWindowsフォームを利用し、多機能なWindows画面を構築できます。

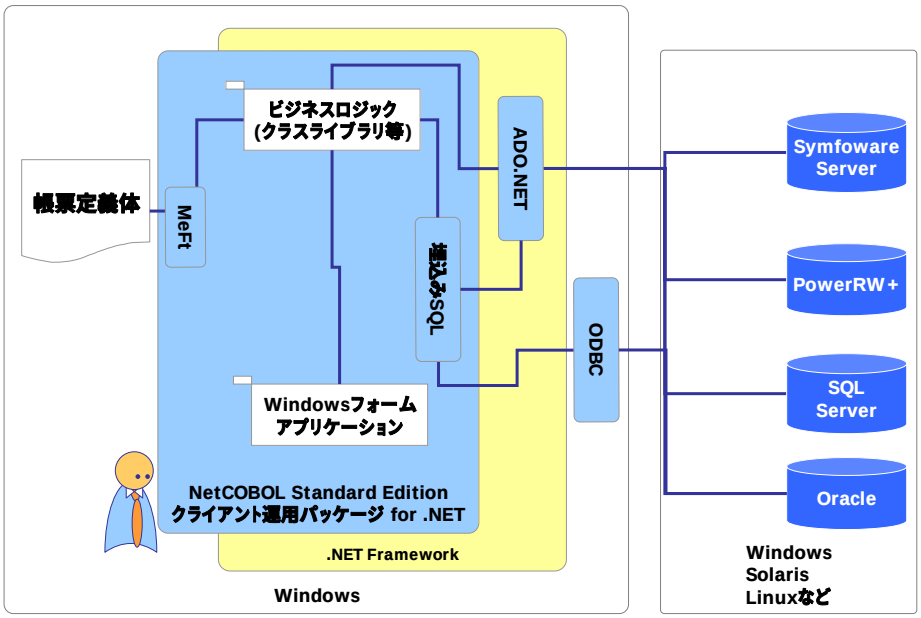


図1.4.1. クライアントサーバ利用時の構成

ADO.NETはデータベースの種類によらない一貫したプログラミングモデルを提供しているため、同じ手続きで様々なデータベースに接続できます。NetCOBOL fot .NETでADO.NET接続する場合、ADO.NETクラスライブラリを利用する方法と埋込みSQLを利用する方法があります。

埋込みSQLは、従来のCOBOLから利用されている機能であるため、従来のCOBOLと高い互換性があります。また、埋込みSQLでは

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ADO.NET接続とODBC接続をサポートしています。

(1) ADO.NETクラスライブラリ

長所

✓ .NET Frameworkのクラスを直接使用するため、多彩な機能を利用できる

✓ 他言語、他コンポーネントと連携しやすい

短所

✓ データアクセスに関する手続きの互換性が無い

(2) 埋込みSQL

長所

✓ 従来のCOBOLプログラムと互換性が高い

短所

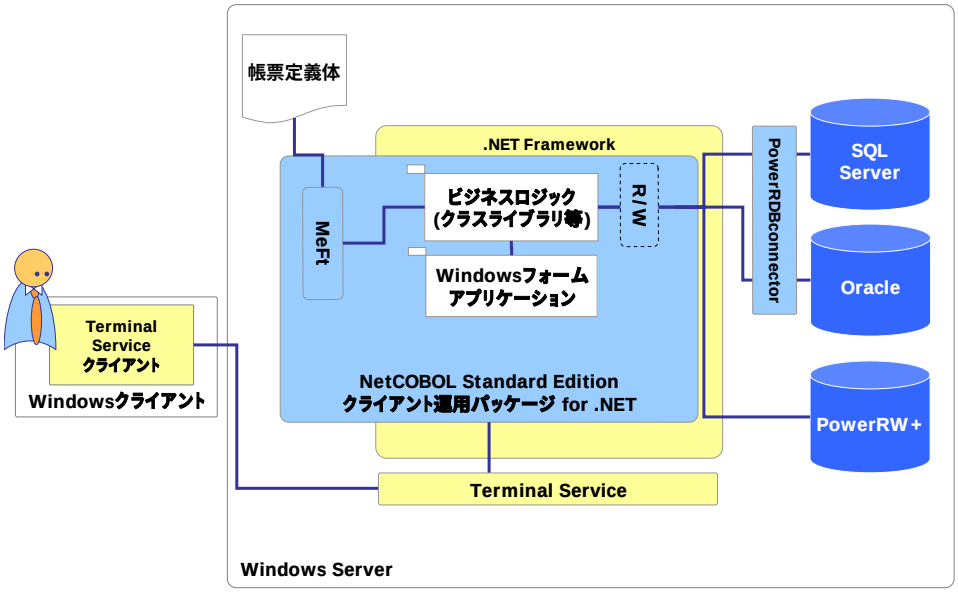
✓ ADO.NETクラスライブラリに比較すると、機能が少ない

1.4.2 Terminal Service

オフコンでの運用イメージに最も近い運用形態がTerminal Serviceです。プログラム資産はスタンドアロンと同様の形式で作成しますが、WindowsのTerminal Serviceを使用し、サーバ上で実行されるプログラムをクライアントで実行するイメージで運用することができます。

このシステムパターンでは、データベースアクセスの方法として、READ/WRITE形式によるアクセスが可能です(ADO.NETや埋め込みSQLも可能)。具体的には、PowerRDBconnectorを使用したSQL Server と Oracleの利用や、PowerRW+ for NetCOBOLの利用が可能です。これらの製品では、COBOLから入出力インタフェースを使用し、通常のファイルのイメージでデータベースにアクセスすることができ、従来のCOBOLとの互換性も高いといえます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



※R/W READ/WRITEインタフェース

図1.4.2 Terminal Service利用時の構成

長所

✓ 運用のイメージがオフコンの場合と類似している

✓ クライアント側にプログラム資源を配置する必要がなく、メンテナンスが容易

短所

✓ クライアント側のプリンタに出力する場合はLPTポートのみしかサポートされない(Citrix MetaFrameではネットワークプリンタに対応)

✓ 画面やオペレーションがネットワークで伝達されるため、回線速度によってはレスポンスが悪くなる

✓ Windows2000の場合、色数が最大256色と限定される(Windows Server 2003でフルカラー対応)

✓ 接続台数が多くなる場合、サーバに多くのリソース(メモリ、CPUなど)が必要となる

1.4.3 Webシステム(.NET)

NetCOBOL for .NETのWebシステムでは、画面に.NET FrameworkのWebフォームを利用します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

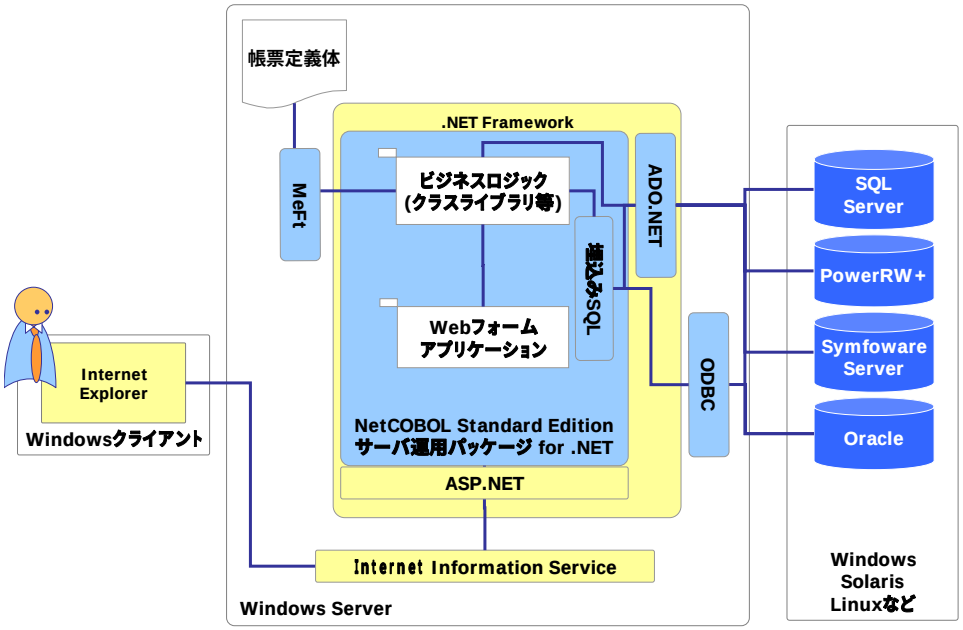


図1.4.3 Webシステム利用時の構成

長所

✓ 表示部分とビジネスロジックを明確に分離できる

✓ VBやC#などの.NET Framework上の他の開発言語との連携が容易

✓ クライアントがブラウザのみで良い

短所

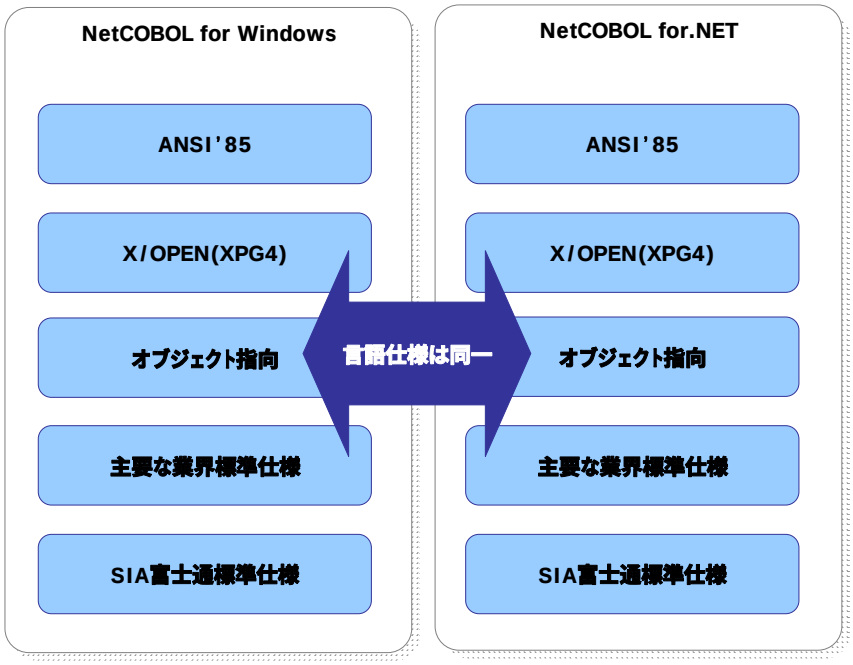
✓ クラスやオブジェクトの知識が必要

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.5 従来のCOBOLと NetCOBOL for .NETの共通点と相違点

1.5.1 言語仕様は同一

NetCOBOL for .NETは従来のCOBOL(NetCOBOL for Windows)と同様、国際規格(ANSI ’85、X/OPEN(XPG4)に準拠しているほか、国際規格2002を先行したオブジェクト指向プログラミング機能や業界主要仕様、富士通標準仕様をサポートしています。



1.5.1 従来のCOBOLとNetCOBOL for .NETの共通点（言語仕様）

NetCOBOL for Windowsでは、手続き型COBOL文法が一般的で、オブジェクト指向プログラミングやクラスを利用したプログラミングは影をひそめがちでしたが、NetCOBOL for .NETではオブジェクト指向プログラミングやクラスの利用が比較的多くなります。

しかし、ソースレベルで互換性があるため、手続き型COBOL文法を利用した既存COBOL資産を.NET Framework上に容易に移行することが可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.5.2 従来COBOLとNetCOBOL for .NETの共通点と相違点 (プレリンパイラ未対応)

NetCOBOL for Windowsでデータベースにアクセスする場合は、ODBC経由、またはデータベースから提供されているプリコンパイラを使用していました。

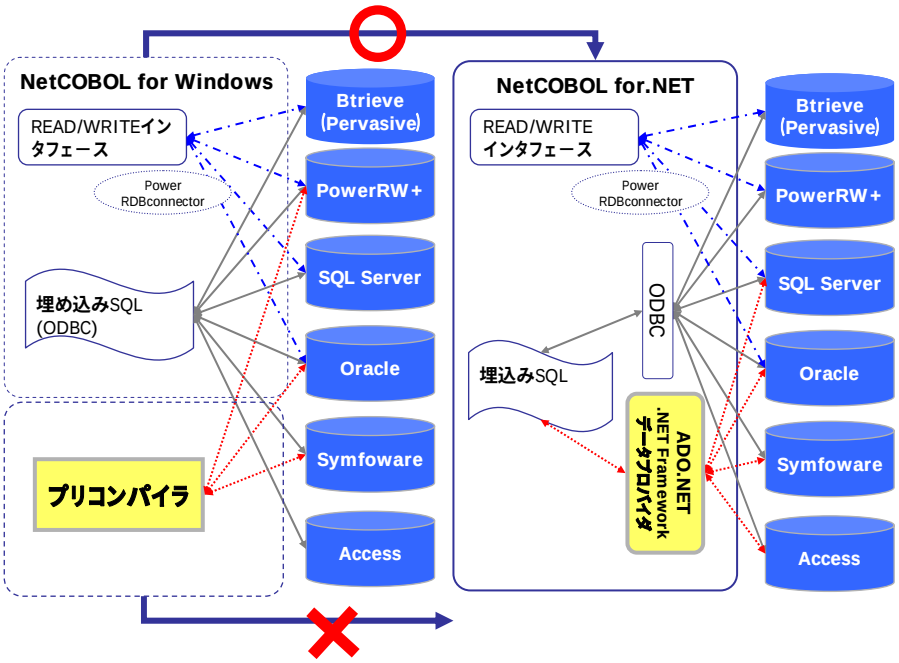


図 1.5.2 従来のCOBOLとNetCOBOL for .NETの相違点(プリコンパイラ)

NetCOBOL for .NETでは、埋込みSQLによるデータベースアクセスやREAD/WRITEインタフェースによるデータアクセスは従来と同様に使用できます(一部制限あり)。埋込みSQLでは、.NET Framework のデータアクセス機能であるADO.NET接続、従来と同様のODBC接続が可能です。しかし、プリコンパイラはNetCOBOL for .NETに対応していません。

ADO.NETでは.NET Frameworkデータプロバイダを使用してデータベースにアクセスします。データベースによってはDB専用のデータプロバイダが提供されており、DB専用プロバイダでは、ODBCより高速なデータ操作が可能になります。その他にも、OLE DB .NETデータプロバイダやODBC.NETデータプロバイダが提供されています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

1.5.3 画面定義体未対応

従来のCOBOLで作成する画面処理のアプリケーションではFORMで作成した画面定義体をフロントとして、MeFtの制御配下で実行していましたが、NetCOBOL for .NETでは画面定義体(FORM)を利用することはできません。

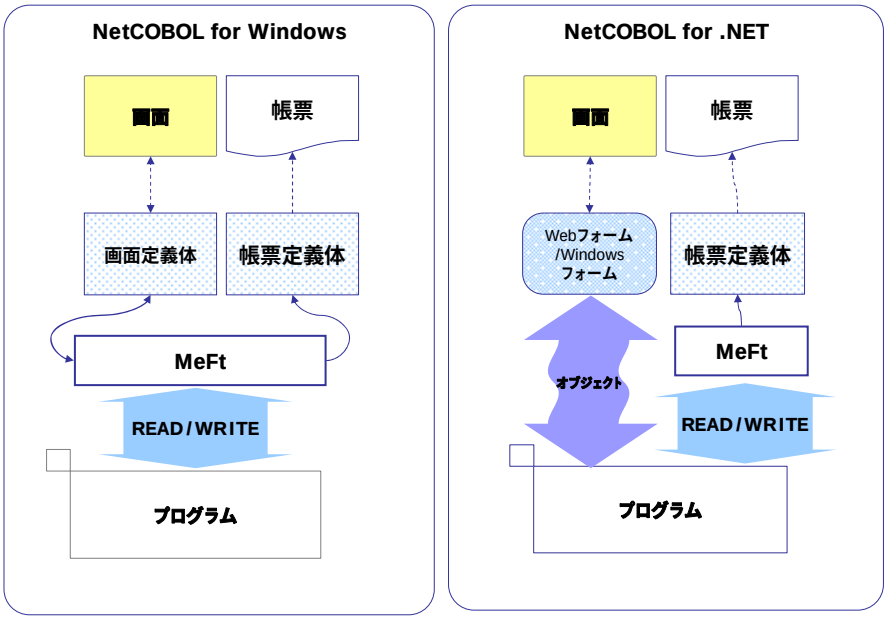


図1.5.3 従来のCOBOLとNetCOBOL for .NETの相違点(画面定義体)

画面はWindowsフォームかWebフォームを利用することになります。これらの機能では、今までのCOBOLアプリケーションでは実現できなかった、よりWindowsライクな画面設計が可能になります。

しかし、画面に対しての入出力に、READ/WRITE文で容易に操作が行えていた従来のCOBOLプログラムに対し、NetCOBOL for .NETではオブジェクト指向による操作になるため、比較的高度なプログラミングスキルが必要になります。なお、帳票定義体はNetCOBOL for .NETでも従来の方式で利用可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第2章 NetCOBOL for .NETを使う

2.1 NetCOBOL for .NETの開発資産
2.2 NetCOBOL for .NETのプログラム開発画面
2.3 プログラム開発の流れ
2.4 NetCOBOL for .NETによる手続き型プログラム作成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.1 NetCOBOL for .NETの開発資産

2.1.1 プログラム開発の単位

NetCOBOL for .NETアプリケーションはVisual Studioの開発環境を用いて開発します。Visual Studioでは、開発資産はソリューションとプロジェクトという2つの単位で管理されます。

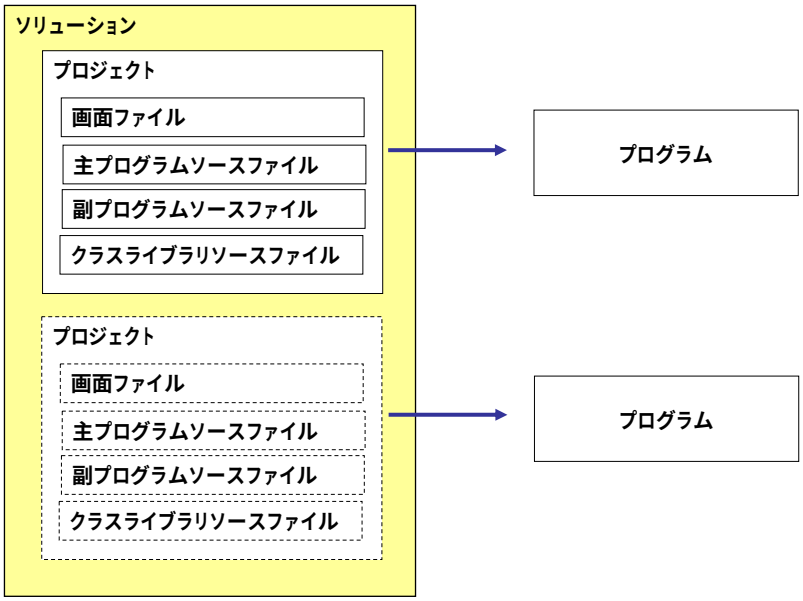


図2.1.1 プログラム開発の単位

ソリューション

プログラムのもっとも大きな管理単位です。ソリューションには一つ以上のプロジェクトが含まれます。関連するプロジェクトをまとめてソリューションとして管理します。通常はソリューションにはプロジェクトしか含まれませんが、ソリューションに共通でプロジェクトに含まれないファイルをソリューション項目として追加することもできます。

プロジェクト

個々のプログラムの管理単位です。NetCOBOL for .NETには、5種類

All Rights Reserved, Copyright FUJITSU LIMITED 2006

のプロジェクトおよび2種類のWebサイトのテンプレート(雛形)が用意されていて、プロジェクトの作成時にプログラムの目的に応じたプロジェクトのテンプレートを選びます。

プロジェクト／Webサイトを構成するファイル

NetCOBOL for .NETのプロジェクト／Webサイトを構成するファイルには、ソースファイルや登録集があります。プロジェクトを構成するファイルの種類は、アプリケーションの形式によって異なります。

2.1.2 ファイルの種類と機能

NetCOBOL for .NETでの開発時、プロジェクト／Webサイトに登録する(登録される)主なファイルについて説明します。

区分	拡張子	機能/役割
COBOLソースファイル	*.cob、*.cbl、*.cobol,*.cobx	プログラム定義
		クラス定義
		インタフェース定義
		デリゲート定義
		ENUM定義
		プログラム原型定義
Web関連ファイル	*.aspx、*.asmx、*.asax、*.ascx	Web フォーム
		Web サービス
		Web ユーザー コントロール
リソースファイル	*.resx	アセンブリリソースファイル
COBOL固有ファイル	*.pmd、*.pxd、*.smd	帳票定義体ファイル
	*.ffd	ファイル定義体ファイル(*.ffd)
	*.cbr(変更可能)	実行用の初期化ファイル
	任意	エントリ情報ファイル
	任意	印刷情報ファイル
	任意	ODBC情報ファイル

図2.1.2 ファイルの種類と機能

COBOLソースファイル

COBOL言語で記述されたプログラムのソースファイルです。プログラム定義やクラス定義を記述します。プログラム(主、副)やクラスなどで利用します。拡張子は、*.cob、*.cbl、*.cobol、*.cobxの4種類

All Rights Reserved, Copyright FUJITSU LIMITED 2006

が使えます。拡張子ごとに制限はありません。一般的に*.cobを利用します(*.cblは登録集として使用するケースが多くなっています)。また、.cobxはデザイナで使用します。

Web関連ファイル

WebフォームやXML Webサービスなどの機能で使用されるファイルです。主にGUIで編集します。Web関連ファイルは、.NET Frameworkで共通に利用されるファイルであり、NetCOBOL for .NET独自のファイルではありません。

リソースファイル

さまざまなデータをアプリケーションとは別に格納しているのがリソースファイルです。リソースファイルを変更してもアプリケーション全体を再コンパイルする必要がありません。リソースファイルは.NET Frameworkの機能として提供されています。

COBOL固有ファイル

.NET Frameworkには用意されておらず、NetCOBOL for .NETで利用するファイルです。通常はプロジェクトには追加せずに利用しますが、目的に応じてプロジェクトに追加することもできます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.2 NetCOBOL for .NETのプログラム開発画面

2.2.1 プログラム開発画面(画面デザイン時)

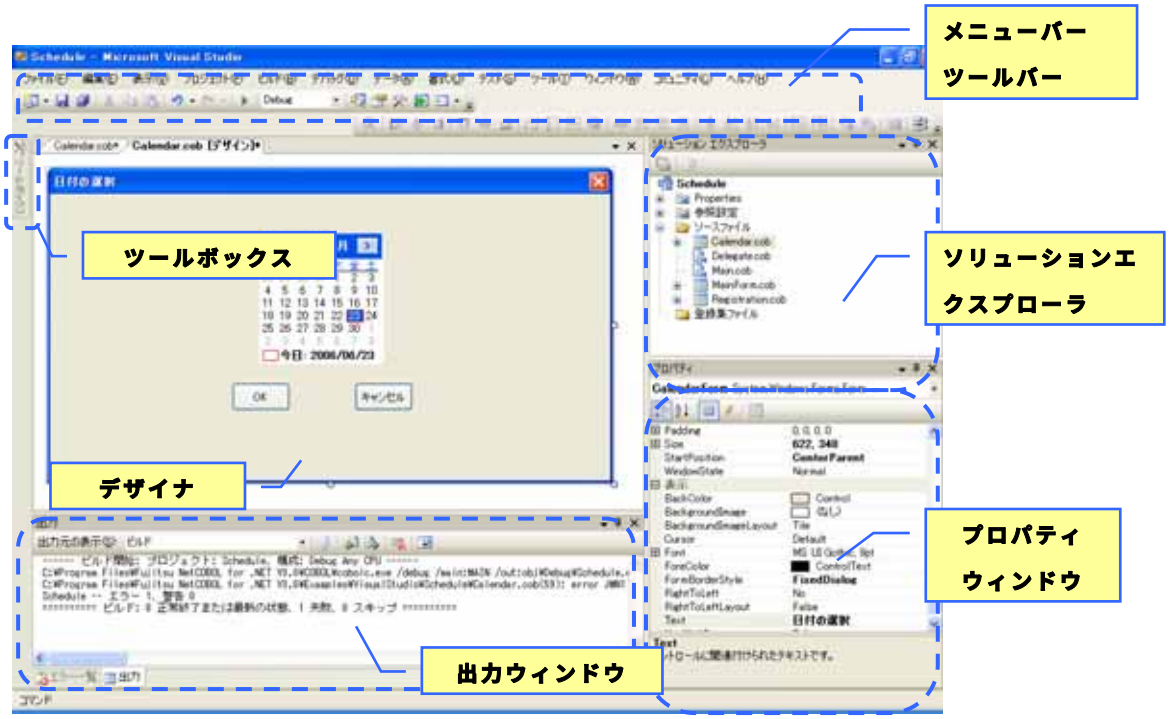


図2.2.1 プログラム開発画面（画面デザイン時）

ソリューションエクスプローラ

ソリューションやプロジェクトで管理されるファイルの一覧を階層的に表示しています。ソリューションエクスプローラに表示されたファイルをクリックし、目的のファイルの編集を行います。

メニューバー/ツールバー

開発を行うために必要なコマンドが登録されています。

デザイナ

画面をデザインするためのウィンドウです。Windowsフォームアプリケーション作成時には「Windows フォーム デザイナ」、Webフォームアプリケーション作成時には「Web フォーム デザイナ」となります(画面例はWindowsフォームデザイナです)。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ツールボックス

デザイナに表示されたプログラムの画面に貼り付けるツールが登録されています。マウスポインタでポイントすると展開します。

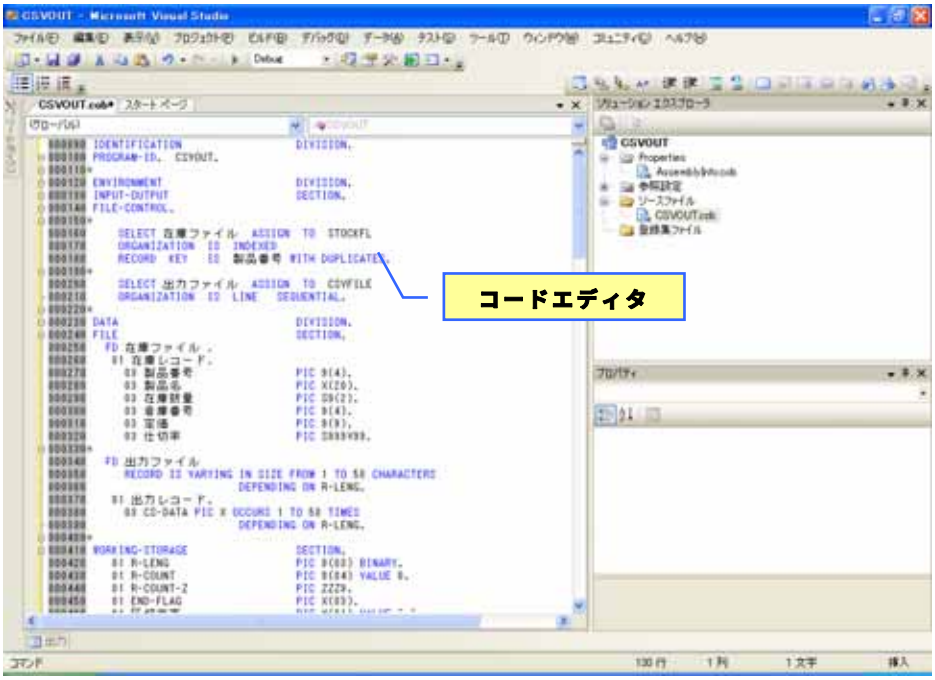
プロパティウィンドウ

ソリューション、プロジェクト、画面など各項目のプロパティや設定を行うためのウィンドウです。

出力ウィンドウ

ビルドの結果やデバッグ出力が表示されるウィンドウです。

2.2.2 プログラム開発画面(コード編集時)



2.2.2 プログラム開発画面(コード編集時)

コードエディタ

COBOLの手続きを記述するエディタです。ソリューションエクスプローラからCOBOLソースを選択したり、デザイナ上のコントロールをダブルクリックしたりすると表示されます。

初期設定では、予約語は「青」、コメント行は「緑」、文字定数は「茶」で表示されます。また、IntelliSense機能により、選択したクラスや集団項目に対する有効なメンバやデータを表示させ、表示され

All Rights Reserved, Copyright FUJITSU LIMITED 2006

た一覧からメンバやデータを選択してプログラムに挿入したり、データの型情報を表示させたりすることができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.3 プログラム開発の流れ

NetCOBOL for .NETでプログラムを新規に作成する場合の大まかな流れを説明します。

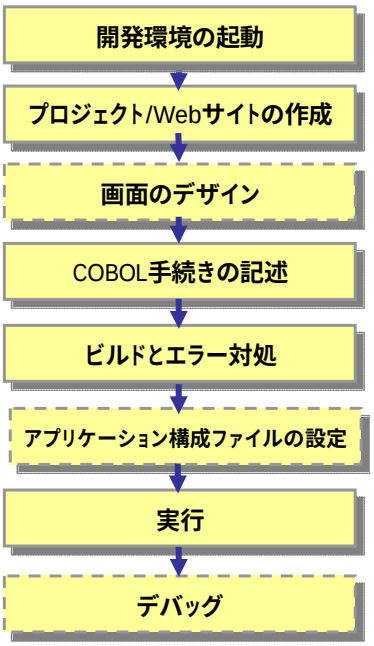


図2.3 プログラム開発の流れ

開発環境の起動

プログラムグループから「NetCOBOL 開発パッケージ for .NET V3.0」を選択し、「NetCOBOL for .NET」を起動します。Visual Studioを別にインストールしている場合には、「Visual Studio 2005」を選択しても問題ありません。

プロジェクト／Webサイトの作成

目的に応じて、プロジェクトまたはWebサイトを作成します。さらにテンプレートから作成するプロジェクトやWebサイトの種類を選択します。

画面のデザイン

プロジェクトやWebサイトで[Windowsアプリケーション]や[ASP.NET Webサイト]を作成する場合、画面を持つアプリケーションとなります。画面を持つアプリケーションの場合、画面に関連する手

All Rights Reserved, Copyright FUJITSU LIMITED 2006

続きを作成する前に、デザイナを使用して画面デザインの作成を行います(クラスライブラリやサブプログラムの作成は画面デザインの前に行ってもかまいません。

COBOL手続きの記述

プロジェクトにCOBOLプログラムやクラスを追加するなどして、COBOL言語によるプログラム手続きを記述します。画面のコントロール(部品)に依存する手続きは、画面に貼り付けられたコントロールをダブルクリックし、コントロールのイベントに目的の処理をCOBOL言語で記述します。

ビルドとエラー対処

プログラムはソリューション単位やプロジェクト単位でビルドすることができます。ビルドは[ビルド]メニューなどから行いますが、ビルドの対象により方法が若干異なります。ビルド後は、文法エラーへの対処を行います。また、文法エラーは「タスク一覧」ウィンドウに表示され、エラー表示行をクリックすると、プログラムのエラー箇所にジャンプします。

アプリケーション構成ファイルの設定

[COBOL実行環境設定ユーティリティ]を起動して表示されるツールで実行環境変数やファイルの割り当てなどの定義を行います。設定した情報はXML形式のファイルに出力されます。
アプリケーション構成ファイルは、Webアプリケーションと、Webアプリケーション以外でファイル名が異なります。
Webアプリケーション以外では、App.configというファイル名で作成され、ビルド時に「exe名.exe.config」で出力フォルダにコピーされます。Webアプリケーションでは「Web.config」となります。
従来の実行用初期化ファイル(COBOL85.CBR)やエントリ情報ファイル、ODBC情報ファイルのインポートも可能です。

実行

[デバッグ]-[開始]やF5キーの押下により、開発環境からプログラムを起動することができます。また、COBOL実行環境ファイルの編集を必要とするプログラムの場合、[COBOL実行環境設定ユーティリティ]により、実行環境ファイルを作成してから実行を行います。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

デバッグ

論理エラーなどが発生した場合のデバッグ方法として、プログラムをデバッグモード(初期値)でビルドすることで、ステップ実行や式のウォッチなどを行うことができます。デバッグ機能のためのフレームワークが.NET Frameworkで提供されていることから、各言語ほぼ共通の操作性でデバッグを行うことができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.4 NetCOBOL for .NETによる手続き型プログラム作成

2.4.1 プログラム概要

ここでは、NetCOBOL for .NET V2.1を使用して、COBOLの索引ファイルを読み込んで、CSV形式のファイルに変換するコンソールアプリケーションの開発手順について説明します。下図はアプリケーションの実行イメージ図です。

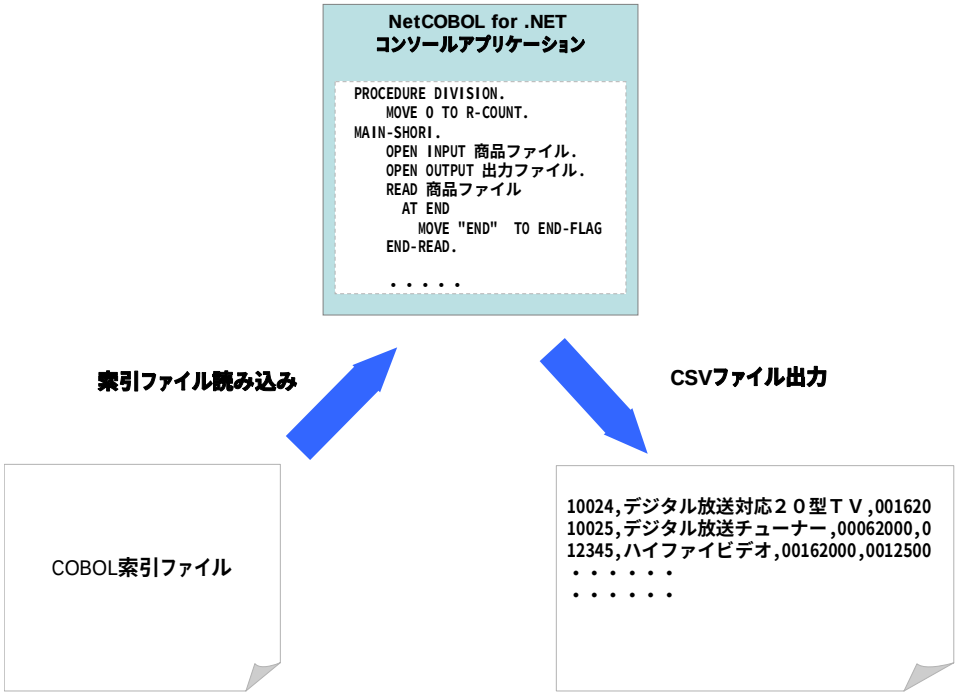


図 2.4.1 コンソールアプリケーションの実行イメージ

それでは、実際に次の手順に従い、アプリケーションを作成していきましょう。

2.4.2 開発環境の起動

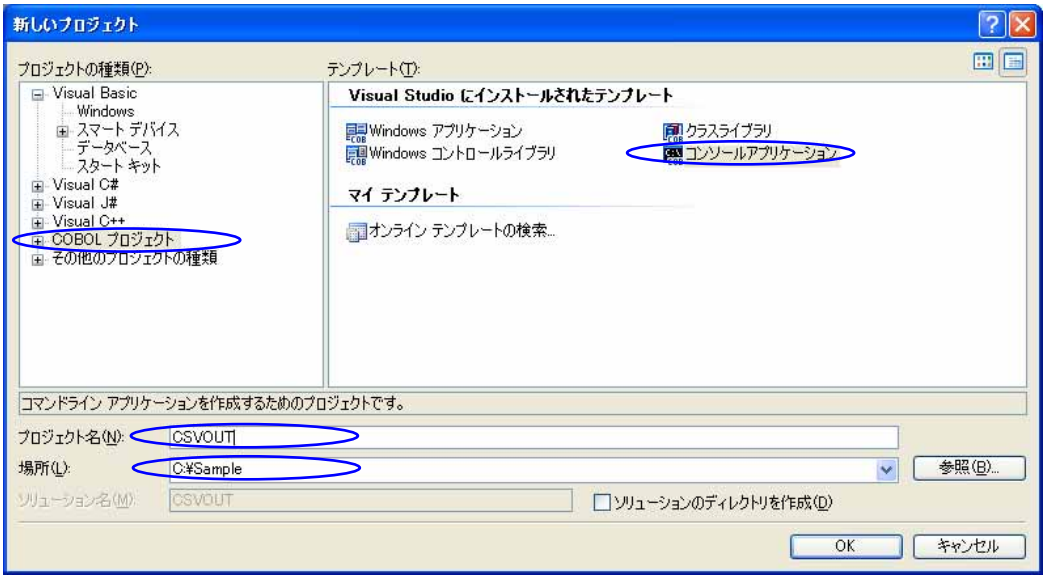
[スタート]メニューの[すべてのプログラム]より「NetCOBOL for .NET」をクリックし、NetCOBOL for .NETの開発環境を起動します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.4.3 プロジェクトの作成

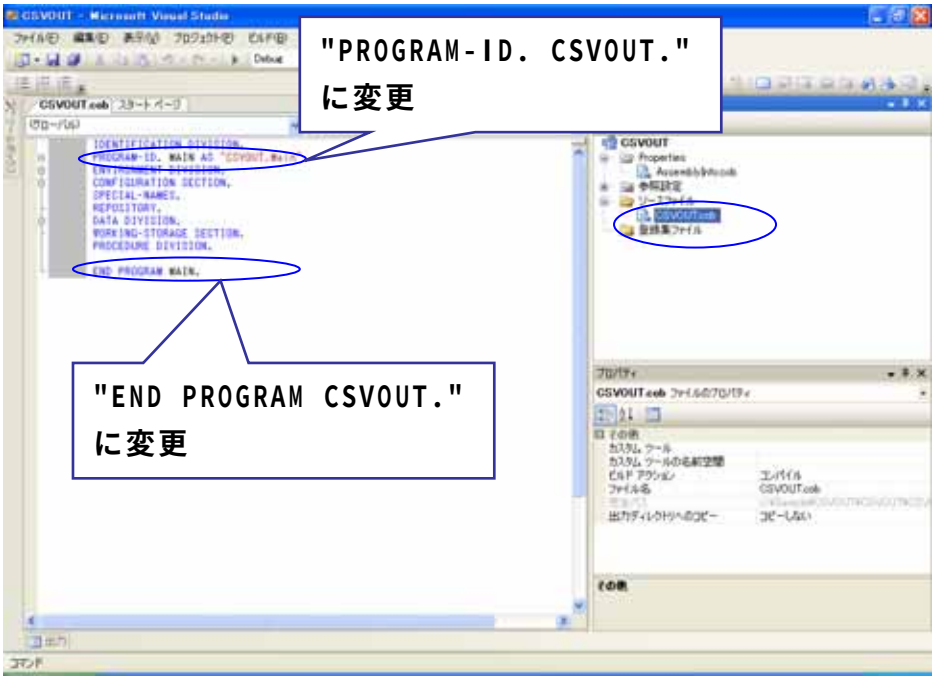
- **手順**1
  起動画面の[ファイル]メニューより[新規作成]-「プロジェクト」を選択します。

- **手順**2
  [プロジェクトの種類]より「COBOLプロジェクト」、[テンプレート]より「コンソールアプリケーション」を選択し、[プロジェクト名]、のテキストボックスに「CSVOUT」などの任意の名称を入力し[場所]のテキストボックスに、プロジェクトの作成フォルダを指定後、[OK]ボタンをクリックします。



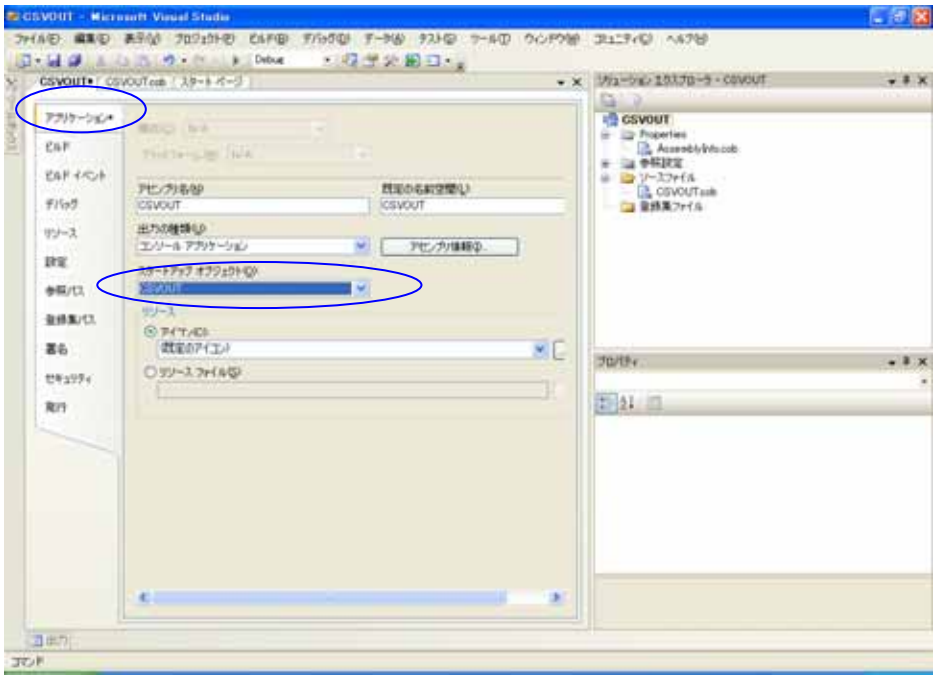
- **手順**3
  手順2で指定したテンプレートに従って、プロジェクトが自動生成されます。ソースファイル名を「CSVOUT.cob」に変更し、雛型で生成されたCOBOLソースのプログラム名（PRGRAM-ID,END PRGRAMの後のプログラム名）を「CSVOUT」に変更します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



- **手順**4

  [ソリューションエクスプローラ]より[プロジェクト]を右クリックし、コンテキストメニューより「プロパティ」をクリックします。
  プロパティから[アプリケーション]タグを選択し、[スタートアップオブジェクト]を「CSVOUT」にします。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

2.4.4 COBOL手続きの編集

CSVOUT.cobに、次のプログラムを入力します。

```
 IDENTIFICATION                    DIVISION.
 PROGRAM-ID.  CSVOUT.
*
 ENVIRONMENT                       DIVISION.
 INPUT-OUTPUT                      SECTION.
 FILE-CONTROL.
*
     SELECT 在庫ファイル  ASSIGN  TO  STOCKFL
     ORGANIZATION  IS  INDEXED
     RECORD  KEY   IS  製品番号 WITH DUPLICATES.
*
     SELECT 出力ファイル  ASSIGN  TO  CSVFILE
     ORGANIZATION  IS  LINE   SEQUENTIAL.
*
 DATA                              DIVISION.
 FILE                              SECTION.
   FD 在庫ファイル .
    01 在庫レコード.
      03 製品番号                 PIC 9(4).
      03 製品名                   PIC X(20).
      03 在庫数量                 PIC S9(2).
      03 倉庫番号                 PIC 9(4).
      03 定価                     PIC 9(9).
      03 仕切率                   PIC S999V99.
*
   FD 出力ファイル
      RECORD IS VARYING IN SIZE FROM 1 TO 50 CHARACTERS
                        DEPENDING ON R-LENG.
    01 出力レコード.
      03 CS-DATA PIC X OCCURS 1 TO 50 TIMES
                        DEPENDING ON R-LENG.
*
 WORKING-STORAGE                   SECTION.
    01 R-LENG                      PIC 9(03) BINARY.
    01 R-COUNT                     PIC 9(04) VALUE 0.
    01 R-COUNT-Z                   PIC ZZZ9.
    01 END-FLAG                    PIC X(03).
    01 区切文字                     PIC X(01) VALUE ",".
    01 位置                         PIC 9(02) VALUE 1.
    01 変換.
      03 仕切率変換                 PIC 999.99.
*
    01 符号２.
      03 符号在庫数量               PIC S9(3) LEADING SEPARATE.
*
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*出力ファイル編集用レコード
    01 CSV-REC.
      03 CSV-DATA1                PIC X(4).
      03 KUGIRIA                  PIC X.
      03 CSV-DATA2.
        05  CSV-DATA21            PIC X OCCURS 1 TO 20 TIMES
                          DEPENDING ON LENG-B.
      03 KUGIRIB                  PIC X.
      03 CSV-DATA3                PIC X(3).
*
      03 KUGIRIC                  PIC X.
      03 CSV-DATA4                PIC X(4).
*
      03 KUGIRID                  PIC X.
      03 CSV-DATA5                PIC X(9).
*
      03 KUGIRIE                  PIC X.
      03 CSV-DATA6                PIC X(7).
*
*出力ファイル各項目長
    01 LENG-A                    PIC 9(03) VALUE 4. *>製品番号
    01 LENG-B                    PIC 9(03) BINARY.  *>製品名
    01 LENG-C                    PIC 9(03) VALUE 3. *>在庫数量
    01 LENG-D                    PIC 9(03) VALUE 4. *>倉庫番号
    01 LENG-E                    PIC 9(03) VALUE 9. *>定価
    01 LENG-F                    PIC 9(03) VALUE 8. *>仕切率
*
*************************************************************
*      データコンバート
*************************************************************
 PROCEDURE DIVISION.
 MAIN-SHORI.
*
      DISPLAY   "*** CSVOUT START ****".
*ファイルのオープン
      OPEN INPUT 在庫ファイル.
      OPEN OUTPUT 出力ファイル.
*ファイルの読込み
      READ 在庫ファイル
         AT END
           MOVE "END"             TO  END-FLAG
      END-READ.
*
      PERFORM DATA-SHORI UNTIL END-FLAG = "END".
*ファイルのクローズ
      CLOSE 在庫ファイル.
      CLOSE 出力ファイル.
      MOVE R-COUNT                TO  R-COUNT-Z.
      DISPLAY R-COUNT-Z "件のデータを変換しました"
      DISPLAY   "*** CSVOUT END   ****".
      EXIT  PROGRAM.
*
*データコンバート(COBOL索引ファイル→CSV)
 DATA-SHORI.
      MOVE 製品番号               TO  CSV-DATA1.
*
      COMPUTE LENG-B = FUNCTION STORED-CHAR-LENGTH(製品名).
      MOVE 製品名                 TO  CSV-DATA2.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

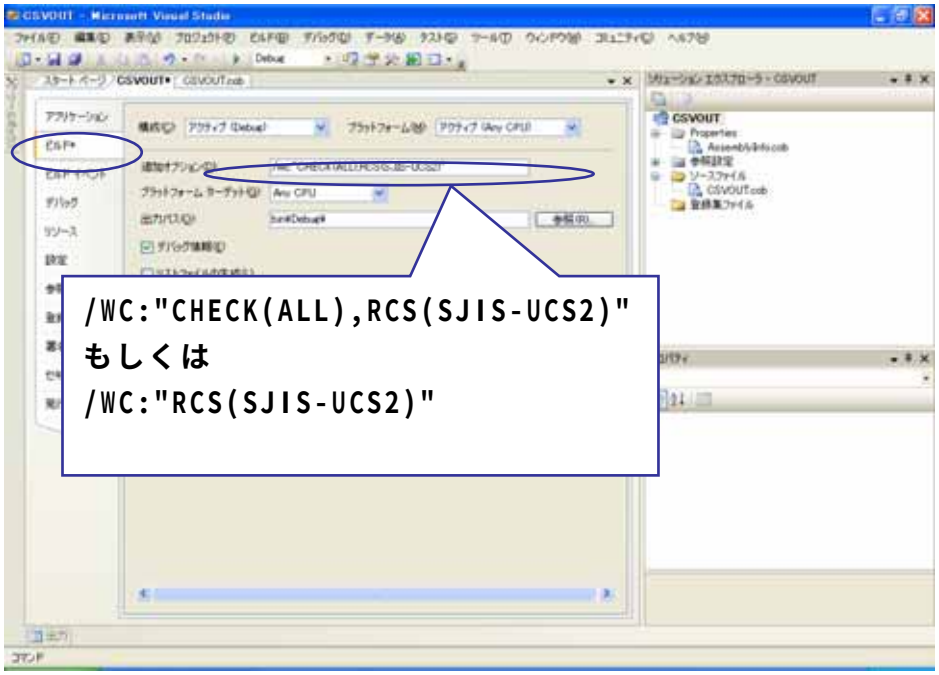
```
*
      MOVE 在庫数量                TO  符号在庫数量.
      MOVE 符号２                  TO  CSV-DATA3.
*
      MOVE 倉庫番号                TO  CSV-DATA4.
*
      MOVE 定価                    TO  CSV-DATA5.
*
      MOVE 仕切率                  TO  仕切率変換.
      MOVE 変換                    TO  CSV-DATA6.
*
      MOVE ","                     TO  KUGIRIA
                                       KUGIRIB
                                       KUGIRIC
                                       KUGIRID
                                       KUGIRIE.
*
      COMPUTE R-LENG = 5 + LENG-A + LENG-B + LENG-C + LENG-D
+ LENG-E + LENG-F.
*
*ファイルの書出し
      WRITE 出力レコード   FROM  CSV-REC.
      COMPUTE R-COUNT = R-COUNT + 1.
*ファイルの読込み
      READ  在庫ファイル  AT END MOVE "END"  TO END-FLAG
      END-READ.
```

2.4.5 翻訳オプションの指定

プログラムの入力が完了したら、翻訳オプションを指定します。

[ソリューションエクスプローラ]ウィンドウよりプロジェクト名(CSVOUT)を右クリックし、コンテキストメニューより「プロパティ」を選択し、プロパティを表示します。「ビルド」タブを選択し、翻訳オプションに「RCS(SJIS-UCS2)」を追加します。なお、Debugモードでは、すでにチェックオプション(CHECK)が設定されていますので、その後に”,”を追加し複数の翻訳オプションを設定します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

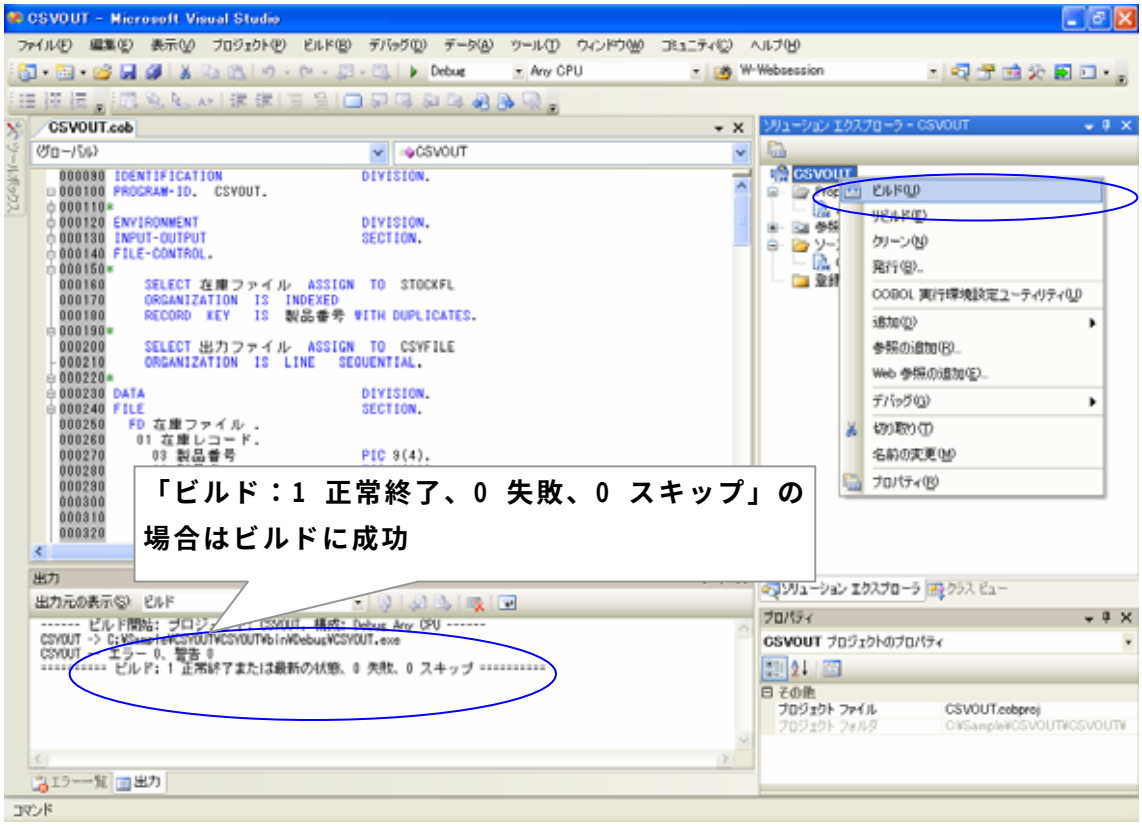


2.4.6 ビルドエラー対処

- **手順**1

  コードの入力が完了したら、ビルドを行います。[ソリューションエクスプローラ]ウィンドウよりプロジェクト名(CSVOUT)を右クリックし、「ビルド」を選択しビルドを実行します。

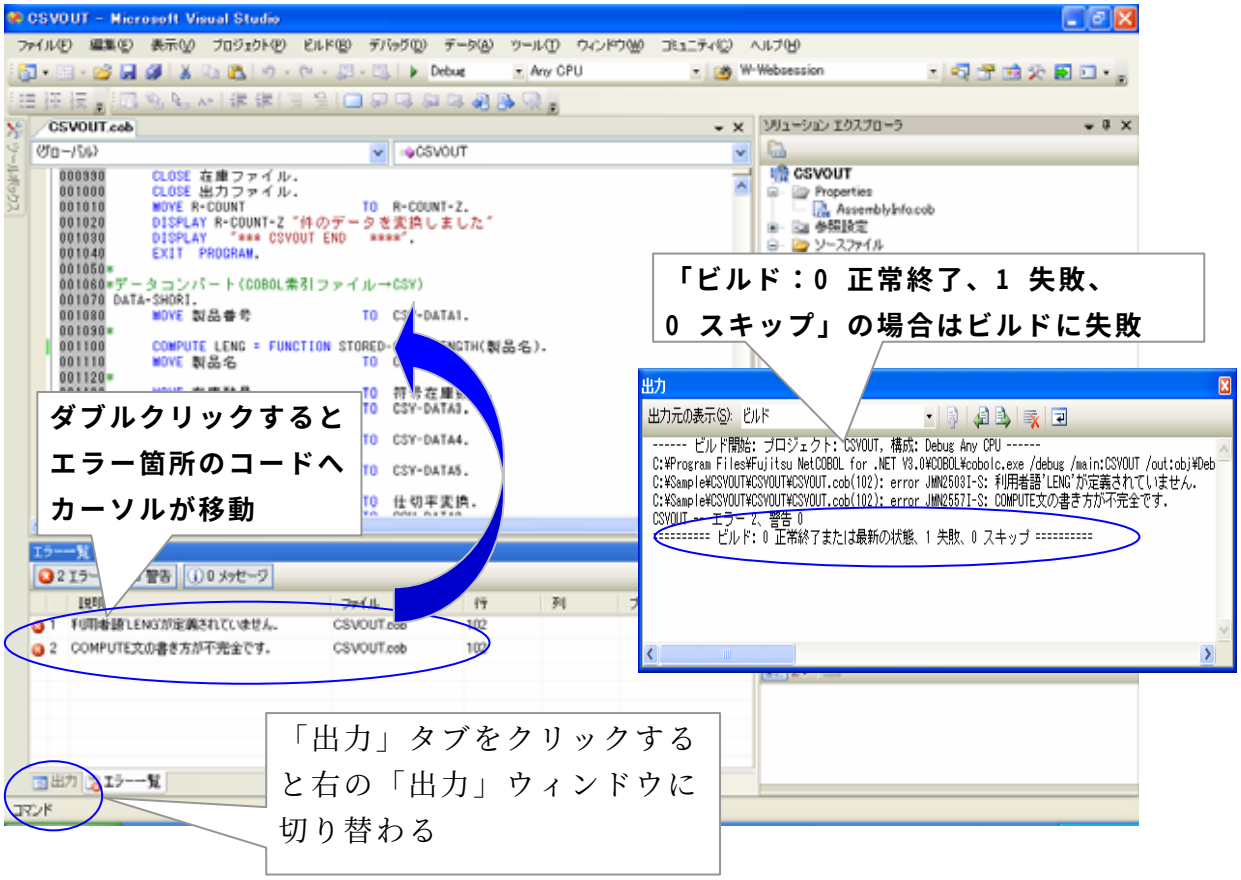
All Rights Reserved, Copyright FUJITSU LIMITED 2006



ビルド成功イメージ

[出力ウィンドウ]にビルド結果が表示されます。上図のようなビルド結果の場合は、ビルドに成功しています。エラー対処は必要ありません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



ビルド失敗イメージ

上図はビルド失敗イメージです。ビルドエラーが表示されるため、メッセージにしたがってエラー対処をおこなってください。

- **手順**2

  ビルドエラー発生時は、エラー対処後、1のビルド処理をエラーがなくなるまで繰り返しおこないます。

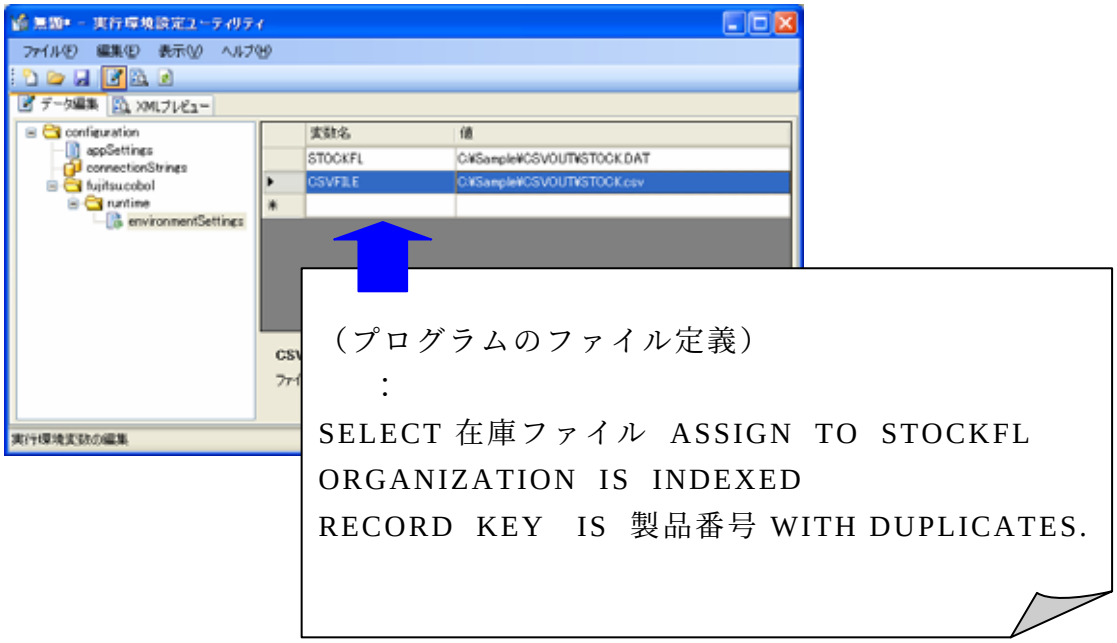
2.4.7 アプリケーション構成ファイルの作成

ビルドが正常に終了したら、いよいよ実行になりますが、その前にプログラムで使用するファイルの割り当てをアプリケーション構成ファイルに定義します。アプリケーション構成ファイルは次のCOBOL実行環境設定ユーティリティを利用することで、GUIで作成することが可能です。作成手順について説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

- **手順**1
  [ソリューションエクスプローラ]ウィンドウよりプロジェクト名(CSVOUT)を右クリックし、[COBOL実行環境設定ユーティリティ]をクリックします。

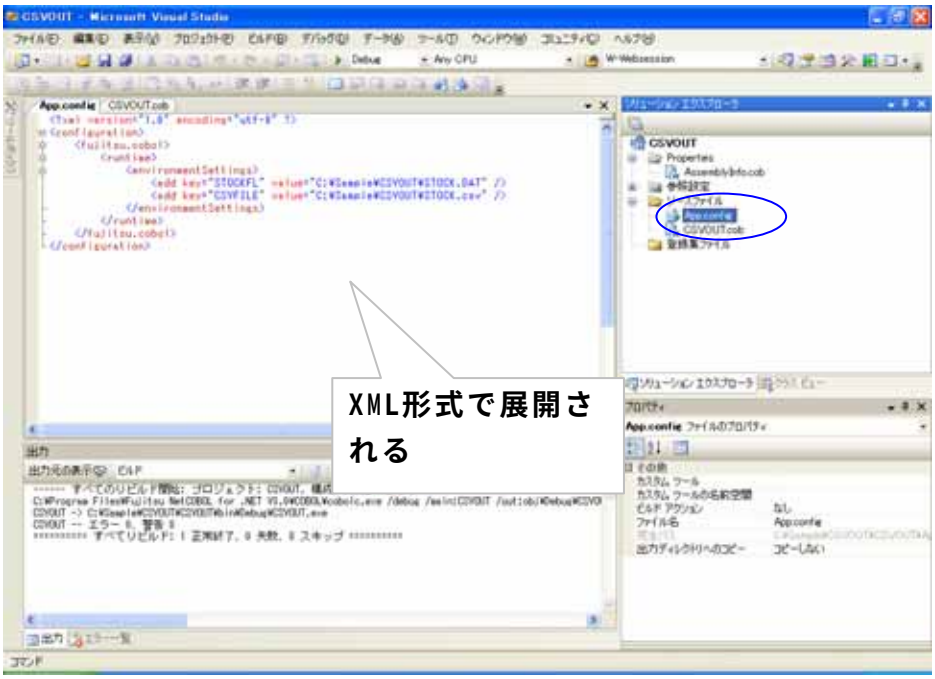
- **手順**2
  [COBOL実行環境設定ユーティリティ]ダイアログボックスが表示されます。
  環境変数情報として、プログラムのファイル識別名と実ファイル名の対応を設定します。



- **手順**3
  環境変数の設定後、[ファイル]メニューより「終了」をクリックすると設定を保存する旨のメッセージボックスが表示されます。「はい」ボタンをクリックし、実行環境設定ユーティリティを終了します。

- **手順**4
  [COBOL実行環境設定ユーティリティ]ダイアログボックスを閉じると、プロジェクトの[ソースファイル]フォルダにファイル名「App.config」のアプリケーション構成ファイルが追加されているのが確認できます。なお、アプリケーション構成ファイルはXML形式で、直接コードウィンドウで編集することも可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



以上が、実行までに必要なプログラム作成や環境設定の手順です。

アプリケーションを実行し、動作を確認してください。「2.4.1 コンソールアプリケーションの実行イメージ」のようにCSV形式のファイルが出力されていれば正常です。

なお、実行時に発生したエラーの対処については、.NET Framework共通のデバッグ機能を利用するとエラー箇所の特定が容易です。[デバッグ]メニューより「ステップイン」機能等を利用し、プログラムコードのどの部分でエラーとなっているのか確認し、必要に応じて対処してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

◇MEMO◇

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第3章 NetCOBOL for .NETの特徴

3.1 クラスとオブジェクトとは
3.2 クラスを使用するプログラムソースの構造
3.3 クラスの利用方法
3.4 オブジェクト指向プログラミングの文法
3.5 NetCOBOL for .NETのデータ型と文字コード
3.6 プログラム間連絡機能

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.1 クラスとオブジェクトとは

3.1.1 クラスとオブジェクトの概要

.NET Frameworkの機能は、クラス(オブジェクト)として提供されます。したがって、NetCOBOL for .NETで、.NET Frameworkから提供されるさまざまな機能を有効に活用するためには、クラスやオブジェクトの基本的な知識が必須です。

しかし、クラスとオブジェクトは、従来の手続き型プログラムとは、まったく異なった概念であるため、従来のCOBOL技術者にとっては、ハードルが高いと感じるケースが多いのも事実です。また、抽象的な概念はなんとなく理解できても、実際にクラスやオブジェクトを使おうとしても具体的な方法がわかりくいこともあります。

ただし、クラスやオブジェクトは、そのイメージ(概念)と基本的な文法さえ正しく理解すれば、それほど難しいものではありません。

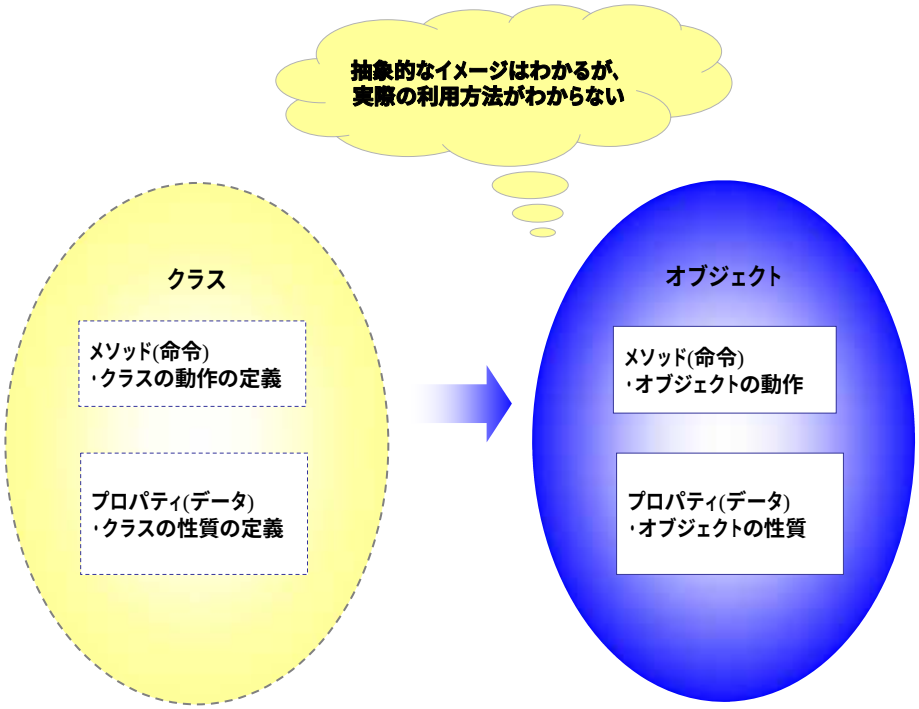


図3.1.1 クラスとオブジェクトとは

また、クラスやオブジェクトは、それぞれ細かな定義や仕様を持っていますが、それぞれを暗記する必要はなく、マニュアルを確認しながらプログラミングするのが、クラスやオブジェクトを利用したプログラミ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ングのスタイルです。

(1) クラスやオブジェクトのイメージ

クラス

命令とデータが定義された設計図(仕様)のようなものです。クラスには複数の命令とデータを定義することができます。したがって、実際にデータは格納されておらず、命令もプログラムから使用できる状態ではありません。

オブジェクト(オブジェクトインスタンス)

(正確には異なりますが)クラスがメモリ上に確保されたもので(インスタンス化)、実際にプログラムから使用できる状態と考えると良いでしょう。

(2) メソッド(命令)とプロパティ(データ)のイメージ

メソッド

クラス(オブジェクト)に定義された命令のことです。メソッドを使用する場合、メソッドが定義されたオブジェクトを指定して使用します。

プロパティ

プロパティとは、クラス(オブジェクト)に定義されたデータを取り出すために用意された方法のことです。事実上、クラスやオブジェクトに定義(格納)されたデータと同義と考えてかまいません。

3.1.2 クラスとオブジェクトとインスタンスの関係

クラスは、「命令」や「データの情報(型など)」が定義された、いわば設計図といえます。命令やデータ自体を扱う場合、この設計図を利用して、実際に動作するモノを作成する必要があります。このモノのことをオブジェクト(オブジェクトインスタンス)と呼びます。クラスとオブジェクトは一対一の関係ではなく、一つのクラスから複数のオブジェクトを作成することができます。その際、プロパティ(データ)はオブジェクトごとに異なる値を持つことができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

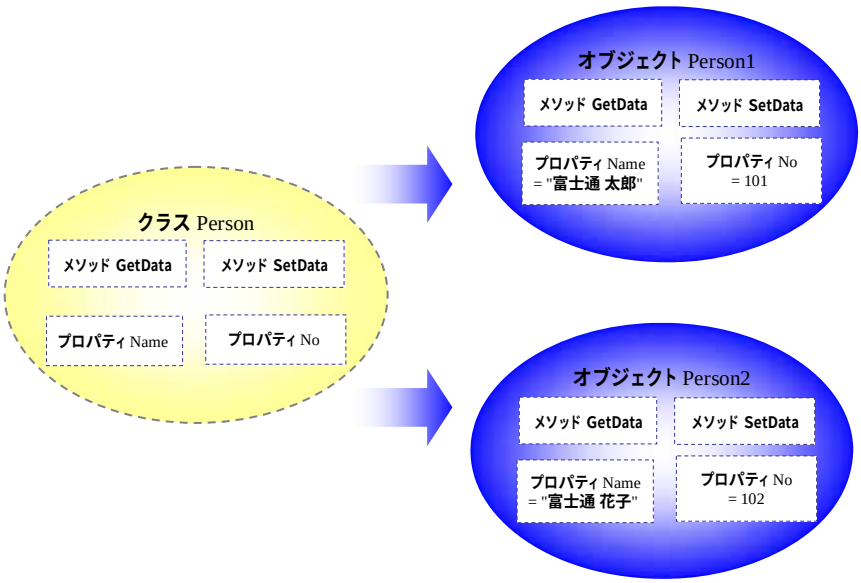


図3.1.2 クラスとオブジェクトとインスタンスの関係

クラスがオブジェクトとして生成され、使用可能な状態になることを「インスタンス化」と呼びます。したがって、クラスから生成されたオブジェクトのことをオブジェクトインスタンスと呼びます。オブジェクトと省略して表現されることも多いのですが、「オブジェクト」はいろいろな意味に使用されることがありますので、オブジェクトインスタンスと表現するのが適切です。

同じクラスから複数のオブジェクトインスタンスを複数作成することができ、オブジェクトインスタンス毎に異なるデータを保持することができます

なお、オブジェクト指向の定義や特徴として「継承」「多態性」「カプセル化」などが挙げられます。しかし、NetCOBOL for .NETによるプログラミングを始める段階で、このような定義を厳密に覚える必要はありません。

3.1.3 クラスの階層とコレクション

実際のクラスやオブジェクトは、複数のクラス(オブジェクト)から成り立っており、階層的な構造をとったり、コレクションと呼ばれるオブ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ジェクトを束ねた概念で扱われたりします。このような構造を理解するためのコツとして、表計算ソフトをイメージすると良いでしょう。

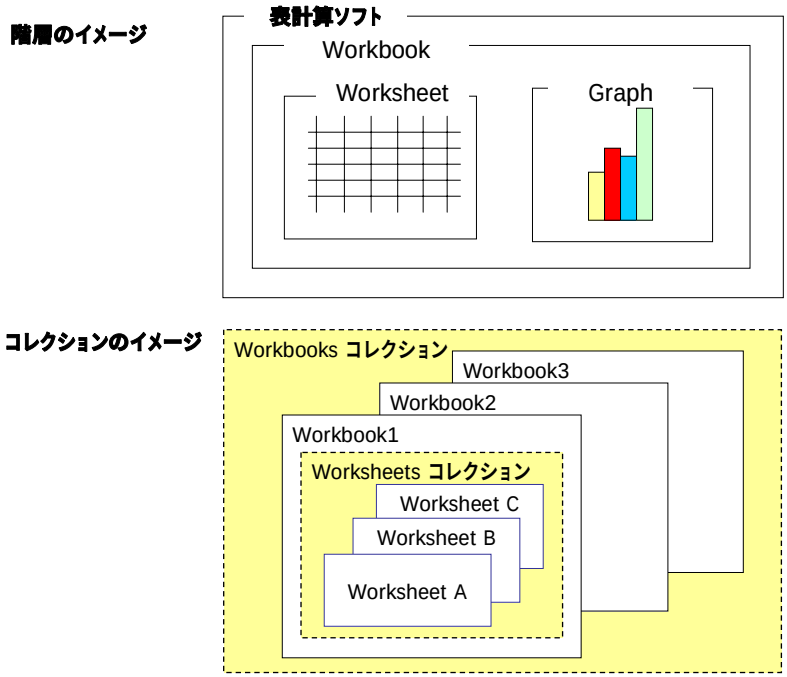


図3.1.3 クラスの階層

たとえば、クラスをMicrosoft Excelをイメージして考えます。Excelで1つの文書はWorkbookという単位で扱われます。これは、Workbookクラスと置き換えることができます。例えば「book1.xls」とか「作業管理表.xls」などといった名前で文書を作成します。作成された文書がいわばオブジェクトになります。したがって、Workbookというクラスから複数のオブジェクトが作成されたことになります。同様に、Excelのシート(ワークシート)も元となるWorksheetクラスから複数のワークシートオブジェクトを作成し、それぞれのワークシートに「WorksheetA」とか「2005年予算」などといった名前が付けられています。

ここで、同じクラスから作成されたオブジェクトをまとめて、配列のようにして扱うことができるとプログラミング上便利です。たとえば、「ワークブック内のすべてのワークシートの名前を調べる」とか「3番目のワークシートの内容を参照する」という処理を行いたい場合、いちいちワークシートの名前を指定せずに、番号で指定できると、繰り返し処理などに便利です。

コレクションとは、同じクラスから作成された複数のオブジェクトを

All Rights Reserved, Copyright FUJITSU LIMITED 2006

配列のようにまとめて扱う概念です。

※「階層構造」は、継承関係を表すことにも利用されますが、ここでは所有関係を「階層化な構造」と表現しています。

(1) クラスの構造とコレクション

- ✓ クラスは階層的な構造をとる場合が多い
- ✓ クラスのインスタンスを作成すると、多くの場合は所有するクラスのインスタンスも作成される

(2) コレクション

- ✓ 同じクラス(オブジェクト)の集まりを「コレクション」と呼ぶ(正確には集まりを管理するオブジェクトのこと)
- ✓ コレクションはオブジェクトの配列のようなもの
- ✓ コレクションの各オブジェクトはインデクサでアクセスする
- ✓ インデクサには”getアクセサ”と”setアクセサ”があり、それぞれ引数を指定するとオブジェクトを特定できる

  例)Worksheets "get_Item" USING 1　　← 1番目を指定
  　Worksheets "get_Item" USING "A"　← A名前を指定

3.1.4 オブジェクト指向の概念

NetCOBOL for .NETは手続き型とオブジェクト指向の2つのプログラミングスタイルを持つ言語です。今まで紹介したように、.NET Frameworkを使用したプログラミングでは、オブジェクト指向を利用します。また、自らクラスを定義することも可能です。

オブジェクト指向には「カプセル化」「継承」「多態」「適合」など、いくつかの概念があります。NetCOBOL for .NETのプログラミングにおいて、それらの概念を詳細に知っておく必要はありませんが、用語自体はマニュアルなどに時々登場します。また、継承はオブジェクト指向を表す特徴的な概念です。ここでは、用語説明的にオブジェクト指向のいくつかの概念について解説します。

(1) カプセル化

カプセル化とは隠蔽とも呼ばれ、あるオブジェクトに対するオブジェクトデータの参照や更新を、そのオブジェクトの持つメソッドだけからのみ可能にすることです。カプセル化により、オブジェクト外に必要な

All Rights Reserved, Copyright FUJITSU LIMITED 2006

情報のみ提供することが可能です。

オブジェクトをカプセル化することで、オブジェクトの独立性を高め、メンテナンス性や再利用性が高まります。

オブジェクト指向としては重要な概念ですが、NetCOBOL for .NETのプログラミングにおいて、カプセル化を強く意識する必要はありません。

(2) 継承(Inheritance)

継承とは、他のクラスの性質(データやメソッドの定義)をそのまま受け継ぐことです。新しいクラスを作る場合に継承を使用すると、既存のクラスの性質をそのまま引き継ぐことができます。さらに新しいデータを追加したり、新しいメソッドを追加したり、メソッドを置き換えたり、さまざまな改造ができます。つまり、既存のクラスを継承すれば、そこからの差分をコーディングするだけで新しいクラスを作ることができます。継承を使用することによって、オブジェクト指向の長所の１つである既存プログラムの再利用を実現できます。継承の親となるクラスをスーパークラス、スーパークラスを継承したクラスをサブクラスや派生クラスと呼びます。.NET Frameworkでは全てのクラスのスーパークラス（基本クラス）としてObjectクラスが用意されています。

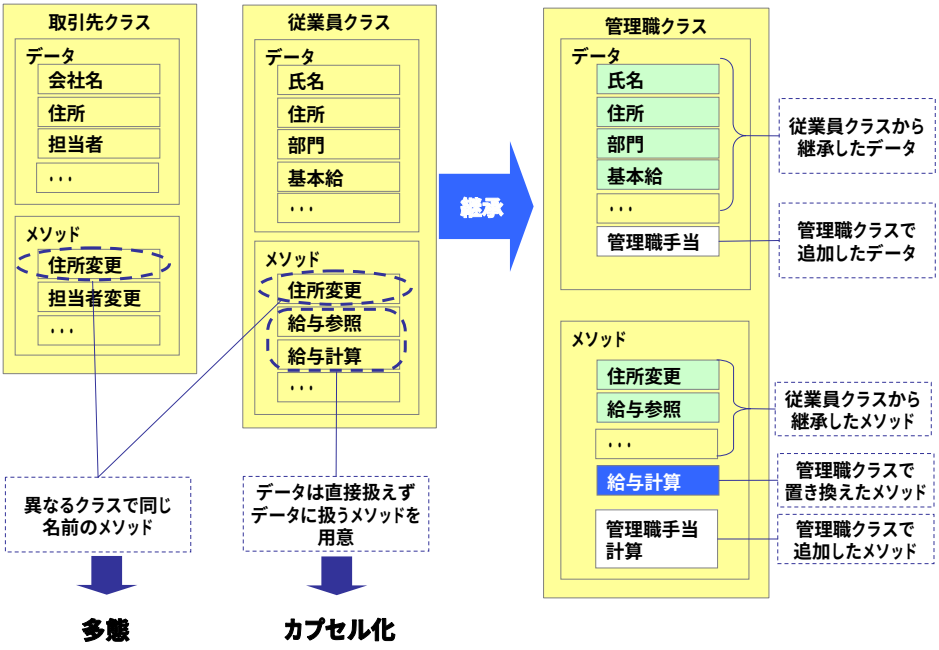


図3.1.4 カプセル化、多態、継承

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(3) 多態(ポリモーフィズム)

多態とは、"1つのものが複数の形態をとることができる" ということです。オブジェクト指向では、同じ名前のメソッドを呼び出したときに、オブジェクトの違いによって、異なる動作をする時などに使われます。

例えば、いくつかの異なるオブジェクトがそれぞれ"SaveData"というメソッドを実装します。メソッドの名前の通り、オブジェクトが扱うデータを保存するための一般的なメソッドとします。しかし、処理の詳細(保存する形式など)はオブジェクトによって異なる場合もあります。

オブジェクト(クラス)を利用する側からすると、メソッドを使用するオブジェクトを変更するだけで、そのオブジェクトに最適な処理を行うことができます。SaveDataメソッドの例では、メソッドを利用するだけで、保存するデータの形式などを意識せずに、オブジェクトに応じた処理ができるということになります。

3.1.5 ジェネリック

ジェネリックとは、特定の型（クラス）に依存せず汎用的な手続きを利用するための仕組みです。

従来のオブジェクト指向プログラミングでは、手続きとその手続きで参照する型を分離することはできませんでした。そのため、型によらない確立された手続きを使用する場合、型毎に同様の手続きを複数用意する、あるいはどのような型にも適合する基本クラスobject型を使用した手続きを用意する、どちらかの方法をとる必要がありました。

例えば、ソートアルゴリズムを考えてみます。ソート対象のデータがstring型（文字列）の場合もint型（整数）の場合も同じソートアルゴリズムを適用したい場合、以下のようなイメージになります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

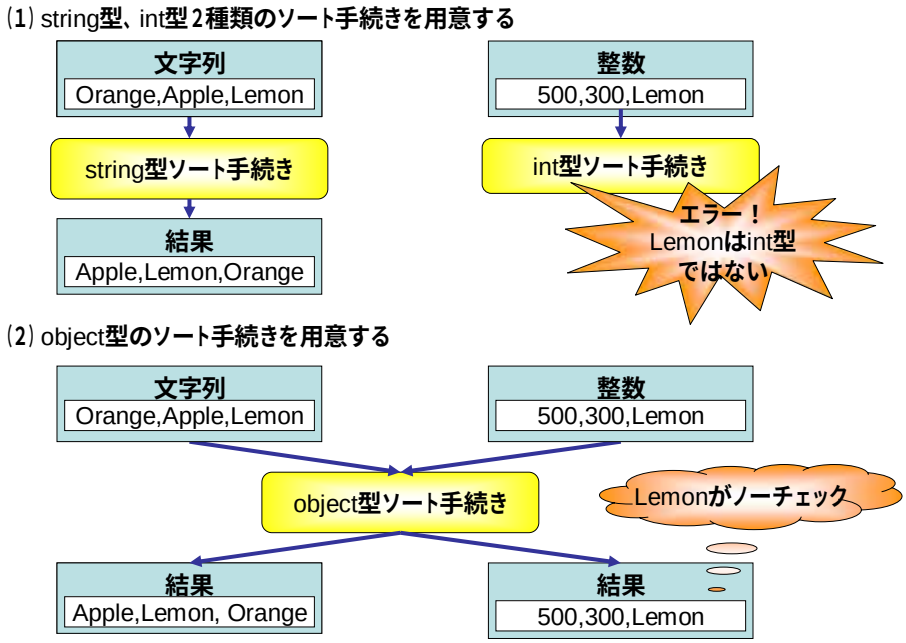


図3.1.5 従来のソート手続き

(1)は複数の同様な手続きを用意する必要があり汎用性に欠けますが、厳密な型チェックが可能です。(2)は手続きを一つにすることができますが、厳密な型チェックができません。

.NET Frameworkでは、このような場合に有効な機能としてジェネリックが用意されています。ジェネリックを使用すると、手続きで参照する型をパラメタ化することができます。型そのものをパラメタとするため、手続き自体は一つ用意するだけでかまいません。さらにパラメタ化された型は通常の型と同様に型チェックが可能です。

先ほどのソートアルゴリズムの例で見てみると以下のようになイメージになります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

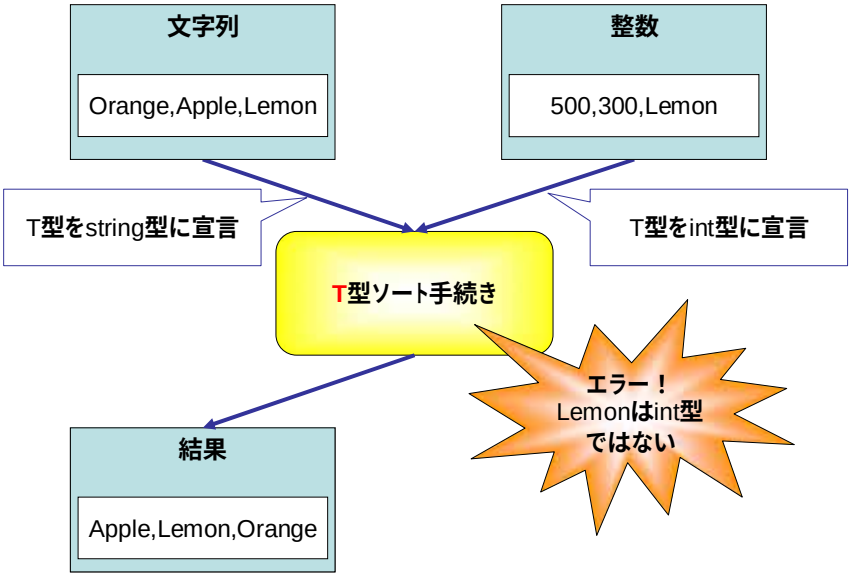


図3.1.6 ジェネリックを使用したソート手続き

パラメタ化した型はT型と表現され、T型をどのような型で使用するかを宣言します。T型をそれぞれstring型,int型として宣言すれば、T型ソート手続きはstring型ソート手続き、int型ソート手続きとして動作します。それぞれの型に合わせた型チェックも行われます。

このようにジェネリックを使用すると、型の概念を持ったまま汎用的な手続きを利用することが可能となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.2 クラスを使用するプログラムソースの構造

クラスを利用したり、クラスを作成したりするCOBOLプログラムソースは従来型のCOBOLソースの構造とは異なるオブジェクト指向COBOLとしての構造となります。以下(図3.2)にオブジェクト指向COBOLで追加された定義を示します。だだし、定義などは新たにオブジェクト指向COBOLとしての文法や構成となりますが、環境部、データ部、手続き部といった基本的なCOBOLの構造は踏襲しています。

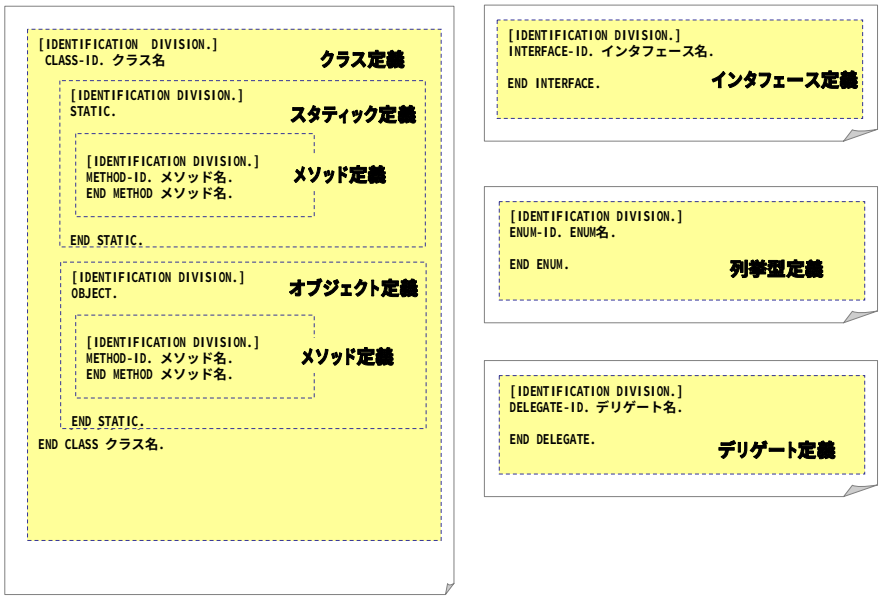


図 3.2 オブジェクト指向COBOLで追加された定義

次に、クラスに関する定義について説明します。

クラス定義

クラス全体の定義部分です。CLASS-IDからEND CLASSまでです。手続き型プログラムのPROGRAM-IDからEND PROGRAMに該当します。クラス定義内に、環境部とスタティック定義、メソッド定義などの手続きを記述します。

スタティック定義

同じクラスから作成されたすべてのオブジェクトインスタンスで共通なデータ(スタティックデータ)やメソッド(スタティックメソッド)を

All Rights Reserved, Copyright FUJITSU LIMITED 2006

記述します。

オブジェクト定義

オブジェクトインスタンスごとに保持するデータやメソッドを定義します。クラス内の処理の大半はオブジェクト定義に記述します。

メソッド定義

スタティック定義やオブジェクト定義内にメソッド(命令、処理)を記述しますが、その記述されたメソッドごとの定義のことをメソッド定義と呼びます。

インタフェース定義

「3.1.4 オブジェクト指向の概念」(3)多態（ポリモーフィズム）を実現する機能の一つです。インタフェース定義には、実装のない（手続きのない）メソッドを定義します。実装は、インタフェースを継承する他のクラスによって行われます。これにより、インタフェースを継承したクラスは同じメソッドを持ち、実装はクラス毎に定義可能となります。

列挙型定義

名前のついた整数からなるデータリストを定義します。

デリゲート定義

メソッドへの内部参照を定義します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.3 クラスの利用方法

3.3.1 クラスを利用する流れ

NetCOBOL for .NETでは、クラスを作成するよりも、外部のクラス(.NET Frameworkや.NETの他言語で作成したクラス)を利用するケースが多いといえます。外部のクラスを利用するためには、COBOLのオブジェクト指向プログラミン機能の約束事に沿って手続きを作成する必要があります。ここでは、外部のクラスを利用する流れを説明します。

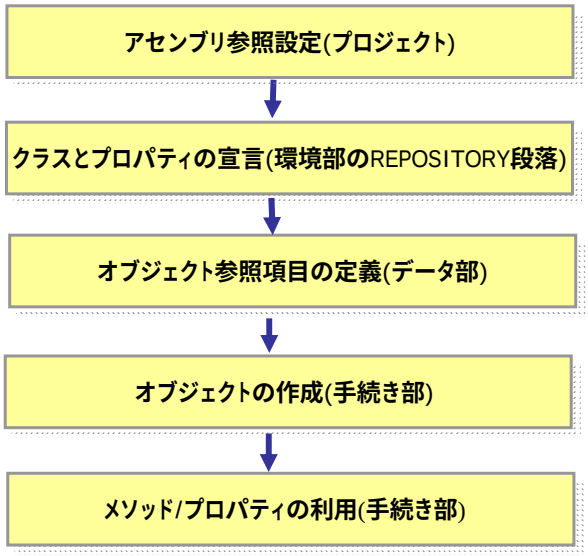


図3.3.1 クラスを利用する流れ

アセンブリの参照設定

クラスはビルド時にクラスの利用に必要な様々な情報とともにアセンブリ(DLLなど)として作成されます。クラスをプロジェクトから利用できるようにするためには、利用したいクラスが格納されたアセンブリをプロジェクトに「参照設定」する必要があります。参照設定は、開発環境上(IDE)で、GUI操作で行います。

また、アセンブリの参照設定はプロジェクト単位で必要となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

クラスとプロパティなどの宣言

プロジェクトで参照設定されたアセンブリ内のクラスは、同じプロジェクトのCOBOLプログラムから利用できるようになります。COBOLプログラムからクラスを利用する最初のステップとして、環境部のリポジトリ段落でクラスの外部名と内部名の関連付けを行います。このとき、クラスの内部名を宣言し、COBOLプログラムではクラスの内部名を指定することになります。また、同時に手続き部で使用するプロパティなども外部名と内部名の関連付けを行います。

オブジェクト参照項目の定義

クラスのインスタンスであるオブジェクトを格納するための「箱」を変数として作成する必要があります。変数の型は「オブジェクト参照項目」となります。データ部で宣言します。

オブジェクトの作成

手続き部でクラスを利用する場合、クラスのインスタンスを作成し、オブジェクトに格納する必要があります。クラスのNEW特殊メソッドを利用して、クラスから実際のオブジェクトを作成します。

メソッド/プロパティの利用

オブジェクトが作成できれば、メソッドとプロパティを利用することができます。メソッド呼出しにはINVOKE文やメソッド行内呼出し、プロパティの設定や取得にはMOVE文やSET文を利用します。

3.3.2 アセンブリの参照設定

プロジェクト(ソリューション)の編集画面のソリューションエクスプローラで、目的のクラスが含まれるアセンブリを参照設定します(プロジェクトを作成する際にテンプレートから選択すると、いくつかのアセンブリがデフォルトで参照設定されています)。

他の.NET対応言語で作成されたアセンブリを参照設定に追加することができます。つまり、つまり、VBやC#などの言語で作成されたクラスを利用することができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

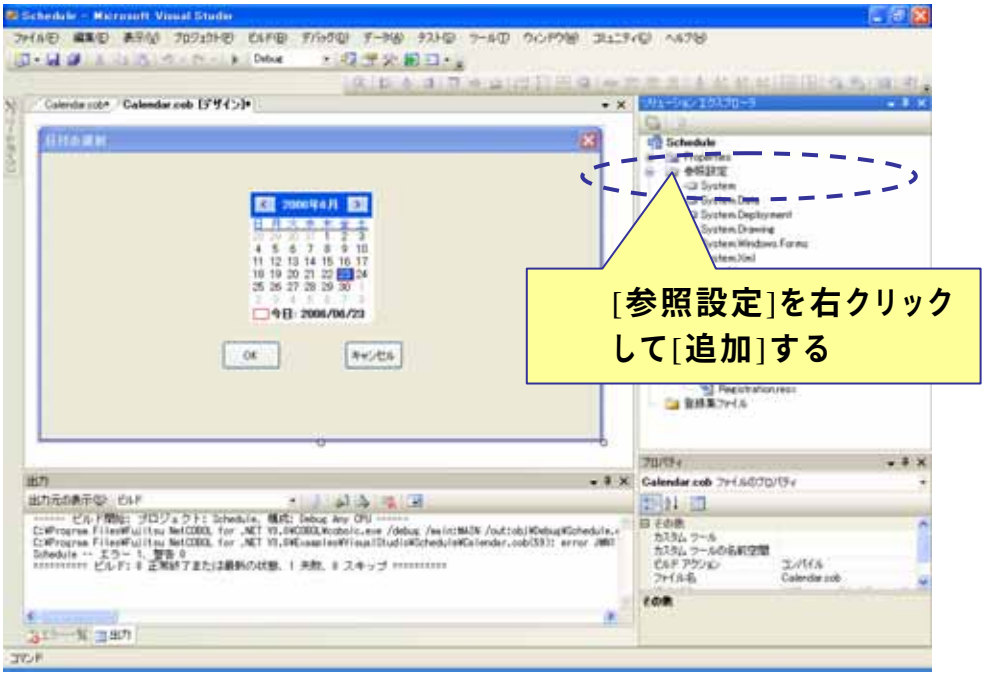


図 3.3.2 アセンブリの参照設定

3.3.3 クラス名やプロパティ名などの宣言

環境部のリポジトリ段落で、使用するクラスやプロパティの内部名を宣言します。

具体的には、CLASS句、PROPERTY句で、クラスの内部名と外部名の宣言(対応付け)を行います。基本的に、クラスやプロパティはREPOSITORY段落で宣言しないと使用することはできません。

また、画面(Windowsフォーム、Webフォーム)を使用するアプリケーションなどではNetCOBOL for .NETが自動的にクラスやプロパティの宣言を書き込むこともあります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

CLASS クラスの内部名 AS "クラス名"

PROPERTY プロパティの内部名 AS "プロパティ名"

(例)

```
 ・・・・・・
REPOSITORY.
    CLASS CLASS-DATAROW AS "System.Data.DataRow"
    CLASS CLASS-DATAROWS AS "System.Data.DataRowCollection"
    CLASS CLASS-DATASET AS "System.Data.DataSet"
    CLASS CLASS-DATATABLE AS "System.Data.DataTable"
    CLASS CLASS-DATATABLES AS "System.Data.DataTableCollection"
    CLASS CLASS-CONNECTION AS "System.Data.SqlClient.SqlConnection"
    PROPERTY PROP-DATAMEMBER AS "DataMember"
    PROPERTY PROP-DATASOURCE AS "DataSource"
・・・・・
```

図3.3.3 クラスの登録とプロパティの登録

クラス指定子

キーワードクラス指定子でクラス名の宣言を行います。ここで指定するクラスの内部名は、このプログラム(クラス)内で有効となります。クラス名(外部クラス名)は名前空間を含む完全名を指定します。

プロパティ指定子

利用するプロパティの内部名を宣言します。クラス内のすべてのプロパティを登録する必要はありません。また、プロパティはクラスに依存せず指定できますので、プロパティ名が同じであれば、どのクラスのプロパティにも使用することができます

その他

リポジトリ段落では、クラス指定子やプロパティ指定子以外にも、インタフェース名を宣言するインタフェース指定子やENUM(列挙型)名を宣言するENUM指定子などを記述する場合があります。
※既定では大文字小文字を区別します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.3.4 オブジェクト参照項目の定義と オブジェクトインスタンスの作成

クラス名を宣言したあとの手続きとして、クラスを格納するための箱を用意する必要があります。この箱を用意する手続きを「オブジェクト参照項目を定義する」と呼びます。オブジェクト参照項目の定義は、データ部でOBJECT REFERENCE句を使用して行います。

オブジェクト参照項目の定義で、オブジェクト(クラス)への参照が定義された変数に、オブジェクトインスタンスを作成することにより、プログラムから、メソッドやデータ(プロパティ)を使用することができるようになります。
クラス(オブジェクト参照が行われたクラス)からオブジェクトインスタンスを作成するにはNEWメソッドを利用します。

オブジェクト参照

レベル オブジェクト変数名 OBJECT REFERENCE クラスの内部名

(例)

```
 ・・・・・・
DATA DIVISION.
WORKING-STORAGE SECTION.
01 strConnection  OBJECT REFERENCE CLASS-STRING.
01 connectDB      OBJECT REFERENCE CLASS-CONNECTION.
・・・・・
```

オブジェクトインスタンスの作成

INVOKE クラスの内部名 "NEW" RETURNING オブジェクト変数名

(例)

```
.
PROCEDURE DIVISION.
 ・・・・・
 INVOKE CLASS-CONNECTION "NEW" USING strConnection RETURNING connectDB.
```

図3.3.4 オブジェクト参照項目の定義とオブジェクトインスタンスの作成

正確には、次のようになります「オブジェクトを作成するには、 コンストラクタを呼び出します。コンストラクタは、実際にはメソッドではありませんが、 形式上は型の"NEW"静的メソッドのように記述します。 NetCOBOL for .NETコンパイラはこの構文を特別扱いして、コンストラクタ呼出しに翻訳します。RETURNING指定に記述されたデータ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

項目には、作成されたオブジェクトへの参照が格納されます。」

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.4 オブジェクト指向プログラミングの文法

3.4.1 メソッドの呼び出し

オブジェクトインスタンスの生成が終わったらメソッドやプロパティを利用できます。まず、メソッドの利用方法について説明します。

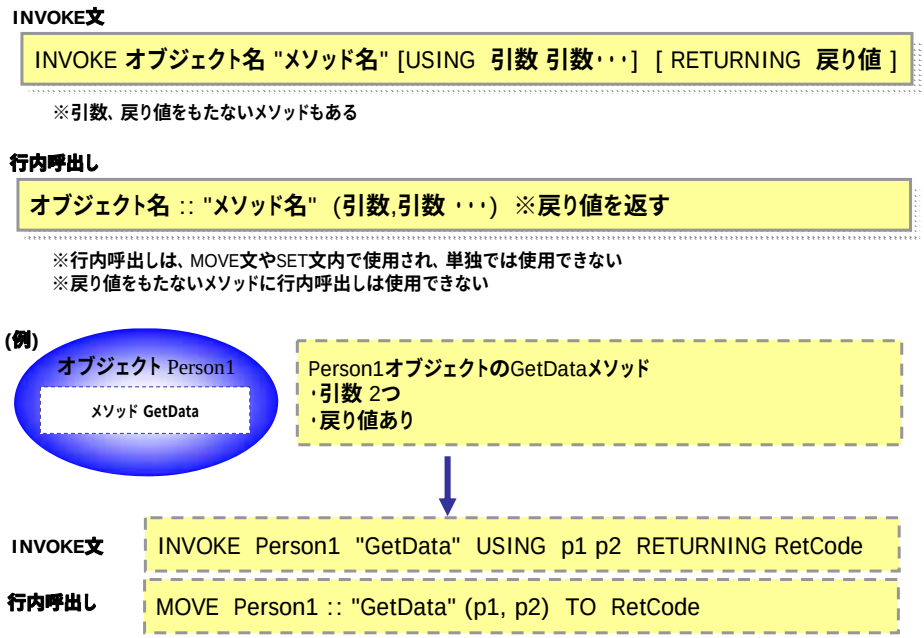


図3.4.1 メソッドの呼び出し

メソッドは通常INVOKE文を利用して呼び出します。構文は次のとおりです。オブジェクト名とメソッド名は必須ですが、USING句とRETURNING句は省略可能です。

```
INVOKE オブジェクト名 "メソッド名" USING 引数列 RETURNING
戻り値
```

また、INVOKE文以外にも「行内呼出し」というメソッド呼び出し方法があります。行内呼出しはメソッドの戻り値を、MOVE文やSET文などで使用する場合に利用できます。戻り値をとらないメソッドには利用できません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.4.2 プロパティの利用

オブジェクトインスタンスの生成が終わったらメソッドやプロパティを利用できます。プロパティの利用方法について説明します。

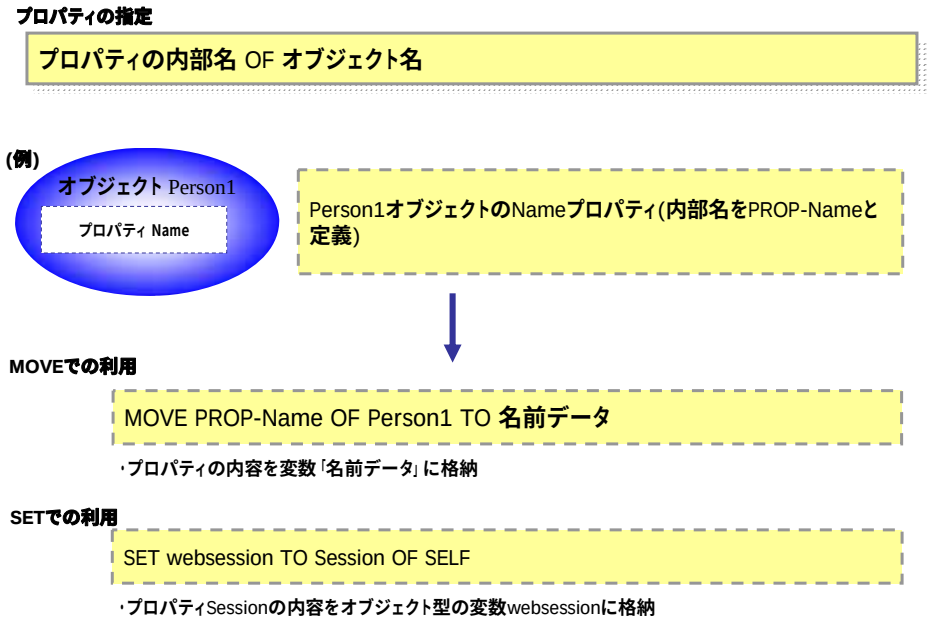


図3.4.2 プロパティの利用

プロパティはオブジェクト名を指定して参照します。代表的な構文は次の通りです。

プロパティの内部名 OF オブジェクト名

プロパティは、MOVE文やSET文を使用して、プロパティの内容を変数に複写したり、プロパティに値を設定したりします。プロパティにより返される値の方が、COBOLの内部表現可能な型の場合MOVE文を使用し、COBOLの内部表現に変換できない型はSET文を利用します。

また、オブジェクトを返すプロパティの場合、メソッドの呼び出しと組み合わせて、INVOKE文や行内呼出しで利用されるケースもあります。

実際のプロパティを使用した例は次にようになります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

txtUserIDオブジェクトのPROP-TEXTプロパティの内容をCOBOL変数のUSERIDに転記する例

```
REPOSITORY.
   PROPERTY PROP-TEXT AS "Text"
 . . . .
WORKING-STORAGE SECTION.
  01 USERID PIC  N(10).
  01 PASSWD PIC  N(10).
  . . . . .
  PROCEDURE DIVISION USING BY VALUE sender e.
      SET USERID TO PROP-TEXT OF txtUserID.
      SET PASSWD TO PROP-TEXT OF txtPassWD.
```

3.4.3 配列とコレクションの利用

.Net FrameworkではCOBOLの表(OCCURS句)と似て異なる概念に配列とコレクションがあります。

(1) 配列

配列は、インデックスを使用してアクセスされる複数の同じ型の変数を含むデータ構造です。その機能を考えると、配列はCOBOLの表によく似ています。しかし、.Net Frameworkの配列は単なる表ではなく配列自体がオブジェクトとなります。そのため、配列を利用するには配列のインスタンス化など「3.3.4 オブジェクト参照項目の定義とオブジェクトインスタンスの作成」で述べたような通常のオブジェクトと同様の記述が必要となります。また、配列の要素にアクセスする場合、取得にはGetメソッド、設定にはSetメソッドを用います。COBOLの表の添字は１から始まりますが、配列のインデックスは０から始まることに注意して下さい。

実際に配列を利用する例は次にようになります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

System.Stringオブジェクトの配列を利用する例

```
REPOSITORY.
   CALSS CLASS-STRING AS "System.String".
   CALSS ARRAY-STRING AS "System.String[]".
   CALSS ARRAY2-CLASS-STRING AS "System.String[,]".
 ・・・・
WORKING-STORAGE SECTION.
  01 WK-STRING  OBJECT REFERENCE CLASS-STRING.
  01 WK-ARRAY   OBJECT REFERENCE ARRAY-STRING.
  01 WK-ARRAY2  OBJECT REFERENCE ARRAY2-STRING..
 ・・・・・
  PROCEDURE DIVISION.
*>8個の要素を持つ１次元文字列配列オブジェクト作成
      INVOKE ARRAY-STRING "NEW" USING 8
                          RETURNING WK-ARRAY.

*>4×8個の要素を持つ２次元文字列配列オブジェクト作成
      INVOKE ARRAY2-STRING "NEW" USING 4 8
                          RETURNING WK-ARRAY2.
....
*>１次元配列の最初の要素を取得
      INVOKE WK-ARRAY "Get" USING 0
                            RETURNING WK-STRING.

*>２次元配列の位置(1,2)の要素を設定
      INVOKE WK-ARRAY2 "Set" USING 1 2
                            RETURNING WK-STRING.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) コレクション

同じクラス（オブジェクト）の集まりを管理するオブジェクトを「コレクション」と言います。

コレクションから要素（オブジェクト）を取り出すには、プロパティにインデックス（添字）を指定するイメージとなります。VBではコレクションにインデックスを指定してオブジェクトを指定することができます。しかし、コレクションに直接インデックスを指定してオブジェクトを参照する文法は、オブジェクトCOBOLには用意されていません。したがって、.NET Frameworkではインデクサと呼ばれる機能を利用してコレクションにインデックスを指定して要素を取得するための方法が用意されています。

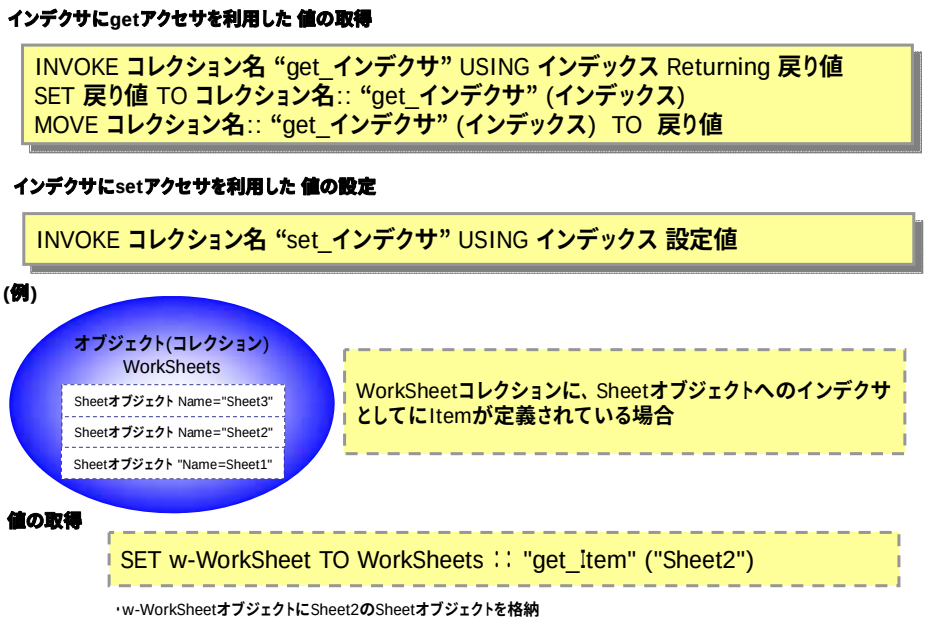


図3.4.3 コレクションの利用

インデクサを利用すると、プロパティの値をメソッド形式で呼び出すことができます。具体的には、インデクサに“get_”や“ set_ ”の接頭辞を指定し、オブジェクトの取得や設定の「メソッド」として利用することになります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

値(オブジェクト)の取得

```
INVOKE コレクション名 "get_インデクサ"
                    USING インデックス RETURNING 戻り値
```

```
SET 戻り値 TO
     コレクション名:: "get_インデクサ" (インデックス)

MOVE コレクション名:: "get_インデクサ" (インデックス)
                                        TO 戻り値
```

値(オブジェクト)の設定

```
INVOKE コレクション名 "set_インデクサ"
                    USING インデックス 設定値
```

また、コレクションの各要素を反復処理で取り出すには、PERFORM文（拡張構文）を使用します。この構文により、Getメソッドやインデックスの換算処理などを記述することなくコレクションの各要素を順次取り出すことが可能です。

反復処理による値の取出し

```
PERFORM VARYING オブジェクト名 THRU コレクション名
....
END-PERFORM
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.5 NetCOBOL for .NETの データ型と文字コード

3.5.1 NetCOBOL for .NETのデータ型

.NET Frameworkでは、すべてのデータをオブジェクトとして扱います。したがって、データを単なるメモリ上の領域として扱う従来の言語とは考え方が異なります。

また、NetCOBOL for .NETで扱うデータ型については、.NET Frameworkでサポートされるデータ型とCOBOL独自のデータ型があり、非常に多くのデータ型が存在することになります。

COBOL言語のみで利用する場合はCOBOL独自データ型を優先して利用して問題ありませんが、クラスなどを利用して他の.NET対応言語と連携する場合は、NETデータ型となります。

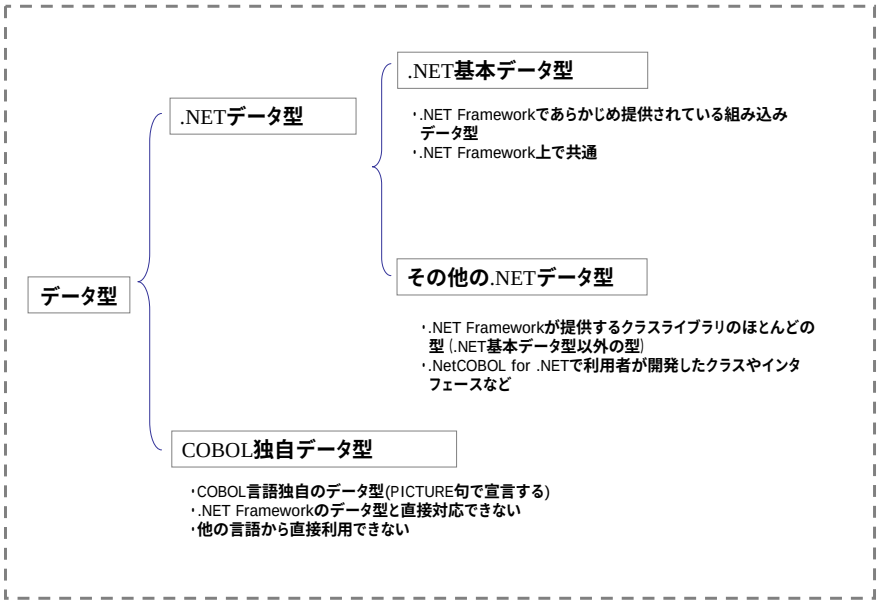


図3.5.1 NetCOBOL for .NETのデータ型

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.5.2 .NET基本データ型とCOBOLデータ型の対応

.NET Frameworkでの型名	COBOL言語上の扱い	説明	CLS準拠	値型
System.Byte	USAGE BINARY-CHAR UNSIGNED	8ビット符号なし整数	Yes	Yes
System.SByte	"System.SByte"型オブジェクト参照	8ビット符号付き整数	No	Yes
System.Int16	USAGE BINARY-SHORT SIGNED	16ビット符号付き整数	Yes	Yes
System.Int32	USAGE BINARY-LONG SIGNED	32ビット符号付き整数	Yes	Yes
System.Int64	USAGE BINARY-DOUBLE SIGNED	64ビット符号付き整数	Yes	Yes
System.UInt16	"System.UInt16"型オブジェクト参照	16ビット符号なし整数	No	Yes
System.UInt32	"System.UInt32"型オブジェクト参照	32ビット符号なし整数	No	Yes
System.UInt64	"System.UInt64"型オブジェクト参照	64ビット符号なし整数	No	Yes
System.Single	USAGE COMP-1	単精度浮動小数点	Yes	Yes
System.Double	USAGE COMP-2	倍精度浮動小数点	Yes	Yes
System.Boolean	"System.Boolean"型オブジェクト参照	ブール値(真または偽)	Yes	Yes
System.Char	PIC N	Unicode文字(16ビット)	Yes	Yes
System.Decimal	"System.Decimal"型オブジェクト参照	96ビット10進値	Yes	Yes
System.IntPtr	"System.IntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号付き整数(32ビットのプラットフォームでは32ビット値、64ビットのプラットフォームでは64ビット値)	Yes	Yes
System.UIntPtr	"System.UIntPtr"型オブジェクト参照	基になるプラットフォームによってサイズが決まる符号なし整数(32ビットのプラットフォームでは32ビット値、64ビットのプラットフォームでは64ビット値)	No	Yes
System.Object	"System.Object"型オブジェクト参照	.NET Frameworkのすべての型のルート	Yes	No
System.String	"System.String"型オブジェクト参照	Unicode文字の文字列	Yes	No

.NET基本データ型は.NET Framework共通のデータ型でCOBOLでも利用することができます。また、COBOLデータ型に置き換えて利用することもできます。

.NET基本データ型と対応しているCOBOLデータ型では、値の受け渡しが可能です。たとえば、「System.Int32」で定義された.NET Frameworkが提供するクラスのプロパティに、「USAGE BINARY-LONG」で定義したCOBOLの変数の内容を転記することができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.5.3 NetCOBOL for .NETの文字コード

.NET Frameworkでは、標準の文字コードはUnicode(UCS-2)です。一方、日本語版のWindowsではシフトJISも多く利用されています。
NetCOBOL for .NETは.NET Framework上の言語であるため、基本的にはUnicodeを採用していますが、シフトJISにも対応しています。翻訳オプションで実行時の英数字類・日本語字類の文字コードを切り替えることが可能です。

処理モード	リソースのコード系	実行時データのコード系		翻訳オプションの指定	プログラム定義	クラス定義
		英数字字類	日本語字類			
処理モード1	シフトJIS	シフトJIS	シフトJIS	SCS(ACP), RCS(SJIS)	○	×
処理モード2		シフトJIS	UCS-2	SCS(ACP), RCS(SJIS-UCS2)	○	○
処理モード3		UTF-8	UCS-2	SCS(ACP), RCS(UTF8-UCS2)	○	○
処理モード4	UTF-8	UTF-8	UCS-2	SCS(ACP), RCS(UTF8-UCS2)	○	○

シフトJIS

MS-DOSやWindowsで広く利用されるコード系。ASCII、半角カナは1バイト、漢字は2バイトで表現される

UCS-2

Unicodeの一種。すべての文字を2バイトで表現される。日本語項目の標準。

UTF-8

Unicodeの一種。ASII文字は1バイト、日本語は3バイトで表現される。英数字項目の標準。

図3.5.3 NetCOBOL for .NETの文字コード

シフトJIS

Windowsで広く利用されているコード体系です。ASCIIや半角カナは1バイト固定で、全角文字は2バイト固定で表現されます。
NetCOBOL for Windowsで動作させていたアプリケーションをそのまま.NET Frameworkへ移植したい場合や、ファイルなどの既存資産を、そのまま(コード変換せずに)利用したい場合などに適しています。

Unicode(UCS-2)

Unicodeの表現形式のひとつです。ASCII文字を含めて1文字は2バイト固定で表現されます。.NET Frameworkの実行時流通コードに採用されているため、他言語や.NET Frameworkと連携する場合には、こ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

のUCS-2表現を使用します。
デフォルトでは、日本語項目(Nタイプ)でUCS-2が利用されます。

Unicode(UTF-8)

Unicodeの表現形式のひとつです。ASCII文字は1バイト、欧州などの国語文字(ギリシャ文字、ロシア文字など)は2バイト、日本語や半角カナ類は3バイトで表現する可変長形式です。このため、文字コードの値だけでなくデータ長の点でシフトJISと互換がありません。
デフォルトでは、英数字項目(Xタイプ)でUTF-8が利用されます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.6 プログラム間連絡機能

3.6.1 サブプログラムの作成と呼び出し

NetCOBOL for .NETのCOBOL形式のサブプログラムは、呼出し元と同一のプロジェクトに作成する場合と、サブプログラムをクラスライブラリとして、呼出し元との異なるプロジェクトに作成する方法があります(サブプログラムをクラスライブラリ形式で作成する方法は4.4を参照)。

サブプログラムを呼び出す場合、従来のCOBOLと同じくCALL文で、「プログラムID」を指定することで呼び出すことができます。引数の設定や戻り値の取得についても従来のCOBOLのサブルーチンの呼び出しと同様です。

呼び出される側のサブプログラムについても、引数の定義(LINKAGE SECTION)をはじめ従来のCOBOL文法と変わりありません。

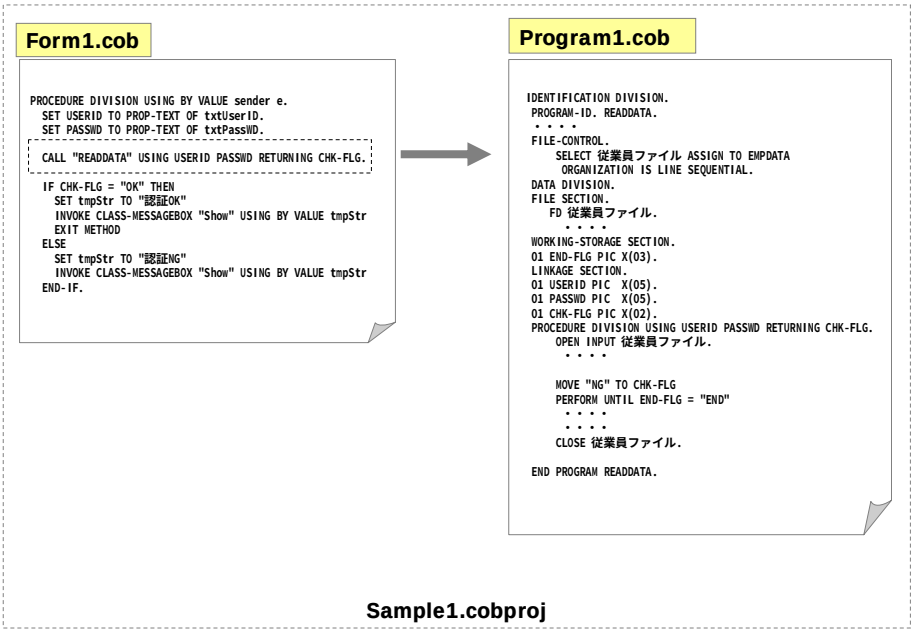


図3.6.1 NetCOBOL for .NET サブプログラムの呼び出し

Windowsフォームアプリケーションのボタンのクリックイベントで、サブプログラムの呼び出している例を紹介します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

【呼び出されるサブプログラムの例】

```
* ---------------------------------------------------------
* 認証処理サブプログラム
*
* パラメタとしてUSERIDとPASSWDを受け取り、従業員ファイルに
* 該当するユーザIDとパスワードが存在するか確認する
* -USERIDとPASSWDが従業員ファイルに存在する→"OK"を返す
* -USERIDとPASSWDが従業員ファイルにない    →"NG"を返す
*   in :USERID PIC  N(05)
*   in :PASSWD PIC  N(05)
*   out:(なし)
*   ret:CHK-FLG PIC X(02)
* ---------------------------------------------------------
 IDENTIFICATION DIVISION.
 PROGRAM-ID. READDATA.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT 従業員ファイル ASSIGN TO EMPDATA
      ORGANIZATION IS LINE SEQUENTIAL.
 DATA DIVISION.
 FILE SECTION.
    FD 従業員ファイル.
    01 EMPDATA-R.
      02 EMPCODE   PIC N(3).
      02 EMPID     PIC N(5).
      02 EMPPWD    PIC N(5).

 WORKING-STORAGE SECTION.
 01 END-FLG PIC X(03).
 LINKAGE SECTION.
 01 USERID PIC  N(05).
 01 PASSWD PIC  N(05).
 01 CHK-FLG PIC X(02).
 PROCEDURE DIVISION USING USERID PASSWD RETURNING CHK-FLG.
*ファイルの読み込み
     OPEN INPUT 従業員ファイル.
     READ 従業員ファイル AT END MOVE "END" TO END-FLG
     END-READ.

     MOVE "NG" TO CHK-FLG
*USERIDとPASSWDがファイルに存在するか確認
     PERFORM UNTIL END-FLG = "END"
       IF EMPID = USERID THEN
         IF EMPPWD = PASSWD THEN
            MOVE "OK" TO CHK-FLG
            EXIT PERFORM
         END-IF
       END-IF

*ファイルの読み込み
       READ 従業員ファイル AT END MOVE "END" TO END-FLG
       END-READ
     END-PERFORM

     CLOSE 従業員ファイル.

 END PROGRAM READDATA
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

【サブプログラムを呼び出す例】

```
* ------------------------------------------------------------
* サブプログラムを呼び出すイベントプロシージャの例
*
* テキストボックスに入力されたユーザIDとパスワードを取得し
* 認証処理のサブプログラムを呼び出す処理
*
* ------------------------------------------------------------
 METHOD-ID. btnLogon1_Click PRIVATE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 USERID PIC  N(5).
 01 PASSWD PIC  N(5).
 01 CHK-FLG PIC X(02).
 01 tmpStr OBJECT REFERENCE CLASS-STRING.
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
* テキストボックスから情報を採取する
     SET USERID TO PROP-TEXT OF txtUserID.
     SET PASSWD TO PROP-TEXT OF txtPassWD.

* サブプログラム呼出し
     CALL "READDATA" USING USERID PASSWD RETURNING CHK-FLG.

* サブプログラムの戻り値で認証を判断
* 戻り値が"OK"
     IF CHK-FLG = "OK" THEN
       SET tmpStr TO "認証OK"
* メッセージボックスの表示
       INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE tmpStr
       EXIT METHOD
* 戻り値が"NG"
     ELSE
       SET tmpStr TO "認証NG"
* メッセージボックスの表示
       INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE tmpStr
     END-IF.

 END METHOD btnLogon2_Click.
```

3.6.2 同一プロジェクト内のクラス定義の利用

NetCOBOL for .NETで、同じプロジェクト内のクラスを呼び出す場合、外部のクラスを利用する方法とほぼ同じ手順になります。ただし、外部のクラスを利用するときに必要な参照設定は不要です。

呼び出される側のクラスについては、PROGRAM-ID段落の替わりにCLASS-ID段落を使用し、呼び出されるメソッドを定義します。引数と戻り値の定義は、手続き型COBOLと同様です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

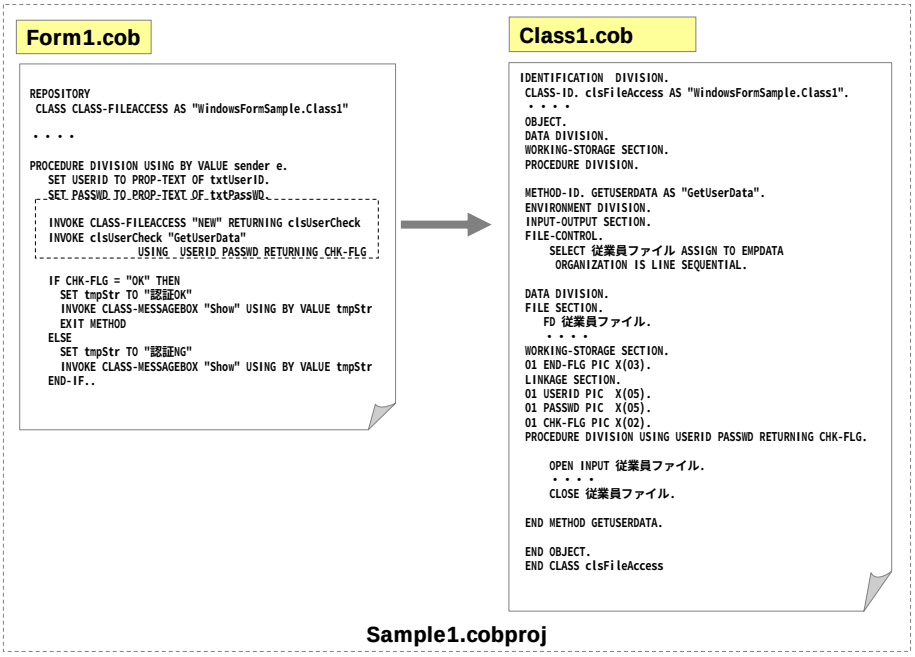


図3.6.2 同じプロジェクト内のクラスの呼び出し

Windowsフォームアプリケーションのボタンのクリックイベントで、クラスを呼び出している例を紹介します。「3.6.1のサブプログラム呼び出し」の例と比較してみてください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

【クラスの例】

```
* ------------------------------------------------------------
* 認証処理クラス
*
* パラメタとしてUSERIDとPASSWDを受け取り、従業員ファイルに
* 該当するユーザIDとパスワードが存在するか確認する
* -USERIDとPASSWDが従業員ファイルに存在する→"OK"を返す
* -USERIDとPASSWDが従業員ファイルにない    →"NG"を返す
*   in :USERID PIC  N(05)
*   in :PASSWD PIC  N(05)
*   out:(なし)
*   ret:CHK-FLG PIC X(02)
* ------------------------------------------------------------
 IDENTIFICATION  DIVISION.
 CLASS-ID. clsFileAccess AS "WindowsFormSample.Class1".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.

 STATIC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.

 END STATIC.

 OBJECT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.

* メソッドの宣言
 METHOD-ID. GETUSERDATA AS "GetUserData".
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT 従業員ファイル ASSIGN TO EMPDATA
      ORGANIZATION IS LINE SEQUENTIAL.

 DATA DIVISION.
 FILE SECTION.
    FD 従業員ファイル.
    01 EMPDATA-R.
      02 EMPCODE   PIC N(3).
      02 EMPID     PIC N(5).
      02 EMPPWD    PIC N(5).
 WORKING-STORAGE SECTION.
 01 END-FLG PIC X(03).
 LINKAGE SECTION.
 01 USERID PIC  N(05).
 01 PASSWD PIC  N(05).
 01 CHK-FLG PIC X(02).

PROCEDURE DIVISION USING USERID PASSWD RETURNING CHK-FLG.
*ファイルの読み込み
     OPEN INPUT 従業員ファイル.
     READ 従業員ファイル AT END MOVE "END" TO END-FLG
     END-READ.


・・・(続く)・・・
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

【 クラスの例(続き) 】

```
     MOVE "NG" TO CHK-FLG
     PERFORM UNTIL END-FLG = "END"
*USERIDとPASSWDがファイルに存在するか確認
       IF EMPID = USERID THEN
         IF EMPPWD = PASSWD THEN
            MOVE "OK" TO CHK-FLG
            EXIT PERFORM
         END-IF
       END-IF

*ファイルの読み込み
       READ 従業員ファイル AT END MOVE "END" TO END-FLG
       END-READ
     END-PERFORM

     CLOSE 従業員ファイル.

 END METHOD GETUSERDATA.

 END OBJECT.
 END CLASS clsFileAccess.
```

【クラスを呼び出す例】

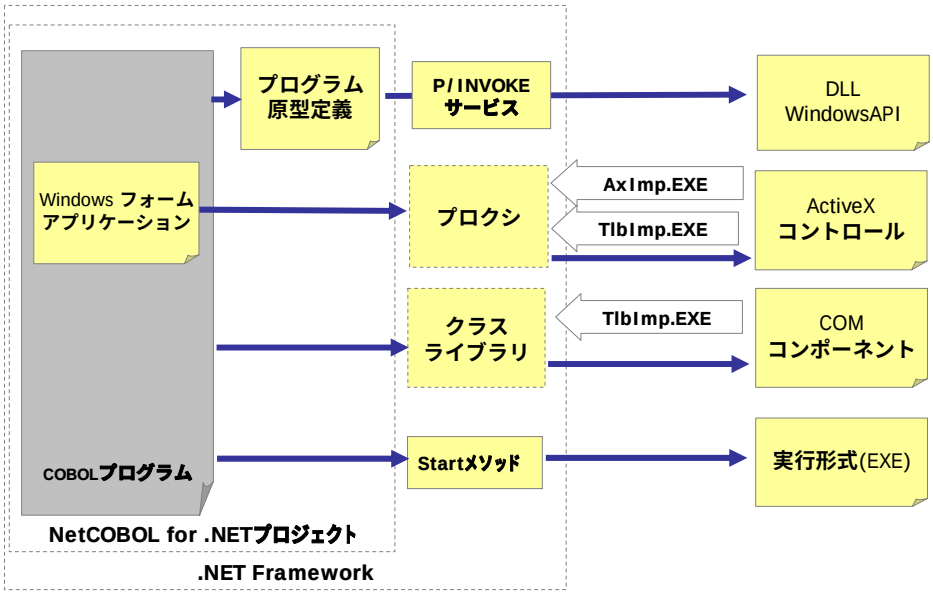
```
* ---------------------------------------------------------
* サブプログラムを呼び出すイベントプロシージャの例
*
* テキストボックスに入力されたユーザIDとパスワードを取得し
* 認証処理のサブプログラムを呼び出す処理
*
* ---------------------------------------------------------
METHOD-ID. btnLogon2_Click PRIVATE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 USERID PIC  N(5).
 01 PASSWD PIC  N(5).
 01 CHK-FLG PIC X(02).
 01 tmpStr OBJECT REFERENCE CLASS-STRING.
 01 clsUserCheck  OBJECT REFERENCE CLASS-FILEACCESS.
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
     SET USERID TO PROP-TEXT OF txtUserID.
     SET PASSWD TO PROP-TEXT OF txtPassWD.
* インスタンスの作成
     INVOKE CLASS-FILEACCESS "NEW" RETURNING clsUserCheck
* メソッドの呼出し
     INVOKE clsUserCheck "GetUserData" USING  USERID PASSWD
RETURNING CHK-FLG
      IF CHK-FLG = "OK" THEN
        SET tmpStr TO "認証OK"
        INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE tmpStr
        EXIT METHOD
      ELSE
        SET tmpStr TO "認証NG"
        INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE tmpStr
      END-IF.

  END METHOD btnLogon2_Click.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

3.6.3 NetCOBOL for .NETの外部プログラムとの連携

.NET Framework上で開発されたアプリケーション(.NET アセンブリ)のことを「マネージコード」と呼びます。逆に、.NET Frameworkで開発されていない(.NET アセンブリではない)アプリケーションのことを、「アンマネージコード」と呼びます。

NetCOBOL for .NETからマネージコードを呼び出すのはクラスライブラリの呼び出しとして行えば良いので、比較的簡単です。しかし、アンマネージコードを呼び出す場合はNetCOBOL for .NETから直接呼び出すことはできず、.NETが提供している特別な機能を利用する必要があります。また、アンマネージコードの種類ごとに呼び出し方法が異なります。



※Startメソッド:System.Diagnostics.ProcessクラスのStartメソッド

図3.6.3 NetCOBOL for .NETの外部プログラムとの連携

DLL/Windows API

NetCOBOL for WindowsやVisual C++などで作成したWindowsのDLLやWindows APIを呼び出す場合、.NET FrameworkのP/INVOKEサービス(プラットフォーム呼び出しサービス)を利用します。

NetCOBOL for .NETでは、呼び出すDLLに応じたプログラム原型定義を作成することで、P/INVOKEサービス経由でDLLを呼び出すことができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

COMコンポーネント(ActiveX DLL)

COMコンポーネントやVisual Basic 6.0で作成したDLL(ActiveX DLL)は、.NET FrameworkのツールであるTlbImp.EXE(タイプライブラリ インポータ)を使用し、.NET Frameworkのクラスライブラリとして変換します。

ActiveXコントロール

Visual BasicやPowerCOBOLなどの画面を構成する部品であるActiveXコントロールを使用する場合、上記のTlbImp.EXEに加えて.NET FrameworkのツールであるAxImp.EXE(Windows フォーム ActiveX コントロール インポータ)を使用し、ActiveXコントロールをWindows Formアプリケーションから利用できるプロクシを作成します。

実行形式プログラム(EXE)

実行形式プログラムは、System.Diagnostics.ProcessクラスのStartメソッドを利用して呼び出すことが可能です。また、Process.WaitForExitメソッドによりプロセスの終了監視が可能です。

3.6.4 P/INVOKE機能の利用方法

NetCOBOL for .NETからP/INVOKE機能を利用して、アンマネージコードのDLLを呼び出す場合、プログラム原型定義の作成が必要です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

NetCOBOL for .NETプログラム

```
CLASS-ID. HELLO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS SYS-STRING AS "System.String"
    PROGRAM MESSAGEBOX AS "MessageBox".
STATIC.
PROCEDURE DIVISION.
METHOD-ID. MAIN.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 HANDLE USAGE BINARY-LONG.
01 MESSAGE-TEXT OBJECT REFERENCE SYS-STRING.
01 CAPTION-TEXT OBJECT REFERENCE SYS-STRING.
01 MSGBOX-TYPE USAGE BINARY-LONG.
PROCEDURE DIVISION.
    MOVE 0 TO HANDLE
    SET MESSAGE-TEXT TO "Hello world!"
    SET CAPTION-TEXT TO "Calling Unmanaged Code"
    MOVE 0 TO MSGBOX-TYPE
    CALL MESSAGEBOX USING HANDLE MESSAGE-TEXT
        CAPTION-TEXT MSGBOX-TYPE
    END-CALL.
END METHOD MAIN.
END STATIC.
END CLASS HELLO.
```

プログラム原型定義

```
PROGRAM-ID. MESSAGEBOX AS "MessageBox"
            IS PROTOTYPE CUSTOM-ATTRIBUTE IS PINVOKE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.

   CUSTOM-ATTRIBUTE PINVOKE
    CLASS DLLIMPORT USING "USER32.DLL"
       PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
       PROPERTY CHARSET IS ANSI OF E-CHARSET.

REPOSITORY.
    CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
    ENUM E-CALLINGCONVENTION
       AS "System.Runtime.InteropServices.CallingConvention"
    PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
    PROPERTY STDCALL AS "StdCall"
    PROPERTY CHARSET AS "CharSet"
    ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
    PROPERTY ANSI AS "Ansi"
    CLASS SYS-STRING AS "System.String".
DATA DIVISION.

LINKAGE SECTION.
01 HWND USAGE BINARY-LONG.
01 MESSAGE-TEXT OBJECT REFERENCE SYS-STRING.
01 CAPTION-TEXT OBJECT REFERENCE SYS-STRING.
01 MSGBOX-TYPE USAGE BINARY-LONG.
PROCEDURE DIVISION USING BY VALUE HWND MESSAGE-TEXT
    CAPTION-TEXT MSGBOX-TYPE.
END PROGRAM MessageBox.
```

```
int MessageBox(
  HWND hWnd,
  LPCTSTR lpText,
  LPCTSTR lpCaption,
  UINT uType
);
```

WindowsAPIの定義

図3.6.4 P/INVOKEサービスの利用方法

プログラム原型定義には、呼び出し方法、関数名、DLL名、引数と戻り値の定義などを行います。NetCOBOL for .NETの手続きからは、プログラム原型定義で定義された関数のインタフェースを呼び出すイメージで記述します。

プログラム原型定義は、一見複雑ですが、サンプルから変更するべき点は多くはありません。ほとんどの場合、関数名、DLL名、引数、戻り値といった関数の呼び出しに必要な点を書き換えるだけで動作します。

実際に関数を呼び出す手続きは、CALL文を利用します。厳密にはプログラム原型を呼び出すための特別なCALL文となりますが、利用方法は通常のCALL文(プログラム間連絡のCALL文)とほぼ同様です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

◇MEMO◇

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第4章 NetCOBOL for .NETによる画面プログラムの作成

4.1 Windowsフォームアプリケーションの概要
4.2 NetCOBOL for .NETとWindowsフォームアプリケーション
4.3 Windowsフォームの代表的なクラスと利用方法
4.4 Windowsフォームからサブプログラムの呼出し
4.5 Windowsフォームアプリケーションの作成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.1 Windowsフォームアプリケーションの概要

4.1.1 Windowsフォームと Windowsフォームアプリケーション

Windowsフォームとは、Windowsアプリケーションのために、.NET Frameworkが提供している画面を作成・実行するためのフレームワークです。Windowsフォームの機能はすべてクラスで提供されており、.NET Framework上の言語であれば、種類に関係なく利用することができます。また、Windowsフォームを利用するアプリケーションをWindowsフォームアプリケーションと呼びます。

WindowsフォームアプリケーションはWindowsの機能を十分に利用できる形式で、コンソールアプリケーションやWebアプリケーションと比較して、画面設計や機能的に自由度が高いアプリケーションを開発することができます。

Windowsフォームアプリケーションはいわゆるイベントドリブン(イベント駆動型)です。「ボタンのクリック」「画面のロード」など、イベントごとに手続きを記述します。
しかし通常は、すべての処理をイベント手続きに記述するのではなく、データアクセスやビジネスロジックをクラスにまとめておき、Windowsフォームから呼び出します。こうすることで、プログラムの再利用性を高めることができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

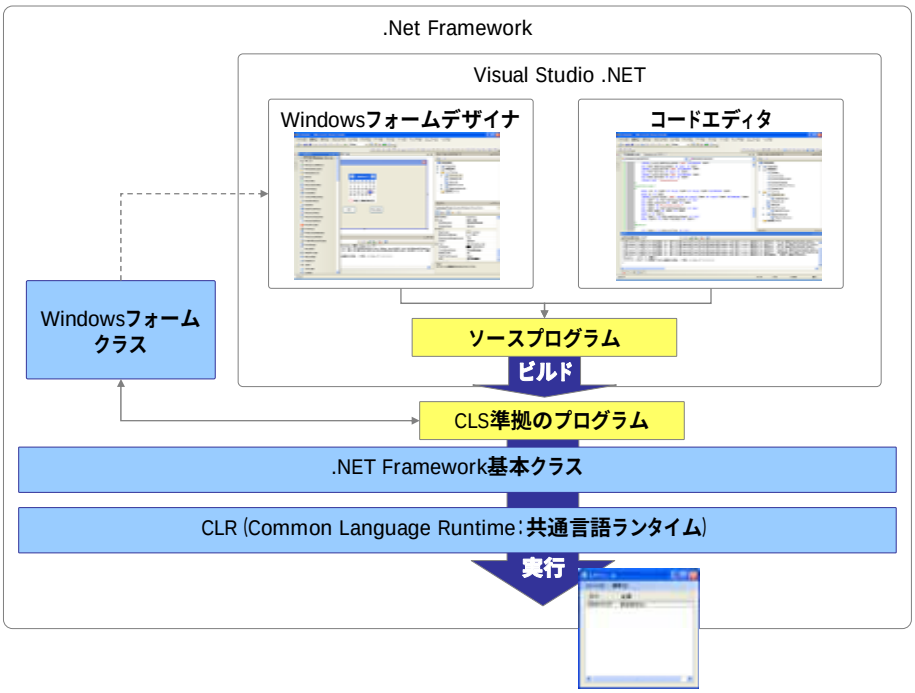


図4.1.1 Windowsフォームアプリケーションの開発イメージ

Windowsフォームを使用したアプリケーション開発の基本的なパターンについて説明します。まず、Visual Studioのデザイナを利用しGUIで画面デザインやプロパティの設定などを行います(コード記述でのプロパティ設定も可能)。その後、Windowsフォームのイベント手続きなどのアプリケーションの動作を、コードエディタで記述します。

コーディング終了後、Visual Studioを利用してビルド/実行/デバッグを行います。この手順は、.NET Frameworkの各言語を通して共通です。

4.1.2 Windowsフォームアプリケーションの運用形式

Windowsフォームアプリケーションの運用形式としては、スタンドアロン、クライアントサーバ、XML Webサービスクライアントがあります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

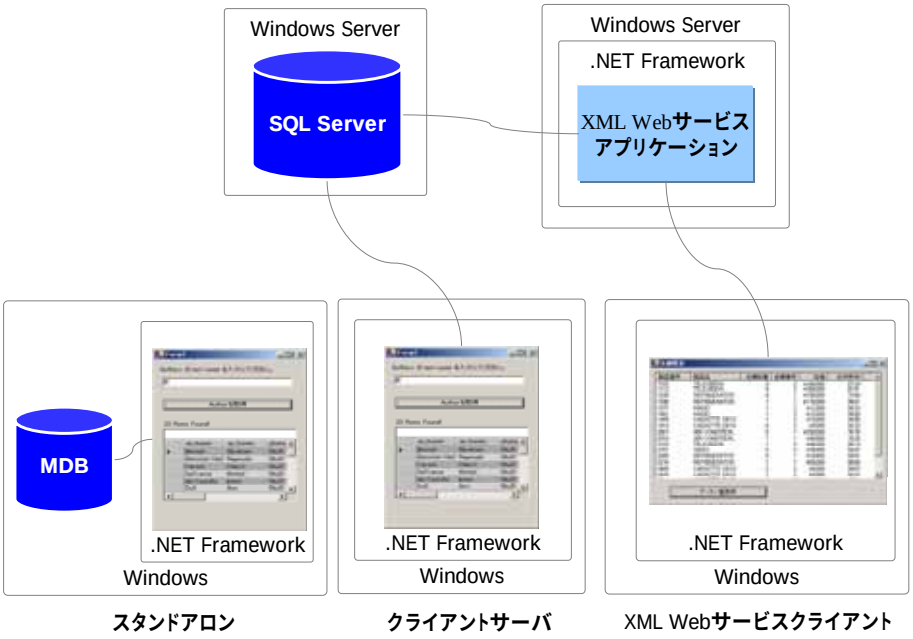


図4.1.2 Windowsフォームアプリケーションの運用形式

スタンドアロンは1台のWindows環境で完結した形式で、データベースなどもローカルに保持しています。

クライアントサーバは、データをサーバ上に配置し、アプリケーションをクライアントに配置する形式です。複数のクライアントでデータを共有できるのが特徴です。

XML Webサービスクライアントは、比較的新しい形式です。Webサービスと呼ばれるビジネスロジックをまとめたアプリケーションをWeb上に配置し、クライアントからXML Webサービスを利用します。3階層のシステムを構築できる利点があります。

XML WebサービスにはXMLやSOAPといった技術が使われていますが、.NET FrameworkにはXML Webサービスへアクセスするためのインタフェースクラスを自動生成する機能が提供されているため、複雑な手続きを記述することなく、通常のクラスを扱うようにXML Webサービスにアクセスすることができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.2 NetCOBOL for .NETと Windowsフォームアプリケーション

NetCOBOLのWindowsフォームアプリケーションの概要

NetCOBOL for .NETでもWindowsフォームアプリケーションを構築することができます。

「4.1 Windowsフォームアプリケーションの概要」で説明したように、Windowsフォームの機能は、.NET Frameworkで提供されるため、NetCOBOL for .NETからも他の言語と同様にWindowsフォームの機能をフルに活用することができます。

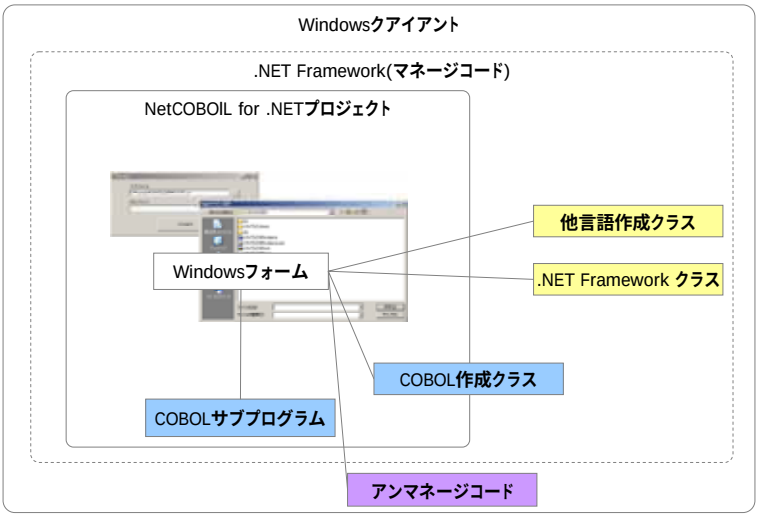


図 4.2.1 Windowsフォームアプリケーションの構成

NetCOBOL for .NETのWindowsアプリケーションからは、クラスだけではなく、COBOLサブグログラムやアンマネージコードを呼び出すことも可能です。

特にCOBOLサブプログラムは、いわゆる手続き型のプログラム形式となりますので、NetCOBOL for Windowsや他のプラットフォームで作成されたCOBOLソースをサブプログラム化し、Windowsフォームから呼び出すことで、既存資産を有効に活用することも可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.2.1 Windowsフォームの構造

Windowsフォームはビジュアルなデザイナでデザインするため、一見バイナリのファイルで構成されているイメージがあります。しかし、画面デザインなどはテキスト形式でファイルに保存されています。また、同時にイベント手続きについても同じファイルに保存されます。定義ファイルは実際にはクラス定義形式のCOBOLソースであり、図4.1.2のようになっています。

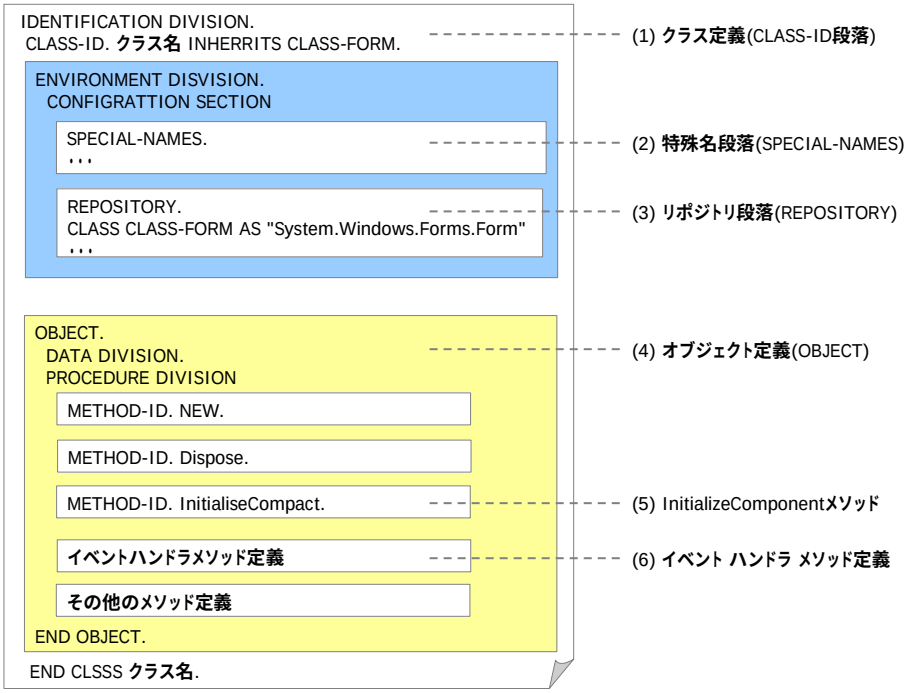


図4.2.2 WindowsフォームのCOBOLプログラム構造

(1) クラス定義(CLASS-ID段落)

クラス定義は、COBOLのオブジェクト指向プログラミング機能で追加された構文です。クラスは、そのクラスから生成されるオブジェクト(インスタンス)の動作と属性を定義したものです。

オブジェクト指向プログラミングの機能には、任意のクラスを継承して新しいクラスを作成する機能があります。Windows アプリケーションでは、.NET Frameworkクラスライブラリに用意されているSystem.Windows.Forms.Formクラスを継承して、新しいWindowsフォームクラスを作成します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) 特殊名段落(SPECIAL-NAMES)

.NET Frameworkの重要な機能の1つである カスタム属性を定義する場合、ここに定義します。カスタム属性を意識して利用することはあり多くはありません。

(3) リポジトリ段落(REPOSITORY)

この環境部(ENVIRONMENT DIVISION)を含むソース単位内で使われるクラス名やプロパティ名を指定します。また、.NET Frameworkのクラス名(外部名)を、ソース単位内で参照するための名前(内部名)と対応付けて指定します。したがって、リポジトリ段落はクラスを利用する上で非常に重要です。

デザイナを利用してフォーム上に配置したコントロールのクラスやプロパティは、デザイナが自動的にリポジトリ段落に必要な情報を追加します。また、Visual Studio .NETエディタのIntelliSense機能でも、コントロールのプロパティの外部名に対応する内部名をリポジトリ段落に追加できます。

リポジトリ段落で使用するクラス指定子やプロパティ指定子については、「3.3.3 クラスの登録とプロパティの登録」を参照してください。

(4) オブジェクト定義(OBJECT)

オブジェクトのデータやメソッドを定義する領域です。Windowsフォームの画面に関連したメソッドやデータ(フォーム上のコントロールへの参照を保持する作業領域やイベントハンドラとなるメソッドなど)は、テンプレートやデザイナで自動的に生成されますが、必要に応じて独自のメソッドやデータを定義することができます。

(5) InitializeComponentメソッド

画面のレイアウトを作成する手続きが記述されます。この手続きはデザイナが自動生成します。

デザイナで設計した画面のレイアウトは、このメソッド内に記述されたCOBOL手続きとコメントによって表現されているため、デザイナ以外でこの手続きを変更すると、デザイナが画面レイアウトを復元できなくなる場合があります。このため、このメソッド内の手続き(コメント行含む)を直接変更してはいけません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(6) イベントハンドラメソッド定義

Windowsフォームで作成するWindowsアプリケーションは、「ボタンがクリックされた」などの動作を契機(イベント)として手続きを実行します。この契機をイベントといいます。従来の上から下に処理が流れていく手続き型プログラミングモデルとは異なり、イベント駆動型プログラミングモデルによってアプリケーションを作成します。

イベントハンドラは、メソッドとして実装されます。イベントハンドラメソッドとコントロールが発行するイベントとの関連付けはデザイナが行っており、InitializeComponentメソッド内に必要な情報が自動生成されます。デザイナはイベントハンドラメソッド定義の最小限の手続きを自動的に生成するので、プログラマはメソッド定義のデータ部や手続き部を記述するだけで済みます。

イベントハンドラのプログラム例

```
METHOD-ID. button1_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 LVItem OBJECT REFERENCE CLASS-LISTVIEWITEM.
01 tmpStr          OBJECT REFERENCE CLASS-STRING.
01 tmpDcml         OBJECT REFERENCE CLASS-DECIMAL.
01 tmpInt          OBJECT REFERENCE CLASS-INT32.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.

PROCEDURE DIVISION USING BY VALUE sender e.
DECLARATIVES.
ERR-001 SECTION.
    USE AFTER EXCEPTION CLASS-EXCEPTION.
    SET tmpStr TO PROP-MESSAGE OF EXCEPTION-OBJECT.
    INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE tmpStr.
    EXIT METHOD.
END DECLARATIVES.
    OPEN INPUT 入力ファイル.
    MOVE SPACE TO ENDFLG.
    START 入力ファイル FIRST RECORD
    PERFORM UNTIL ENDFLG = "END"
      READ 入力ファイル NEXT RECORD
      AT END MOVE "END" TO ENDFLG
      NOT AT END
        SET tmpInt TO 製品番号
        SET tmpStr TO tmpInt::"ToString"("0000")
        INVOKE CLASS-LISTVIEWITEM "NEW" USING BY VALUE tmpStr RETURNING LVItem

        SET tmpStr TO 製品名
        INVOKE PROP-SUBITEMS OF LVItem "Add" USING BY VALUE tmpStr.

  ・・・中略・・・
        INVOKE Items OF listView1 "Add" USING LVItem .
```

Button1のクリックイベント

System.Windows.Forms.ListViewItem クラスの参照設定

System.Windows.Forms.MessageBox クラスのShowのメソッドによるメッセージボックスの表示

ListViewItemクラスのインスタンスを作成し、アイテム(データ)を格納

ListViewItemクラスのインスタンスにアイテムを追加

ListViewコントロールにデータを追加 (ListViewItemオブジェクト内のデータを表示)

図4.2.3 ListViewコントロールにデータを表示する例(一部)

このサンプルリストは、ファイルから読み取ったデータをListViewコントロールに表示する例ですが、各所にWindowsフォームのクラス、すなわち.NET Frameworkのクラスライブラリを利用しているのがわかります。Windowsフォームアプリケーションを作成するということは、.NET FrameworkのWindowsフォームのクラスライブラリを活用す

All Rights Reserved, Copyright FUJITSU LIMITED 2006

るということに他なりません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.3 Windowsフォームの代表的なクラスと利用方法

4.3.1 名前空間とは

Windowsフォームの代表的なクラスを知る前に、名前空間について説明します。

.NET Frameworkのクラスは、関連するクラスがグループ化され、階層化して管理されています。この階層化されたクラスの各階層を名前空間と呼びます。それぞれの名前空間には、それぞれの名前空間で使用できるクラスが定義されています。

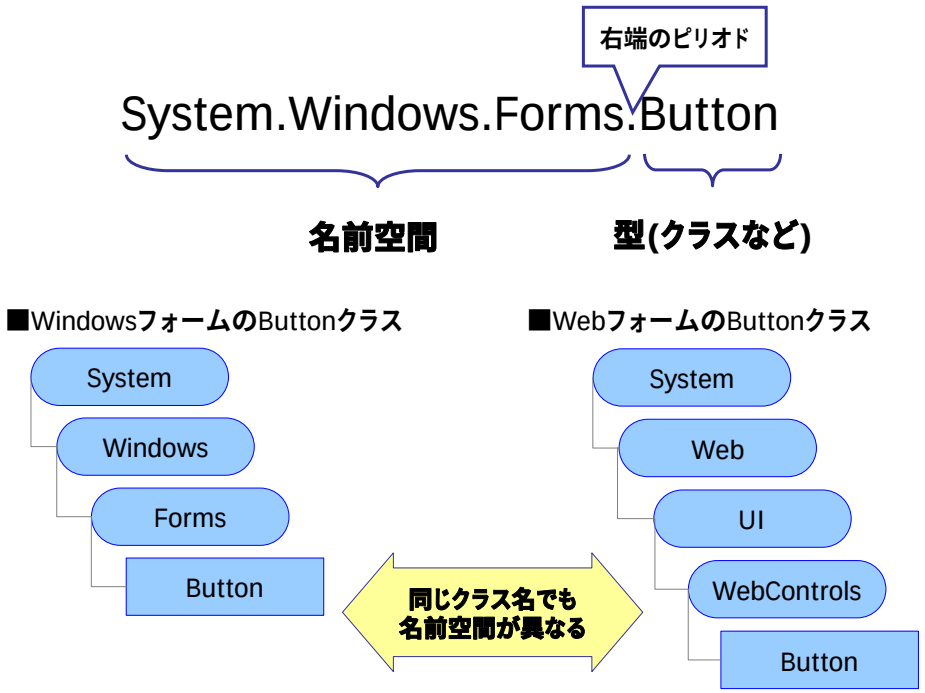


図4.3.1 名前空間とクラス

名前空間はクラスの階層を表すとともに、クラスを識別することになります。.NET Frameworkプラットフォーム上には大量のクラスが提供されているため、特徴が似た同じ名前のクラスがある場合があります。そのようなクラスを名前空間で識別することになります。

.NET Frameworkのリファレンスでは、クラスが、どの名前空間に属するか明記されています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.3.2 System.Windows.Forms名前空間

Windowsフォームで中心となる名前空間はSystem.Windows.Forms名前空間です。System.Windows.Forms名前空間にある幾つかのコントロールについて簡単な知識があれば、Windowsフォームアプリケーションを作成することはできます。しかし、System.Windows.Forms名前空間の詳細を知ることで、より高度なWindowsフォームアプリケーションを構築することができます。

図4.3.2 System.Windows.Formsのリファレンス

System.Windows.Forms名前空間を.NET Frameworkのリファレンスで参照すると、多少わかりにくい解説が続いています。ここを正確に覚える必要はありません。System.Windows.Forms名前空間は以下のような機能が用意されていることを雰囲気でつかんでいれば問題ありません。

■ Control,User Control,and Form

Windowsフォームに関する最も基本的な機能です。

ウィンドウ(フォーム)そのものの機能や、各コントロールに共通する基本機能を提供するクラスなどです。ユーザが独自のコントロールを

All Rights Reserved, Copyright FUJITSU LIMITED 2006

作成するための機能もあります。

- Controls

  Windowsフォームに貼り付ける各コントロールのクラスを提供します。コントロールとは、データ入力のテキストボックスなど、情報表示のラベル、リストボックスなど、イベント(動作)を発生させる部品のことです。

- Components

  コントロールとしては用意されていないメニューはヘルプ機能などです。

- Common Dialog Boxes

  Windowsで共通化されたコモンダイアログを呼び出す機能です。コモンダイアログとは、例えば「ファイルを開く」「フォントの選択」「色の選択」など多くのプログラムで利用される機能を、一貫したユーザインタフェースを実現するためにWindowsが提供しているものです。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.3.3 代表的なクラス

(1) Formクラス

名前空間:System.Windows.Forms

アプリケーションのトップレベルウィンドウとなるクラスです。Formクラスでは、さまざまなウィンドウを作成できます。通常はデザイナでウィンドウ(フォーム)の作成とプロパティの設定などを行うため、プログラムの手続きでFormクラスを使用することはあまりありません。ウィンドウの表示位置やサイズの変更、モーダル/モードレスの設定変更などを使用したい場合はFormクラスを利用します。

なお、作成されたFormオブジェクト内からFormクラス(自分自身)のオブジェクトやメソッドを利用する場合は、オブジェクト名に定義済みのオブジェクトの「SELF」を使用します。

■ 使用例

Formのタイトル(Textプロパティ)を指定した文字列に変更する例

```
【クラス宣言】
  IDENTIFICATION DIVISION.
  CLASS-ID. Form1 AS "WinFormSample1.Form1"
      INHERITS CLASS-FORM.

【リポジトリ段落】
  REPOSITORY.
      CLASS CLASS-FORM AS "System.Windows.Forms.Form"
      PROPERTY PROP-TEXT AS "Text"

【手続き部(Form_Load)】
 * フォームのTextプロパティ(タイトルの)変更
      SET PROP-TEXT OF SELF TO N"フォームサンプル".
```

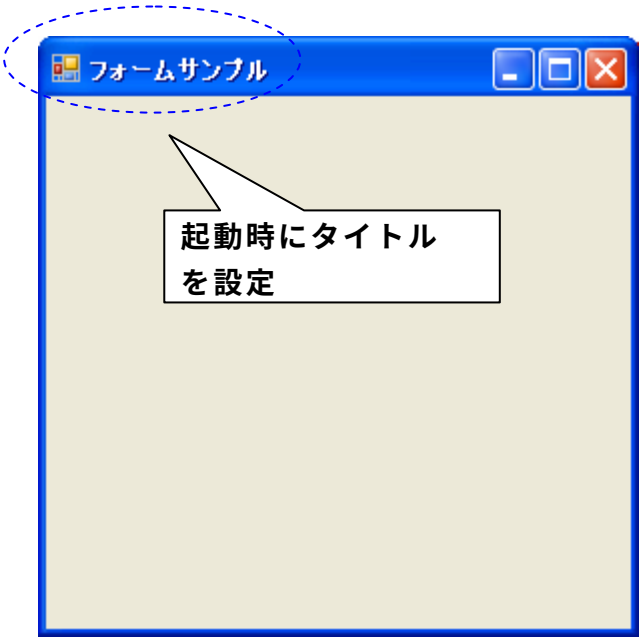
■ 解説

クラス宣言では、System.Windows.Forms.Formクラスを継承したForm1クラスの宣言を行っています。Formクラスの宣言は通常はデザイナが自動生成します。

リポジトリ段落では、参照するクラスの内部名の設定をおこないますが、Formに関連する主なクラスやプロパティはデザイナが自動生成します。

手続き部では、Formクラスのオブジェクトである自分自身(SELF)のTextプロパティに値を設定しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



4.3.3(1) Formクラスの使用例

(2) Labelクラス

名前空間:System.Windows.Forms

LabelクラスはLabelコントロールを表すクラスです。Labelコントロールは、フォーム上に配置された部品の説明や操作の解説などに利用されます。大半のプロパティはデザイナで設定されるケースがほとんどですが、Textプロパティは状況によってプログラムから設定することもあります。Textプロパティの利用方法は、Formプロパティのサンプルとほぼ同様です。

(3) TextBoxクラス

名前空間:System.Windows.Forms

TextBoxクラスはTextBoxコントロールを表すクラスです。Labelが表示専用なのに対し、TextBoxコントロールにはキーボードなどから単一行や複数行のデータの入力が可能です。また、PasswordCharプロパティを利用すると、入力された文字を直接表示せず、別の文字(＊など)で文字の個数だけを表示させるように設定することができます。

プログラムからは、Textプロパティの値の設定や参照、Enabledプロパティを利用した入力の可否の制御、KeyPressイベントやKeyDownイベントを利用した入力データの確認などの処理がよく行われます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ 使用例

TextBoxコントロールに入力された文字列を取得する例

```
【リポジトリ段落】
  REPOSITORY.
    CLASS CLASS-TEXTBOX AS "System.Windows.Forms.TextBox"
    PROPERTY PROP-TEXT AS "Text"

【OBJECT定義のDATA DIVISION】
WORKING-STORAGE SECTION.
    01 txtUserID OBJECT REFERENCE CLASS-TEXTBOX.
    01 txtPassword OBJECT REFERENCE CLASS-TEXTBOX.

【メソッドのDATA DIVISION】
WORKING-STORAGE SECTION.
    01 USER-ID   PIC N(10).
    01 USER-PWD  PIC N(10).
【メソッドの手続き部】
    SET USER-ID TO PROP-TEXT OF txtUserID.
    SET USER-PWD TO PROP-TEXT OF txtPassword.
```

■ 解説

TextBoxコントロールをデザイナで追加すると、リポジトリ段落にSystem.Widnows.Forms.TextBoxクラスが追加されます。また、OBJECT定義のDATA DIVISIONには、TextBoxコントロールに設定されたname(ID)プロパティの内容と同じ名前でオブジェクト変数が追加されます。したがって、クラス内で追加されたTextBoxコントロールが使用できることになります。

手続き部では、TextBoxコントロールに入力された文字列を変数に取り込む処理を記述しています。

なお、この例は、TextBoxコントロールのMaxLengthプロパティで、入力できる文字数を10文字に制限していることを前提としています。

(4) Buttonクラス

名前空間:System.Windows.Forms

ButtonクラスはButtonコントロールを表すクラスです。Buttonコントロールはコマンドボタンとも呼ばれ、クリックして処理を開始する際に利用されます。

プログラムからは、Enabledプロパティを利用した、クリックの可否の制御が行われることがあります。また、通常はClickイベントにButtonコントロールがクリックされた場合の手続きを記述します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ 使用例

Buttonコントロールをクリック不可にする例

```
【リポジトリ段落】
  REPOSITORY.
    CLASS CLASS-BUTTON AS "System.Windows.Forms.Button"
    PROPERTY PROP-ENABLED AS "Enabled"

【OBJECT定義のDATA DIVISION】
WORKING-STORAGE SECTION.
   01 btnLogin OBJECT REFERENCE CLASS-BUTTON.

【メソッドの手続き部】
    SET PROP-ENABLED OF btnLogin TO B"0".
```

■ 解説

Buttonコントロールをデザイナで追加すると、リポジトリ段落にSystem.windows.Forms.Buttonクラスが追加されます。また、System.Boolean構造体型というtrueもしくはfalseを保持するクラスも追加されます。

OBJECT定義のDATA DIVISIONには、Buttonコントロールに設定されたname(ID)プロパティの内容と同じ名前でオブジェクト変数が追加されます。したがって、クラス内で追加されたButtonコントロールが使用できることになります。

手続き部では、ButtonコントロールのEnabledプロパティには、System.Boolean型の値を設定する必要がありますが、falseを意味するB”0”を代入しています。

(5) MessageBoxクラス

名前空間:System.Windows.Forms

MessageBoxクラスはメッセーボックスを表示するためのクラスです。System.Windows.Forms名前空間に属しますが、コントロールではありありません。メッセージボックスを表示するためのメソッドはMessageBoxクラスの静的メソッドとして提供されており、MessageBoxクラスのインスタンスを生成することなく呼び出すことができます。

プログラムからは、Showメソッドを利用してメッセージボックスを表示する処理を記述しますが、パラメータによりメッセージ文のほか、メッセージボックスのタイトルやボタンの形状、アイコンの形式などを指定することもできます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ 使用例

メッセージボックスを表示する例

```
 【リポジトリ段落】
REPOSITORY.
    CLASS CLASS-MESSAGEBOX
AS "System.Windows.Forms.MessageBox"
    ENUM ENUM-MESSAGEBOXBUTTONS
 AS "System.Windows.Forms.MessageBoxButtons"
    PROPERTY PROP-YESNOCANCEL AS "YesNoCancel"
    ENUM ENUM-DIALOGRESULT
 AS "System.Windows.Forms.DialogResult"
    PROPERTY PROP-YES AS "Yes"
    PROPERTY PROP-NO AS "No"
    PROPERTY PROP-CANCEL AS "Cancel"


 【メソッドのDATADIVISION】
WORKING-STORAGE SECTION.
    01 WK-MESSAGEBOXBUTTONS
   OBJECT REFERENCE ENUM-MESSAGEBOXBUTTONS.
 01 WK-DIALOGRESULT
   OBJECT REFERENCE ENUM-DIALOGRESULT.

 【メソッドの手続き部】
 SET WK-MESSAGEBOXBUTTONS TO
       PROP-YESNOCANCEL OF ENUM-MESSAGEBOXBUTTONS.

 INVOKE CLASS-MESSAGEBOX "Show"
       USING N"クリック!" N"Sample" WK-MESSAGEBOXBUTTONS
       RETURNING  WK-DIALOGRESULT.

    EVALUATE  WK-DIALOGRESULT
    WHEN PROP-YES OF ENUM-DIALOGRESULT
      INVOKE CLASS-MESSAGEBOX "Show"
        USING N"「はい」がクリック" N"クリック結果"
    WHEN PROP-NO  OF ENUM-DIALOGRESULT
      INVOKE CLASS-MESSAGEBOX "Show"
        USING N"「いいえ」がクリック" N"クリック結果"
    WHEN PROP-CANCEL OF ENUM-DIALOGRESULT
      INVOKE CLASS-MESSAGEBOX "Show"
        USING N"「キャンセル」がクリック" N"クリック結果"
 END-EVALUATE.
```

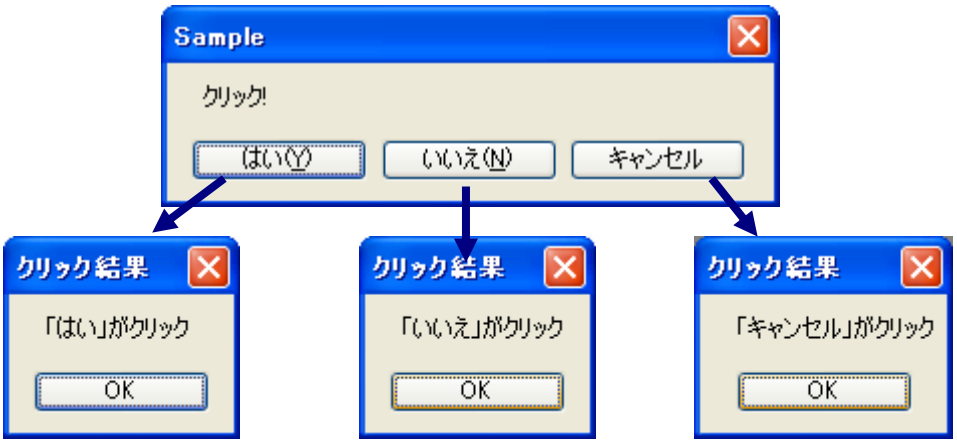
■ 解説

単純にメッセージボックスを表示するのみであれば、リポジトリ段落に、MessageBoxクラスの内部名を指定し、手続き部でShowメソッドを利用するだけでメッセージボックスを表示できます。

「はい」「いいえ」などのボタンを表示したり、アイコンを表示したりする場合、メッセージ以外のパラメータを指定する必要があります。例では、「はい」「いいえ」「キャンセル」の3つのボタンを表示するために、MessageboxButtons列挙体のYesNoCancelメンバを指定しています。また、戻り値としてDialogResult列挙体が返されます。したがって、リポジトリ段落にMessageboxButtons列挙体と

All Rights Reserved, Copyright FUJITSU LIMITED 2006

DialogResult列挙体およびそれぞれの列挙体のメンバ(プロパティ)の内部名を宣言しています。
手続き部では、メッセージボックスを表示していますが、列挙体の参照は直接メソッドのパラメータに指定できないため、列挙体の参照を格納する列挙型変数を利用しています。



4.3.3(2) MessageBoxクラスの使用例

(6) OpenFileDialogクラス、SaveFileDialogクラス

名前空間:System.Windows.Forms

OpenFileDialogクラスはOpenFileDialogコントロールを表すクラスで、SaveFileDialogクラスはSaveFileDialogコントロールを表すクラスです。それぞれのクラスは、ファイルを選択するダイアログボックスを表示する機能を持ちます。

これらのクラスはWindowsのコモンダイアログボックスを表示する機能をもち、これによりWindows標準のUIを利用することができます。これらのクラスは一般のコントロールのようにフォーム上に貼り付いた形で表示されません(ただし、デザイン時にはこれらのクラスを使用していることがわかるようにデザイナ上に表示されます)。

OpenFileDialogクラスとSaveFileDialogは非常に良く似ていますが、OpenFileDialogは主に既存のファイルを開くことを目的としているのに対し、SaveFileDialogは保存するファイルを選択することを目的としているため、メソッドの動作やプロパティが異なります。

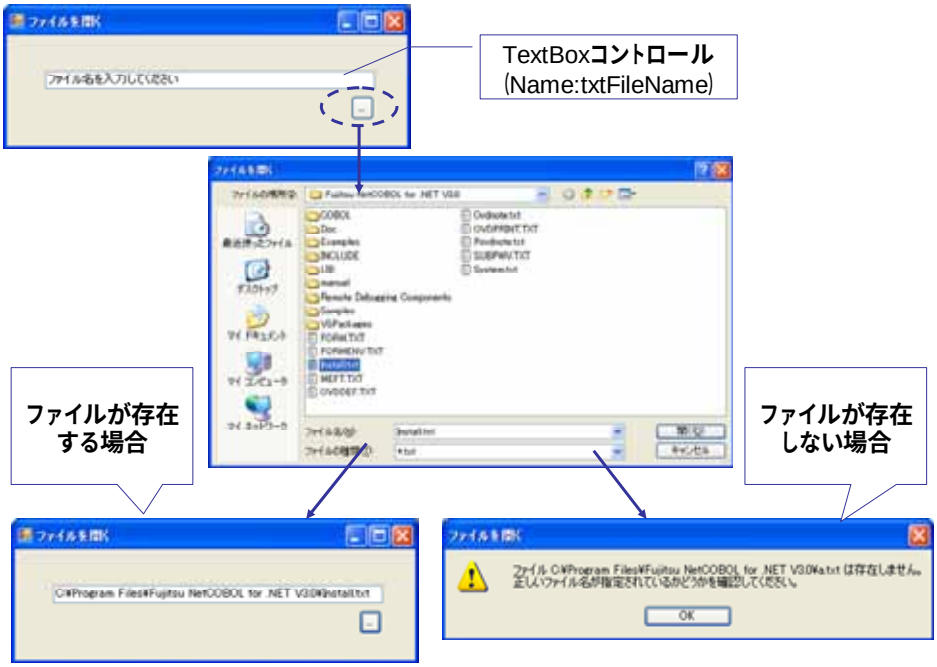
プログラムからは、ShowDialogメソッドでファイルを開く(保存する)ダイアログを表示し、FileNameプロパティで、ダイアログで指定されたファイル名をパス付きで受け取ります。また、Filterプロパティを指定しておくと、開くファイルの拡張子を指定することができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ 使用例

メッセージボックスを表示する例

```
【リポジトリ段落】
REPOSITORY.
     CLASS CLASS-OPENFILEDIALOG
 AS "System.Windows.Forms.OpenFileDialog"
     ENUM ENUM-DIALOGRESULT
 AS "System.Windows.Forms.DialogResult"
     PROPERTY PROP-OK AS "OK"
     PROPERTY PROP-FILENAME AS "FileName"
     PROPERTY PROP-FILTER AS "Filter""

【メソッドのDATADIVISION】
WORKING-STORAGE SECTION.
   01 WK-DIALOGRESULT OBJECT REFERENCE
                      ENUM-DIALOGRESULT.

【メソッドの手続き部】
    SET  PROP-FILTER OF openFileDialog1 TO
                     N"*.txt|*.txt".
    INVOKE openFileDialog1 "ShowDialog"
           RETURNING WK-DIALOGRESULT.
    IF WK-DIALOGRESULT = PROP-OK OF ENUM-DIALOGRESULT
                                 THEN
        SET PROP-TEXT OF txtFileName
             TO PROP-FILENAME OF openFileDialog1
    END-IF.
```



4.3.3(3) OpenFileDialogクラスの使用例

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ 解説

リポジトリ段落には、OpenFileDialogクラスへの参照が自動的に追加されます。ShowDialogメソッドの戻り値のための、DialogResult列挙体の宣言と、OpenFileDialogクラスからファイル名を受け取るためのFileNameプロパティ、ファイルタイプを指定するFilterプロパティを宣言しています。

メソッドのDATA DIVISIONでは、ShowDialogメソッドの戻り値を受け取るための変数を宣言しています。

メソッドの手続き部では、ファイルを開くダイアログで使用するフィルタ(拡張子の指定)の設定後、ShowDialogメソッドを呼出し、OKがクリックされた場合、ファイル名をTextBoxコントロール(txtFileName)のテキストプロパティに代入する処理を記述しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.4 Windowsフォームから サブプログラムの呼出し

4.4.1 サブプログラムの構造

サブプログラムを作成する場合、呼出し元プログラムと同じプロジェクト内にサブプログラムを作成する方法と、呼出し元プログラムと別のプロジェクトとしてサブプログラムを作成する(サブプログラムを独立して作成する)方法があります。

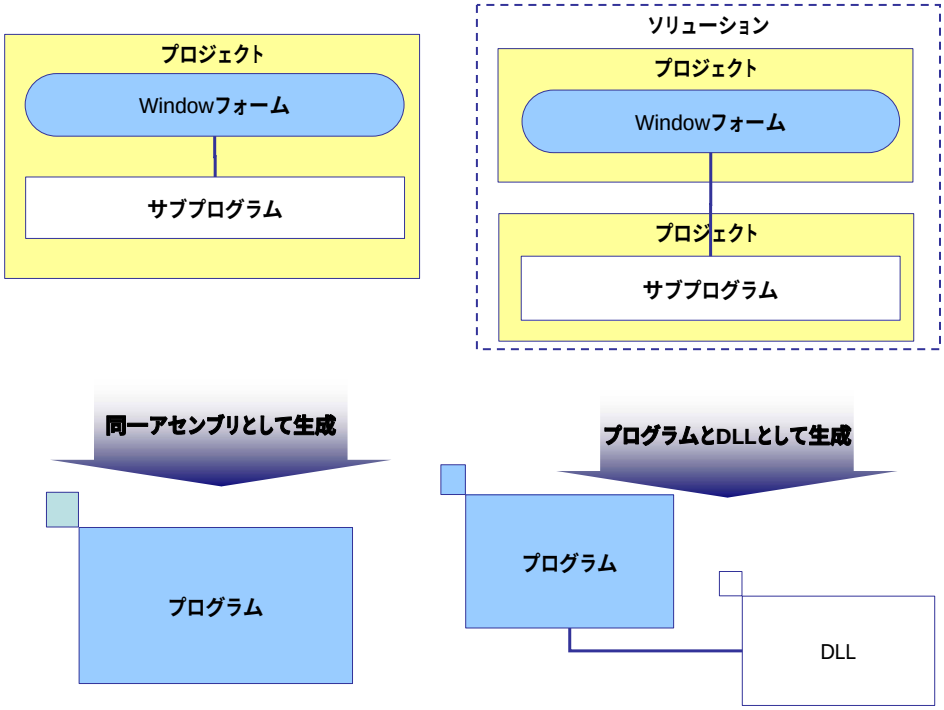


図 4.4.1 サブプログラムの構造

呼出し元と同じプロジェクトにサブプログラムを作成する場合、呼出し元と同じプログラムとして生成されるため、他のプログラムなどからの再利用が難しくなります。ただし、内部的にサブプログラムが結合されているため、実行時にオーバヘッドが発生することがありません。

サブプログラムを独立して作成した場合、独立したDLLとして作成されるため、複数のプログラムから呼出しが可能です。ただし、プログラムの実行時にサブプログラムであるDLLが呼び出されるため、サブプログラムの呼出しがオーバヘッドとなることがあります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.4.2 サブプログラムの呼出し方法

(1) 同一プロジェクトのサブプログラムを呼び出す

プロジェクトにサブプログラムが追加(新規作成)されている場合、プロジェクト内のCOBOLプログラムからCALL文でサブプログラム名を指定して呼び出すことができます。特別な設定は必要ありません。

ただし、開発環境上でサブプログラムを新規作成すると、PROGRAM-ID段落に、”プロジェクト名.ファイル名”の形式の外部名が付加されます。CALL文には正確な外部名を指定する必要があります。

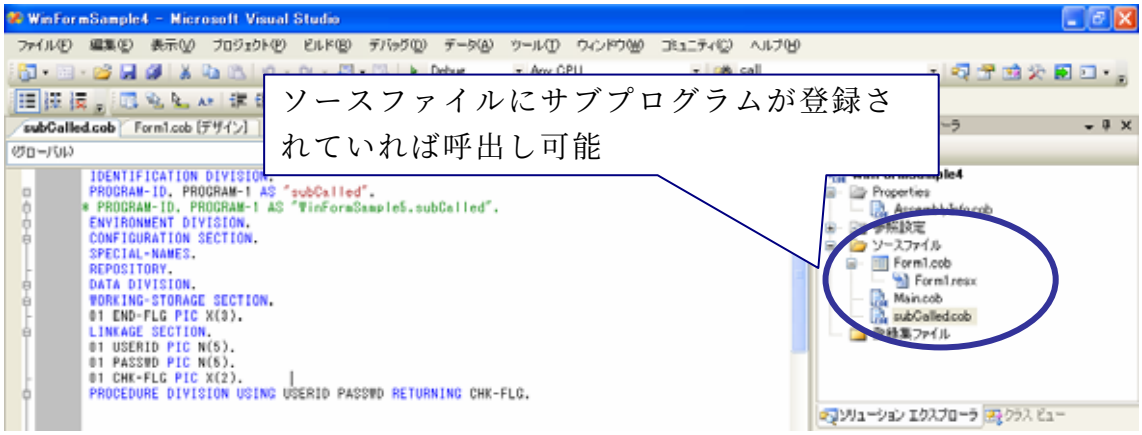


図 4.4.2 (1) プロジェクト内のサブプログラム

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) 独立したプロジェクトのサブプログラムを呼び出す

独立したプロジェクトのサブプログラムは事前にDLLとして作成されている必要があります。サブプログラムのDLLを呼び出すには、呼出し元の手続きを含むプロジェクトの参照設定にサブプログラムが格納されたDLLを追加する必要があります。同一プロジェクトのサブプログラムを呼び出す場合と異なるのは、この参照設定のみです。したがって、CALL文で簡単にサブプログラムを呼出すことができます。

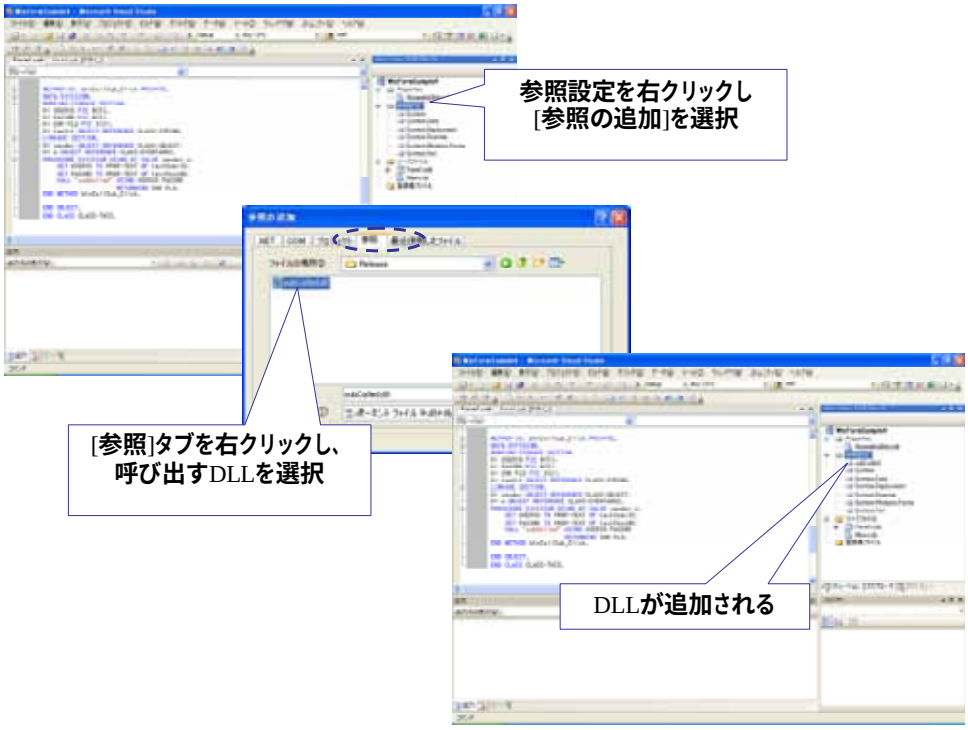


図4.4.2 (2) 参照設定の追加

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(3) サブプログラム呼出しのサンプル

次の例は、サブプログラムを呼び出すメソッドの例と呼び出されるサブプログラムの例です。

```
【サブプログラムを呼び出すメソッド例】
METHOD-ID. btnCallSub1_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 USERID PIC  N(5).
01 PASSWD PIC  N(5).
01 CHK-FLG PIC X(2).
01 tmpStr OBJECT REFERENCE CLASS-STRING.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
* ユーザIDとパスワードは5文字以内に制限しているとする。
    SET USERID TO PROP-TEXT OF txtUserID.
    SET PASSWD TO PROP-TEXT OF txtPassWD.

    CALL "subCalled" USING USERID PASSWD
                           RETURNING CHK-FLG.
* 外部名を明に指定している場合
*   CALL "WinFormSample5.subCalled"
*                          USING USERID PASSWD
*                          RETURNING CHK-FLG.
```

```
【呼び出されるサブプログラム(一部)】
IDENTIFICATION DIVISION.
PROGRAM-ID. subCalled.
* 外部名を明に指定している例
* PROGRAM-ID. subCalled AS "WinFormSample5.subCalled".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
REPOSITORY.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 END-FLG PIC X(3).
LINKAGE SECTION.
01 USERID PIC  N(5).
01 PASSWD PIC  N(5).
01 CHK-FLG PIC X(02).
PROCEDURE DIVISION USING USERID PASSWD RETURNING CHK-FLG.

 ・・・・・・・
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

4.5 Windowsフォームアプリケーションの作成

それでは、実際にWindowsフォームアプリケーションの作成を行ってみます。

4.5.1 プログラム概要

COBOLの索引ファイルを読み込んでCSV形式のファイルに変換するWindowsフォームアプリケーションを作成します。ファイルの指定には、コモンダイアログを使用しGUIで選択ができるようにします。

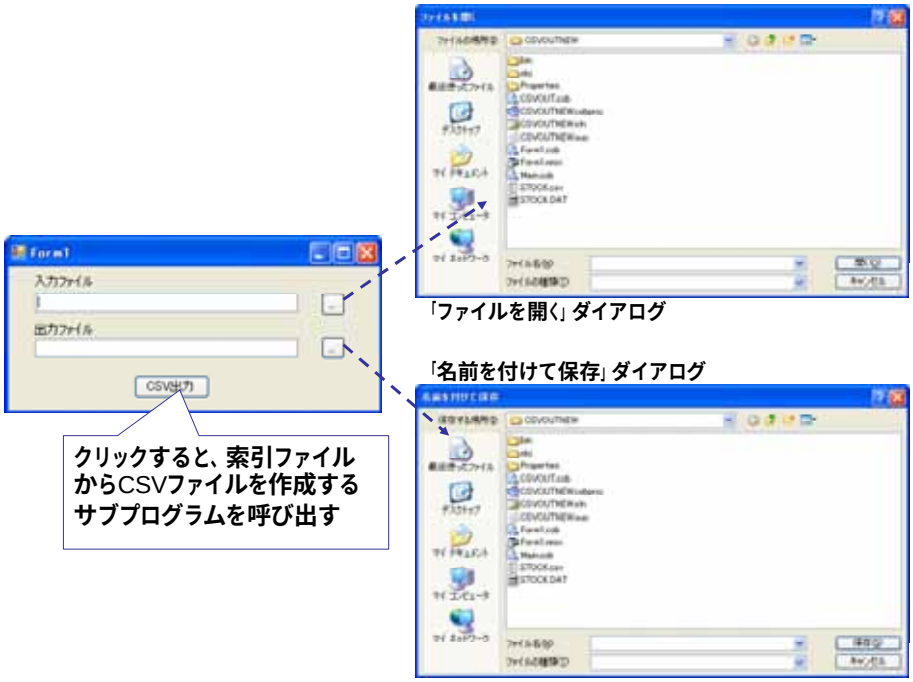


図4.5.1 Windowsフォームアプリケーションの実行イメージ

作成するWindowsフォームアプリケーションは、ボタンをクリックすると、2章で作成したコンソールアプリケーション(索引ファイルをCSVファイルに変換するプログラム)をサブプログラムとして活用し、Windowsフォームから呼び出せるようにします。

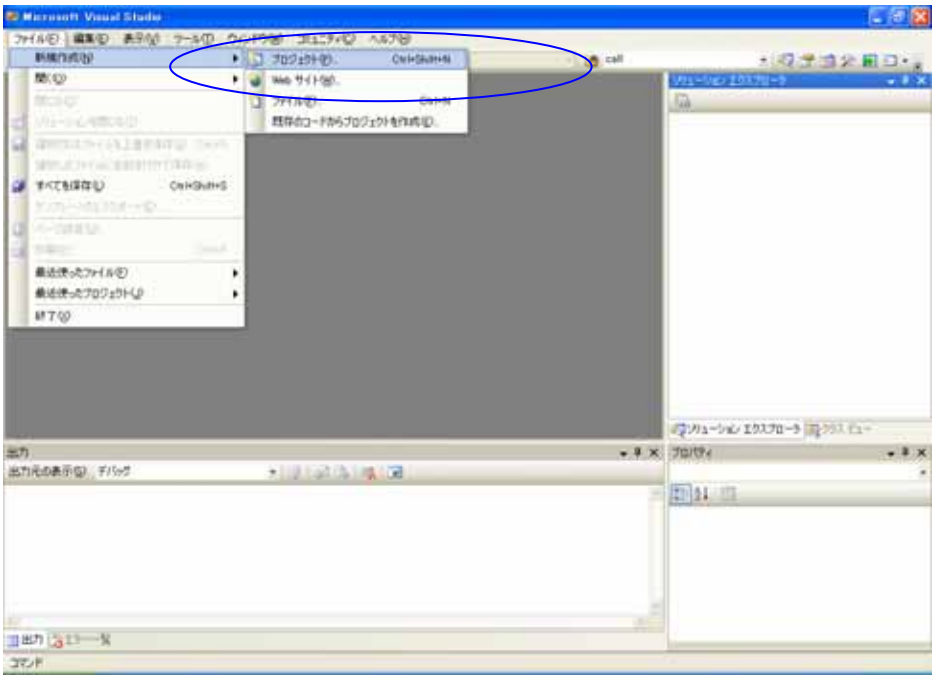
4.5.2 プロジェクトの作成

- **手順**1

  NetCOBOL for .NETを起動し、起動画面の[ファイル]メニューより

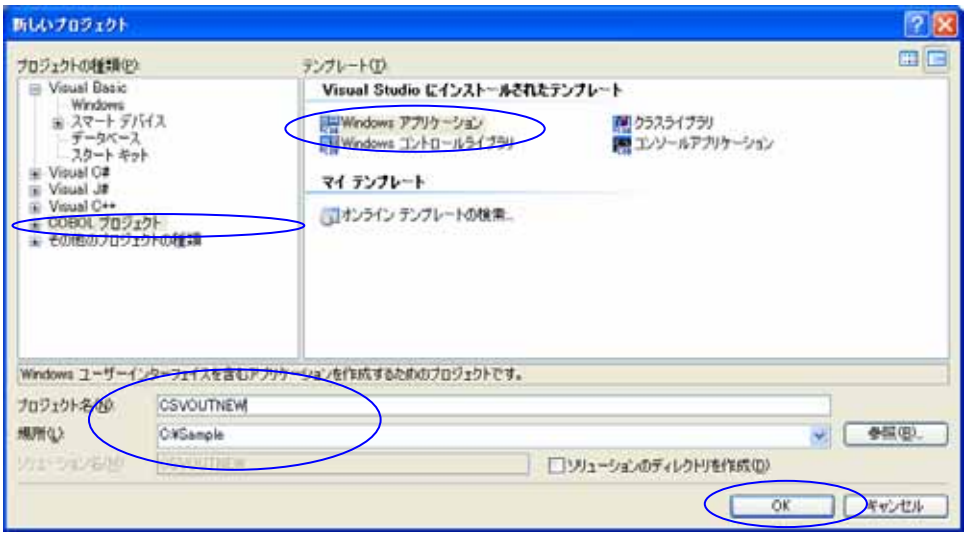
All Rights Reserved, Copyright FUJITSU LIMITED 2006

[新規作成]-[プロジェクト]を選択します。



- **手順**2

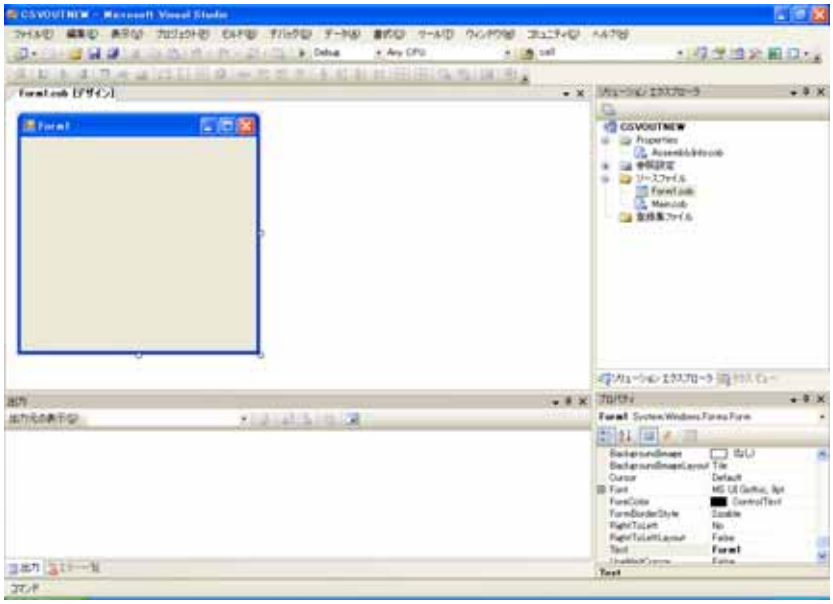
  [プロジェクトの種類]より「COBOLプロジェクト」、[テンプレート]フレームより「Windowsアプリケーション」を選択し、[プロジェクト名]、[ソリューション名]の両テキストボックスに「CNVFILEWIN」など任意の名称を入力し[場所]のテキストボックスに、プロジェクトの作成フォルダを指定後、「OK」ボタンをクリックします。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

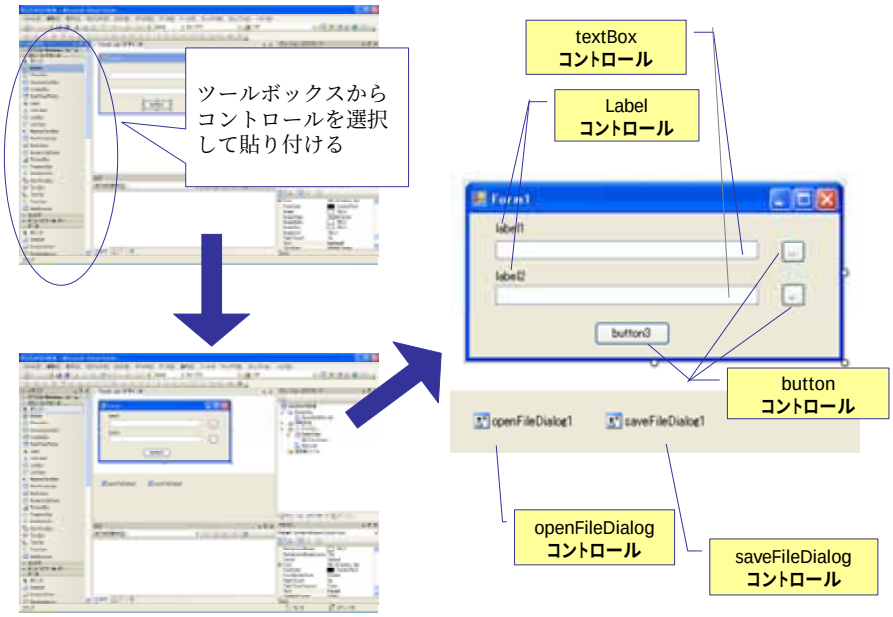
- **手順**3

  手順2で指定したテンプレートに従って、プロジェクトが自動生成されます(生成直後の画面イメージです)。Windowsフォームデザイナウインドウに初期状態のWindowsフォームが表示されています。



- **手順**4

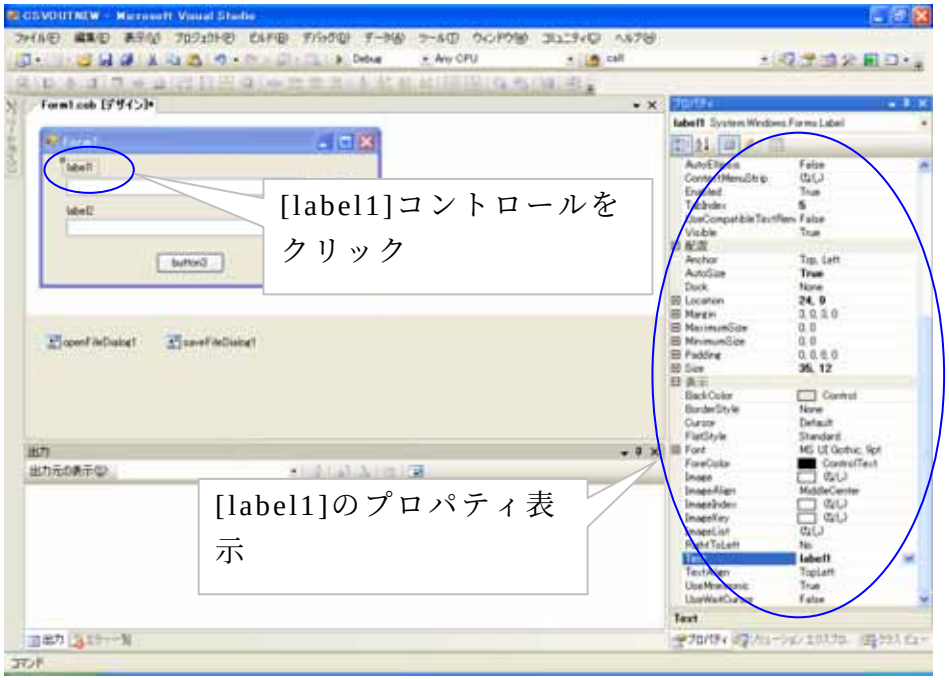
  Windowsフォーム内に、「ツールボックス」からコントロールを選択し、下図のように配置します。



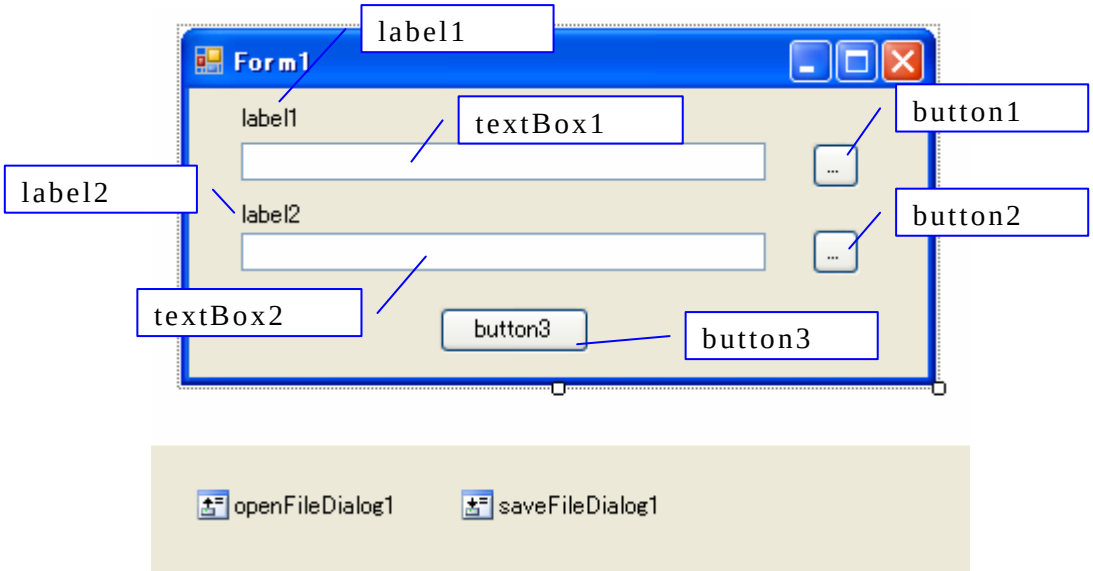
All Rights Reserved, Copyright FUJITSU LIMITED 2006

- **手順**5

コントロールの配置が完了したら、各コントロールのプロパティを設定します。プロパティの設定は、コントロールにクリックすると表示されるプロパティウィンドウで行ってください。



各コントロールに設定するプロパティは次のとおりです。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

コントロール	プロパティ	値
label1	Text	入力ファイル
label2	Text	出力ファイル
textBox1	Name	txtInfile
textBox2	Name	txtOutfile
button1	Name	btnOpen
	Text	"..."
button2	Name	btnSave
	Text	"..."
button3	Name	btnOut
	Text	"CSV出力"
openFileDialog1	InitialDirectory	"."
saveFileDialog1	InitialDirectory	"."

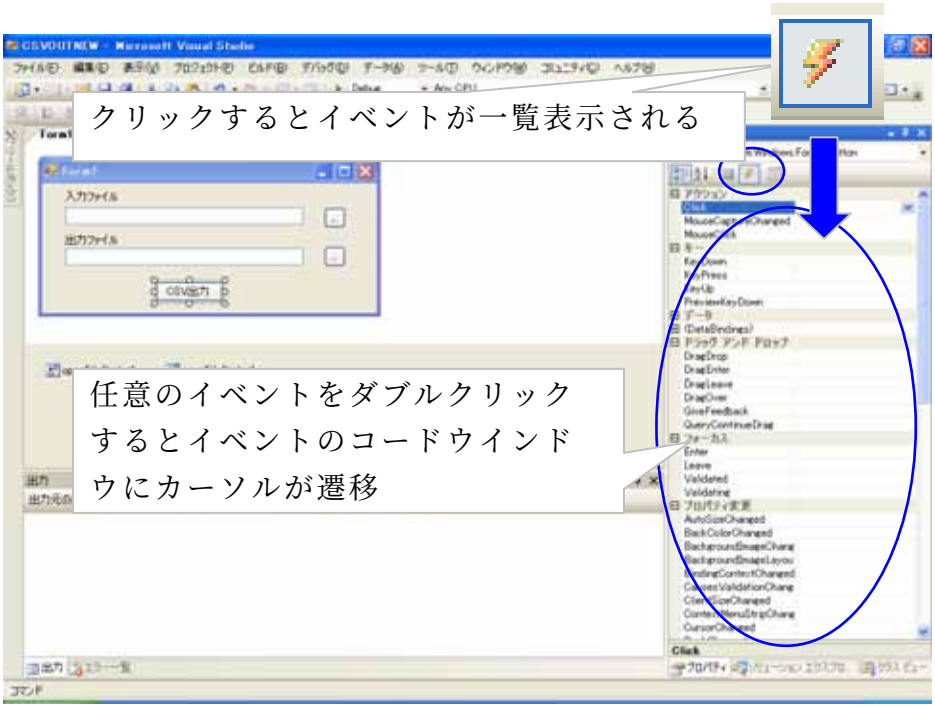
- **手順**6

  プロパティの設定が完了したら、コードエディタウィンドウで、コーディングを行います。

  コーディングが必要なのは、buttonコントロール(btnOpen、btnSave、btnOut)のClickイベントです。フォーム上のbuttonコントロールを直接ダブルクリックするとClickイベントのコードエディタにカーソルが遷移するので、処理を追加します。

  なお、他のイベントのコーディングを行う際は、下図のように対象となるコントロールを選択した状態で、プロパティウィンドウ内のイベントのアイコンをクリックするとイベント一覧が表示されます。ここで、該当のイベントをさらにクリックすると選択したイベントのコードエディタにカーソルが遷移します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



また、リポジトリ段落にクラス名やプロパティ名を宣言したり、データ部に作業領域を定義したりする必要があるかも知れません。なお、コントロールを設定するために必要なクラス名やプロパティ名の宣言はコントロールをフォーム上に配置したタイミングで自動的に生成されます。
次のソースを参考に、コーディングを行ってください。

■ btn0ut_Clickイベントに次の処理を追加

```
METHOD-ID. btn0ut_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  W-ENVVALUE  PIC X(240).
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    *>環境変数にファイルを設定
     DISPLAY "SYOPF" UPON ENV-NAME.
     SET W-ENVVALUE TO PROP-TEXT OF txtInfile.
     DISPLAY W-ENVVALUE UPON ENV-VALUE.

     DISPLAY "CSVFILE" UPON ENV-NAME.
     SET W-ENVVALUE TO PROP-TEXT OF txt0utfile.
     DISPLAY W-ENVVALUE UPON ENV-VALUE.

     *>ファイル変換呼び出し
   CALL "CNVFILE".
   INVOKE CLASS-MESSAGEBOX "Show"
          USING SELF N"処理が終了しました".
 END METHOD btn0ut_Click.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnOpen_Clickイベントに次の処理を追加

```
METHOD-ID. btnOpen_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-RET OBJECT REFERENCE ENUM-DIALOGRESULT.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
     *>ダイアログ表示
     INVOKE openFileDialog1 "ShowDialog" USING SELF
RETURNING W-RET.
     IF W-RET = PROP-OK OF ENUM-DIALOGRESULT THEN
       SET PROP-TEXT OF txtInfile TO PROP-FILENAME
 OF openFileDialog1
     END-IF.
END METHOD btnOpen_Click.
```

■ btnSave_Clickイベントに次の処理を追加

```
METHOD-ID. btnSave_Click PRIVATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-RET OBJECT REFERENCE ENUM-DIALOGRESULT.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    *>ダイアログ表示
    INVOKE saveFileDialog1 "ShowDialog"
           USING SELF RETURNING W-RET.
    IF W-RET = PROP-OK OF ENUM-DIALOGRESULT THEN
      SET PROP-TEXT OF txtOutfile TO
          PROP-FILENAME OF saveFileDialog1
END-IF.
END METHOD btnSave_Click.
```

なお、上記イベントの処理で必要なクラス、プロパティ、機能を追加します。

■ SPECIAL-NAMES句に次のソースを追加

```
SPECIAL-NAMES.

ENVIRONMENT-NAME  IS  ENV-NAME
ENVIRONMENT-VALUE IS  ENV-VALUE.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ REPOSITORY句に次の処理を追加

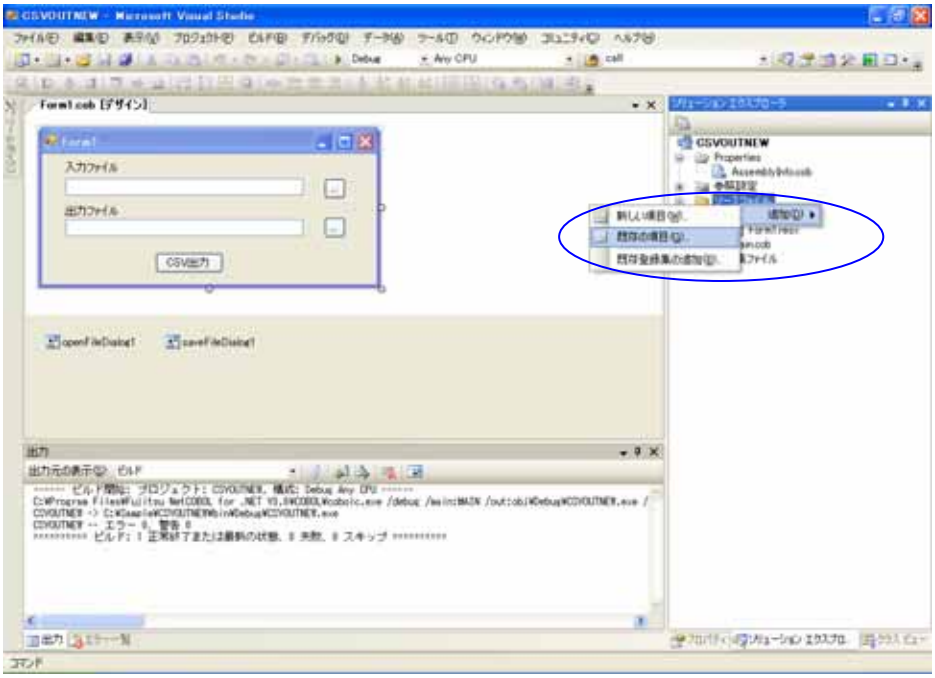
```
REPOSITORY.
                    :
ENUM ENUM-DIALOGRESULT AS
     "System.Windows.Forms.DialogResult"
                    :
CLASS CLASS-MESSAGEBOX AS
     "System.Windows.Forms.MessageBox"
        :
PROPERTY PROP-FILENAME AS "FileName"
PROPERTY PROP-OK AS "OK".
```

以上でソースの追加は完了です。

➢ **手順7**

次に、第2章で作成したファイル変換のコンソールアプリケーションのソースプログラムを、Windowsフォームから呼び出せるようにサブルーチンとして追加します。

[ソリューションエクスプローラ]ウィンドウより[ソースファイル]フォルダを右クリックし[追加]-[既存項目の追加]を選択し、表示されるダイアログボックスで、該当のプログラム(CSVOUT.cob)を指定してください。



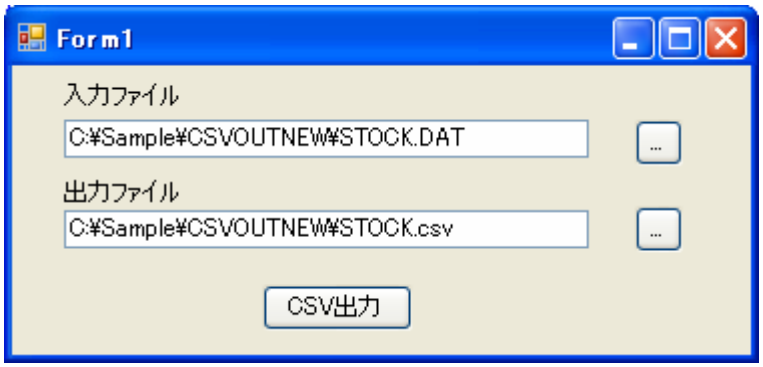
All Rights Reserved, Copyright FUJITSU LIMITED 2006

- **手順**8

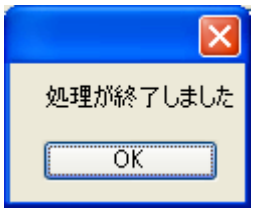
  [ビルド]メニューから[ソリューションのビルド]を実行して翻訳エラーなどがないか確認します。

- **手順**9

  ビルドが正常実行したらいよいよ実行です。[デバッグ]メニューから[デバッグなしで開始]をクリックするとWindowsフォームが起動します。入力ファイル、出力ファイルを指定して、[CSV出力]ボタンを押してください。

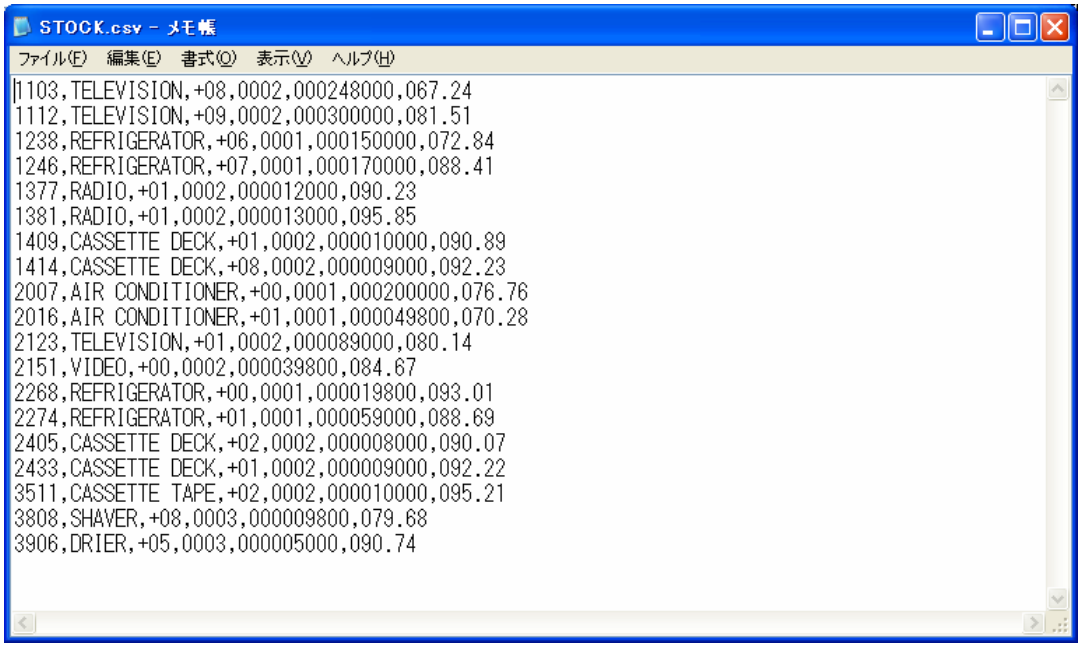


[CSV出力]ボタンを押すと次のメッセージボックスが表示されます。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

CSVファイルの出力結果は下図のとおりです。




```
1103,TELEVISION,+08,0002,000248000,067.24
1112,TELEVISION,+09,0002,000300000,081.51
1238,REFRIGERATOR,+06,0001,000150000,072.84
1246,REFRIGERATOR,+07,0001,000170000,088.41
1377,RADIO,+01,0002,000012000,090.23
1381,RADIO,+01,0002,000013000,095.85
1409,CASSETTE DECK,+01,0002,000010000,090.89
1414,CASSETTE DECK,+08,0002,000009000,092.23
2007,AIR CONDITIONER,+00,0001,000200000,076.76
2016,AIR CONDITIONER,+01,0001,000049800,070.28
2123,TELEVISION,+01,0002,000089000,080.14
2151,VIDEO,+00,0002,000039800,084.67
2268,REFRIGERATOR,+00,0001,000019800,093.01
2274,REFRIGERATOR,+01,0001,000059000,088.69
2405,CASSETTE DECK,+02,0002,000008000,090.07
2433,CASSETTE DECK,+01,0002,000009000,092.22
3511,CASSETTE TAPE,+02,0002,000010000,095.21
3808,SHAVER,+08,0003,000009800,079.68
3906,DRIER,+05,0003,000005000,090.74
```

以上が、Windowsフォームアプリケーションによるファイル変換プログラムの作成手順です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第5章 データアクセス

5.1 NetCOBOL for .NETのデータアクセス
5.2 ADO.NETクラスライブラリでのデータアクセス
5.3 埋込みSQLでのデータアアクセス
5.4 READ/WRITEインタフェース
5.5 データアクセスプログラムの作成
5.6 SQL CLRストアドプロシージャの作成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.1 NetCOBOL for .NETのデータアクセス

5.1.1 NetCOBOL for .NETでアクセスできるデータ

NetCOBOL for .NETには、READ/WRITEインタフェースと埋込みSQLというデータアクセス方法が用意されています。さらに、.NET Frameworkのクラスライブラリであるの.NETを利用することも可能です。したがって、大きくわけて3種類のデータアクセス方法があります。

ただし、NetCOBOL for Windowsなどでサポートされているプレコンパイラ(Oracle Pro*COBOLなど)はサポートされていませんので注意が必要です。

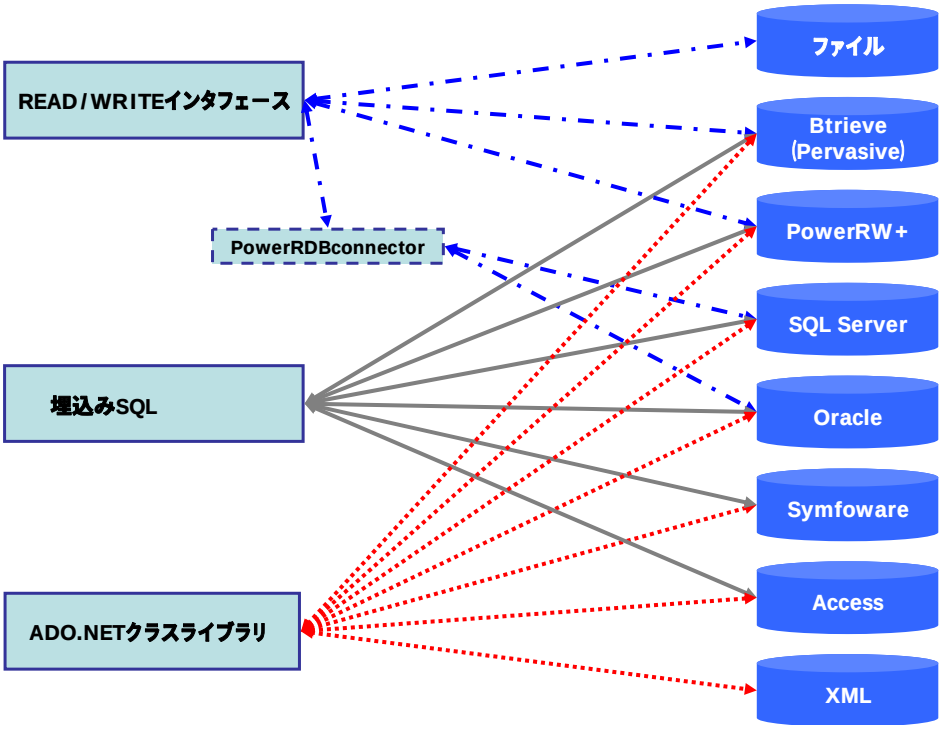


図5.1.1 データアクセスの種類

データベースによっては、複数のアクセス方法を選択できることになるため、システムを設計する時点でアクセス方法ごとの特徴や機能を把握しておく必要があります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.1.2 データアクセスの概要

(1) ADO.NETクラスライブラリ

ADO.NETクラスライブラリは.NET Frameworkで提供されるデータアクセス機能を提供します。多くの種類のデータベースやXMLファイルにアクセスでき、機能も豊富です。NetCOBOL for .NETからADO.NETの各クラスを利用してプログラミングすることが可能です。しかし、従来のCOBOL資産との互換性はありません。

(2) 埋込みSQL

.NET Frameworkデータプロバイダ（ADO.NET）経由あるいはWindowsで汎用的なデータベースアクセスソフトウェアであるODBCドライバを経由した接続方法です。NetCOBOL for .NET上ではSQL文を利用して記述します。ADO.NETプロバイダあるいはODBCドライバが提供されているデータベースであれば、原則として接続することができます。埋込みSQLは、NetCOBOL for Windowsでもサポートされている形式であるため、比較的互換性もあります。

(3) READ/WRITEインタフェース

もともとCOBOLデータファイルにアクセスするためのREAD/WRITEインタフェース(入出力文)ですが、NetCOBOL for .NETからREAD/WRITEインタフェースをそのまま利用できるデータベースとしてPowerRW+(富士通)とBtrieveがあります。

また、PowerRDBconnectorを利用することで、SQL ServerやOracleデータベースにもアクセスすることができます。従来のファイルアクセスと同様の記述でデータベースアクセスが可能なため、資産の互換性が高いのが特徴です。ADO.NETは.NET Frameworkで提供されるデータアクセスのためのクラスライブラリです。多くの種類のデータベースやXMLファイルにアクセスでき、機能も豊富です。NetCOBOL for .NETからADO.NETの各クラスを利用してプログラミングすることが可能です。しかし、従来のCOBOL資産との互換性はありません。

続いて、アクセス方法ごとの詳細について説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.2 ADO.NETクラスライブラリでのデータアクセス

5.2.1 ADO.NETの概要

ADO.NETは.NET Frameworkで提供されるデータアクセス機能を提供します。ADO.NETは非常に多機能で、様々な種類のデータにアクセスすることができます。また、接続を受け持つ部分とデータの操作を受け持つ部分が分離されており、データの種類に依存したプログラミングを行う必要がほとんどありません。

また、ADO.NETはWindowsアプリケーションやWebアプリケーションのコントロールと簡単に連携することが可能で、プログラミングの量を減らすことが可能です。

従来のデータ操作は、常に接続(セッション)をベースに行われてきました。しかし、クライアントとの接続がページ単位で切断されるのが通常のWebシステムや、Webサービス(SOAP)とよばれるコンポーネント間の結合が緩やかな処理方式などが広まるにつれ、接続を維持する形式のデータアクセス方式に無理が生じるようになりました。

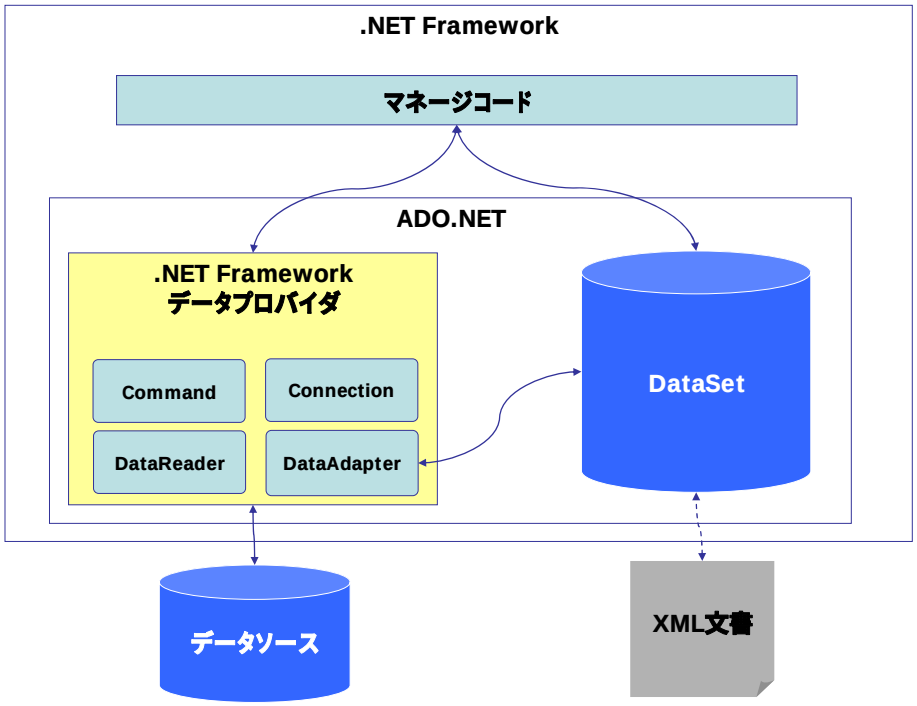


図5.2.1 ADO.NETの2つのコンポーネント

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ADO.NETは.NET Framework データプロバイダというデータソースへの処理を受けもつコンポーネントと、データセット(DataSet)というデータソースとの接続に依存しないコンポーネントに分離されており、WebシステムやWebサービスに最適なデータアクセスのための方式です。

■ .NET Frameworkデータプロバイダ

.NET Frameworkデータプロバイダは、データソースとの接続と比較的簡単なデータアクセス機能およびDataSetとの橋渡しの役割があります。

■ DataSet

DataSetは、データを格納するいわば箱と、データにアクセスするための豊富な機能が提供されるコンポーネントです。データベースとの接続部分は.NET Frameworkデータプロバイダの役割であり、DataSetはデータベースの種類に依存しません。

なお、ADO.NETはデータベース以外にもXMLファイルやテキストファイルなどさまざまなデータにアクセスできるため、アクセスする対象のデータをデータソースと呼んでいます。

5.2.2 NetCOBOL for .NETとADO.NET

NetCOBOL for .NETからADO.NETを利用する場合、埋込みSQLを利用する方法とADO.NETクラスライブラリを利用する方法があります。埋込みSQLを利用する方法は、「5.3 埋込みSQLでのデータアクセス」を参照して下さい。ここでは、ADO.NETクラスライブラリを利用する方法について説明します。

ADO.NETクラスライブラリを利用する場合、手続き型のCOBOL文法でデータにアクセスする方法とは異なるため、従来のCOBOL資産とのデータアクセスに関する互換性が失われます。

逆に、Visual Studioの他の言語との互換性が高くなります。例えば、VBで利用できないNetCOBOL for .NETのデータを、ADO.NETのデータ(DataSet)として作成しなおし、VBに渡すなどの処理が可能になります。また、READ/WRITEや埋込みSQLにない豊富なADO.NETの機能が利用でき、接続できるデータの種類や接続パターンが増えるなど、データアクセスの選択肢が増えるのも大きな利点です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

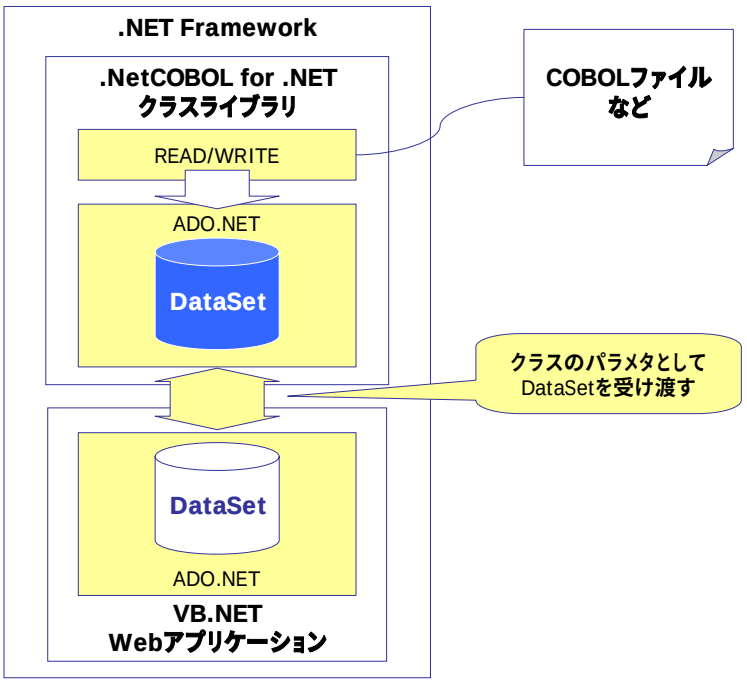


図5.2.2 DataSetを使用したデータ連携イメージ

例えば図の連携イメージでは、WebアプリケーションをVBで作成し、データのCOBOLファイルはクラスライブラリ化したNetCOBOL for .NETでアクセスしています。Webアプリケーションとクラスライブラリ間のデータの連携にDataSetを使用します。DataSetはWebアプリケーションのコントロールとも簡単に連携できるため、Webアプリケーション側で複雑なデータ処理を行う必要はありません。

5.2.3 .NET Frameworkデータプロバイダの種類

.NET Frameworkデータプロバイダは、プログラムとデータソースを、結ぶ橋渡しを行います。したがって、データソースとの接続方法やデータソースの種類ごとに、4種類の.NET Frameworkデータプロバイダが用意されています。また、Oracle社などのデータベースベンダが、専用の.NET Frameworkデータプロバイダを提供している場合もあります。

次に、各データプロバイダの特徴について説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

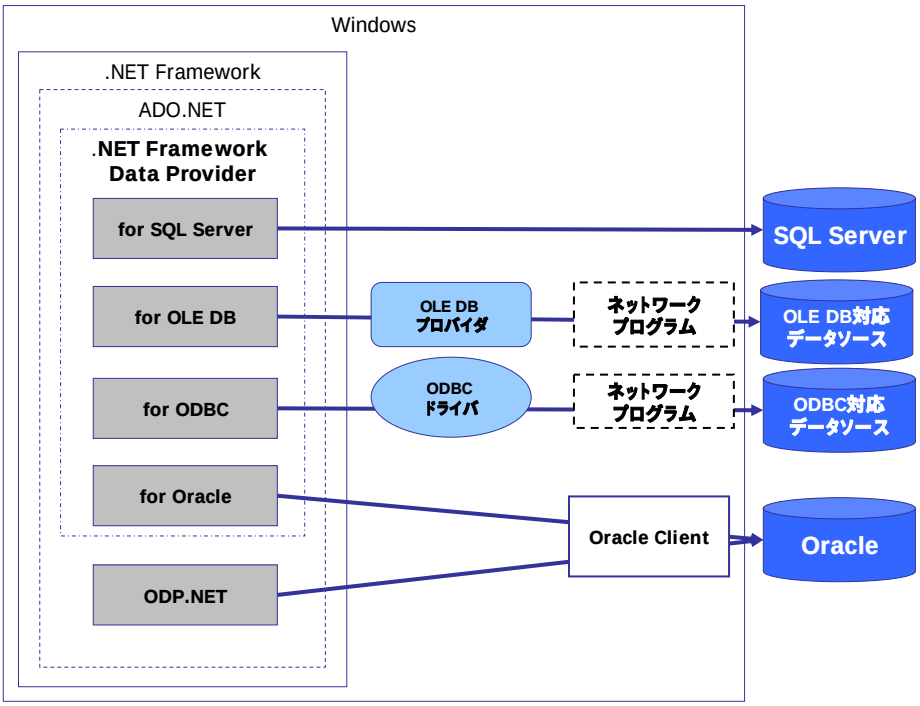


図5.2.3 .NET Framework データプロバイダとデータソース

(1) .NET Framework Data Provider for SQL Server

Microsoft SQL Server V7.0以降に接続するための専用のデータプロバイダです。.NET Framework Data ProviderとSQL Serverの間は、独自のプロトコルで接続されるため、高速な処理が期待できます。

SQL Serverへの接続は、.NET Framework Data Provider for OLE DBや.NET Framework Data Provider for ODBCを利用することも可能です。しかし、これらの接続方法は、OLE DBやODBC Driverといったデータアクセスのための層(ミドルウェア)が必要になるため、処理速度が落ちメモリ使用量が増加します。

(2) .NET Framework Data Provider for OLE DB

COMベースのデータベースアクセス技術であるOLE DBの機能を利用するための.NET Frameworkデータプロバイダです。ADO.NET以前にはADO(ActiveX Data Object)がデータアクセス技術の中心でしたが、ADOによるデータアクセス技術で中核となっているのがOLE DBです。OLE DBはOLE DBプロバイダとよばれるデータベースごとのミドルウェアを介してデータベースに接続します。したがって、OLE DBプロバイダが提供されているデータソースであれば、.NET Framework Data

All Rights Reserved, Copyright FUJITSU LIMITED 2006

Provider for OLE DBを使用することで接続することができます。また、SQL Server 6.5以前の場合、.NET Framework Data Provider for SQL Serverが未サポートのため、OLE DBによる接続が必要となります。

(3) .NET Framework Data Provider for ODBC

Windowsにおいてもっとも汎用的に利用されているデータアクセス技術がODBCです。ほとんどのデータベースやデータソースでODBCドライバが提供されています。したがって、.NET Framework Data Provider for ODBCを利用することで、ほとんどのデータソースに接続することができます。ただし、ODBCは古い技術のため、機能やパフォーマンスで期待通りの成果を得られないケースもあるので注意が必要です。

(4) .NET Framework Data Provider for Oracle

Windowsを問わず、Solaris、Linuxなどで広く利用されているデータベースであるOracleに接続するための専用のデータソースです。接続のためにはOracle Client(Oracle Net)が必要となります。Oracle Client R8.1.7以降がサポートされています。したがって、Oracle Clientのバージョンが古い場合は、ODBCなどによる接続を検討する必要があります。

(5) Oracle Data Provider for .NET

Oracle社が提供する.NET Frameworkデータプロバイダです。Oracle社では、ADO.NETでOracleにアクセスする場合、Oracle Data Provider for .NETでの接続を推奨しています。なお、埋込みSQLを利用する場合は、この接続方法は選択できません。

5.2.4 .NET Framework データプロバイダのクラス

.NET Frameworkデータプロバイダのクラスは、使用する.NET Frameworkデータプロバイダで実装されており、それぞれ接頭辞が異なります。しかし、クラスの体系と機能はほぼ同じですので、接頭辞を変更するだけで互換があります。ただし、.NET Frameworkデータプロバイダの種類によりサポートされるメソッドやプロパティが異なる場合もあるので完全には互換していません(実装により異なります)。

.NET Frameworkデータプロバイダには、Connection、Command、DataReader、DataAdapterの4つの代表的なクラスがあります。それぞれのクラスの概要について次に説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(1) Connectionクラス

データソースへ接続するための機能を提供するクラスです。データベースへの接続はConnectionオブジェクトとして生成されます。Connectionオブジェクトインスタンスを生成する際に、データソースへ接続するための文字列(接続文字列)を指定します。Openメソッドで接続を開き、CloseメソッドやDisposeメソッドで接続を解放します。

(2) Commandクラス

データソースに対して、SQLステートメント(Transact-SQLステートメント)やストアドプロシージャを呼び出すためのクラスです。行を返すSQL(SELECT)を発行するExecuteReaderメソッド、行を返さないSQL(INSERT、DELETE、UPDATEなど)を発行するExecuteNonQueryメソッドなどがあります。

ExecuteReaderメソッドは、結果をDataReaderオブジェクトとして生成します。

(3) DataReaderクラス

データソースから前方向にデータを読み込むための機能を提供します。DataReaderオブジェクトを作成するには、CommandクラスのExecuteReaderメソッドを使用します。

レコードを次の行に位置づけるReadメソッド、カレント行の指定した列のデータを取得するメソッドはGetの後にデータの型名が付いたメソッドが用意されています(GetString、GetInt32など)。また、実装ごとにデータソースのデータ型に応じたメソッドが用意されているケースもあります(GetSQLInt32、GetOracleNumberなど)。

(4) DataAdapterクラス

DataSetへデータソースのデータを格納したり、DataSetのデータをデータソースへ反映したりするための機能を提供するクラスです。

データソースのデータをDataSetに反映させるFillメソッド、DataSetのデータをデータソースに反映させるUpdateメソッドなどが用意されています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.2.5 .NET Frameworkデータプロバイダの使用例

次のサンプルは、.NET Framework Data Provider for SQL Serverを使用し、SQL Serverのテーブル(STOCK)のコピーをDataSetに作成しています。他の種類の.NET Frameworkデータプロバイダを使用する場合も基本的な手順は同じです。

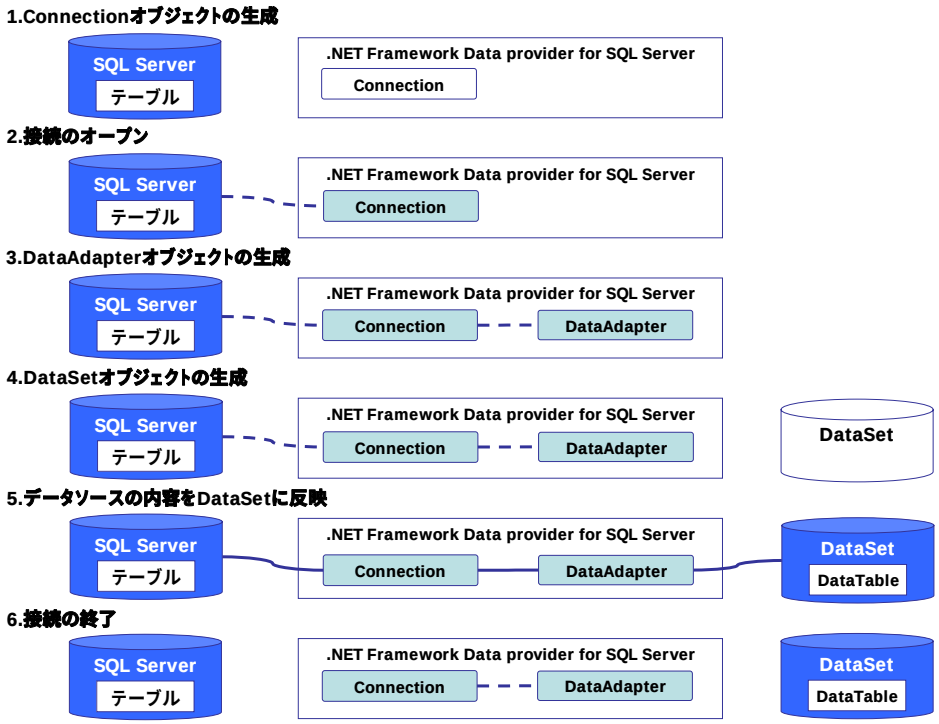


図5.2.4 サンプルの動作イメージ

1. Connectionオブジェクトの生成
   ConnectionクラスのNewメソッドを使用してConnectionオブジェクトを作成します。接続文字列を指定することもできますが、接続はオープンされません。Connectionオブジェクト生成後、ConnectionStringプロパティに接続文字列を設定することも可能です。

2. 接続のオープン
   ConnectionオブジェクトのOpenメソッドを使用して接続をオープンします。Openを省略した場合、DataAdapterのFillメソッドの使用時に自動的に接続がオープンされます。

3. DataAdapterオブジェクトの生成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

DataAdapterクラスのNewメソッドを使用してDataAdapterオブジェクトを生成します。この時点で、パラメタとしてSQL文やConnectionオブジェクトを指定することができます。もしくは、DataAdapterオブジェクト生成後に、Commandクラスを使用してSelectCommandプロパティなどで値を設定することも可能です。

4. DataSetオブジェクトの生成
   DataSetクラスのNewメソッドを使用してDataSetオブジェクトを生成します。DataSetオブジェクトを生成しただけでは、データは格納されません。

5. データソースの内容をデータセットに反映
   DataAdapterのFillメソッドを使用して、データソースであるSQL ServerのデータをDataSetに反映します。接続が開かれていない場合は、自動的に接続を開き、処理を行ったあとに自動的に接続を終了します。

6. 接続の終了
   Openメソッドで接続を開いている場合、ConnectionオブジェクトのCloseメソッドを使用して接続を終了します。

■ サンプルリスト

```
*リポジトリ段落
 REPOSITORY.
      ・・・・
     CLASS CLASS-DATASET AS
                "System.Data.DataSet"
     CLASS CLASS-DATATABLE AS "System.Data.DataTable"
     CLASS CLASS-DATATABLES AS
                "System.Data.DataTableCollection"
      ・・・・・
     CLASS CLASS-CONNECTION AS
                "System.Data.SqlClient.SqlConnection"
     CLASS CLASS-DBADAPTER AS
                "System.Data.SqlClient.SqlDataAdapter"
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
* メソッドのデータ部
 WORKING-STORAGE SECTION.
 01 Con OBJECT REFERENCE CLASS-CONNECTION.
 01 Adp OBJECT REFERENCE CLASS-DBADAPTER.
 01 tmpDS OBJECT REFERENCE CLASS-DATASET.
 01 tmpTables OBJECT REFERENCE CLASS-DATATABLES.
 01 tmpTable OBJECT REFERENCE CLASS-DATATABLE.
 01 tmpRows OBJECT REFERENCE CLASS-DATAROWS.
 01 tmpRow OBJECT REFERENCE CLASS-DATAROW.
 01 conStr OBJECT REFERENCE CLASS-STRING.
 01 sqlStr OBJECT REFERENCE CLASS-STRING.
 01 tmpStr OBJECT REFERENCE CLASS-STRING.

*メソッドの手続き部
 PROCEDURE DIVISION USING BY VALUE sender e.

* 接続文字列の編集
    SET conStr TO "Persist Security Info=False;Integrated
        Security=SSPI;database=sqltmp;server=netsv".

* Connectionオブジェクトの作成(接続文字列を指定)
    INVOKE CLASS-CONNECTION "NEW"
                 USING BY VALUE conStr RETURNING Con.

* 接続を開く
   INVOKE Con "Open".

* SQLの編集
    SET sqlStr TO "SELECT * FROM STOCK".

* DataAdapterオブジェクトの生成(SQL文とConnectionオブジェクト
* を指定)
    INVOKE CLASS-DBADAPTER "NEW"
                 USING BY VALUE sqlStr Con RETURNING Adp.

* DataSetオブジェクトの作成
    INVOKE CLASS-DATASET "NEW" RETURNING tmpDS.

* Fillメソッドによりデータソースの内容をDataSetに反映
    INVOKE Adp "Fill" USING tmpDS.

* データソースとの接続を終了
    INVOKE Con "Close".
```

5.2.6 DataSet

DataSetはADO.NETで非接続型のデータアクセス機能を提供する、いわばメモリ上のミニチュアデータベースです。.NET Frameworkデータプロバイダは、データソースの種類により実装が異なりますが、DataSetはデータソースの種類に依存しない一貫したプログラミングモデル(プログラミングインタフェース)を提供します。したがって、DataSetに対する処理は、どのようなデータベースであれ同じ記述となります（図5.2.5）。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

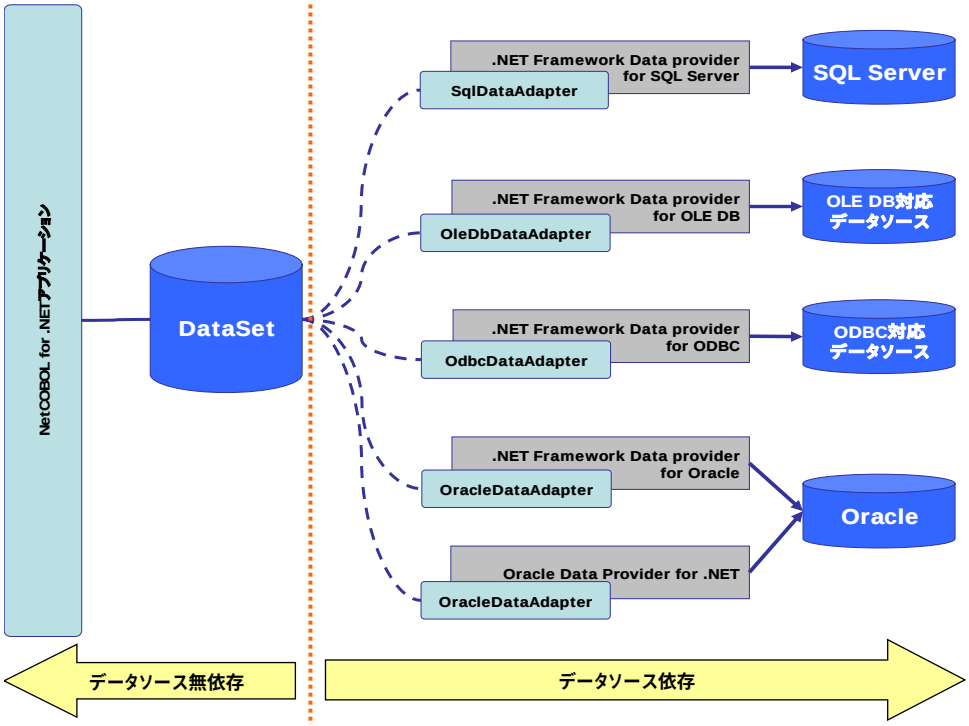


図5.2.5 DataSetと.NET Frameworkデータプロバイダのイメージ

DataSetは、階層化されたクラスから構成されており、データベースやMicrosoft Excelによく似ています。DataSetを利用するプログラムを作成する場合、この階層化されたイメージを理解しておく必要があります（図5.2.6）。また、常にオブジェクトはコレクションで管理されますので、コレクションの概念(「3.1.3 クラスの階層とコレクション」参照)をつかんでおくことが必要となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

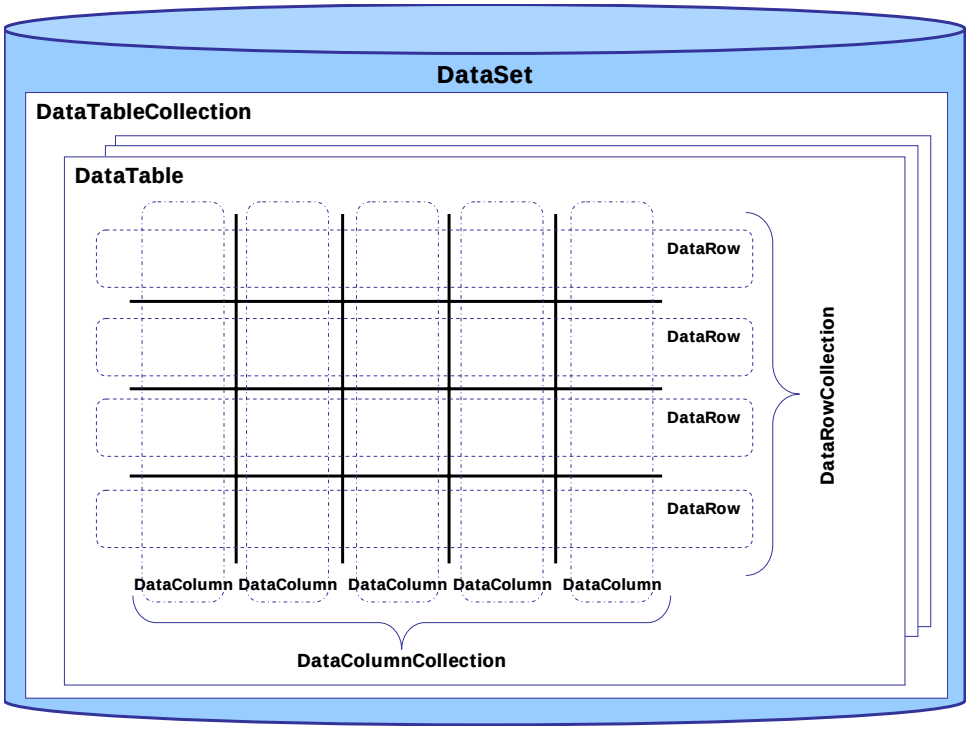


図5.2.6 DataSetのクラス構成のイメージ

5.2.7 DataSetのクラス

(1) DataSetクラス

DataSetの最上位のクラスであり、DataSetそのものです。

(2) DataTableCollectionクラス

DataSet内のテーブルのコレクションを表すクラスです。つまり、DataTableCollectionにはDataSet内の全てのDataTableオブジェクトが格納されています。DataSet.TablesプロパティでDataTableCollectionオブジェクトへの参照が返されます。

(3) DataTableクラス

DataTableはメモリ上のテーブルを表します。DataTableオブジェクトは.NET FrameworkデータプロバイダのDataAdapterクラスを使用して作成することもできますが、独自に作成することもできます。

DataTableオブジェクトの中で、DataColumnオブジェクトとDataRowオブジェクトがもっとも重要なコンポーネントです。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(4) DataColumnCollectionクラス、DataColumnクラス

DataColumnはDataTableの列を表します。DataTableにデータを格納するための箱のようなものをスキーマとよびますが、スキーマを作成するにはDataColumnCollectionにDataColumnオブジェクトを追加する必要があります。また、DataColumnCollectionオブジェクトの参照はDataTable.Columnsプロパティで返されます。

(5) DataRowCollectionクラス、DataRowクラス

DataRowはDataTable内の行を表します。DataTableのデータの操作はDataRowオブジェクトを利用して行われます。

5.2.8 DataSetの使用例

「5.2.5 .NET Frameworkデータプロバイダの使用例」に引き続くDataSetの使用例です。

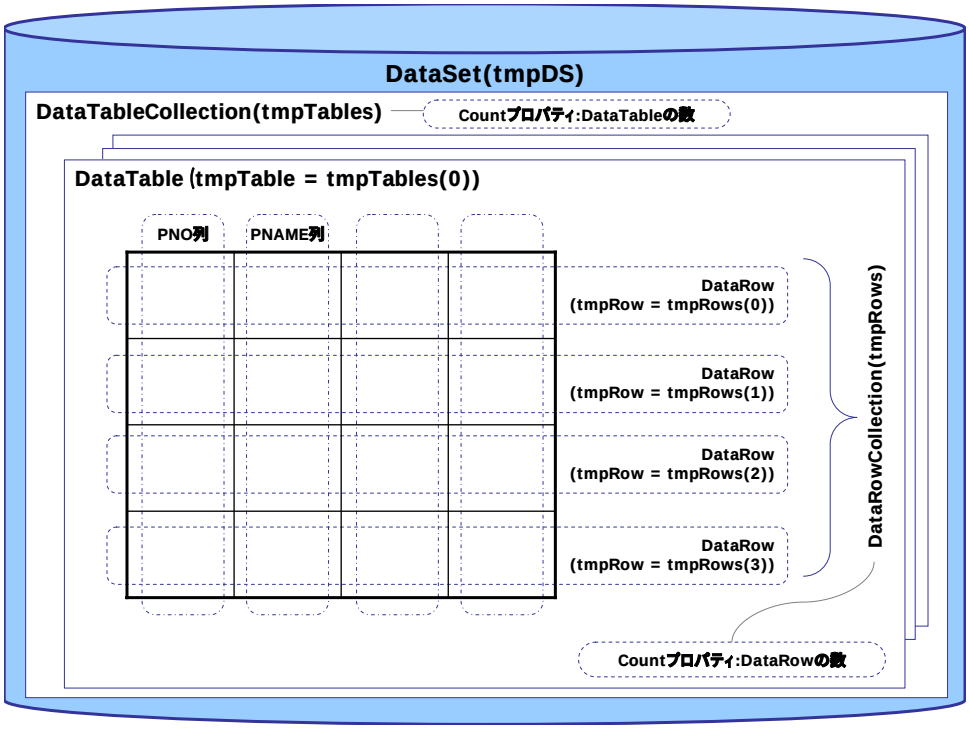


図5.2.7 DataSetからデータをアクセスするイメージ

この例では、作成済みのDataSetから、DataTableCollection、DataTable、DataRowCollection、DataRowの各オブジェクトを使用して、目的のデータを取得するまでの手順を紹介しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ サンプルリスト

```
*リポジトリ段落(差分のみ)
 REPOSITORY.
      ・・・・
     CLASS CLASS-DATAROW AS "System.Data.DataRow"
     CLASS CLASS-DATAROWS AS
                "System.Data.DataRowCollection"
     CLASS CLASS-DATASET AS "System.Data.DataSet"
     CLASS CLASS-DATATABLE AS "System.Data.DataTable"
     CLASS CLASS-DATATABLES AS
                "System.Data.DataTableCollection"
      ・・・・・
     PROPERTY PROP-COLUMNS AS "Columns"
     PROPERTY PROP-COUNT AS "Count"
     PROPERTY PROP-ROWS AS "Rows"
     PROPERTY PROP-TABLES AS "Tables"

* メソッドのデータ部(差分のみ)
 WORKING-STORAGE SECTION.
・・・・
 01 tmpTables OBJECT REFERENCE CLASS-DATATABLES.
 01 tmpTable OBJECT REFERENCE CLASS-DATATABLE.
 01 tmpRows OBJECT REFERENCE CLASS-DATAROWS.
 01 tmpRow OBJECT REFERENCE CLASS-DATAROW.
 01 tmpStr OBJECT REFERENCE CLASS-STRING.
 01 tmpObj OBJECT REFERENCE CLASS-OBJECT.
・・・

* メソッドの手続き部(差分のみ)

* データセットのTablesプロパティから
* DataTablesCollectionオブジェクトを生成
    SET tmpTables TO PROP-TABLES OF tmpDS.

* DataTableCollectionオブジェクトのCountプロパティを取得
    MOVE PROP-COUNT OF tmpTables TO itemCount.

* Countプロパティが0であれば
    IF itemCount = 0 THEN
      SET tmpStr TO "データがありません"
      INVOKE CLASS-MESSAGEBOX "Show" USING BY VALUE
tmpStr
      EXIT METHOD
    END-IF.

* DataTableCollectionオブジェクトの最初(0番目)の
* DataTableオブジェクトを取得
    SET tmpTable TO tmpTables::"get_Item"(0)

* DtaTableオブジェクトのRowsプロパティから
* DataRowCollectionオブジェクトを取得
    SET tmpRows TO PROP-ROWS OF tmpTable.

* DataRowCollectionオブジェクトのCountプロパティを取得
    MOVE PROP-COUNT OF tmpRows TO itemCount.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
* 行数分(Countプロパティｎ値)ループ
    PERFORM VARYING I FROM 0 BY 1 UNTIL I >= itemCount


* I行目の行データ(DataRowオブジェクト)を取得
      SET tmpRow TO tmpRows::"get_Item"(I)

* DataRowオブジェクトから列名から、"PNO"のデータ(オブジェク
ト)を取得
      SET tmpObj TO tmpRow::"get_Item"("PNO")

* DataRowオブジェクトから列名から、"PNAME"のデータ(オブジ
ェクト)を取得
      SET tmpObj TO tmpRow::"get_Item"("PNAME")

  ・・・・
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.3 埋込みSQLでのデータアクセス

5.3.1 埋込みSQLのイメージ

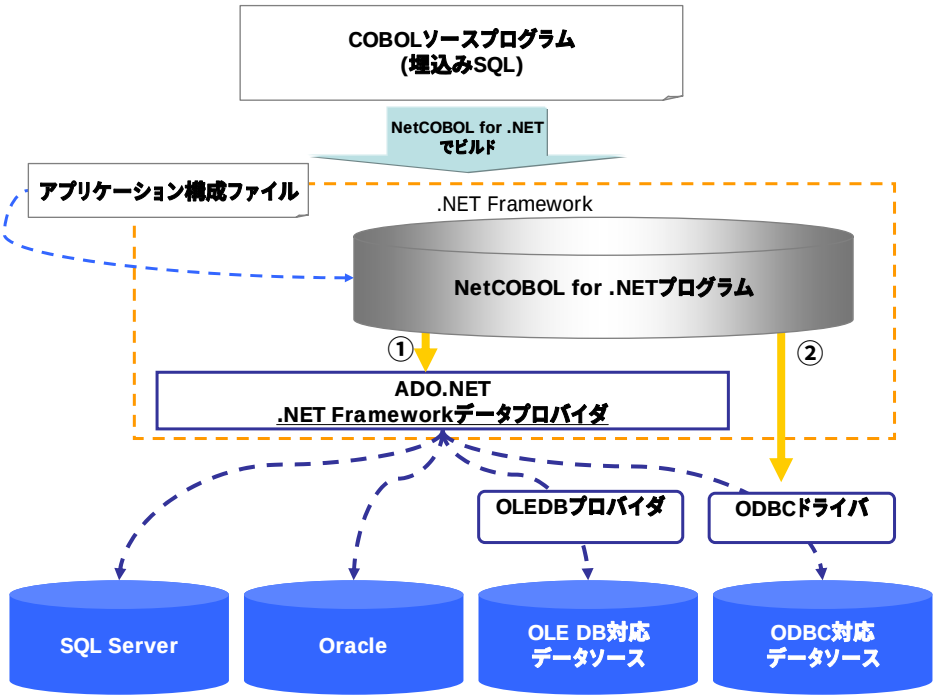


図5.3.1 埋込みSQLのイメージ図

埋込みSQLでのデータベース接続では、.NET Frameworkデータプロバイダを利用して接続する方法（図5.3.1 ①）とODBCドライバを利用して接続する方法（図5.3.1 ②）があります。.NET Frameworkでは.NET Frameworkデータプロバイダを利用して接続する方法が一般的です。ここでは.NET Frameworkデータプロバイダを利用して接続する方法について説明します。

データベースに接続するための定義については、アプリケーション構成ファイルに記述します。プログラムは実行時にアプリケーション構成ファイルを優先します。したがって、データベースの接続先に依存しないCOBOLソースプログラムを作成することが可能です。

データベースの処理手続きは、COBOLソースプログラムの中にSQL先頭子(EXEC SQL)とSQL終了子(END-EXEC)で囲んで記述します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.3.2 埋込みSQLプログラム作成手順

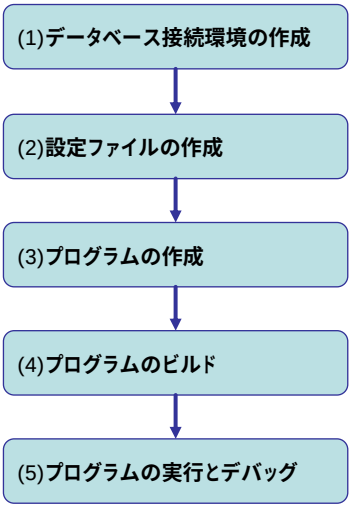


図5.3.2 プログラム作成手順

(1) データベース接続環境の作成

埋込みSQLのプログラムは、実行時にはじめて.NET Frameworkデータプロバイダを利用してデータベースに接続します。したがって、データベース接続環境の作成や設定などはいつ行ってもかまいません。

しかし、環境設定で作成した接続名をプログラム内で使用するケースもあるため、できればデータベース接続の環境設定から順に行ったほうが確実です。

(2) 設定ファイルの作成

アプリケーション構成ファイルに、プログラムが利用するデータソースやユーザなどの情報を登録します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(3) プログラムの作成

COBOLソースプログラムのSQL先頭子とSQL終了子の間に埋込みSQLによるデータベース処理手続きを記述します。大きく分けて次の処理が可能です。

■ コネクションの管理(接続、変更、切断)

コネクションとは、データベースをアクセスするためにクライアントとサーバの間を結んだ接続関係のことです。この接続関係を結ぶことにより、クライアントからサーバのデータベースをアクセスするSQL文を実行することができます。
コネクションを接続するにはCONNECT文、コネクションを変更するにはSET CONNECTION文、コネクションを切断するにはDISCONNECT文を使用します。

■ データ操作(参照、更新、変更、削除など)

データを操作するためには、SELECT文、INSERT文、UPDATE文およびDELETE文などのSQL文を利用します。カーソルを定義し、FETCH文によってデータを取り出すこともできます。
これらのSQL文を動的に生成することも可能です(動的SQL)。動的SQL文では、NetCOBOL for .NETによるSQL文法のチェックは行いません。したがって、データベースに依存するSQL文を使用することも可能となります。ただし、使用するSQL文がデータベースに依存する場合、NetCOBOL for .NETでは動作保証していませんので、利用者側の責任において利用することとなります。

■ ストアドプロシージャの呼出し

ストアドプロシージャとは、データベースに登録された処理手続きのことです。通常ストアドプロシージャは、データベース依存のSQL文で記述します。一般的には、ストアドプロシージャを利用することで、「処理手続きの実行速度の向上」、「クライアントーサーバ間の通信負荷の減少」、「 開発/保守の生産性の向上 」、「セキュリティの向上」などの効果があります。
NetCOBOL for .NETではCALL文でストアドプロシージャを呼び出すことができます。
なお、SQL Server 2005ではストアドプロシージャをCOBOLで記述することができます。詳細は「5.6 SQL CLRストアドプロシージャの作成」で説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ トランザクション処理

トランザクションは、データベースに対するデータ操作の一貫性を保証する単位です。 トランザクションは、最初のSQL文を実行したときに開始され、COMMIT文またはROLLBACK文を実行したときに終了します。

なお、NetCOBOL for .NETの埋込みSQLでは、SQL情報の@SQL_COMMIT_MODEをMANUAL（デフォルト）に設定することにより、接続と同時にトランザクションが開始されます。

(4) プログラムのビルド

ビルド時に、埋込みSQL用の特別な翻訳オプションやリンクオプションは必要ありません。

なお、動的SQLでないSQL文はビルド(翻訳)時にチェックされます。しかし、動的SQLはチェックが行われませんので、ビルド時に文法エラーを検出することができません。

(5) プログラムの実行とデバッグ

埋込みSQLを使用したプログラムを実行する場合は、アプリケーション構成ファイルに記述された接続情報に従ってデータベースに接続します。そのため、データベースが参照できない場合、プログラムのソースとともに設定ファイルの内容を確認する必要があります。

また、動的SQLは、プログラムが実行される場合に初めて利用されます。動的SQLは翻訳されずそのままデータベースに渡されるため、動的SQLに誤りがある場合、データベースのエラーからエラーの原因を確認する必要があります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.4 READ/WRITEインタフェース

5.4.1 アクセスするデータを切り替える仕組み

READ/WRITEインタフェースを利用したプログラムでは、ファイルやデータベースの種類が異なってもプログラムの手続きはほぼ同一です。したがって、環境(COBOL初期化ファイルなど)を変更するだけで、アクセスするファイルやデータベースを切り替えることができます。ほとんどの場合、プログラム修正は不要です。

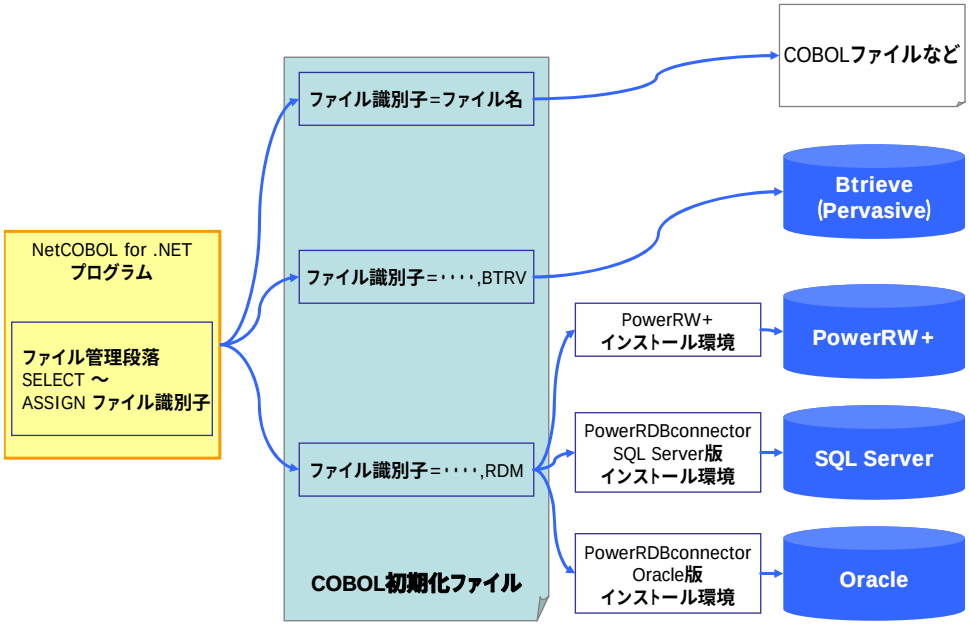


図5.4.1 データを切り替える仕組み

READ/WRITEインタフェースによるCOBOLファイルへのアクセスでは、ファイル管理段落(FILE-CONTROL段落)のSELECT句のASSIGN句でファイル識別子を指定し、COBOL初期化ファイルで、ファイル識別子と実ファイル名を対応付けるのが一般的です。

READ/WRITEインタフェースを使用したデータベースへのアクセスでは、ファイル識別子に指定するファイル名の後にキーワードを指定することで接続先を切り替えます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.4.2 データごとのアクセス方法

(1) Btrieveファイルの場合

ファイル名の後に「,BTRV」を付け加えるとBtrieveファイルへアクセスします。プログラムの記述はほとんど変更する必要はありませんが、SEPARATE指定のない符号付き外部10進項目を88コンソーシアム形式で扱う際に、型変換の処理が必要になる場合があります(データの変換機能が用意されています)。

(2) PowerRW+の場合

PowerRW+がインストールされている環境で、ファイル名に、アクセスするRDMファイルのファイル定義体を指定し、その後に「,RDM」を付け加えるとPowerRW+を利用したデータアクセスを行います。

PowerRW+は、もともとCOBOLからREAD/WRITEインタフェースによる接続を想定したデータベースですので、ほとんどのCOBOLデータ型に対応しています。したがって、ファイルをアクセスするプログラムをほぼ無修正で利用することが可能です。

複数のアプリケーションから一つのファイルをアクセスするケースが想定されるため、トランザクション機能が用意されています。トランザクション機能を組み込む場合は、ファイルアクセスの前後にトランザクションサブルーチンの呼び出し処理を追加する修正が必要です。

なお、PowerRW+はマルチスレッドの動作に対応していないため、ASP.NET(Webアプリケーション) で利用することはできません。

(3) PowerRDBconnector(SQL Server/Oracle)の場合

PowerRDBconnectorインストールされている環境で、ファイル名の後に「,RDM」を付け加えると、SQL ServerかOracleにアクセスすることができます(PowerRDBconnectorのインストールオプションで、SQL ServerとOracleのどちらを使用するか選択します)。

接続するデータベースサーバの情報は、PowerRDBconnectorの動作環境ファイルであるDBIO_ENVファイルに記述します。

データベースのデータ型とCOBOLのデータ型は、一度PowerRDBconnectorのデータに置き換えられ対応付けられます。したがって、ほとんどの場合、ファイルをアクセスするプログラムをほぼ無修正で利用することが可能です。

複数のアプリケーションから一つのファイルをアクセスするケースが

All Rights Reserved, Copyright FUJITSU LIMITED 2006

想定されるため、トランザクション機能が用意されています。トランザクション機能を組み込む場合は、ファイルアクセスの前後にトランザクションサブルーチンの呼び出し処理を追加する修正が必要です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.5 データアクセスプログラムの作成

5.5.1 プログラムの概要

(1) プログラムの実行イメージ

データアクセスプログラムの作成例として、5.5節では埋込みSQLを使用したプログラムを紹介します。この例では、データベースに格納された「商品テーブル」「在庫テーブル」「社員テーブル」にアクセスするサブプログラムを作成します。また、各クラスの動作を確認するためのコンソールアプリケーションも合わせて作成します。

ここでは、データベースとしてSQL Serverを利用していますが、ADO.NETに対応しているデータベースであればSQL Server以外のデータベースでも利用でき、プログラムの修正も必要ありません（アプリケーション構成ファイルの設定変更のみ必要）。ただし、次節「5.6 SQL CLRストアドプロシージャの作成」で説明するストアドプロシージャに関してはSQL Serverでしか利用できません。

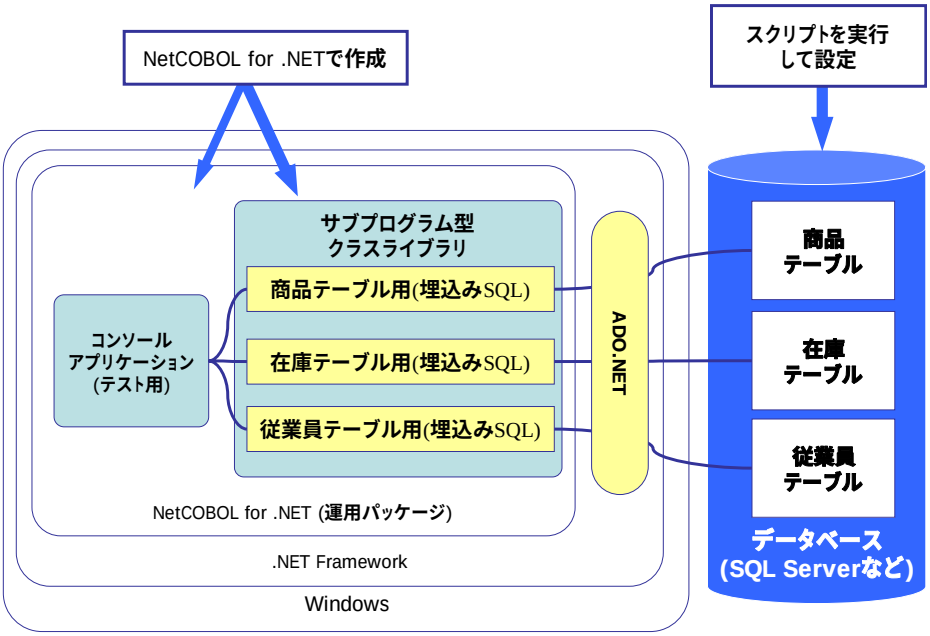


図5.5.1(1) プログラムの運用イメージ

なお、ここで作成したクラスライブラリは7章でWebアプリケーションを使用するときに再利用します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) プログラムの構成

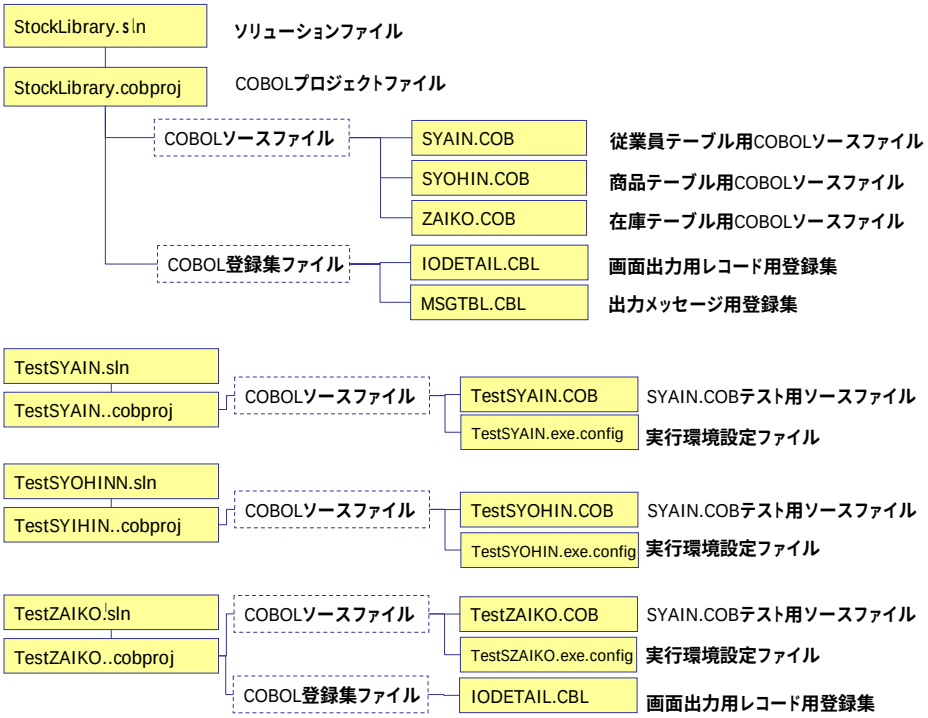


図5.5.1(2) サンプルプログラムの構成ファイル

データベースアクセスを行うためのサブプログラムを集めたクラスライブラリは、同じプロジェクトに作成します。また、プログラムの実行に必要な登録集もプロジェクトに登録します。

また、作成したサブプログラムを検証するため、サブプログラムごとに簡単なコンソールアプリケーションを作成します。各プログラムは独自のコンソールアプリケーションとして作成するため、ソースファイルとともに、実行時に必要な設定を格納した実行環境設定ファイルを作成します。実行環境設定ファイルには、埋込みSQLで使用する接続文字列設定などの情報を格納します。

(3) サブプログラムの仕様

3種類のサブプログラムの仕様について説明します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SYAIN.COBサブプログラム

社員コードとパスワードを受け取り、社員テーブルを読み込んで認証処理を行います。認証の結果は戻り値で返します。また、エラー発生時のメッセージもパラメタとして返します。

(パラメタと戻り値)

パラメタ	IN/OUT	型	意味
P-EmployeeCode	IN	N(6)	社員コード
P-Password	IN	N(6)	パスワード
P-Msg	OUT	N(20)	例外メッセージ
戻り値	IN/OUT		意味
R-ReturnCode	OUT	BINARY-LONG	認証OK:0 社員コード誤り:2 パスワード誤り:3

■ SYOHIN.COBサブプログラム

商品コードのみ格納された商品レコードを受け取り、商品テーブルを検索して、該当する商品コードのレコードを読み込み、商品レコードとして呼出し元に返します。

(パラメタと戻り値)

パラメタ	IN/OUT	型	意味
P-SYOHINREC	IN/OUT	集団項目	商品レコード
P-SYOCD	IN/OUT	PIC 9(5)	商品コード
P-SYONA	OUT	PIC N(20)	商品名
P-BAIKA	OUT	PIC 9(8)	売価
P-GENKA	OUT	PIC 9(8)	原価
P-Msg	OUT	N(20)	例外メッセージ
戻り値	IN/OUT	型	意味
R-ReturnCode	OUT	BINARY-LONG	正常終了:0 商品コードなし:5

■ ZAIKO.COBサブプログラム

指定された商品コードの在庫状況について、在庫テーブルを読み込み、画面出力用の1ページ分(10行)のレコードを生成し、呼出し元に返します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(パラメタと戻り値)

パラメタ	IN/OUT	型	意味
M-AREA	IN/OUT	IODETAIL.CBL	パラメタ用レコード
P-Start	IN/OUT	BINARY-LONG	開始行
P-Cnt	OUT	PIC 9(3)	件数
P-Next	OUT	PIC 9(1)	次レコード有無
P-Msg	OUT	N(20)	例外メッセージ
戻り値	**IN/OUT**	**型**	**意味**
R-ReturnCode	OUT	BINARY-LONG	正常終了:0 商品コードなし:5

5.5.2 プログラム作成手順

(1) データベース環境設定

利用するデータベースを構築します。今回はデータベース構築のためのスクリプト(cobdb.sql)を提供しています。SampleDataフォルダ内のsetupcobdb.batをコマンドラインから実行するとSQL Server 2005 Express上に構築されます。

(2) プロジェクトの作成とファイルの追加

サブプログラムが格納されるクラスライブラリ用のプロジェクトを作成し、サブプログラム用のCOBOLソースファイルなどを追加します。

(3) COBOL手続きの編集

埋込みSQLを使用してデータベースにアクセスするCOBOL手続きを編集します。

(4) プログラムの実行とデバッグ

プログラミングが完了したら通常は、ビルド後に実行/デバッグを行います。しかし、今回作成するのはクラスライブラリのため、そのままでは実行およびデバッグができません。したがって、クラスライブラリを呼び出すための簡単なコンソールアプリケーションを作成しながら、クラスライブラリの動作確認を行います。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.5.3 プロジェクトの作成とファイルの追加

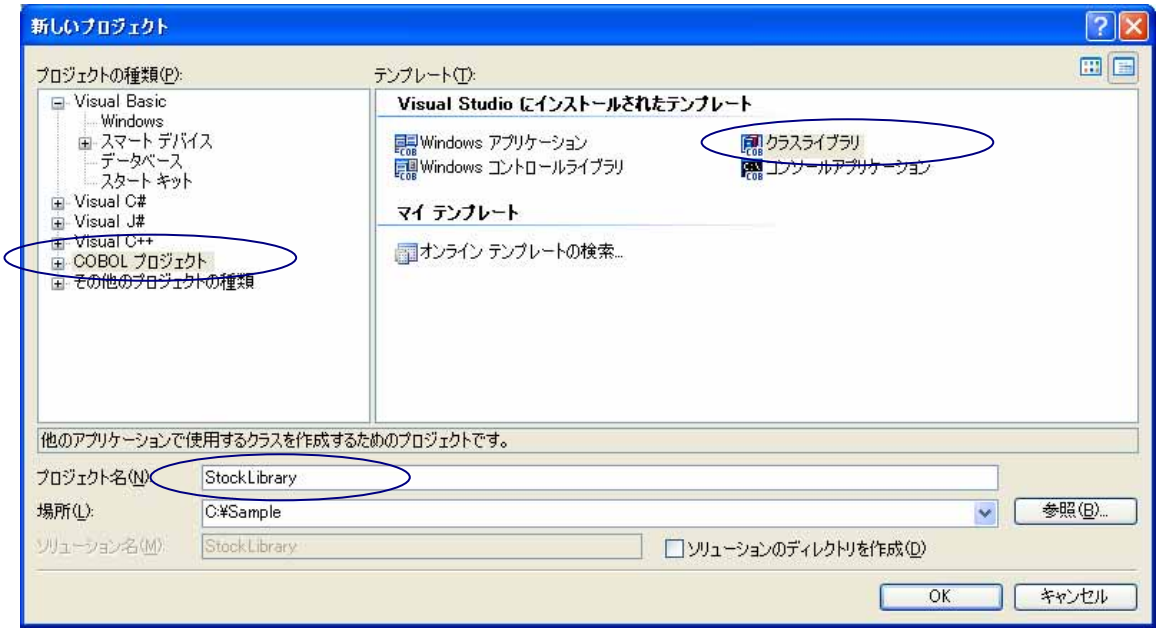
はじめに、データベースにアクセスするサブプログラムを管理するプロジェクトを作成します。

- **手順**1

  NetCOBOL for .NETを起動し、[ファイル]メニューより[新規作成]-[プロジェクト]を選択します。

- **手順**2

  [新しいプロジェクト]ダイアログボックスが表示されます。[プロジェクトの種類]より[COBOLプロジェクト]を選択し、[テンプレート]より[クラスライブラリ]を選択します。プロジェクト名に「StockLibrary」と入力しOKボタンを押します。



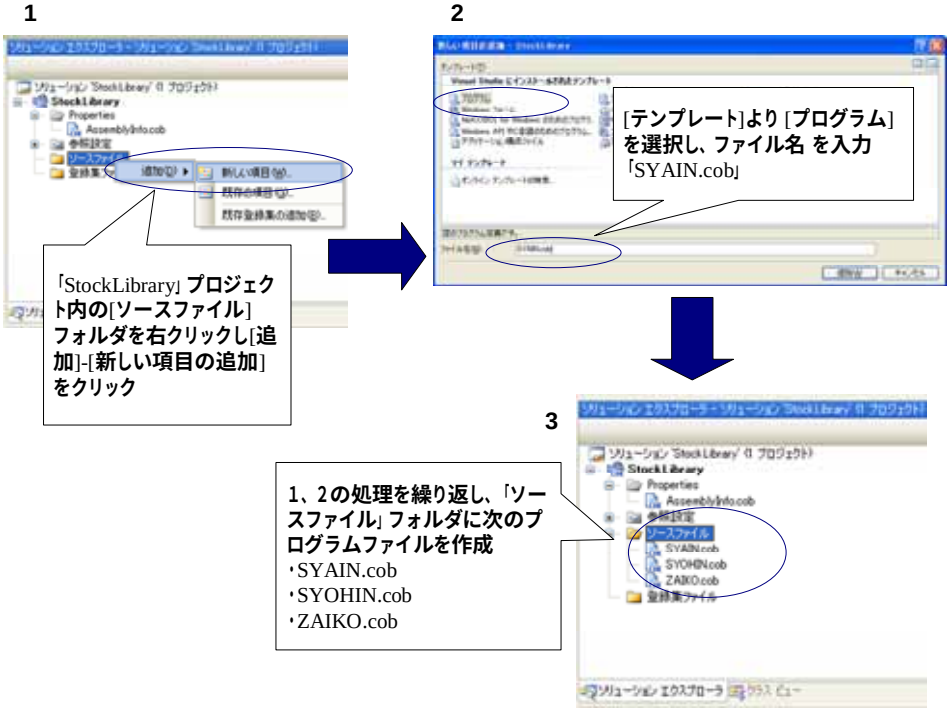
  作成したプロジェクトのソースファイルにはClass1.cobが自動生成されますが、今回は使用しません。ソリューションエクスプローラからClass1.cobを右クリックし、削除を選んで削除してください。

- **手順**3

  作成した[StockLibrary]プロジェクトにソースプログラムを追加します。[ソリューションエクスプローラ]の[StockLibrary]プロジェクトより[ソースファイル]フォルダを右クリックし[追加]-[新しい項目の追加]をクリックし、下図のようにソースプログラム(SYAIN.COB、

All Rights Reserved, Copyright FUJITSU LIMITED 2006

SYOHIN.COB、ZAIKO.COB)を新規に追加してください。



5.5.4 COBOL手続きの編集

次に追加したプログラムをコーティングします。
作成するプログラムは次の3本です。

✓ SYAIN.COB
✓ SYOHIN.COB
✓ ZAIKO.COB

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SYAIN.COB

(社員テーブルにアクセスし認証を行う)

```
*****************************************************
* プログラム名 : SYAIN
* 概要         : 社員テーブルにアクセスし認証を行う。
*
*****************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYAIN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.
*=====================================================
*  ホスト変数宣言
*=====================================================
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE     PIC X(5).
 01 SQLMSG       PIC X(128).

 01 SYAPF-REC.
   02  SYACD      PIC S9(9) COMP-5.
   02  SYANA      PIC N(20).
   02  PASS       PIC N(6).
   02  BUSYO      PIC S9(9) COMP-5.
   02  SDATE      PIC S9(9) COMP-5.
   02  SNAME      PIC N(20).

 01  S-SYACD      PIC S9(9) COMP-5.
     EXEC SQL END DECLARE SECTION END-EXEC.

 COPY MSGTBL.
 LINKAGE                          SECTION.
 01  P-EmployeeCode           PIC N(06).
 01  P-Password               PIC N(06).
 01  P-Msg                    PIC N(20).
 01  R-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION USING
                    P-EmployeeCode   *>社員コード
                    P-Password       *>パスワード
                    P-Msg            *>メッセージ
                    RETURNING
                    R-ReturnCode. *>戻り値(エラーコード)

     PERFORM MAIN-SEC.
     PERFORM END-SEC.
     EXIT PROGRAM.

 MAIN-SEC SECTION.
*    レコードがなかった場合の動作
     EXEC SQL
     WHENEVER NOT FOUND GO TO :P-NOTFOUND
     END-EXEC.

*    DB接続
     EXEC SQL CONNECT TO DEFAULT END-EXEC.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*    接続確認
     IF SQLSTATE = "00000" OR SQLSTATE = "01000" THEN
         MOVE FUNCTION DISPLAY-OF(P-EmployeeCode)
              TO S-SYACD

*        SQL実行
         EXEC SQL
         SELECT SYACD, PASS
             INTO :SYACD, :PASS FROM SYAPF
             WHERE SYACD = :S-SYACD
         END-EXEC
*        パスワード比較
         IF SQLSTATE = "00000" AND
            PASS = P-Password THEN
             MOVE 0 TO R-ReturnCode
         ELSE
             MOVE 3 TO R-ReturnCode
         END-IF
     END-IF.

 END-SEC SECTION.
*    DBクローズ
     EXEC SQL DISCONNECT DEFAULT END-EXEC.
*    メッセージ設定
     IF R-ReturnCode > 1 THEN
         MOVE MSG-DSP(R-ReturnCode) TO P-Msg
     END-IF.

 NOTFOUND-SEC SECTION.
 P-NOTFOUND.
*    社員コードが見つからなかった場合の処理
     MOVE 2 TO R-ReturnCode.
     PERFORM END-SEC.
 END PROGRAM SYAIN.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SYOHIN.COB

(商品テーブルにアクセスし、商品レコードの取得を行う)

```
*******************************************************
* プログラム名 : SYOHIN
* 概要         : 商品テーブルにアクセスし、
*                商品レコードの取得を行う。
*
*******************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYOHIN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*=====================================================
*  ホスト変数宣言
*=====================================================
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE     PIC X(5).
 01 SQLMSG       PIC X(128).

 01 SYOPF-REC.
   02  SYOCD      PIC S9(9) COMP-5.
   02  SYONA      PIC N(20).
   02  BAIKA      PIC S9(9) COMP-5.
   02  GENKA      PIC S9(9) COMP-5.

 01  S-SYOCD      PIC S9(9) COMP-5.
     EXEC SQL END DECLARE SECTION END-EXEC.

 COPY MSGTBL.

 LINKAGE                        SECTION.
 01  P-SYOHINREC.
     02  P-SYOCD             PIC 9(5).
     02  P-SYONA             PIC N(20).
     02  P-BAIKA             PIC 9(8).
     02  P-GENKA             PIC 9(8).
 01  P-Msg                   PIC N(20).
 01  R-ReturnCode            BINARY-LONG.
 PROCEDURE DIVISION USING
                    P-SYOHINREC   *>商品レコード
                    P-Msg         *>メッセージ
                    RETURNING
                    R-ReturnCode. *>戻り値(エラーコード)

     PERFORM MAIN-SEC.
     PERFORM END-SEC.
     EXIT PROGRAM.

 MAIN-SEC SECTION.
*    レコードがなかった場合の動作
     EXEC SQL
     WHENEVER NOT FOUND GO TO :P-NOTFOUND
     END-EXEC.

*    DB接続
     EXEC SQL CONNECT TO DEFAULT END-EXEC.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*     接続確認
      IF SQLSTATE = "00000" OR SQLSTATE = "01000" THEN
          MOVE P-SYOCD OF P-SYOHINREC TO S-SYOCD

*         SQL実行
          EXEC SQL
          SELECT SYOCD, SYONA, BAIKA, GENKA
              INTO :SYOCD, :SYONA, :BAIKA, :GENKA
              FROM SYOPF WHERE SYOCD = :S-SYOCD
          END-EXEC

*         取得データの転記
          IF SQLSTATE = "00000" THEN
              MOVE SYOCD TO P-SYOCD
              MOVE SYONA TO P-SYONA
              MOVE BAIKA TO P-BAIKA
              MOVE GENKA TO P-GENKA
          END-IF
      END-IF.

 END-SEC SECTION.
*     DBクローズ
      EXEC SQL DISCONNECT DEFAULT END-EXEC.
*     メッセージ設定
      IF R-ReturnCode > 1 THEN
          MOVE MSG-DSP(R-ReturnCode) TO P-Msg
      END-IF.

 NOTFOUND-SEC SECTION.
 P-NOTFOUND.
*     商品コードが見つからなかった場合の処理
      MOVE 5 TO R-ReturnCode.
      PERFORM END-SEC.
 END PROGRAM SYOHIN.
```

■ ZAIKO.COB

(入出庫テーブルから入出庫明細を取得する)

```
*****************************************************
* プログラム名 : ZAIKO
* 概要         : 入出庫テーブルから入出庫明細を取得する。
*
*****************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ZAIKO.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 INPUT-OUTPUT SECTION.

 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*=========================================================
*  ホスト変数宣言
*=========================================================
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE     PIC X(5).
 01 SQLMSG       PIC X(128).

 01 SYOIOPF-REC.
   02  IODATE     PIC S9(9) COMP-5.
   02  SYOCD      PIC S9(9) COMP-5.
   02  NODATE     PIC S9(9) COMP-5.
   02  SYUBAN     PIC S9(9) COMP-5.
   02  TUBAN      PIC S9(9) COMP-5.
   02  KUBUN      PIC S9(9) COMP-5.
   02  IOSU       PIC S9(9) COMP-5.

 01  S-SIODATE    PIC S9(9) COMP-5.
 01  S-EIODATE    PIC S9(9) COMP-5.
 01  S-SYOCD      PIC S9(9) COMP-5.
     EXEC SQL END DECLARE SECTION END-EXEC.
 01  W-IODATE     PIC 9(8).

 01  W-DENPYO.
   03  W-SYUBAN   PIC 9(8).
   03  W-TUBAN    PIC 9(4).

*---コードテーブル----------------
   01  S-CODE.
     03  S-TANKA                PIC  X(4)  VALUE "0001".
     03  S-GENKA                PIC  X(4)  VALUE "0002".
     03  S-MUKA                 PIC  X(4)  VALUE "0003".
     03  S-NYUKO                PIC  X(4)  VALUE "0001".
     03  S-SYUKO                PIC  X(4)  VALUE "0002".
*
     03  S-TANGEN               PIC  X(4)  VALUE "0000".
     03  S-NYUSYU               PIC  X(4)  VALUE "0000".

*---伝票区分コードテーブル--------
   01  KUBUN-AREA.
*    01～10 **
     03  FILLER                  PIC  N(20) VALUE
           NC"                                        ".
*    11～20 **
     03  FILLER                  PIC  N(20) VALUE
           NC"売上                          売訂    ".
*    21～30 **
     03  FILLER                  PIC  N(20) VALUE
           NC"売返                          売訂    ".
*    31～40 **
     03  FILLER                  PIC  N(20) VALUE
           NC"仕入送仕状仕加仕浮借浮仕委仕値増仕訂    ".
*    41～50 **
     03  FILLER                  PIC  N(20) VALUE
           NC"仕返送返状返加返浮返浮戻委戻値引仕訂    ".
*    51～60 **
     03  FILLER                  PIC  N(20) VALUE
           NC"貸出持出社出預出持越展出振出他出出訂    ".
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*    61～70 **
     03  FILLER                    PIC  N(20) VALUE
            NC"貸戻持戻社戻預戻持戻展戻振入他入入訂    ".
*    70～80 **
     03  FILLER                    PIC  N(20) VALUE
            NC"返出振出                                ".
*    81～90 **
     03  FILLER                    PIC  N(20) VALUE
            NC"返入振入                        他入入訂".
*    91～100**
     03  FILLER                    PIC  N(20) VALUE
            NC"                                        ".
   01 KUBUN-TBL REDEFINES  KUBUN-AREA.
     03  KUBUN-ID          OCCURS 100 TIMES.
       05  FILLER                  PIC  N(02).

 01  FILE-STS-AREA.
   03  FILE-STS1                PIC  X(02).
   03  FILE-STS2                PIC  X(06).

*---フラグ関連--------------------
   01  FLG-AREA.
     03  END-FLG                  PIC  X(03).
     03  MEISAI-FLG               PIC  X(03).

*---プログラム作業領域------
   01  W-AREA.
     03  W-KEY.
       05  W-Y                    PIC  9(04).
       05  W-M                    PIC  9(02).
       05  W-SYOCD                PIC  9(05).
     03  W-IOYMD.
       05  W-IOY                  PIC  9(04).
       05  W-IOM                  PIC  9(02).
       05  W-IOD                  PIC  9(02).
     03  W-RECCNT                 PIC  9(03).

 COPY MSGTBL.

 01  LINEMAX                   PIC 9(03) VALUE 10.
 LINKAGE SECTION.
 COPY IODETAIL.
 01 P-Start                    BINARY-SHORT.
 01 P-Cnt                      PIC 9(03).
 01 P-Next                     PIC 9(01).
 01 P-Msg                      PIC N(20).
 01 R-ReturnCode               BINARY-LONG.
 PROCEDURE DIVISION USING
                    M-AREA         *>明細領域
                    P-Start        *>開始行
                    P-Cnt          *>件数
                    P-Next         *>次のレコードがあるか
                                   *>どうか(ある場合は1)
                    P-Msg          *>メッセージ
                    RETURNING
                    R-ReturnCode.  *>戻り値(エラーコード)
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
     PERFORM INIT-SEC.
     IF R-ReturnCode = 0 THEN
         PERFORM MAIN-SEC
     END-IF.
     PERFORM END-SEC.
     EXIT PROGRAM.

*-----------------------------------------------------
 INIT-SEC SECTION.
     MOVE 0 TO R-ReturnCode
     MOVE SPACE TO MEISAI-FLG.
     MOVE SPACE TO END-FLG.

     MOVE 0 TO W-RECCNT.
     MOVE 0 TO P-Cnt.

*=====================================================
*  カーソルを宣言します
*=====================================================
     EXEC SQL
       DECLARE CUR1 CURSOR FOR SELECT * FROM SYOIOPF
         WHERE (IODATE BETWEEN :S-SIODATE
         AND :S-EIODATE) AND SYOCD = :S-SYOCD
     END-EXEC.

*    DB接続
     EXEC SQL CONNECT TO DEFAULT END-EXEC.
*    接続確認
     IF SQLSTATE NOT = "00000" AND
        SQLSTATE NOT = "01000" THEN
         MOVE 9 TO R-ReturnCode
         MOVE MSG-DSP(R-ReturnCode) TO P-Msg
         EXIT PROGRAM
     END-IF.
*---入出庫レコードの検索条件------
     MOVE    M-SYOCD  TO    S-SYOCD.
     MOVE    M-Y      TO    W-IOY.
     MOVE    M-M      TO    W-IOM.
     MOVE    1        TO    W-IOD.
     MOVE    W-IOYMD  TO    W-IODATE.
     MOVE    W-IODATE TO    S-SIODATE.
     MOVE    31       TO    W-IOD.
     MOVE    W-IOYMD  TO    W-IODATE.
     MOVE    W-IODATE TO    S-EIODATE.

*-----------------------------------------------------
 END-SEC SECTION.
*    DBクローズ
     EXEC SQL DISCONNECT DEFAULT END-EXEC.
*    メッセージ設定
     IF R-ReturnCode > 1 THEN
         MOVE MSG-DSP(R-ReturnCode) TO P-Msg
     END-IF.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*-----------------------------------------------------
 FETCH-SEC SECTION.
*    フェッチ
     EXEC SQL FETCH CUR1 INTO :SYOIOPF-REC END-EXEC.
     IF SQLSTATE = "00000" THEN
*        読込件数カウントアップ
         COMPUTE W-RECCNT = W-RECCNT + 1
     ELSE
*        レコードがない場合の動作
         IF SQLSTATE = "02000" THEN
             MOVE "END" TO MEISAI-FLG
         ELSE
             MOVE "ERR" TO END-FLG
         END-IF
     END-IF.

*-----------------------------------------------------
 MAIN-SEC SECTION.
*    カーソルオープン
     EXEC SQL OPEN CUR1 END-EXEC.
     PERFORM FETCH-SEC.
*    ループ
     PERFORM UNTIL (END-FLG = "ERR") OR
                   (MEISAI-FLG = "END") OR
                   (MEISAI-FLG = "MAX")
         IF W-RECCNT >= P-Start THEN
             COMPUTE P-Cnt = P-Cnt + 1
             PERFORM CODE-SEC
         END-IF
*        明細行数を越えたら終了
         IF P-Cnt >= LINEMAX THEN
             MOVE "MAX" TO MEISAI-FLG
*            最後に1件読んで次の行があるか確認する
         END-IF
         PERFORM FETCH-SEC
     END-PERFORM.
*    カーソルクローズ
     EXEC SQL CLOSE CUR1 END-EXEC.

     IF MEISAI-FLG = "MAX" THEN
         MOVE 1 TO P-Next
     END-IF
     IF MEISAI-FLG = "END" AND P-Cnt = 0 THEN
         *>商品コードはあったが、該当する年月のレコードなし
         MOVE 8 TO R-ReturnCode
     END-IF.

*-----------------------------------------------------
 CODE-SEC SECTION.
*    区分コード処理
     EVALUATE KUBUN OF SYOIOPF-REC

     WHEN    ZERO
         CONTINUE

*    *** 納品系処理区分（１１～２９）
     WHEN    11  THRU  19
         EVALUATE   KUBUN OF SYOIOPF-REC
             WHEN    11
                 MOVE    S-TANKA  TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
            WHEN    19
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE

    WHEN    21  THRU  29
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    21
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
        END-EVALUATE

*   ***仕入系処理区分（３１～４９）
    WHEN    31  THRU  39
        EVALUATE   KUBUN OF SYOIOPF-REC
            WHEN    39
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-GENKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            END-EVALUATE

    WHEN    41  THRU  49
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    49
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-GENKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE

*   *** 社内良品系処理区分（５１～６９）
    WHEN    51  THRU  59
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    51
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    52
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    59
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE
*
    WHEN    61  THRU  69
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    61
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
            WHEN    62
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    69
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
        END-EVALUATE

*    ***社外不良品系処理区分（７１～８９）
     WHEN    71  THRU  79
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    71  THRU  72
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    79
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE
*
     WHEN    81  THRU  89
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    81  THRU  82
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    89
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
        END-EVALUATE
     END-EVALUATE

     MOVE IODATE OF SYOIOPF-REC TO M-NODATE(P-Cnt)
     MOVE SYUBAN OF SYOIOPF-REC TO W-SYUBAN
     MOVE TUBAN OF SYOIOPF-REC  TO W-TUBAN
     MOVE W-DENPYO              TO M-DENBAN(P-Cnt)
     MOVE KUBUN OF SYOIOPF-REC  TO M-KUBUN(P-Cnt)
     MOVE KUBUN-ID(KUBUN OF SYOIOPF-REC)
                                TO M-KUBUNN(P-Cnt)

     EVALUATE S-TANGEN
     WHEN S-TANKA
        MOVE M-TANKA  TO M-KAKAKU(P-Cnt)
     WHEN S-GENKA
        MOVE M-GENKA  TO M-KAKAKU(P-Cnt)
     WHEN S-MUKA
        MOVE 0        TO M-KAKAKU(P-Cnt)
     END-EVALUATE
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
    IF S-NYUSYU = S-NYUKO THEN
        MOVE IOSU OF SYOIOPF-REC TO M-ISU(P-Cnt)
    ELSE
        MOVE IOSU OF SYOIOPF-REC TO M-OSU(P-Cnt)
    END-IF
END PROGRAM ZAIKO.
```

➢ **手順**2

手順1のプログラムで使用する登録集をコーディングします。
作成する登録集は、次の2本です。

✓ IODETAIL.CBL
✓ MSGTBL.CBL

ソリューションエクスプローラより[StockLibrary]の[登録集ファイル]フォルダを右クリックし、[追加]-[新しい項目の追加]で表示されるダイアログボックス内で[テンプレート]より「登録集」を選択しファイル名を入力してください。

２本の登録集を次のようにコーディングします。

■ IODETAIL.CBL

```
01  M-AREA.
  03  M-Y                          PIC  9(04).
  03  M-M                          PIC  9(02).
  03  M-SYOCD                      PIC  9(05).
  03  M-TANKA                      PIC  9(08).
  03  M-GENKA                      PIC  9(08).
  *>明細10レコード
  03  M-DENPYO OCCURS 10.
    05  M-NODATE                   PIC  9(08).
    05  M-DENBAN                   PIC  9(12).
    05  M-KUBUN                    PIC  9(02).
    05  M-KUBUNN                   PIC  N(02).
    05  M-KAKAKU                   PIC  9(08).
    05  M-ISU                      PIC  9(06).
    05  M-OSU                      PIC  9(06).
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ MSGTBL.CBL

```
*---メッセージテーブル------------
   01  MSG-AREA.
*    NO.01
     03  FILLER                    PIC  N(20) VALUE
            NC"                                        ".
*    NO.02
     03  FILLER                    PIC  N(20) VALUE
            NC"社員コードに、間違いがあります          ".
*    NO.03
     03  FILLER                    PIC  N(20) VALUE
            NC"パスワードに、間違いがあります          ".
*    NO.04
     03  FILLER                    PIC  N(20) VALUE
            NC"年月の値に誤りがあります                ".
*    NO.05
     03  FILLER                    PIC  N(20) VALUE
            NC"商品コードに、間違いがあります。        ".
*    NO.06
     03  FILLER                    PIC  N(20) VALUE
            NC"前頁はありません                        ".
*    NO.07
     03  FILLER                    PIC  N(20) VALUE
            NC"次頁はありません                        ".
*    NO.08
     03  FILLER                    PIC  N(20) VALUE
            NC"該当の入出庫明細がありません            ".
*    NO.09
     03  FILLER                    PIC  N(20) VALUE
            NC"データベースにアクセスできません        ".
*--
   01 MSG-TBL   REDEFINES  MSG-AREA.
     03  MSG-DSP OCCURS 9 TIMES.
       05  FILLER                  PIC N(20).
```

以上で、クラスライブラリ(StockLibrary)を構成するファイルの編集は完了です。

➢ **手順**3

手順1、2で作成したクラスライブラリ(StockLibrary)の動作を検証するため、それぞれのサブルーチンを呼び出すコンソールアプリケーションを別々のソリューションで作成します。作成するコンソールアプリケーションは次の3本です。

✓ TestSYAIN.cob(TestSYAIN.cobproj)
✓ TestSYOHIN(TestSYOHIN.cobproj)
✓ TestZAIKO(TestZAIKO.cobproj)

All Rights Reserved, Copyright FUJITSU LIMITED 2006

コンソールアプリケーションのプロジェクト作成手順については、第2章の[2.4.3 プロジェクトの作成]を参考に作成し、次のソースリストをもとにコーディングを行ってください。

■ TestSYAIN.cob

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. Program1 AS "TestSYAIN.Program1".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  WK-EmployeeCode           PIC N(06).
 01  WK-Password               PIC N(06).
 01  WK-Msg                    PIC N(20).
 01  WK-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION.
*    社員コードとパスワードの設定
     MOVE NC"200007" TO WK-EmployeeCode.
     MOVE NC"200007" TO WK-Password.

     DISPLAY "社員コード:"  WK-EmployeeCode
             " パスワード:" WK-Password
             "で認証を行います。"
*    認証用のSYAINプログラム呼び出し
     CALL "SYAIN" USING WK-EmployeeCode WK-Password
                        WK-Msg
                  RETURNING WK-ReturnCode.
     IF WK-ReturnCode  =  0 THEN
*        認証OK
         DISPLAY "認証に成功しました"
     ELSE
         DISPLAY "認証に失敗しました： " WK-Msg
     END-IF
 END PROGRAM Program1.
```

■ TestSYOHIN.COB

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. Program1 AS "TestSYOHIN.Program1".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  商品レコード.
     02  商品コード  PIC 9(5).
     02  商品名      PIC N(20).
     02  売価        PIC 9(8).
     02  原価        PIC 9(8).
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
 01 W-Msg            PIC N(20).
 01 W-ReturnCode     BINARY-LONG.
 PROCEDURE DIVISION.
*    商品コード設定
     MOVE 10024 TO 商品コード.
     DISPLAY "商品コード:" 商品コード
             " の情報を検索します。"
*    商品レコード検索
     CALL "SYOHIN" USING 商品レコード W-Msg
                   RETURNING W-ReturnCode.

     DISPLAY 商品コード " " 商品名 " " 売価 " " 原価
 END PROGRAM Program1.
```

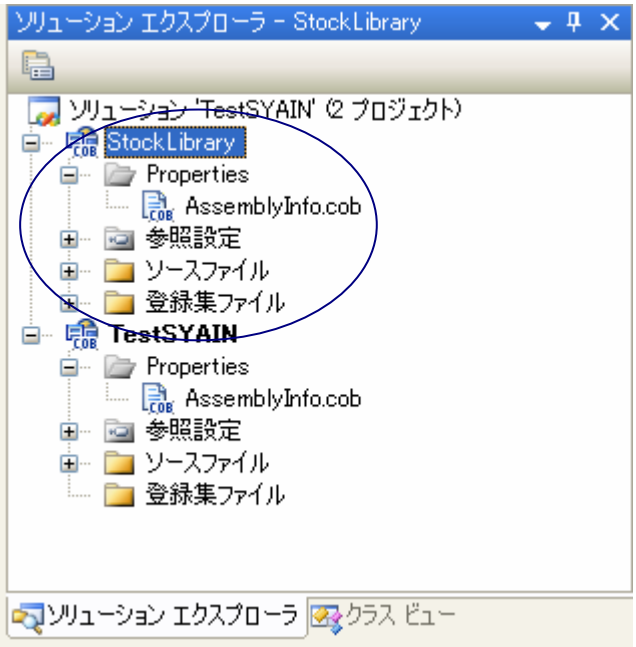
■ TestZAIKO.cob

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. Program1 AS "TestZAIKO.Program1".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 COPY IODETAIL.
 01 W-Msg                     PIC N(20).
 01 W-Next                    PIC 9(01).
 01 W-ReturnCode              BINARY-LONG.
 01 W-Cnt                     PIC 9(03).
 01 W-Start                   BINARY-SHORT.
 01 W-IDX                     BINARY-LONG.
 PROCEDURE DIVISION.
     INITIALIZE M-AREA

*    商品コード設定
     MOVE 10024 TO M-SYOCD.
*    日付設定
     MOVE 2004 TO M-Y.
     MOVE 4 TO M-M.
*    開始位置
     MOVE 1 TO W-Start.
     DISPLAY M-Y "年" M-M "月 商品コード:" M-SYOCD
             "の入出庫情報を検索します。"
*    入出庫検索
     CALL "ZAIKO" USING M-AREA W-Start
                        W-Cnt W-Next W-Msg
                  RETURNING W-ReturnCode.

     PERFORM VARYING W-IDX  FROM 1 BY 1
                            UNTIL W-IDX > 10
         DISPLAY M-NODATE(W-IDX) " " M-DENBAN(W-IDX)
             " " M-KUBUN(W-IDX)  " " M-KUBUNN(W-IDX)
             " " M-ISU(W-IDX)    " " M-OSU(W-IDX)
     END-PERFORM.
 END PROGRAM Program1.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

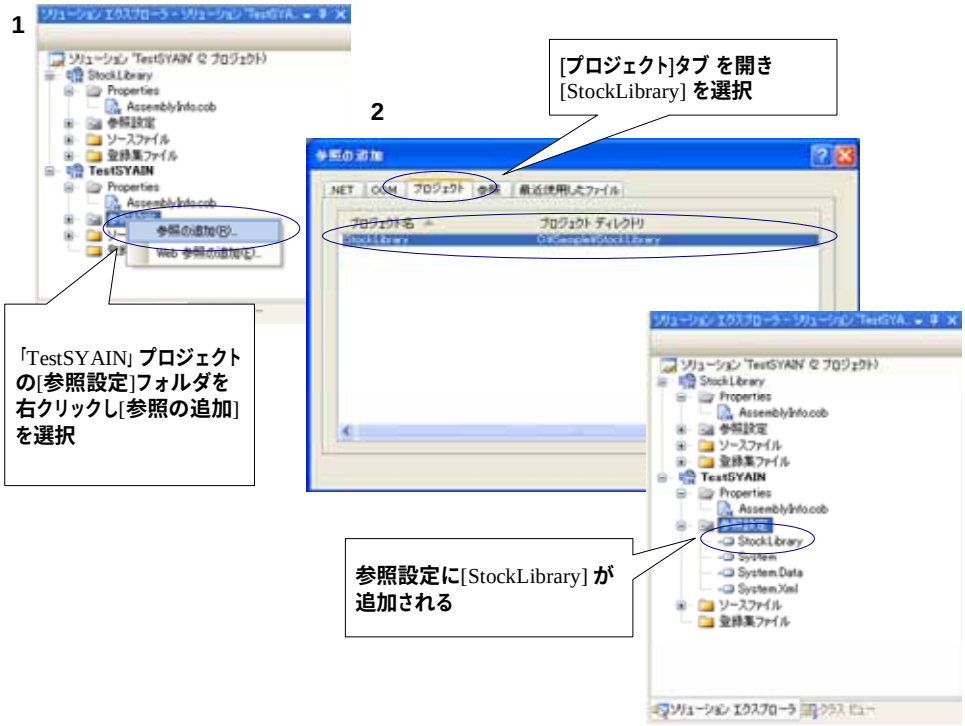
- **手順**4

手順3で作成した各コンソールアプリケーションのソリューションに手順2で作成したクラスライブラリ(StockLibrary)のプロジェクトを追加し、コンソールアプリケーションからクラスライブラリを呼び出せるよう、さらに[参照設定]に追加します。
[ソリューションエクスプローラ]よりソリューションを右クリックし、[追加]-[既存のプロジェクト]を選択し、[既存プロジェクトの追加]ダイアログボックスより「StockLibrary.cobproj」を選択します。
下図は、プロジェクトの追加イメージです。



次に、クラスライブラリ(StockLibrary)の参照設定を各コンソールアプリケーションで行います。
下図のように[ソリューションエクスプローラ]より、コンソールアプリケーションのプロジェクト(例：TextSYOHIN.cobproj)内の[参照設定]フォルダを右クリックし[参照]を選択し、手順にしたがって追加してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



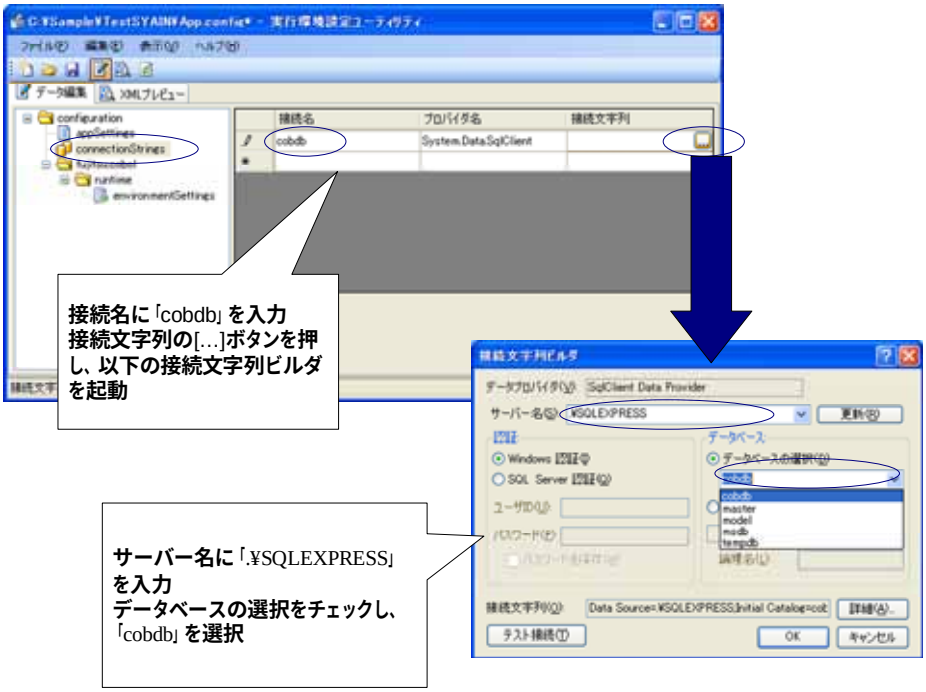
5.5.5 ビルドおよびアプリケーション構成ファイルの作成

検証用の3本のコンソールアプリケーションにそれぞれクラスライブラリ(StockLibrary)の参照設定を行ったあとは、ソリューションのビルドを行います。ビルドで発生したエラーの対処を行い、ビルドが正常に終了したことを確認後、次に、ソリューションエクスプローラよりコンソールアプリケーションのプロジェクトを右クリックし[COBOL実行環境設定ユーティリティ]を起動します。このユーティリティを使用してデータベース接続に必要な設定をGUI形式で指定し、アプリケーション構成ファイル(拡張子：.config)に反映させます。

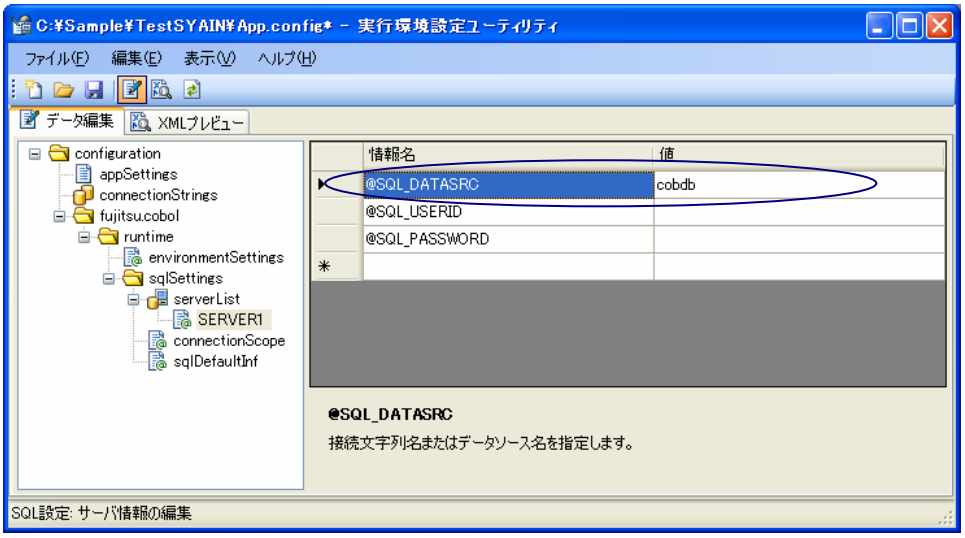
- **手順**1

  データベース接続文字列に該当する「connectionStrings」を変更します。デフォルトで挿入されている「LocalSqlServer」は、接続名を右クリックし[設定の削除]で削除しておきます。次に以下の設定を行います。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



次に、[編集]メニューより[SQL設定の追加]をクリックすると雛形が表示されますので、情報名[@SQL_DATASRC]の値を「cobdb」とします。



上記の編集が完了すると、プロジェクトの[ソースファイル]ファオルダにアプリケーション構成ファイルApp.configが追加されます。最後にソリューションのビルドを行いコンソールアプリケーションを作成します。

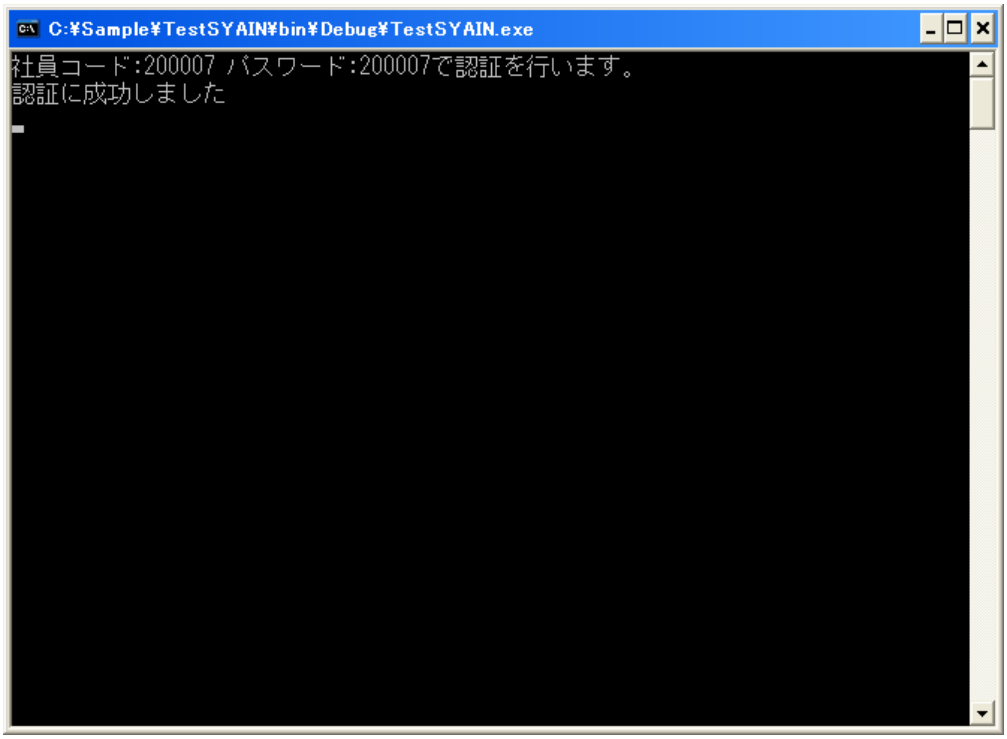
All Rights Reserved, Copyright FUJITSU LIMITED 2006

以上で、クラスライブラリを呼び出すコンソールアプリケーション(TestSYAIN、TestSYOHIN、TestZAIKO)を実行するために必要なファイルやプログラムの作成は完了です。

5.5.6 実行

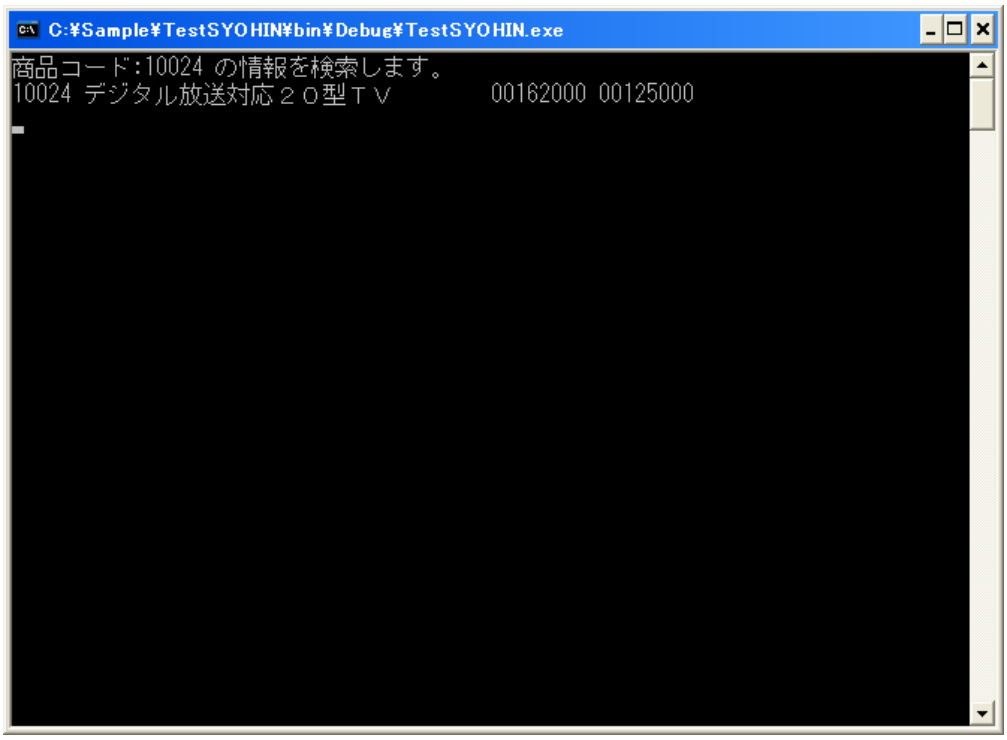
各コンソールアプリケーションを実行し、クラスライブラリ(StockLibrary)として作成したサブルーチンの動作について確認してください。

■ TestSYAINの実行イメージ

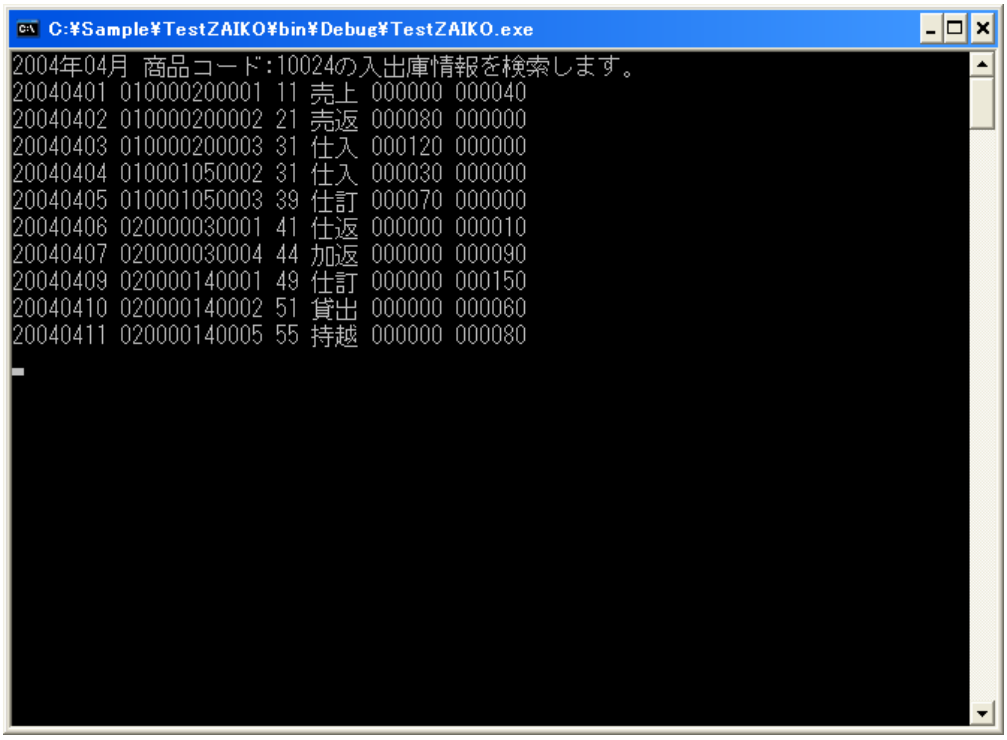


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ TestSYOHINの実行イメージ



■ TestZAIKOの実行イメージ



All Rights Reserved, Copyright FUJITSU LIMITED 2006

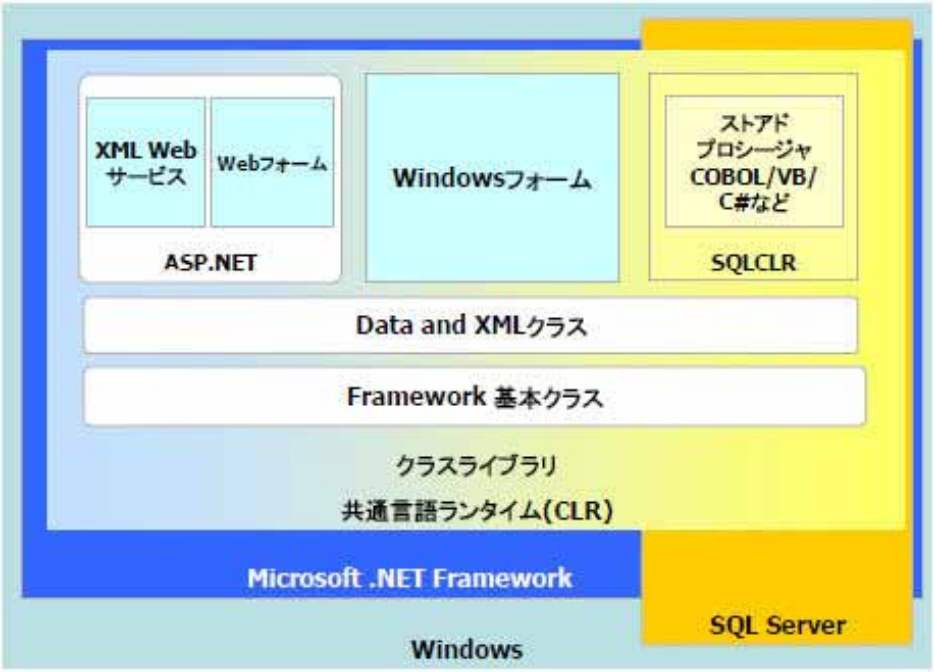
5.6 SQL CLRストアドプロシージャの作成

5.6.1 SQL CLRとは

これまで、SQL Serverでストアドプロシージャを記述する場合、Transact-SQL言語を使用して記述するしかありませんでした。

SQL CLRでは、.NET Frameworkの共通言語ランタイム(CLR)のコンポーネントがSQL Server 2005に統合されたことにより、マネージコードによるデータベースオブジェクト(ストアドプロシージャやユーザ定義関数など)が利用可能になりました。

これにより、.NET Frameworkで利用可能な言語を使用したデータベースオブジェクトを記述することが可能となり、COBOLで作成したプログラムをストアドプロシージャとして、SQL Serverプロセス内で実行させることができるようになりました。



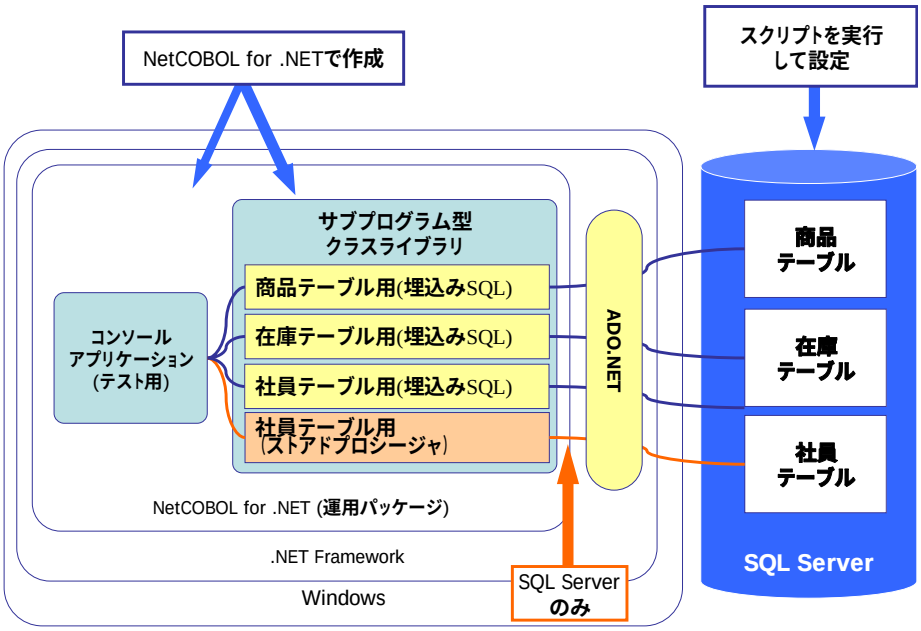
SQL CLRストアドプロシージャは、ADO.NETを用いてSQL Server 2005上のデータに直接アクセスすることができます。このため、NetCOBOLでも、クラスライブラリや、埋込みSQLを使用したデータアクセスが可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.6.2 プログラムの概要

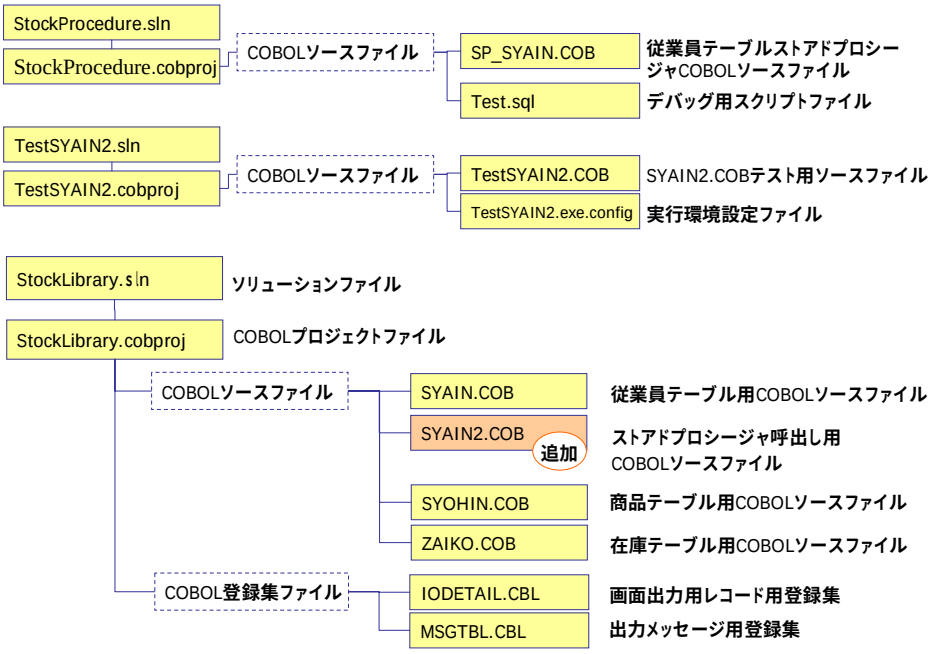
(1) プログラムの実行イメージ

SQL CLRストアドプロシージャの作成例として、埋込みSQLを使用したプログラムを紹介します。この例では、5.5節「データアクセスプログラムの作成」で紹介した、「社員テーブル」にアクセスするサブプログラム(SYAIN.COB)の一部をストアドプロシージャとして作成します。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) プログラムの構成



ストアドプロシージャ用のソリューションを作成します。

また、実行検証用に5.5節で作成した[StockLibrary]ソリューションにストアドプロシージャ呼出し用のCOBOLソース(SYAIN2.cob)を追加し、コンソールアプリケーションを作成します。

5.6.3 プログラム作成手順

(1) データベース環境設定

5.5.2章で構築したデータベースを使用します。データベースの構築はcobdb.sqlを実行してください。付属のsetupcobdb.batを実行することにより、SQL Server 2005 Express上に構築されます。

SQL CLRを使用する場合は、あらかじめNetCOBOLランライムアセンブリ(Fujitsu.COBOL.dll)をデータベースに登録する必要があります。cobdb.sqlスクリプトには、ランタイムアセンブリを登録するスクリプトが含まれています。ランタイムアセンブリのインストール先が異なる場合は、環境に合わせてNetCOBOLランタイムアセンブリのパスを変更してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) プロジェクトの作成とファイルの追加

SQL Serverプロジェクトを作成し、ストアドプロシージャのためのCOBOLソースファイルを追加します。

(3) COBOL手続きの編集

埋込みSQLを使用してデータベースにアクセスするCOBOL手続きを編集します。

(4) プログラムの配置

プログラミングが完了したら、ビルド後にSQL Serverにストアドプロシージャを配置します。

(5) プログラムの実行とデバッグ

デバッグや実行はプロジェクト内に自動的に作成されるテストスクリプトや、簡単な呼び出しプログラムを作成し動作確認を行います。

5.6.4 プロジェクトの作成とファイルの追加

SQL CLRストアドプロシージャを管理するプロジェクトを作成します。

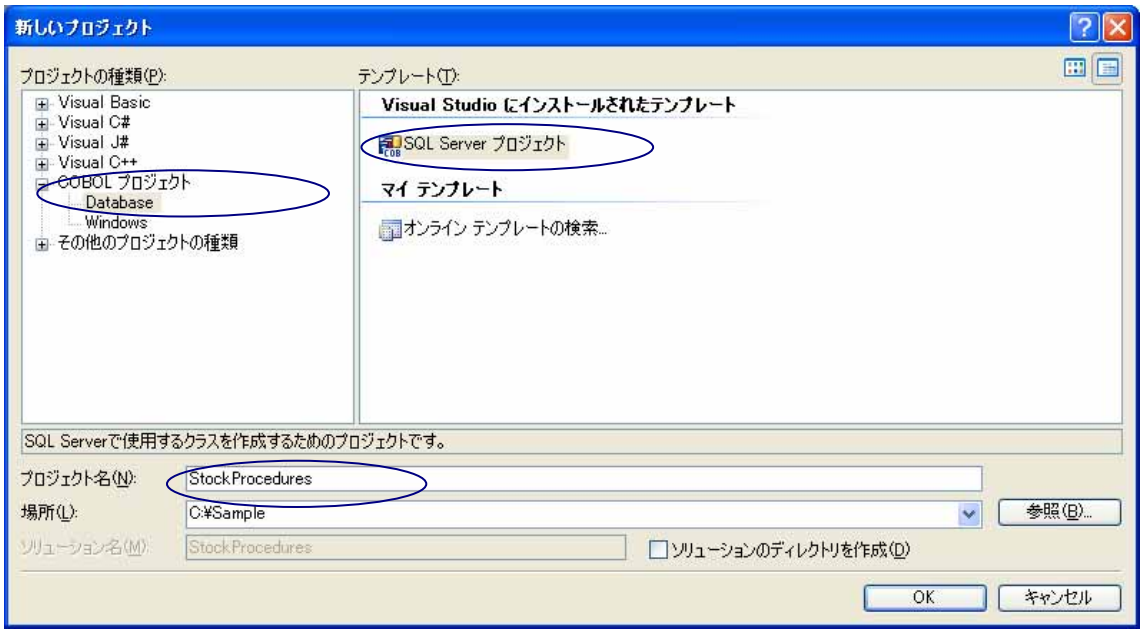
- **手順**1

  NetCOBOL for .NETを起動し、[ファイル]メニューより[新規作成]-[プロジェクト]を選択します。

- **手順**2

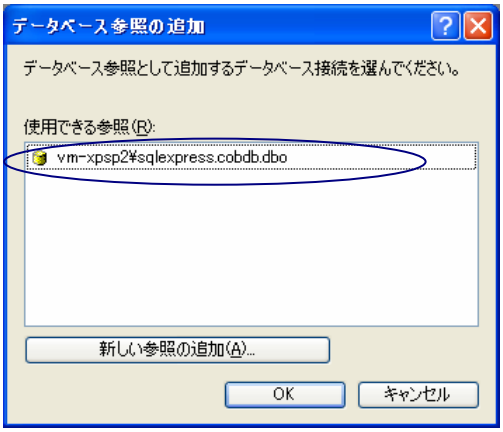
  [新しいプロジェクト]ダイアログボックスが表示されます。プロジェクトの種類より[COBOLプロジェクト]-[Database]を選択し、テンプレートより[SQL Serverプロジェクト]を選択します。プロジェクト名に「StockProcedures」と入力しOKボタンを押します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

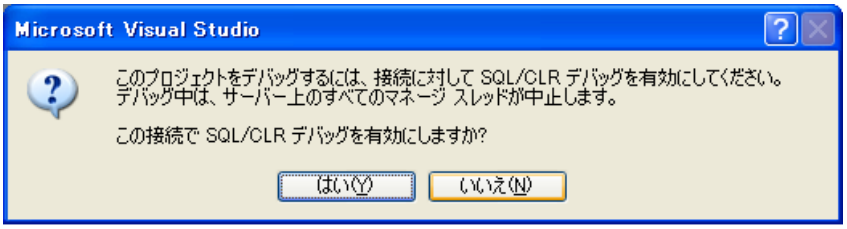


➢ **手順**3

[データベース参照の追加]ダイアログボックスが表示されるので、構築したデータベースを選択し、OKボタンを押します。



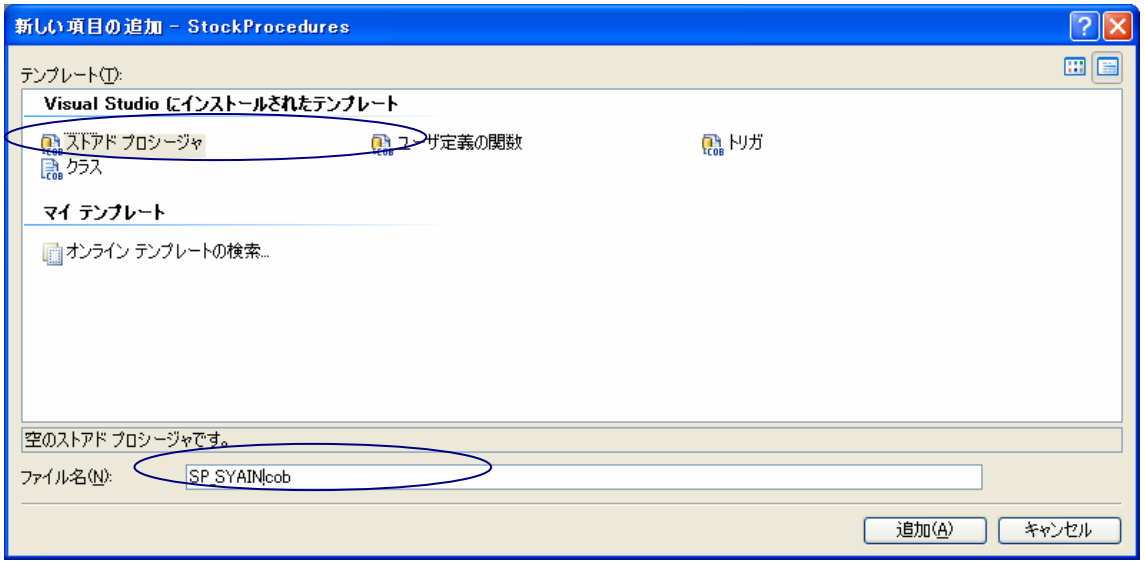
続いて以下のようなメッセージボックスが表示されます。SQL CLRデバッグを行うため、ここでは[はい]ボタンを押します。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

- **手順4**

  [StockProcedure]プロジェクトにソースプログラムを追加します。ソリューションエクスプローラの[StockProcedure]プロジェクトより[ソースファイル]フォルダを右クリックし、[追加]-[新しい項目の追加]をクリックします。[新しい項目の追加]ダイアログボックスが表示されますので、テンプレートから[ストアドプロシージャ]を選択し、ファイル名を「SP_SYAIN.cob」とします。OKボタンを押すと、[ソースファイル]フォルダに「SP_SYAIN.cob」が追加されます。



5.6.5 COBOL手続きの編集

次に追加したプログラムをコーディングします。5.5.4節で作成した埋込みSQLで社員テーブルにアクセスするプログラム(SYAIN.cob)の一部をストアドプロシージャ化しています。作成するプログラムは次のプログラムです。

✓ SP_SYAIN.COB

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SP_SYAIN.COB

(社員テーブルにアクセスし認証を行う)

```
*******************************************************
* プログラム名 : SP_SYAIN
* 概要         : 社員テーブルにアクセスし認証を行う
*                ストアドプロシージャ
*
*******************************************************
 IDENTIFICATION DIVISION.
 CLASS-ID. CLASS-1 AS "StockProcedures.SP_SYAIN".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     CUSTOM-ATTRIBUTE CA-SQLPROCEDURE CLASS CLASS-SQLPROCEDURE
     .
 REPOSITORY.
     CLASS CLASS-SQLPROCEDURE AS
       "Microsoft.SqlServer.Server.SqlProcedureAttribute"
     CLASS CLASS-STRING AS "System.String".
 STATIC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 PROCEDURE DIVISION.

 METHOD-ID. PROCEDURE-1 AS "SP_SYAIN"
                        CUSTOM-ATTRIBUTE CA-SQLPROCEDURE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*==============================================================
*  ホスト変数宣言
*==============================================================
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE     PIC X(5).
 01 SQLMSG       PIC X(128).

 01 SYAPF-REC.
   02  SYACD      PIC S9(9) COMP-5.
   02  SYANA      PIC N(20).
   02  PASS       PIC N(6).
   02  BUSYO      PIC S9(9) COMP-5.
   02  SDATE      PIC S9(9) COMP-5.
   02  SNAME      PIC N(20).

 01  S-SYACD      PIC S9(9) COMP-5.
     EXEC SQL END DECLARE SECTION END-EXEC.
 01  W-PASS       PIC N(20).
 LINKAGE SECTION.
 01  P_EmployeeCode           BINARY-LONG.
 01  P_Password               OBJECT REFERENCE CLASS-STRING.
 01  R_ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION USING BY VALUE
                     P_EmployeeCode   *>社員コード
                     P_Password       *>パスワード
                     RETURNING
                     R_ReturnCode.    *>戻り値(エラーコード)
     MOVE P_EmployeeCode TO S-SYACD
```

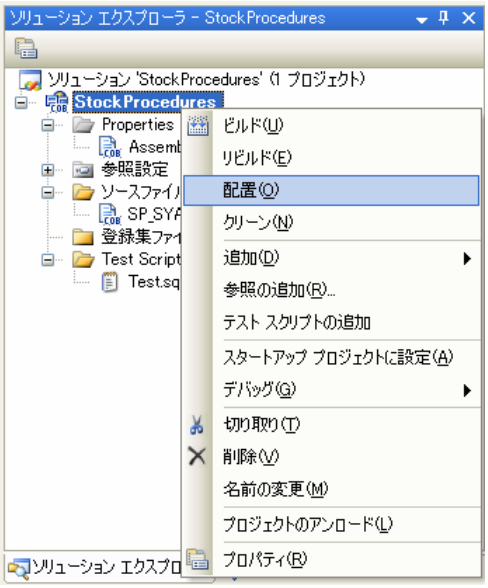

All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*     SQL実行
      EXEC SQL
      SELECT SYACD, PASS
          INTO :SYACD, :PASS FROM SYAPF WHERE SYACD = :S-SYACD
      END-EXEC
*     パスワード比較
      SET W-PASS TO P_Password
      IF SQLSTATE = "00000" AND PASS = W-PASS THEN
          MOVE 0 TO R_ReturnCode
      ELSE
          MOVE 3 TO R_ReturnCode
      END-IF
 END METHOD PROCEDURE-1.

 END STATIC.

 END CLASS CLASS-1.

 PROCEDURE DIVISION.
```

5.6.6 ビルドと配置

プログラムの作成を行ったあとは、ソリューションのビルドを行います。エラーの対処などを行いビルドが正常に終了したならば、SQL Serverへの配置を行います。

SQL Serverへの配置は、ソリューションエクスプローラよりStockProceduresプロジェクトを右クリックし、[配置]を選択します。



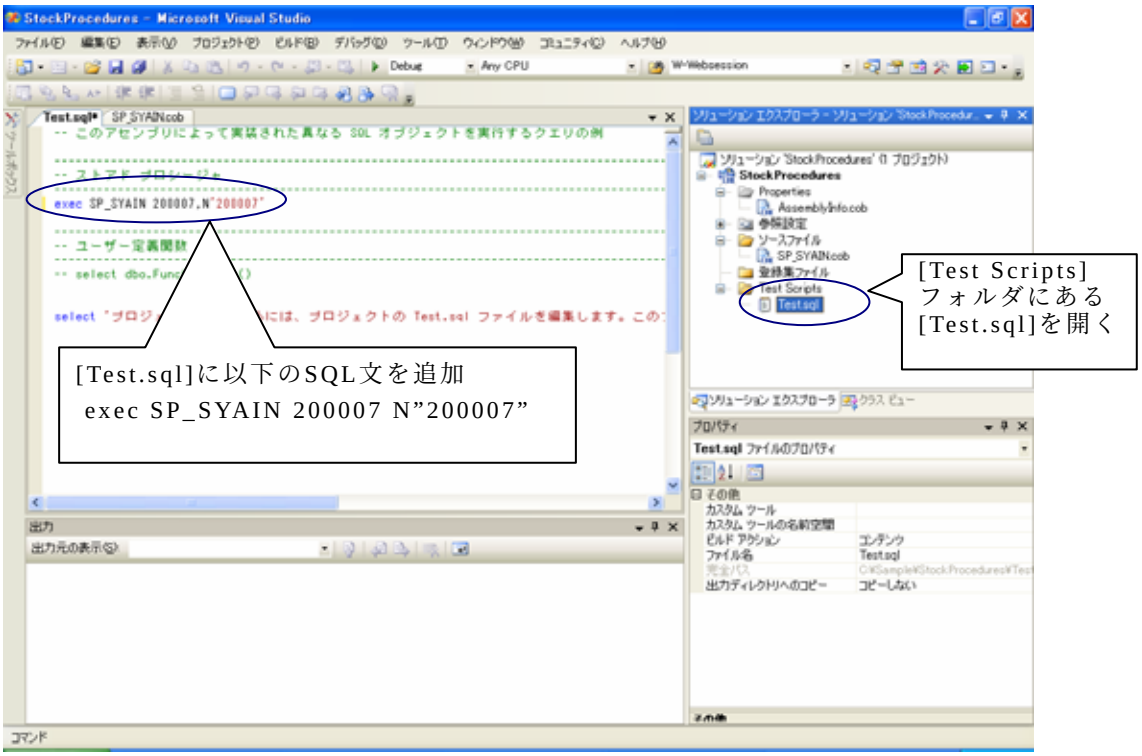
All Rights Reserved, Copyright FUJITSU LIMITED 2006

5.6.7 デバッグ

COBOLで作成したストアドプロシージャをデバッグするには、以下の手順で行います。

- **手順**1

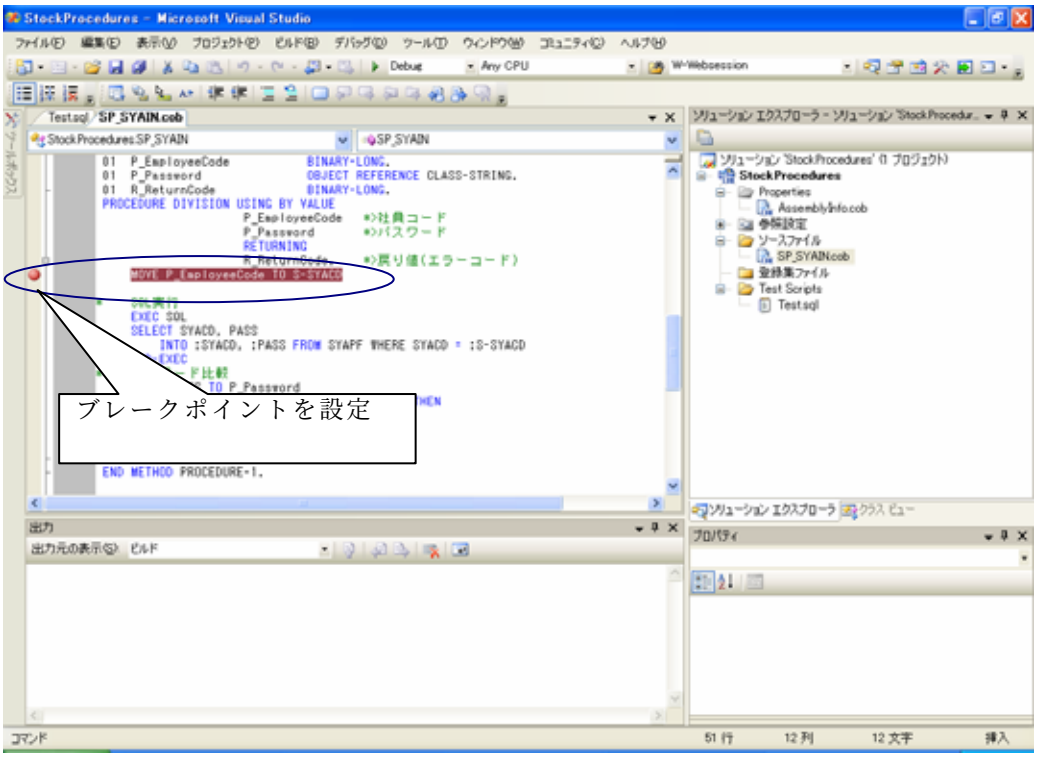
  デバッグを行う場合は、まず、ソリューションエクスプローラのツリーに表示されている、Test.sqlファイルを開いてストアドプロシージャを呼び出すSQL文を記述します。



- **手順**2

  次に、SP_SYAIN.cobを開き、手続き部に記述されている文にブレークポイントを設定します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



- **手順**3

  コンソールアプリケーションやWindowsアプリケーションと同じように、デバッグを開始し動作を確認します。

5.6.8 実行

それでは、実際にCOBOLで作成したストアドプロシージャを呼び出してみましょう。

- **手順**1

  COBOLで作成したストアドプロシージャSP_SYAINを呼び出すサブプログラムを作成します。

  5.5章で作成した、SYAIN.cobをもとに以下のファイルを[StockLibrary]ソリューションに追加します。

  ✓ SYAIN2.COB

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SYAIN2.COB

```
*********************************************************
* プログラム名 : SYAIN2
* 概要         : SP_SYAINを呼び出してし認証を行う。
*
*********************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYAIN2.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.
*=========================================================
*  ホスト変数宣言
*=========================================================
     EXEC SQL BEGIN DECLARE SECTION END-EXEC.
 01 SQLSTATE     PIC X(5).
 01 SQLMSG       PIC X(128).
 01 SQLINFOA.
    02 SQLERRD  PIC S9(9) COMP-5 OCCURS 6 TIMES.
 01  S-SYACD      PIC S9(9) COMP-5.
 01  S-PASS       PIC N(6).
     EXEC SQL END DECLARE SECTION END-EXEC.

 COPY MSGTBL.
 LINKAGE                        SECTION.
 01  P-EmployeeCode           PIC N(06).
 01  P-Password               PIC N(06).
 01  P-Msg                    PIC N(20).
 01  R-ReturnCode             BINARY-LONG.

 PROCEDURE DIVISION USING
                     P-EmployeeCode *>社員コード
                     P-Password     *>パスワード
                     P-Msg          *>メッセージ
                     RETURNING
                     R-ReturnCode.  *>戻り値(エラーコード)

     PERFORM MAIN-SEC.
     PERFORM END-SEC.
     EXIT PROGRAM.

 MAIN-SEC SECTION.
*    レコードがなかった場合の動作
     EXEC SQL
       WHENEVER NOT FOUND GO TO :P-NOTFOUND
     END-EXEC.

*    DB接続
     EXEC SQL CONNECT TO DEFAULT END-EXEC.
*    接続確認
     IF SQLSTATE = "00000" OR SQLSTATE = "01000" THEN
         MOVE FUNCTION DISPLAY-OF(P-EmployeeCode)
                       TO S-SYACD
         MOVE P-Password TO S-PASS
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*         SQL実行
          EXEC SQL
          CALL "SP_SYAIN" (:S-SYACD, :S-PASS)
          END-EXEC
*         ストアドプロシージャの戻り値を設定
          IF SQLSTATE = "00000"
            MOVE SQLERRD(1) TO R-ReturnCode
          ELSE
            MOVE 9 TO R-ReturnCode
          END-IF
      END-IF.

 END-SEC SECTION.
*     DBクローズ
      EXEC SQL DISCONNECT DEFAULT END-EXEC.
*     メッセージ設定
      IF R-ReturnCode > 1 THEN
          MOVE MSG-DSP(R-ReturnCode) TO P-Msg
      END-IF.

 NOTFOUND-SEC SECTION.
 P-NOTFOUND.
*     社員コードが見つからなかった場合の処理
      MOVE 2 TO R-ReturnCode.
      PERFORM END-SEC.
 END PROGRAM SYAIN2
```

- **手順**2

  次に、手順1で作成したサブプログラム(SYAIN2.cob)の動作を検証するため、サブプログラムを呼び出すコンソールアプリケーションを別ソリューションで作成します。作成方法は5.5.4節で作成した[TestSYAIN]ソリューションを参考にしてください。作成するコンソールアプリケーションは次のプログラムです。

  ✓ TESTSYAIN2.cob

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ TESTSYAIN2.COB

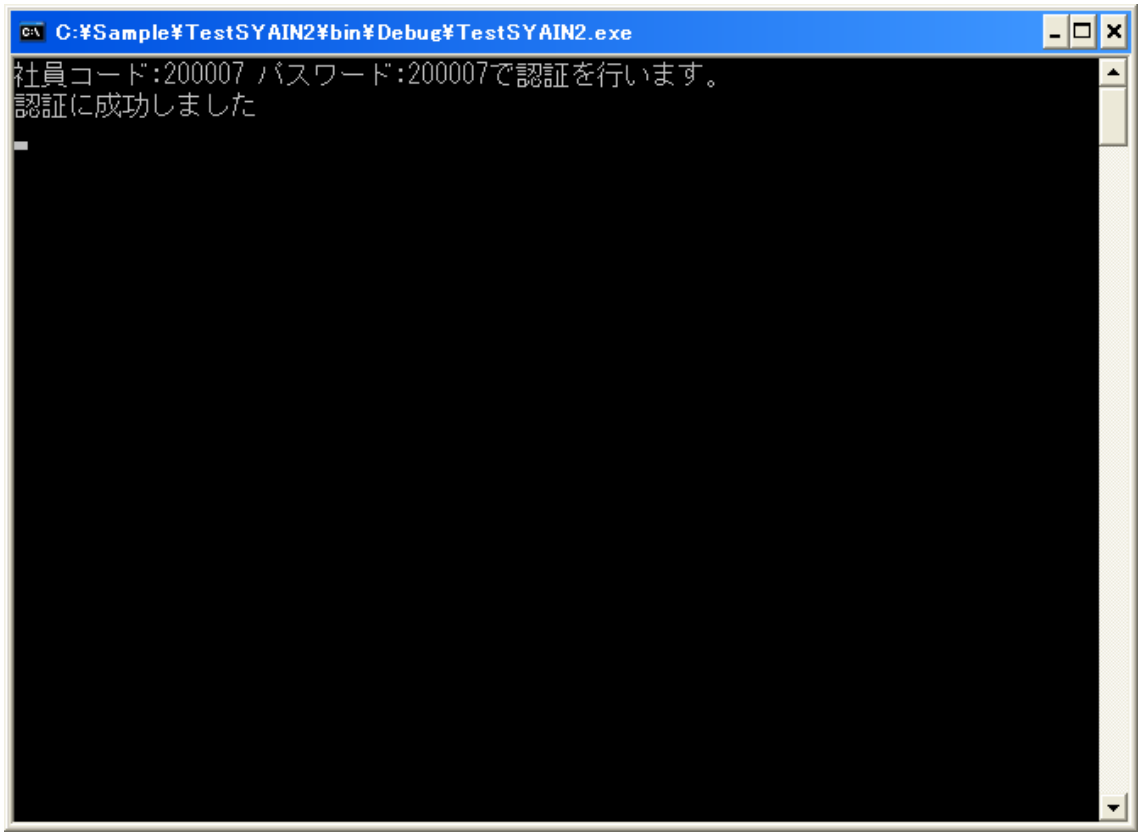
```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. Program1 AS "TestSYAIN2.Program1".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  WK-EmployeeCode           PIC N(06).
 01  WK-Password               PIC N(06).
 01  WK-Msg                    PIC N(20).
 01  WK-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION.
*    社員コードとパスワードの設定
     MOVE NC"200007" TO WK-EmployeeCode.
     MOVE NC"200007" TO WK-Password.

     DISPLAY "社員コード:" WK-EmployeeCode " パスワード:"
WK-Password
         "で認証を行います。"
*    認証用のSYAINプログラム呼び出し
     CALL "SYAIN2" USING WK-EmployeeCode WK-Password WK-
Msg
         RETURNING WK-ReturnCode.
     IF WK-ReturnCode  =  0 THEN
*        認証OK
         DISPLAY "認証に成功しました"
     ELSE
         DISPLAY "認証に失敗しました： " WK-Msg
     END-IF
 END PROGRAM Program1.
```

➢ **手順**3

手順2で作成したソリューションに[StockLibrary]プロジェクトを追加します。さらに、5.5.5節および5.5.6節と同様の手順で、実行環境設定を行いTestSYAIN2を実行します。

■ TestSYAIN2の実行イメージ

All Rights Reserved, Copyright FUJITSU LIMITED 2006



All Rights Reserved, Copyright FUJITSU LIMITED 2006

◇MEMO◇

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第6章 印刷処理

6.1 印刷処理の概要
6.2 行単位の印刷方法
6.3 ページ単位の印刷方法
6.4 オーバレイ定義体との重畳印刷

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.1 印刷処理の概要

6.1.1 NetCOBOL for .NETの印刷形態について

NetCOBOL for .NETを使用して行う印刷形態には次の方法があります。

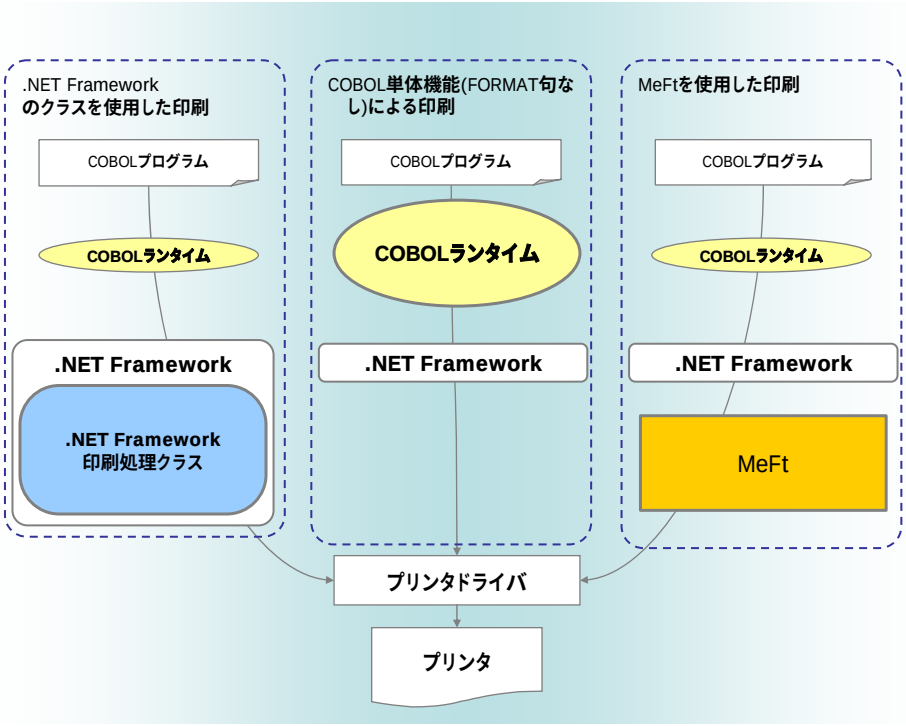


図 6.1.1(1) NetCOBOL for .NETで可能な印刷方法

.NET Frameworkのクラスを使用した印刷

.NET Frameworkクラスライブラリの System.Drawing.Printing 名前空間に含まれる PrintDocumentクラスを使用し、Printメソッドで印刷を行う方法です。また、テキストファイルの印刷は、StreamReaderクラスを使用してレコードを読み取り、System.Drawing.GraphicsクラスのDrawStringメソッドを使用することで実現できます。

COBOL単体機能を使用した印刷

NetCOBOL for Windowsと同様にFOMAT句なしの印刷ファイルの定

All Rights Reserved, Copyright FUJITSU LIMITED 2006

義を行い、行送りやページ替えなど行う行単位のデータを出力する帳票印刷です。

MeFtを使用した印刷

FORMやPowerFORMで作成した帳票定義体を使用する場合の帳票印刷です。
NetCOBOL for Windowsと同様にFORMAT句つき印刷ファイルや表示ファイルの定義を行います。

このように、NetCOBOLを使用した印刷処理の形態として、3通りの方法があります。しかし、プラットフォームで共通なプログラムでは、下図のように、COBOL単体機能を使用した印刷とMeFtを使用する印刷(FORMで作成した帳票定義体を使用する印刷)の2通りの方法が一般的です。

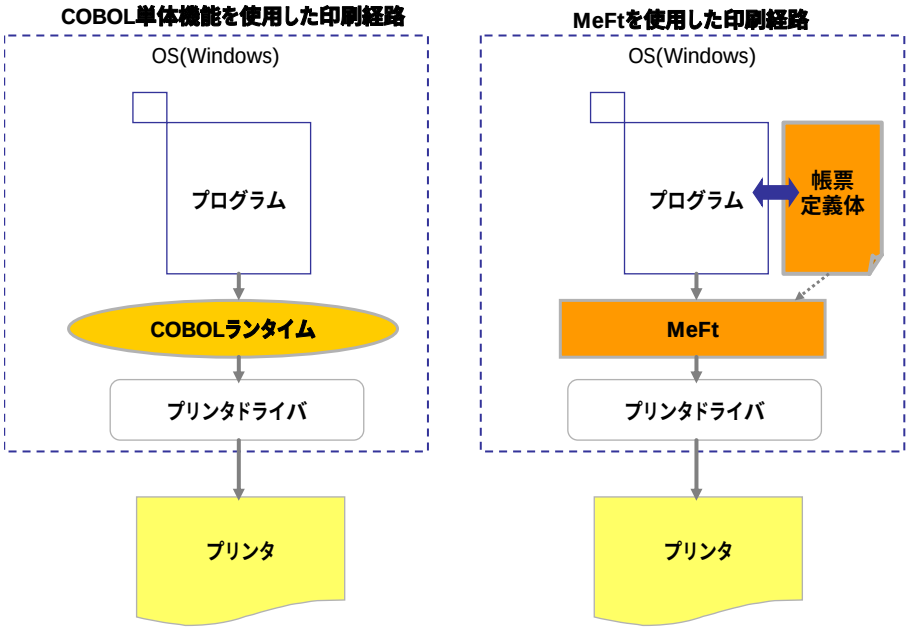


図 6.1.1(2) 印刷経路の違いについて

COBOL単体機能を使用した印刷の場合、NetCOBOL for .NETが印刷処理を行うのに対し、MeFtを使用した印刷の場合、MeFtが印刷処理を行います。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.1.2 印刷機能の概要

NetCOBOLの印刷プログラムで定義する印刷ファイルには、「FORMAT句なし印刷ファイル」「FORMAT句付き印刷ファイル」「表示ファイル」の3パターンがあり、定義ごとに使用できる機能が異なります。

機能	FORMAT句なし		FORMAT句付き	表示ファイル
	行(※1)	オーバレイ(※2)		
行単位のデータの印刷	○	○	○	×
フォームオーバレイとの合成印刷	×	○	○	○
帳票定義体を使った帳票印刷	×	×	○	○
プログラムの記述が簡単	◎	◎	○	◎
帳票形式の印刷が簡単	○	◎	◎	◎
他システムで作成した既存の帳票定義体の利用	×	×	◎	◎
プログラム中で各種印刷情報を指示する	×	◎	◎	○

※1 行単位のデータを印刷する場合
※2 フォームオーバレイ及びFCBを使用して印刷する場合

図6.1.2 印刷機能の概要

FORMAT句なし印刷ファイル

FORMAT句なし印刷ファイルでは、行単位のデータを印刷装置に出力することができます。このとき、論理ページの大きさの指定や行送り、ページ替えを指定することもできます。また、FCBを使用した印刷時の詳細情報(ページ制御)の指定や、フォームオーバレイを利用した印刷が可能です。

FORMAT句付き印刷ファイル

FORMAT句付き印刷ファイルとは、プログラムのファイル定義でFORMAT句を指定した印刷ファイルのことです。FORMAT句付き印刷ファイルでは、パーティション形式の帳票定義体を使って、帳票形式のデータを印刷することができます。また、フォームオーバレイパターンおよびFCBを使った帳票印刷を行うこともできます。FORMAT

All Rights Reserved, Copyright FUJITSU LIMITED 2006

句付き印刷ファイルによる帳票印刷では、実行時にMeFtが必要となります。

表示ファイル

表示ファイルでは、帳票定義体に定義した帳票形式のデータを印刷することができます。FORMAT句付き印刷ファイルとの相違点は、パーティション形式以外の帳票定義体も使用できることです。ただし、行レコードの印刷やフォームオーバレイパターンおよびFCBをプログラムから変更することはできません。表示ファイルによる帳票印刷では、実行時、MeFtが必要になります。

6.1.3 電子帳票への出力

業務のオンライン化やWeb化に伴い、出力先はプリンタ以外の、電子帳票やPDFに対して行うケースが増えてきています。NetCOBOLの印刷プログラムでも、ソフトと連携することによって、電子帳票やPDFに対しての出力を可能としています。

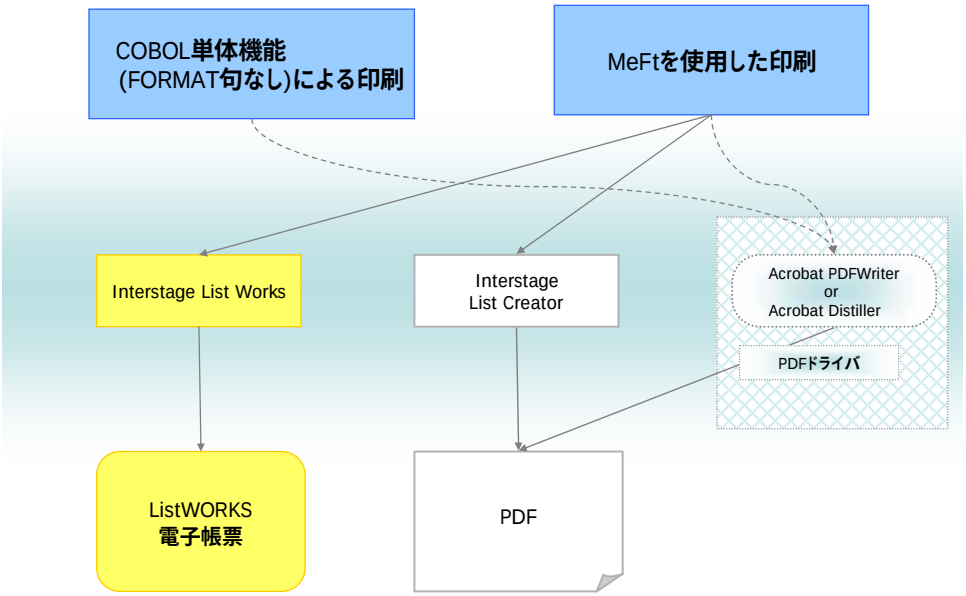


図 6.1.3 電子帳票への出力

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ListWORKS電子帳票

Interstage List Worksを使用してList Works独自形式の電子帳票に出力することが可能です。この場合、プログラムを変更せずに、実行環境ファイルによる設定で簡単に出力することができます。

PDF

Interstage List CreatorのPDF変換機能を使用してPDFへの出力が可能です。この場合、プログラムを変更せずに、実行環境ファイルによる設定で簡単にPDFへ出力することができます。

なお、PDF出力において、Acrobat PDFWriterやAcrobat Destillerから提供されているプリンタドライバを使用したPDF出力も技術的には可能ですが、利用者側で動作検証を行ったうえで使用してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.2 行単位の印刷方法

この章では、COBOL単体機能を使用したFORMAT句なし印刷ファイルによる印刷方法について説明します。

6.2.1 FORMAT句なし印刷ファイル (行単位の印刷、プログラミング)

FORMAT句なし印刷ファイルでは、行単位のデータを印刷装置に出力することが可能です。このとき、論理ページの大きさを指定したり、行送りやページ替えを指定したりすることもできます。FORMAT句なし印刷ファイルは、COBOLファイルシステムのファイル(レコード順や索引)と同様にSELECT句で印刷ファイルとして定義し、手続き部でOUTPUT指定のOPEN文を実行し、創成処理を行います。

データ部

レコードの定義およびファイル管理記述項に指定したデータ名の定義を記述。レコードの定義は、ファイル記述項とレコード記述項で記述。

また、印字文字の大きさ、書体、形態、方向および間隔の値をCHARACTERTYPE句に指定。CHARACTER TYPE句に印字モード名を指定した場合、特殊名段落のPRINTING MODE句で印字モード名に関連付けて印字属性を定義。

手続き部

次の順序で入出力文を実行します。
1.OUTPUT指定のOPEN文: 印刷処理の開始
2.WRITE文: データの出力
3.CLOSE文: 印刷処理の終了

[WRITE文]
1回のWRITE文は、1行のデータを出力します。WRITE文のADVANCING指定にPAGEを記述すると、ページ替えが行われます。また、行数を記述すると、指定した行数が行送りされます。ADVANCING指定には、AFTER指定とBEFORE指定があり、データの出力をページ替えまたは行送りの後に行うか、前に行うかを指定します。

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
  PRINTING MODE MIDASHI1 IS FOR ALL
                IN SIZE 012.0 POINT
                AT PITCH 05.00 CPI.
                FONT F0102.
 INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT 印刷ファイル
        ASSIGN       TO  PRINTER.
DATA DIVISION.
 FILE SECTION.
 FD  印刷ファイル.
 01  印刷レコード       PIC X(40).
WORKING-STORAGE SECTION.
01 見出し１ CHARACTER TYPE IS MIDASHI1
           PRINTING POSITION IS 36
           PIC N(06)
           VALUE NC”帳票サンプル”.
PROCEDURE DIVISION.
   OPEN OUTPUT 印刷ファイル.
   ACCEPT 印刷レコード.
   IF 印刷レコード(1:FUNCTION LENG("/END")) = "/END"
     GO TO 終了処理.
   WRITE 印刷レコード FROM 見出し１
                 AFTER ADVANCING 1 LINE.
   WRITE 印刷レコード.
   GO TO 繰り返し位置.

   CLOSE  印刷ファイル.
END PROGRAM.
```

図6.2.1 FORMAT句なし印刷ファイル(行単位の印刷、プログラミング)

FORMAT句なし印刷ファイルでは、次の指定をプログラムで指定することが可能です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

✓ 論理的な1ページの大きさ(ファイル記述項のLINAGE句)
✓ 文字の大きさ、書体、形態、方向および間隔(データ記述項のCHARACTER TYPE句)
✓ 行送りやページ替え(WRITE文のADVANCING指定)

環境部(ENVIRONMENT DIVISION)

機能名と呼び名の対応付けおよび印刷ファイルの定義を記述します。

データ部

レコードの定義およびファイル管理記述項に指定したデータ名の定義を記述します。レコードの定義は、ファイル記述項とレコード記述項で記述します。

手続き部(PROCEDURE DIVISION)

次の順序で入出力文を実行します。

1. OUTPUT指定のOPEN文: 印刷処理の開始
2. WRITE文: データの出力
3. CLOSE文: 印刷処理の終了

6.2.2 FORMAT句なし印刷ファイル(行単位の印刷、実行)

印刷処理の実行には、プログラムのASSIGN句に指定した印刷ファイル名に対して、実際で使用するプリンタの割り当てが必要です。プリンタの割当てを行う箇所は、ファイル管理記述項のASSIGN句の記述内容によって異なります。ここでは、まず、プログラムまたは実行用の初期化ファイルでの印刷装置の指定方法やASSIGN句の記述内容と出力先の関係について例を用いて説明します。なお、印刷ファイルの出力先をプリンタ以外にした場合、出力内容は保証されません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

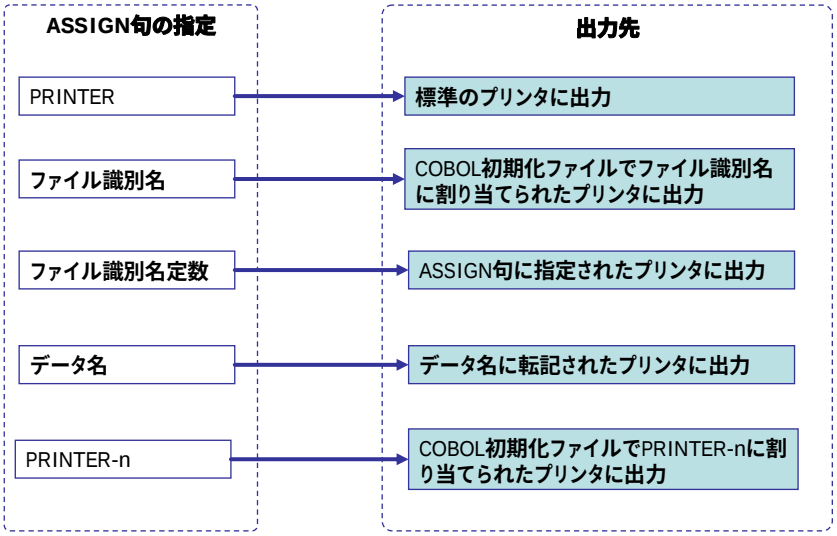


図 6.2.2 FORMAT句なし印刷ファイル(行単位の印刷、実行)

印刷装置の指定は、以下の2種類の方法で設定することができます。

プリンタ名

プリンタ名とは、[プロパティ]ダイアログの[全般]シートのプリンタマークの右横のエディットボックスに表示されているプリンタ名(先頭に"PRTNAME:"を付ける)

ローカルプリンタポート名("LPTn:")、またはシリアルポート名

("COMn:")nは、1～9の範囲です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.3 ページ単位の印刷方法

6.3.1 FCBとI制御レコード

帳票印刷のプログラムでは、数種類の用紙サイズや帳票定義体をプログラムから動的に切り替えたり、印刷データとオーバレイ定義体を重畳して印刷したり、プログラム設計で定義した論理ページと出力するプリンタドライバの情報が一致しているかなどを判断する処理が必要となります。

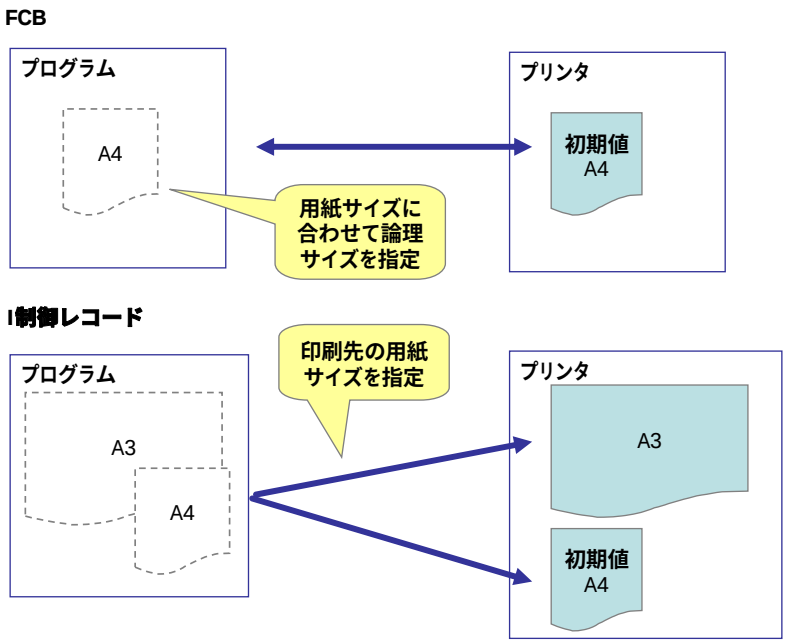


図 6.3.1(1) FCBとI制御レコード

NetCOBOLでは、FCBやI制御レコードといった機能を利用することでこれらの機能を実現できます。

とくに、FCBの指定はFORMAT句なし(定義体を使用しない)印刷ファイルを使用するプログラムでは必須項目です。物理用紙サイズとFCBの指定が合致していない場合、意図した設計どおりの印刷結果が得られない場合があります。

また、I制御レコードは帳票の情報をCOBOLプログラムから切り替えたい場合に使用するレコード形式のことを言います。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(1) FCBの指定例

FCBでは、1ページ分の行数、行間隔および印字開始位置を定義します。この定義は、COBOLランタイムシステムがページ面のフォーマットを認識するためのもので、プリンタドライバに指示するためのものではありません。

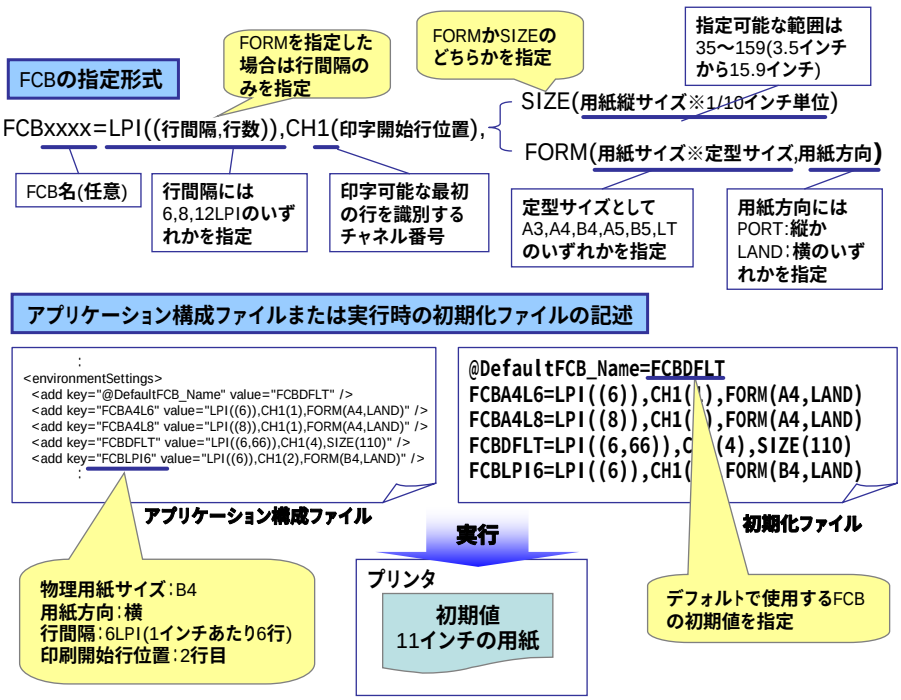


図6.3.1(2) FCBの指定例

FCBを省略した場合、COBOLランタイムシステムは、連帳用紙を想定したサイズや行数、行間の情報(「LPI((6,66)),CH1(4),SIZE(110) 」)をもとに動作します。

FCBは、アプリケーション構成ファイルまたは実行時の初期化ファイルで指定します。複数の帳票を切り替えて出力する場合は、FCBも複数定義し、初期値として使用するFCBを環境変数情報の「@DefaultFCB_Name」に割り当てます。

FCBの指定形式の内、CH情報(チャネル番号)として、1～12が用意されています。チャネル番号には決まった行位置があらかじめ関連付けられていて、チャネル番号を指定した行位置にスキップする動作をするのですが、実際は、CH1固定での指定になり、図の説明もチャネル番号1に限定しています。なお、行間隔の単位であるLPIは、1インチあたりに

All Rights Reserved, Copyright FUJITSU LIMITED 2006

出力する行数を指定します。

(2) I制御レコードの指定例

I制御レコードは、I制御データとして作業場所節に定義したレコードフォーマットに対して、用紙サイズや印刷形式などの情報を設定し、I制御レコードのWRITE文を実行することで、プリンタドライバに指示をおこないます。I制御レコードのWRITE文の実行は、印刷データを出力するためではなく、プリンタドライバに印刷形式の情報を通知するために必要です。実際の印刷データの出力は、I制御レコードのWRITE文の実行後におこないます。

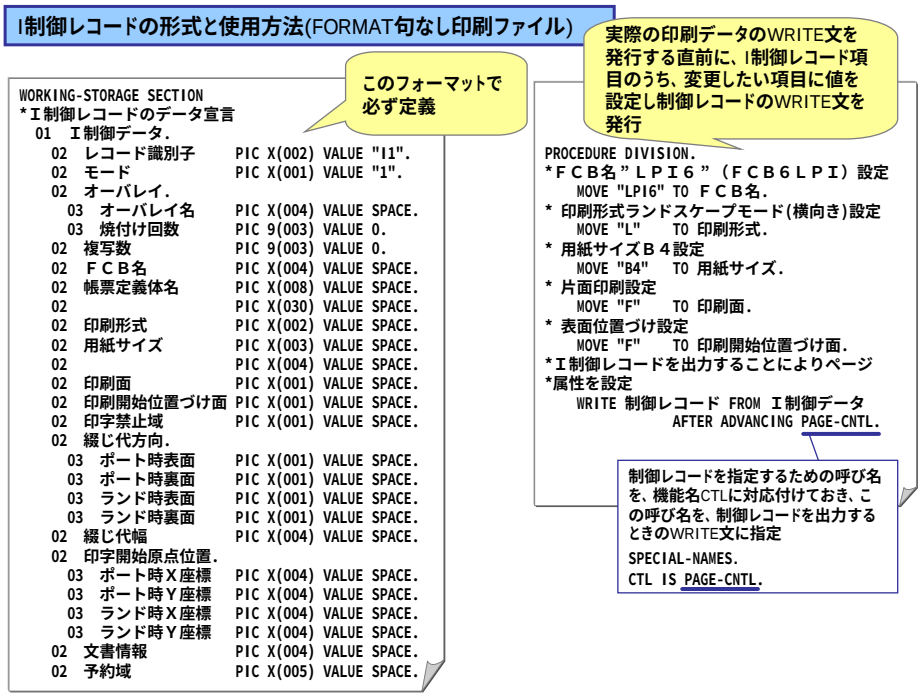


図 6.3.1(3) I制御レコードの指定例(FORMAT句なし印刷ファイル)

Ｉ制御レコードによる用紙サイズや印刷形式などの指定がない(空白や0)場合、COBOLランタイムシステムはプリンタドライバに対して、これらの情報の指示を行わないため、結果としてファイルのオープン時のプリンタドライバの情報(状態)が採用されます。

I制御レコード項目のうち、印刷形式(PRT-FORM)は、単票用紙(カット紙)についての印刷形式を指定する項目です。 また、I制御レコードの機能を使用した場合でも給紙口の指定を行なうことはできません。この場合、I制御レコードで用紙サイズを指定し、プリンタドライバの給紙

All Rights Reserved, Copyright FUJITSU LIMITED 2006

口の指定を自動にしておくことで用紙サイズの情報をもとにドライバ側に給紙口を判断させる方法で対処してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.4 オーバレイ定義体との重畳印刷

6.4.1 オーバレイ定義体とは

オーバレイ定義体とは、帳票定義体作成ツールの「PowerFORM」または「FORM」と「FORM」のオプション製品である「FORMオーバレイオプション」を組み合わせて作成された定義体のことを指し、従来のプレプリント用紙(見出しや枠があらかじめ印刷された専用用紙)に替わって利用します。

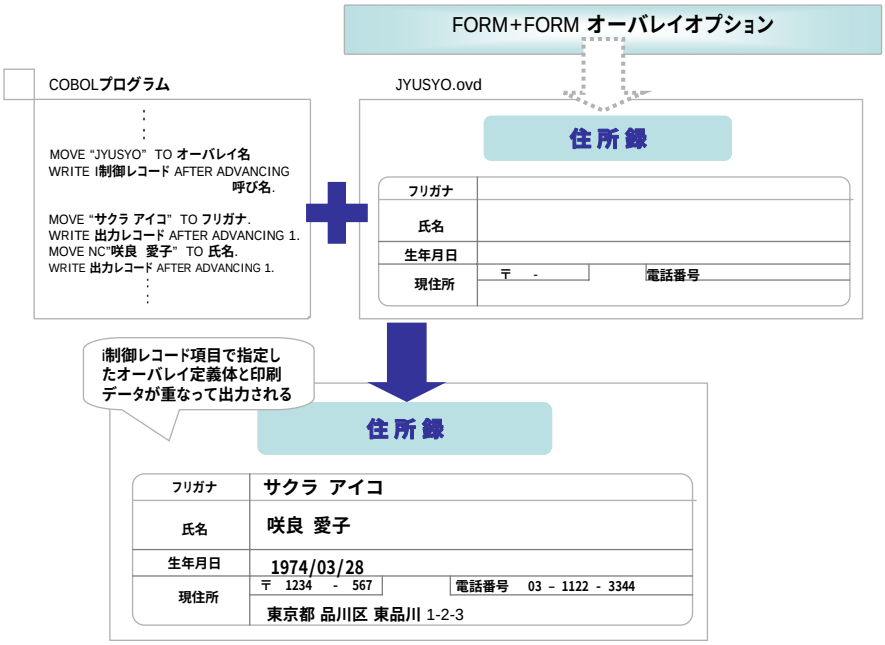


図 6.4.1 オーバレイ定義体との重畳イメージ

COBOLの印刷データと重畳(ちょうじょう)するオーバレイ定義体を作成したら、I制御レコードの、オーバレイ名を格納する項目にオーバレイ定義体名を転記し、I制御レコードをWRITE後、オーバレイ定義体の設計に従って、COBOLデータをWRITEします。上図は、COBOLプログラムとオーバレイ定義体を重畳して出力する場合のイメージ図です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

6.4.2 オーバレイ定義体の活用

オーバレイ定義体を使用することで、プレプリント用紙による帳票出力とどのような面で、効率化や便宜性を図ることができるのでしょうか。

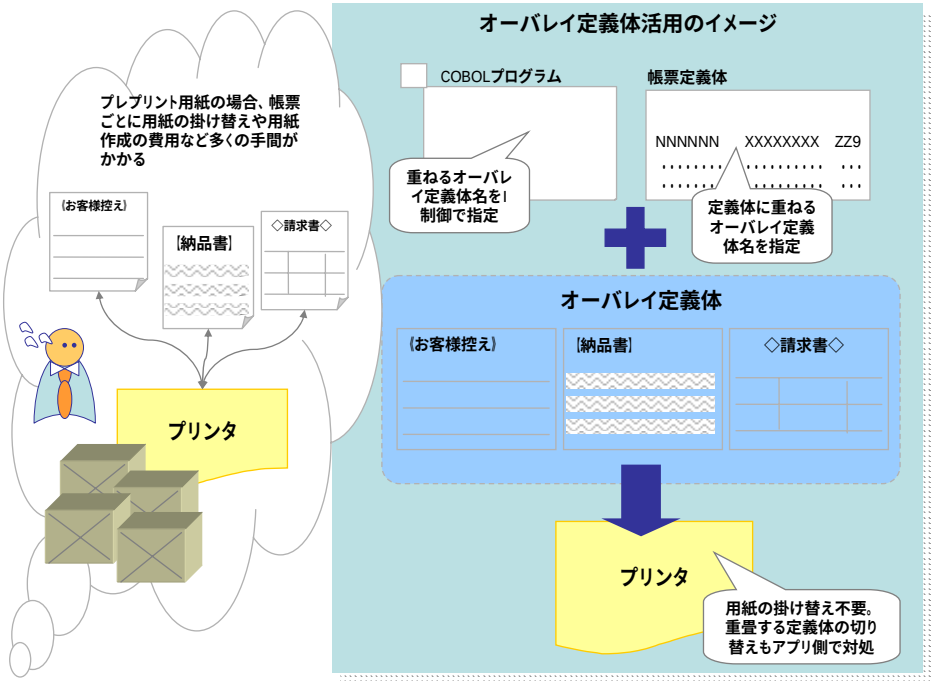


図 6.4.2 オーバレイ定義体の活用

プレプリント用紙による出力処理の場合、出力する帳票によってプレプリント用紙を都度掛け替える手間が発生します。また、プレプリント用紙の作成、印刷において費用の問題が発生します。しかし、オーバレイ定義体を使用することで、プレプリント用紙と同じような帳票を容易に作成することが可能です。また、出力したい帳票に応じて、プログラム側や帳票定義体側で重畳するオーバレイ定義体の指定ができ、またプログラム側でオーバレイ定義体を切り替えて出力することもできます。

6.4.3 印刷データとオーバレイ定義体の重畳方法

作成したオーバレイ定義体を実際の印刷データと重畳する場合、関連付けが必要になります。ここでは帳票印刷のプログラム形式によって、使用するオーバレイ定義体名の関連付けの指定箇所が異なる点について

All Rights Reserved, Copyright FUJITSU LIMITED 2006

説明します。

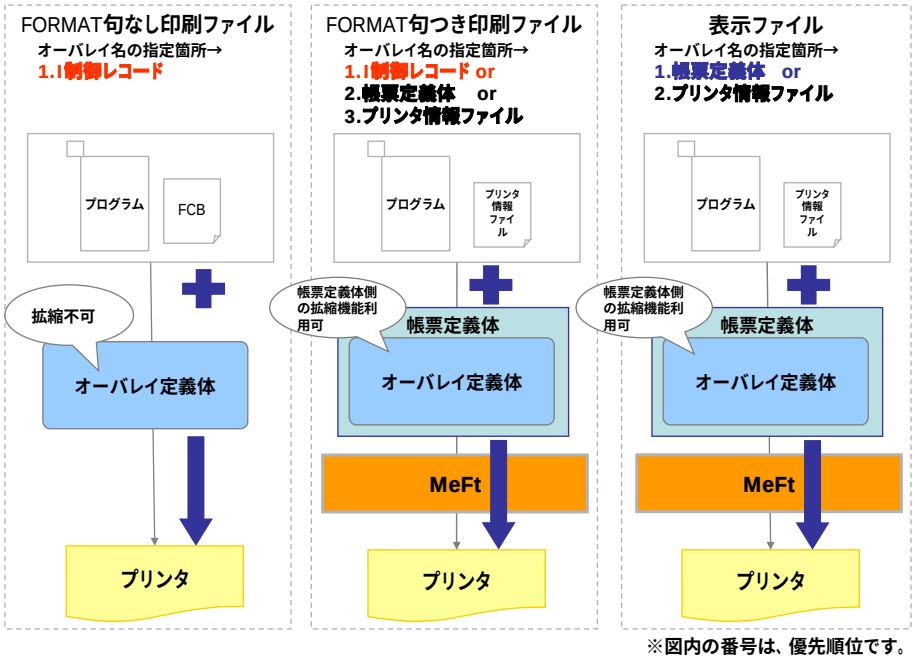


図 6.4.3 プログラムとオーバレイ定義体の関連付けのパターン

FORMAT句なし印刷ファイル

オーバレイ定義体名をプログラムのI制御レコード内の項目に設定しておくことで、関連付けが行われ、実行時、印刷データと重畳出力されます。

FORMAT句つき印刷ファイル

オーバレイ定義体名を次のいずれかに設定しておくことで、関連付けが行われ、実行時、印刷データと重畳出力されます。
なお、オーバレイ定義体名の情報が重複した場合の優先順位は、I制御レコード、帳票定義体、プリンタ情報ファイルの順番となります。

表示ファイル

オーバレイ定義体名を次のいずれかに設定しておくことで、関連付けが行われ、実行時、印刷データと重畳出力されます。表示ファイルではFORMAT句付き印刷ファイルのようにI制御レコードを使用できないため、I制御レコードで指定することはできません。
なお、オーバレイ定義体名の情報が重複した場合の優先順位は、帳票定義体、プリンタ情報ファイルの順番となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

第7章 Webアプリケーションの作成

7.1 ASP.NETの概要
7.2 NetCOBOL for .NETとWebフォームアプリケーション
7.3 ASP.NETのクラスとWebフォームアプリケーション作成のポイント
7.4 Webページの遷移方法とWebページ間の情報共有方法
7.5 Webフォームアプリケーションの作成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.1 ASP.NETの概要

7.1.1 WebフォームとASP.NET

ASP.NET は、.NET Framework 上でWebシステムを開発するためのプラットフォームです。ASP.NETで動作する画面をWebフォームとよび、ASP.NETで作成したアプリケーションをWebサイトやASP.NET Webフォームアプリケーションと呼びます。NetCOBOL for .NETでもASP.NETを利用してWebシステムを開発/運用することができます。

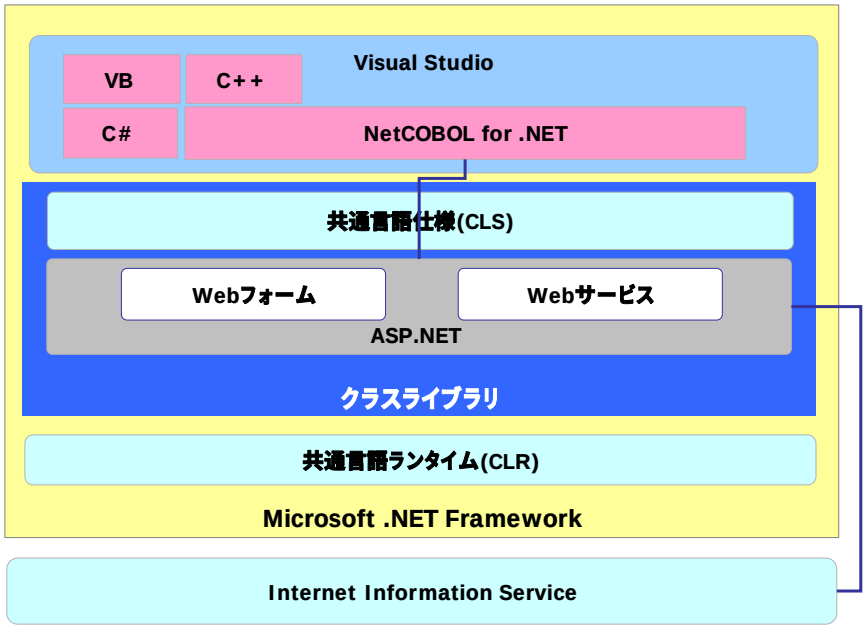


図7.1.1.(1) ASP.NETの構成イメージ

Webフォームアプリケーションは、Webの機能を十分に利用できる形式で、プログラミングの自由度が高く、ビジュアルなアプリケーションを作成することが可能です。処理のほとんどをサーバ上で実行し、クライアントは基本的にブラウザしか必要ありません。このような形式をシン(Thin)クライアントと呼ぶこともあります。

Webフォームアプリケーションは4章で説明したWindowsフォームアプリケーションとよく似ています。Webフォームアプリケーションは、いわゆるイベントドリブン(イベント駆動型)です。「ボタンのクリッ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ク」「画面のロード」など、イベントごとに手続きを記述します。

しかし通常は、すべての処理をイベント手続きに記述するのではなく、データアクセスやビジネスロジックをクラスにまとめておき、Webフォームから呼び出します。こうすることで、プログラムの再利用性を高めることができます。

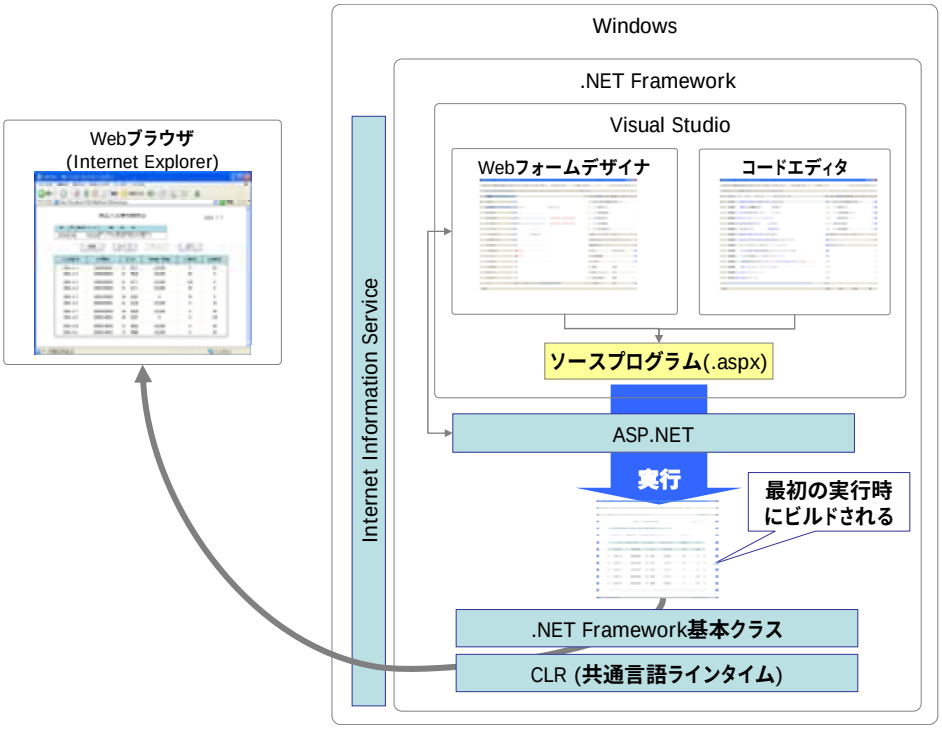


図7.1.1.(2) Webフォームアプリケーションの開発/実行

ASP.NETアプリケーションの運用には、Microsoft Internet Information Service (IIS)で構成されたWebサーバが必須であり、他のインターネットサーバを利用することはできません。ただし、ASP.NETアプリケーションの開発・デバッグでは、IISは必ずしも必須ではありません。

なお、Internet Information Serviceでは、Web アプリケーション環境として、 Active Server Pages (ASP) が用意されていますが、ASP.NETはASPを大幅に拡張/変更しており、互換性はありません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.1.2 Webフォームアプリケーションの運用形式

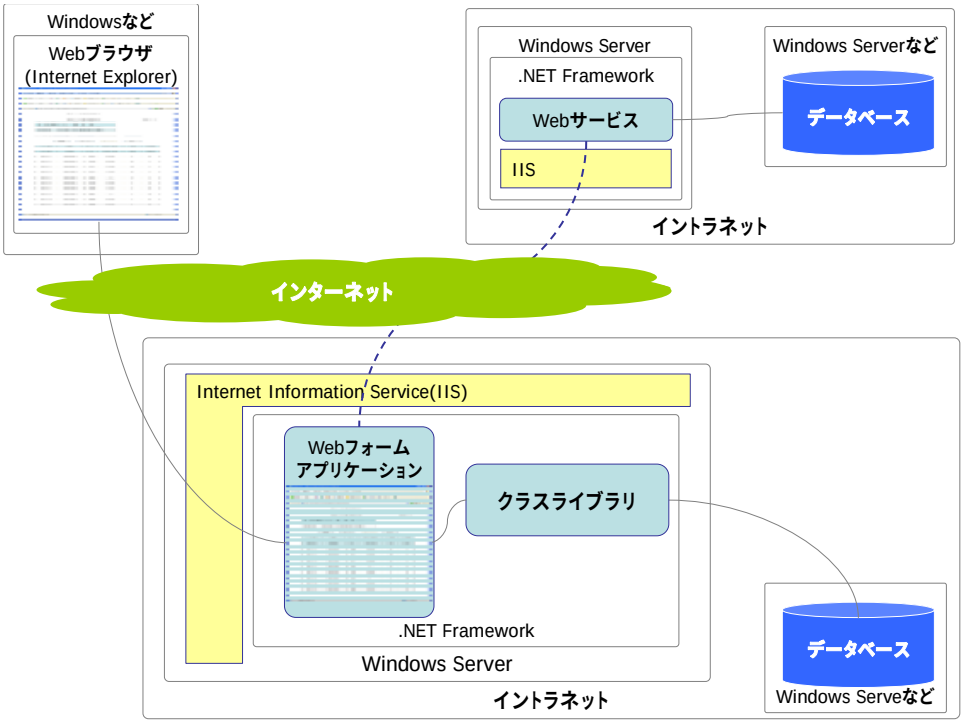


図7.1.2 Webフォームアプリケーションの運用形式

Webアプリケーションの運用形式は、Webサーバとデータサーバおよびwebクライアントの組み合わせが一般的です。また、ビジネスロジックはクラスライブラリにまとめ、Webサーバのフロント部分と分離することにより、いわゆる3階層のシステムを構築できます。

さらに、クラスライブラリをWebサービス化すると、ビジネスロジックを別サーバに分離することができます。つまり、Webフォームアプリケーションが Webサービスクライアントとなります。

Webサービスは、WebサービスとWebサービスクライアントの間を、インターネットを介して接続することも可能です。例えば、「1.1.6 NetCOBOL for .NETの活用イメージ」のようなシステムが可能となります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.2 NetCOBOL for .NETと Webフォームアプリケーション

7.2.1 NetCOBOLのWebフォームアプリケーションの概要

当然のことながら、NetCOBOL for .NETでもWebフォームアプリケーションを構築することができます。

「7.1 Webフォームアプリケーションの概要」で説明したように、Webフォームの機能は、.NET FrameworkのASP.NETで提供されるため、NetCOBOL for .NETからも他の言語と同様にWebフォームの機能を活用することができます。

NetCOBOL for .NETのWebアプリケーションからは、他の言語で作成したクラスを呼び出すことも可能です。しかし、どちらかというと逆のパターンである、NetCOBOL for .NETでビジネスロジックをクラスライブラリとして作成し、VBなどの言語でWebフォームアプリケーションを作成することの方が現実的です。これは、VBやC#のほうがクラスやオブジェクトを扱うのに向いていることと、COBOLの場合、既存の膨大な資産があるため、クラスライブライ化することで、過去の資産を効果的に活用することができるためです。

7.2.2 Webフォームアプリケーションの作成イメージ

NetCOBOL for .NETで、Web フォームアプリケーションを構成するファイルとして、.aspxファイルと.aspx.cobxファイルがあります。

.aspxファイルは、ボタンやラベルなど画面レイアウトや情報を格納するファイルで、Webページデザイナ（Webフォームデザイナとも呼ばれる）で作成します。拡張子は.aspxですが、内容はコントロール要素を含んだHTMLベースのテキストファイルです。クライアントからIISを通して.aspxファイルの要求があると、ASP.NETはWebフォームのコントロールをHTMLに変換してクライアントに送信します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

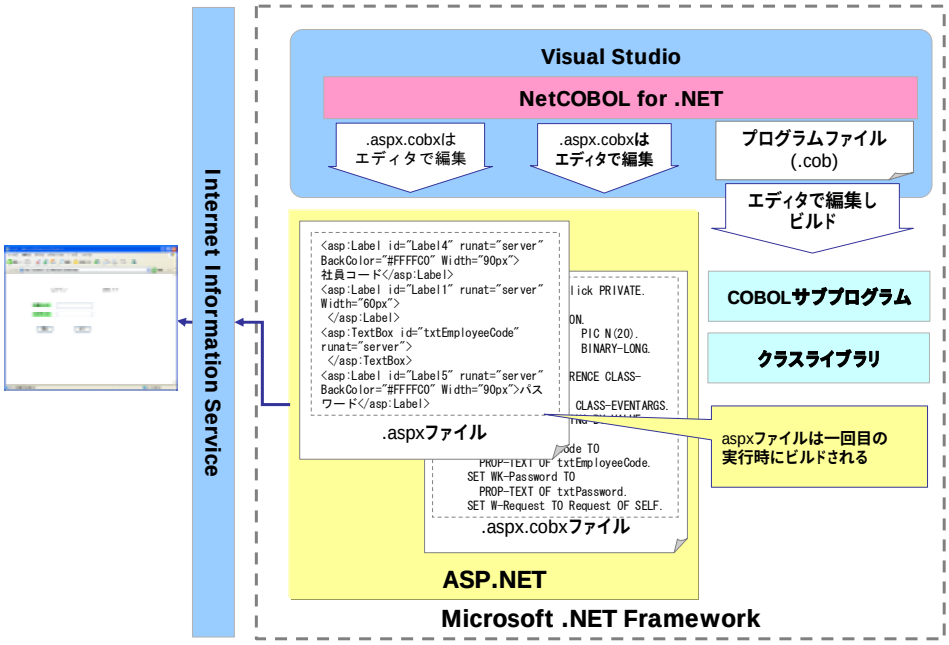


図7.2.2 Webフォームアプリケーションの作成イメージ

.aspx.cobxファイルは、ボタンが押されたときのイベント手続き(メソッド)などの「.aspxファイルに従属する手続き」を格納するファイルです。つまり、発生したイベントにかかわる処理をCOBOLのオブジェクト指向プログラミング機能を利用して記述します。ファイルとしては、COBOLで作成したクラス型のファイルになります。クライアントなどからイベント手続きの実行要求があると(例:ボタンのクリック)、イベント手続きに記述された処理を実行します。

通常、.aspxおよび.aspx.cobxファイルは、ユーザーがページを要求したときに動的にビルドされますが、これらをあらかじめプリコンパイルしておくこともできます。

7.2.3 NetCOBOL for .NETでのWebフォームアプリケーション開発（画面の作成）

Webフォームのデザインは、Webページデザイナで行います。Web

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ページデザイナはWindowsフォームデザイナと非常に良く似ています。したがって、Webアプリケーションであることを意識せずに画面のデザインが可能です。

(メニューから).aspxファイルを新規に作成したり、ソリューションエクスプローラで.aspxファイルを選択したりすると、Webページデザイナで.aspxファイルが開きます。

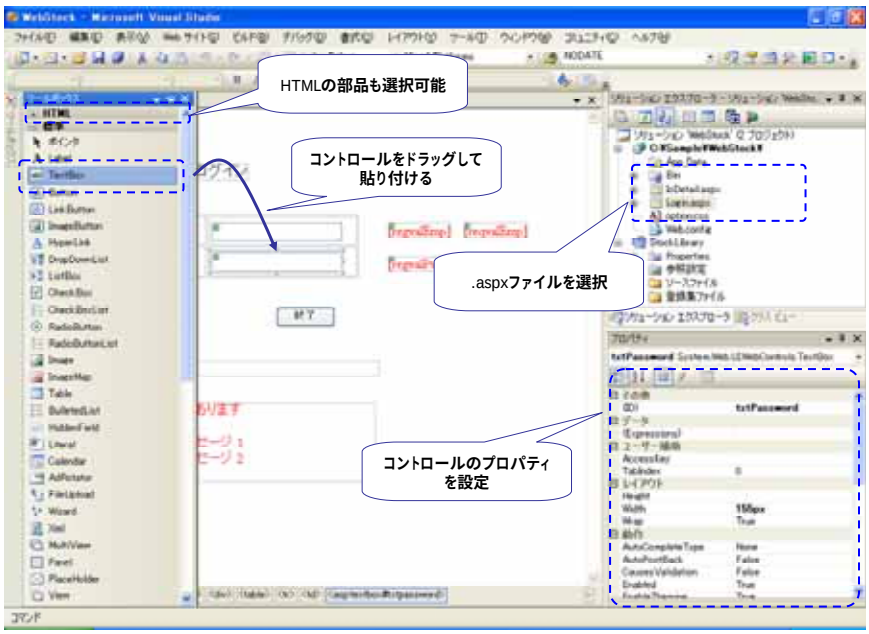


図7.2.3 Webフォームアプリケーションの開発(画面の作成)

WebページデザイナはWindowsフォームデザイナに似たGUIによる画面デザイン機能を提供しています。例えば、ツールボックスから必要なWebフォームのコントロールを貼り付けたり、HTMLの部品を貼り付けたりする操作をマウスによるGUI操作で行います。コントロールのプロパティ(属性、データ)はプロパティウィンドウで編集することができます。

7.2.4 Webフォームアプリケーション開発 (イベント手続きの編集)

Webページデザイナによる.aspxファイルの編集結果は、HTML形式で保存されます。デザイナ下部の[ソース]タブをクリックすると、ソース

All Rights Reserved, Copyright FUJITSU LIMITED 2006

（HTML）表示に切り替えることができます。

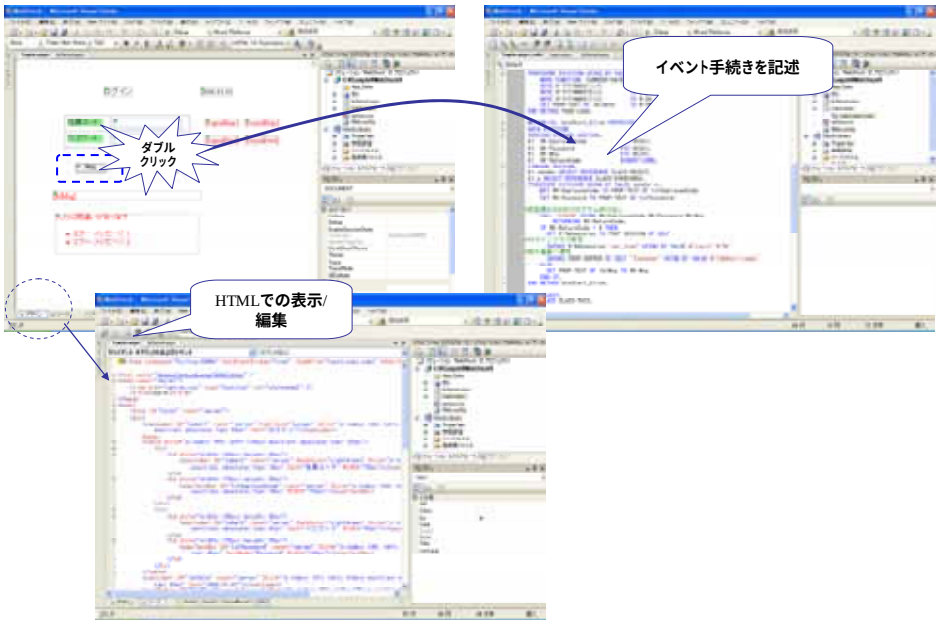


図7.2.4 Webフォームアプリケーションの開発

イベントハンドラ(イベント発生時に実行される手続き)は、.aspx.cobxファイルに記述しますが、コントロールをダブルクリックすると、コードウィンドウ(エディタ)が表示され、目的の手続きの入力箇所が表示されます。コントロールをダブルクリックすることで、イベント処理に必要な定義やメソッドの雛形が作成されます。[表示]メニュー-[コード]などでコードウィンドウを開いた場合は、定義や雛形が作成されませんので注意する必要があります。

また、すでにイベント手続きを作成済みのコントロールについて、デザイナ画面でダブルクリックを行っても多重定義は行われません。コントロールの削除を行った場合、コードエディタでイベントハンドラを削除します。コントロールを削除せずに、コントロールのイベントハンドラを削除した場合、コントロールのプロパティリストの「イベント」タブからイベントとイベント手続きの関連付けを削除します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.3 ASP.NETのクラスと Webフォームアプリケーション作成ポイント

7.3.1 ASP.NETのクラス(名前空間)

Webフォームアプリケーションを作成する場合、Webならではの制御や、通信に関する処理を行うことが必要な場合があります。そのようなWebに関連する処理は.NET Frameworkのクラスで提供されています。

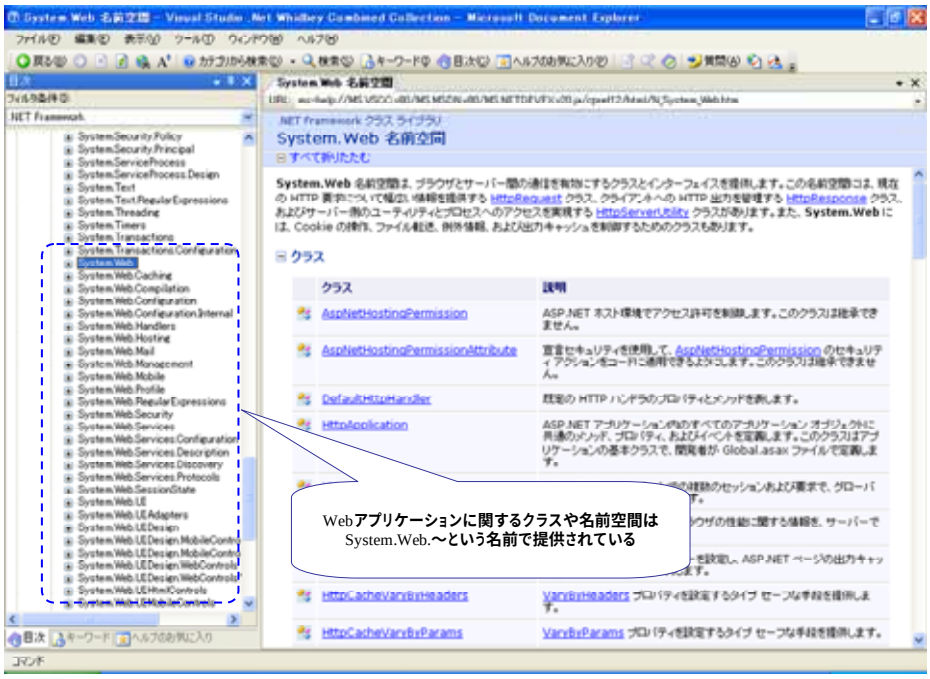


図7.3.1 Webアプリケーションに関するクラスのリファレンス

したがって、効果的なWebフォームアプリケーションを作成する際には、.NET Frameworkのクラスで提供される機能を調べながらプログラミングします。そこで事前に、.NET Frameworkのクラスの概要を知っておくと、リファレンスの検索が楽です。

次に、代表的な名前空間とクラスについて説明します。

(1) System.Web名前空間

System.Web名前空間では、WebブラウザとWebサーバ間の通信に関

All Rights Reserved, Copyright FUJITSU LIMITED 2006

するクラスが提供されています。

System.Web名前空間には、現在のHTTP要求について幅広い情報を提供する HttpRequest クラス、クライアントへの HTTP 出力を管理する HttpResponseクラス、およびサーバ側のユーティリティとプロセスへのアクセスを実現するHttpServerUtilityクラスがあります。

また、 System.Webクラスには、Cookieの操作、ファイル転送、例外情報、および出力キャッシュを制御するためのクラスもあります。

(2) System.Web.SessionState名前空間

System.Web.SessionState名前空間では、ブラウザとサーバ間のセッションの状態に関するクラスが提供されます。特にHttpSessionStateクラスは、Webページ間の情報の受け渡しに利用される非常に重要なクラスです。

(3) System.Web.UI名前空間

System.Web.UI名前空間では、Webフォームアプリケーションの画面(ページ)処理に関するクラスが提供されます。

たとえば、すべてのASP.NETサーバコントロールが共有するプロパティ、メソッド、およびイベントが提供されたControlクラスや、Windowsフォームの画面(.aspxファイル)を表すPageクラスなどがあります。

(4) System.Web.UI.WebControls 名前空間

System.Web.UI.WebControls 名前空間では、Webページに Webサーバコントロールを作成するときに使用するクラスが提供されます。つまり、Webフォーム上に貼り付けられたテキストボックスなどの部品に関するクラスの集まりです。

Webサーバコントロールはサーバ上で実行され、Webクライアント側にはHTMLタグとして送信されます。

7.3.2 Webフォーム画面の処理概要

Webフォームアプリケーションを作成する場合、「プログラムから画面情報へのアクセス」と「Webページの遷移とWebページ間の情報の共

All Rights Reserved, Copyright FUJITSU LIMITED 2006

有」がポイントになります。

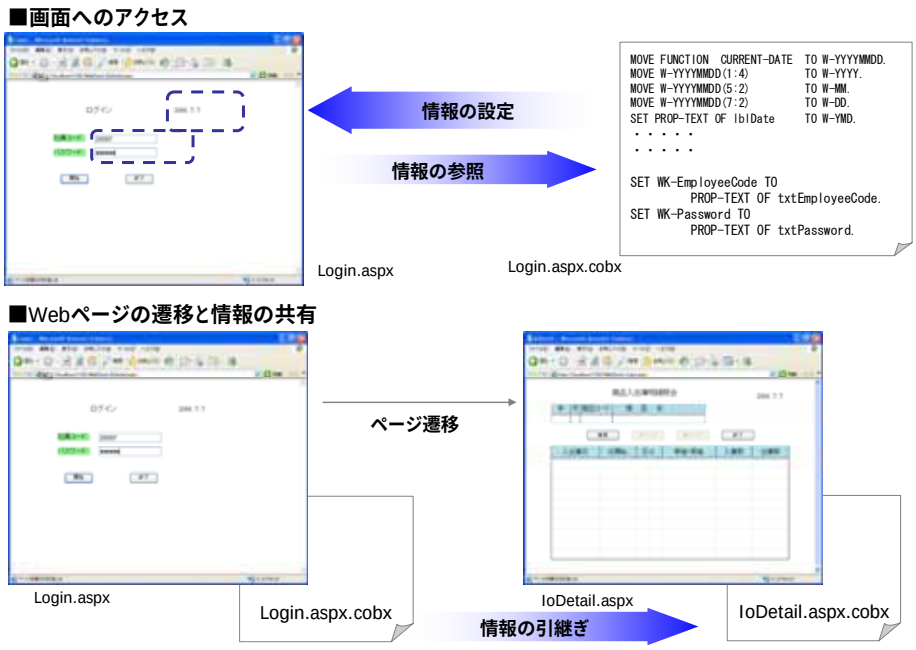


図7.3.2(1) Webフォーム画面の処理画面概要

(1) プログラムから画面情報へのアクセス

Webアプリケーションでは、Webブラウザから情報を受け取り、サーバで要求された処理を行い、動的にWebページを作成しクライアントに送信することが必要です。

Webページへの情報の入出力は、通常Webコントロールを利用して行います。したがって、Webコントロールの操作方法が重要です。

Webコントロールに入力された情報を受け取ったり、Webコントロールに情報を表示したりする方法は、Windowsアプリケーションの場合とほとんど同じで、オブジェクトへのアクセスとなります。したがって、SET文やMOVE文とINVOKE文でプロパティを参照したり、メソッドを呼び出したりすることになります。

使用するWebコントロールは.NET FrameworkのASP.NETのクラスとして提供されますので、ASP.NETの機能を把握する必要があります。逆に、HTMLやXML、Javaなどの知識はまったく必要ありません。

例えば、Webコントロールを利用して作成した入力用のテキストボックス(TextBox1、TextBox2)に、Webブラウザから入力された文字列を

All Rights Reserved, Copyright FUJITSU LIMITED 2006

取得するサンプルは次のようになります。この部分の手続きについては、Windowsアプリケーションの場合とまったく同じです。

```
WORKING-STORAGE SECTION.
01  WK-EmployeeCode            PIC N(06).
01  WK-Password                PIC N(06).

LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    SET WK-EmployeeCode TO PROP-TEXT OF TextBox1.
    SET WK-Password TO PROP-TEXT OF TextBox2.
```

(2) Webページの遷移とWebページ間の情報の共有

一般的なWebアプリケーションでは、認証処理でWebシステムへのログイン情報を別の画面に引き継いだり、ショッピングカートのようにWebページ間で情報を共有したりする必要があります。

しかし、Webシステムは、1画面ごとにサーバークライアント間の接続が切断され、画面ごとに情報が独立しているため(メモリを共有しない)、情報を共有するための特別な仕組みが必要となります。

ASP.NETでは"System.Web.SessionState.HttpSessionState"クラスや"Control.ViewState"プロパティによりページ間の情報の共有やページ情報の保存が可能になっています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

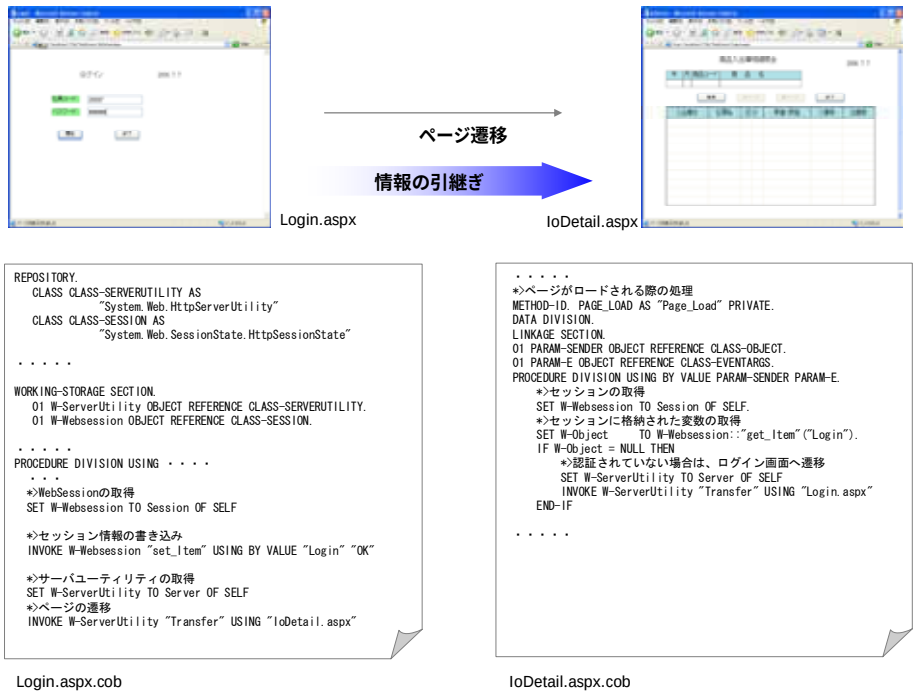


図7.3.2(2) ページ遷移時の処理

Webの画面を別の画面に遷移させるには、System.Web.HttpResponseクラスのRedirectメソッドか、System.Web.HttpServerUtilityクラスのTransferメソッドを使用します。

また、ページ間の情報の引渡しは、ResponseRedirectメソッドでURLを指定する際、クエリ文字列を指定し、HttpRequest.QueryStringプロパティで取得するか、HttpSessionStateクラスを利用します(それぞれ、RedirectメソッドでもTransferメソッドのどちらの遷移方法でも利用できます)。

これらのページ処理で利用されるクラスは、組み込みオブジェクトとして提供されており、プログラムでインスタンスの作成を行うことなく利用できます。

(補足)組み込みオブジェクト

ASP.NETではWeb機能の代表的なクラスを組み込みオブジェクトとして提供しています。組み込みオブジェクトとしてはApplication、Sever、Session、Response、Requestなどがあります。

Webフォーム(System.Web.UI.Pageクラス)がロードされると、ASP.NETによりこれらの組み込みオブジェクトのインスタンスが自動的

All Rights Reserved, Copyright FUJITSU LIMITED 2006

に作成されます。さらにこれらのインスタンスは、ベースとなるPageオブジェクトのプロパティから参照可能となります。したがって、明示的にインスタンスを作成することなく、組み込みオブジェクトを利用することができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.4 Webページの遷移方法と Webページ間の情報共有方法

7.4.1 クライアントサイドのページ遷移処理(Redirect)

System.Web.HttpResponse.Redirectを使用したページ遷移処理について説明します。

Redirectメソッドは、ブラウザにコマンドを送信し、一旦ブラウザ側に制御を戻します。続いて、ブラウザ側から指定したURLにジャンプします。したがって、ブラウザの履歴やアドレスバーのURLが更新されます。しかし、パフォーマンスが低下するなどの欠点があります。

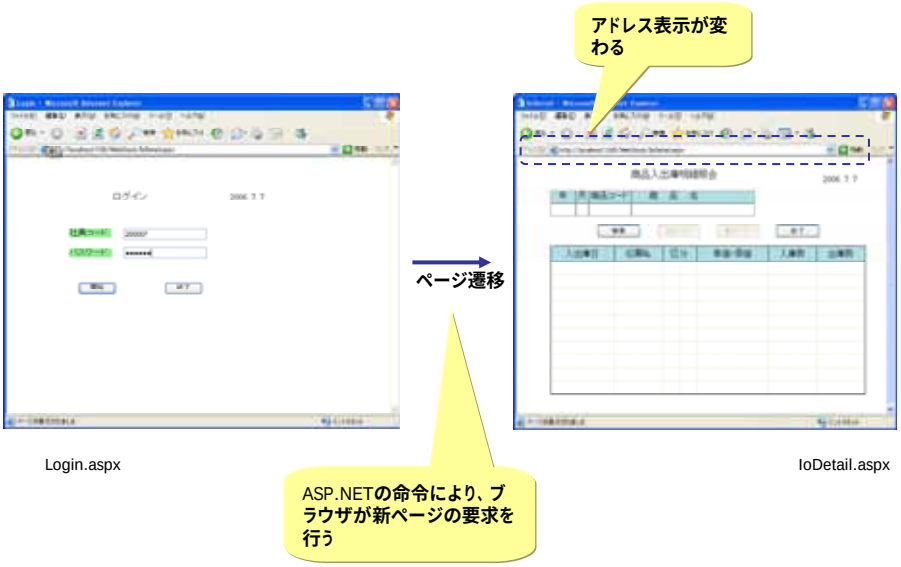


図7.4.1 クライアントサイドのページ遷移処理

Redirectメソッドの利用方法

WebフォームはSystem.Web.UI.Pageクラスを継承したオブジェクトであり、Responseプロパティを参照することで、Webアプリの応答を管理しているHttpReposenseオブジェクトへアクセスすることができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

HttpResponseオブジェクトのRedirectメソッドを呼び出すことにより、別のページへ遷移することができます。その際に、クエリ文字列を指定することで、遷移先ページへ情報を渡すことができます。
NetCOBOL for .NETで自分自身のプロパティを参照する場合は、プロパティ取得対象オブジェクトとしてSELFを記述します。

Redirectメソッドの利用例

(リポジトリ段落)

```
PROPERTY PROP-RESPONSE AS "Response"
```

(手続き部)

```
INVOKE PROP-RESPONSE OF SELF "Redirect"
 USING N"IoDetail.aspx?Param=ABC"
```

(補足)クエリ文字列
クエリ文字列は、ページのURLの最後に追加される情報です。一般的な例を次に示します。
http://www.contoso.com/listwidgets.aspx?category=basic&price=100

上記のURLパスのクエリ文字列は、疑問符(?)で始まり、"category"と"price"という 2つの属性値を含んでいます。

7.4.2 クエリ文字列の取得

.aspxページにクエリ文字列つきURLがリダイレクトされた場合のクエリ文字列の取得方法について説明します。

.aspxページにクエリ文字列が付いたURLがリダイレクトされた場合、ASP.NETではSystem.Web.HttpRequestオブジェクトに格納されます。HttpRequestオブジェクトからクエリ文字列を取得するためのプロパティとしてQueryStringが用意されています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

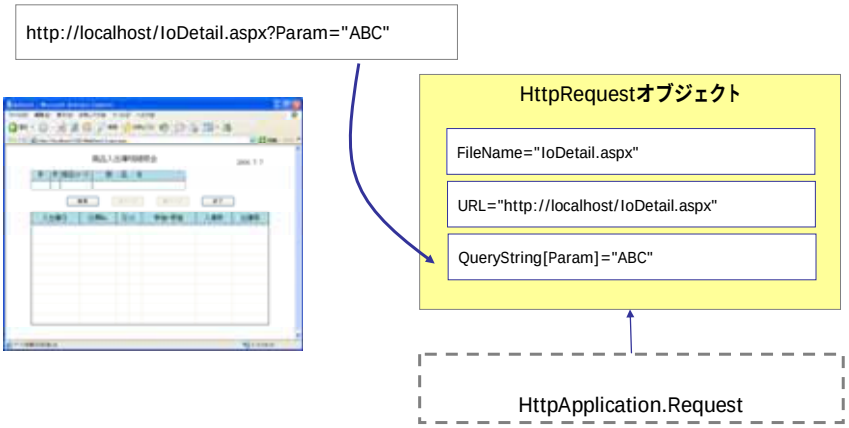


図7.4.2 クエリ文字列の取得方法

QueryStringプロパティの利用方法

Webフォームは System.Web.UI.Pageクラスを継承したオブジェクトであり、Requestプロパティを参照することで、Httpクライアントからの情報を管理しているHttpRequestオブジェクトへアクセスすることができます。

HttpRequestオブジェクトのQueryStringプロパティを参照することにより、クエリ文字列として格納された情報を参照することができます。

NetCOBOL for .NETで自分自身のプロパティを参照する場合は、プロパティ取得対象オブジェクトとしてSELFを記述します。したがって、HttpRequest.QueryStringプロパティは次の手順で呼び出すことができます。

QureyStringプロパティの利用例

(リポジトリ段落)

```
CLASS CLASS-STRING AS "System.String"
PROPERTY PROP-QUERYSTRING AS "QueryString"
PROPERTY PROP-REQUEST AS "Request"
PROPERTY PROP-TEXT AS "Text"
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

(データ部)

```
01 WORK-STRING OBJECT REFERENCE CLASS-STRING.
```

(手続き部)

```
SET WK-QUERYSTRING TO PROP-QUERYSTRING OF
       PROP-REQUEST OF SELF.
SET WORK-STRING TO
       WK-QUERYSTRING::"get_Item"(N"Param").
```

※例にはありませんが、クエリ文字列を受け取る処理には、セキュリティ対策のためデータのチェック処理を加えることが重要です。

7.4.3 サーバサイドのページ遷移処理(Transfer)

System.Web.HttpServerUtility.Transferメソッドを使用したページ遷移について説明します。

Transferメソッドは、ブラウザ側に制御を戻さずに直接別の画面へ遷移します。したがって、ブラウザの履歴やアドレスバーのURLは更新されません。一方、「7.4.1 クライアントサイドのページ遷移処理（Redirect）」で述べたRedirectメソッドは、一旦ブラウザ側に制御を戻し、ブラウザ側から別の画面へ遷移しました。この違いから、Transferメソッドの方がRedirectメソッドよりパフォーマンスが良いという利点があります。しかし、Transferメソッドは同一Webアプリケーション内のWebフォームにしか遷移することはできないという制限があります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

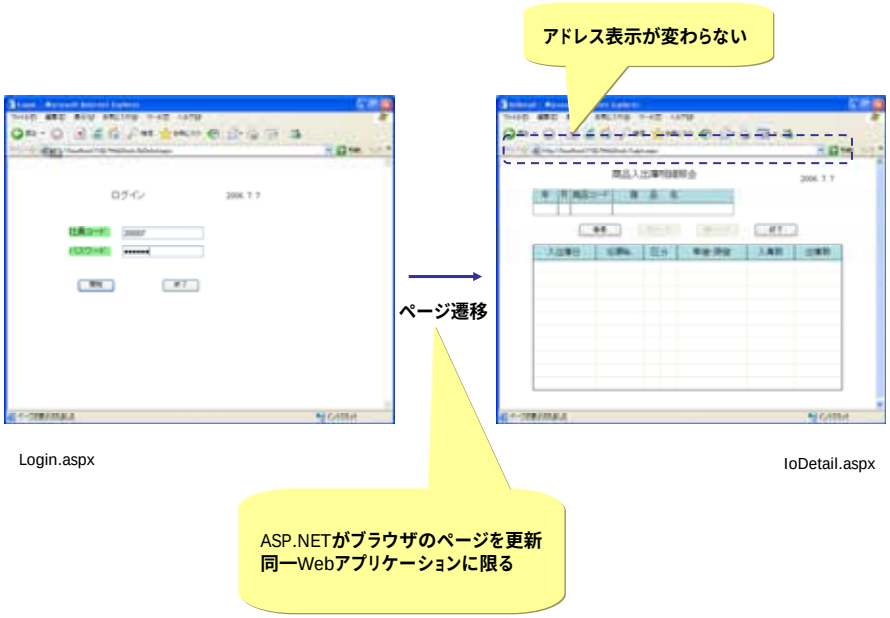


図7.4.3 サーバサイドのページ遷移処理(Transfer)

Transferメソッドの利用方法

WebフォームはSystem.Web.UI.Pageクラスを継承したオブジェクトであり、Serverプロパティを参照することで、Web要求を処理するHttpServerUtilityオブジェクトへアクセスすることができます。HttpServerUtilityオブジェクトのTransferメソッドを呼び出すことにより、現在のURLページのパスを利用して新しいページを実行することができます。

NetCOBOL for .NETで自分自身のプロパティを参照する場合は、プロパティ取得対象オブジェクトとしてSELFを記述します。したがって、Transferメソッドは次の手順で呼び出すことができます。

Transferメソッドの利用例

(リポジトリ段落)

```
PROPERY PROP-SERVER AS "Server"
```

(手続き部)

```
INVOKE PROP-SERVER OF SELF "Transfer"
                USING BY VALUE N"IoDetail.aspx"
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.4.4 ページ遷移時のデータ取得(Sessionオブジェクト)

System.Web.SessionState.HttpSessionStateクラスを利用したWebページ間のデータ共有方法について説明します。

7.4.1および7.2.2で説明した、クエリ文字列を使用したページ間の情報の受け渡しでは、ブラウザに文字列が表示されるなどのセキュリティ上の問題があり、重要な情報の受け渡しには不適切です。
そのため、ASP.NETでは、Webクライアントに依存しない情報を受け渡すための方法として、HttpSessionStateクラスが用意されています。

ページを受け渡す側でHttpSessionStateオブジェクトに情報を設定します。続いて、ページを受け取る側でHttpSessionStateオブジェクトから情報を取得します。HttpSessionStateオブジェクトに格納された情報は、セッションが保持されている間維持されます。

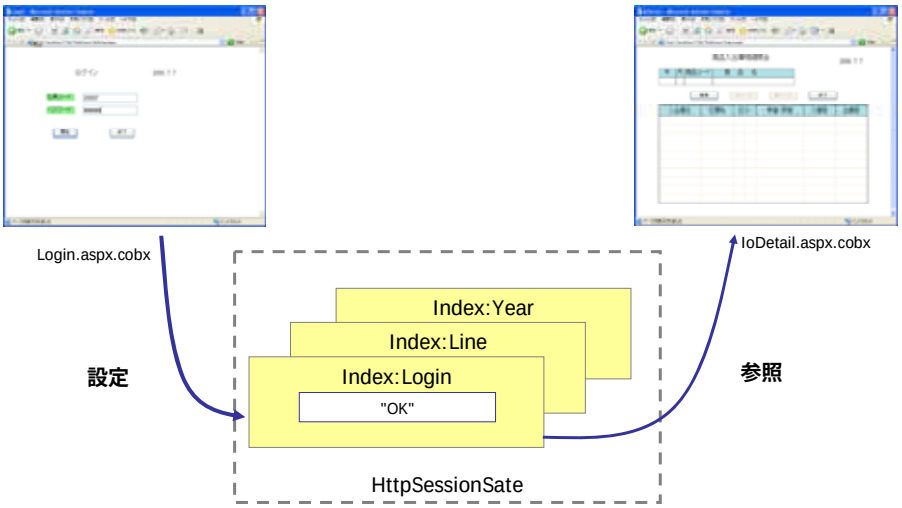


図 7.4.4 ページ遷移時のデータ取得(Sessionオブジェクト)

(1) HttpSessionStateオブジェクトへのデータ格納

HttpSessionStateオブジェクトへのデータ格納方法

WebフォームはSystem.Web.UI.Pageクラスを継承したオブジェクトであり、Sessionプロパティを参照することで、セッションの情報を

All Rights Reserved, Copyright FUJITSU LIMITED 2006

管理できるHttpSessionStateオブジェクトへアクセスすることができます。セッションのデータは、HttpSessionStateオブジェクトのItemインデクサを介してコレクションに格納されています。コレクションにはItemインデクサにsetアクセサを使用してデータを格納することができます。NetCOBOL for .NETで自分自身のプロパティを参照する場合は、プロパティ取得対象オブジェクトとしてSELFを記述します。したがって、HttpSessionStateを利用したデータの格納は次の手順で呼び出すことができます。

HttpSessionStateオブジェクトへのデータ格納例

(リポジトリ段落)

```
PROPERTY PROP-SESSION AS "Session"
```

(手続き部)

```
INVOKE PROP-SESSION OF SELF "set_Item"
              USING BY VALUE N"認証" N"OK"
```

(2) HttpSessionStateオブジェクトからのデータの取得

HttpSessionStateオブジェクトからのデータ所得方法

HttpSessionStateオブジェクトからItemインデクサを利用し格納されたデータを取得することができます。データの取得にはgetアクセサを利用します。

HttpSessionStateオブジェクトの取得は、HttpSessionStateオブジェクトへのデータの格納の場合と同様にPageオブジェクトのSessionプロパティを利用します。

HttpSessionStateオブジェクトからのデータ取得例

(リポジトリ段落)

```
PROPERTY PROP-SESSION AS "Session"
```

(データ部)

```
01 W-Object OBJECT REFERENCE CLASS-OBJECT.
```

(手続き部)

```
INVOKE PROP-SESSION OF SELF "get_Item"
       USING BY VALUE N"Login" RETURNING W-Object
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.5 Webフォームアプリケーションの作成

それでは、実際にWebフォームアプリケーションの作成を行ってみます。

7.5.1 プログラム概要

最初にログイン画面で認証を行い、次画面で年、月、商品コードを元に商品の入出庫明細照会を行うWebフォームアプリケーションを作成します。

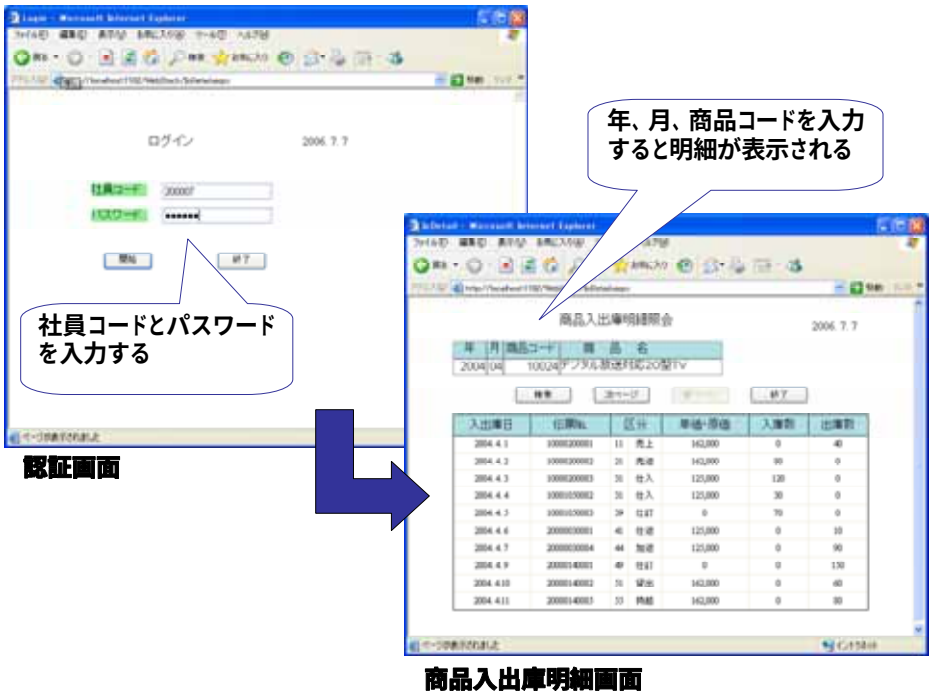


図7.5.1 Webフォームアプリケーションの実行イメージ

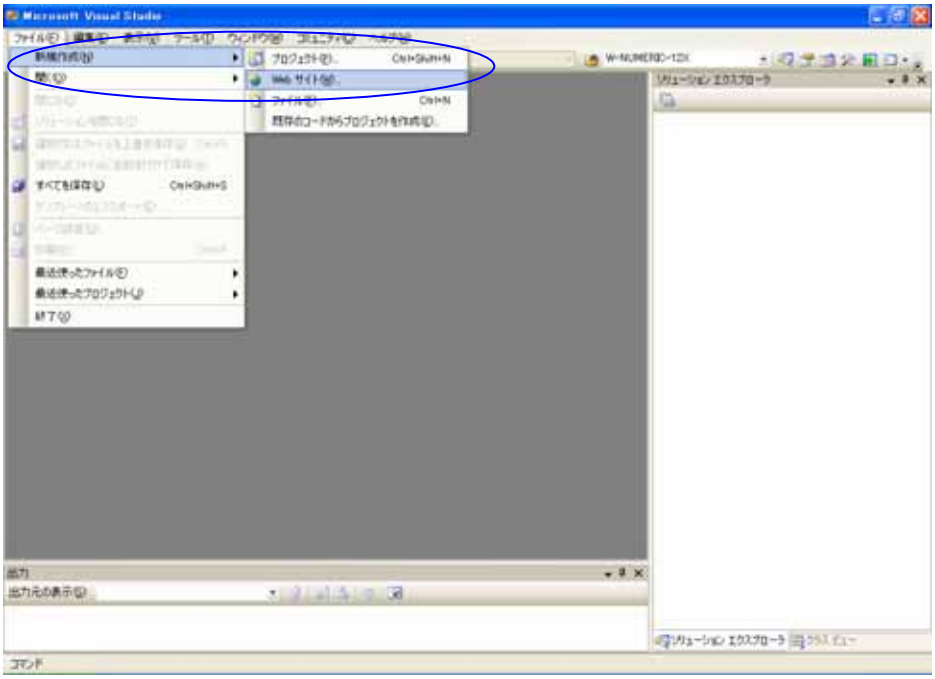
作成するWebフォームアプリケーションの流れについて説明します。まず、ログイン画面で[社員コード]と[パスワード]を入力し、[開始]ボタンを押すと、入力した値を社員テーブル(SYALF)と突合せ、認証が可能な場合は、商品入出庫明細画面に遷移します。年、月、商品コードを入力すると、商品テーブル(SYOPF)から商品コードを元に検索し、見つかった場合は次に入出庫テーブル(SYOIOPF)を検索します。該当する明細があればテーブルに表示されます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.5.2 Webサイトの作成

➢ **手順**1

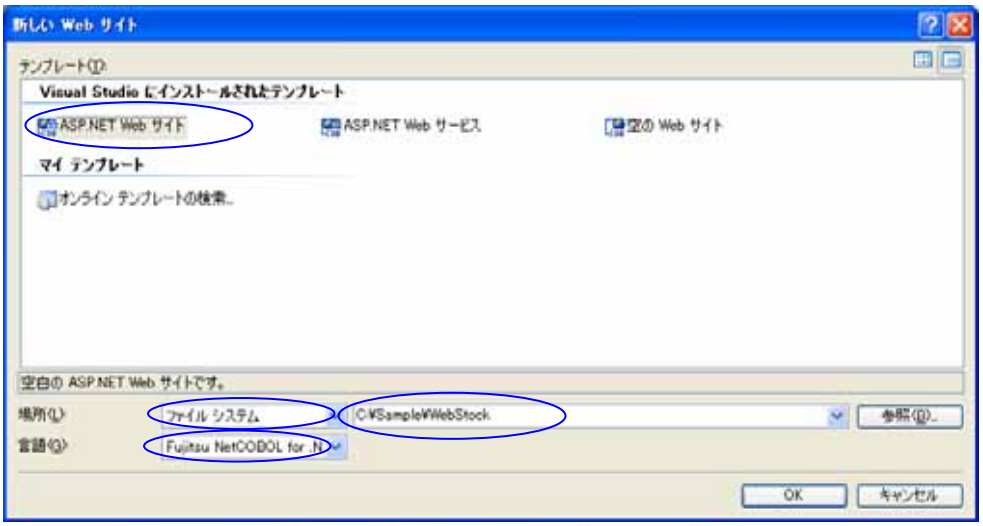
NetCOBOL for .NETを起動し、起動画面の[ファイル]メニューより[新規作成]-[Webサイト]を選択します。



➢ **手順**2

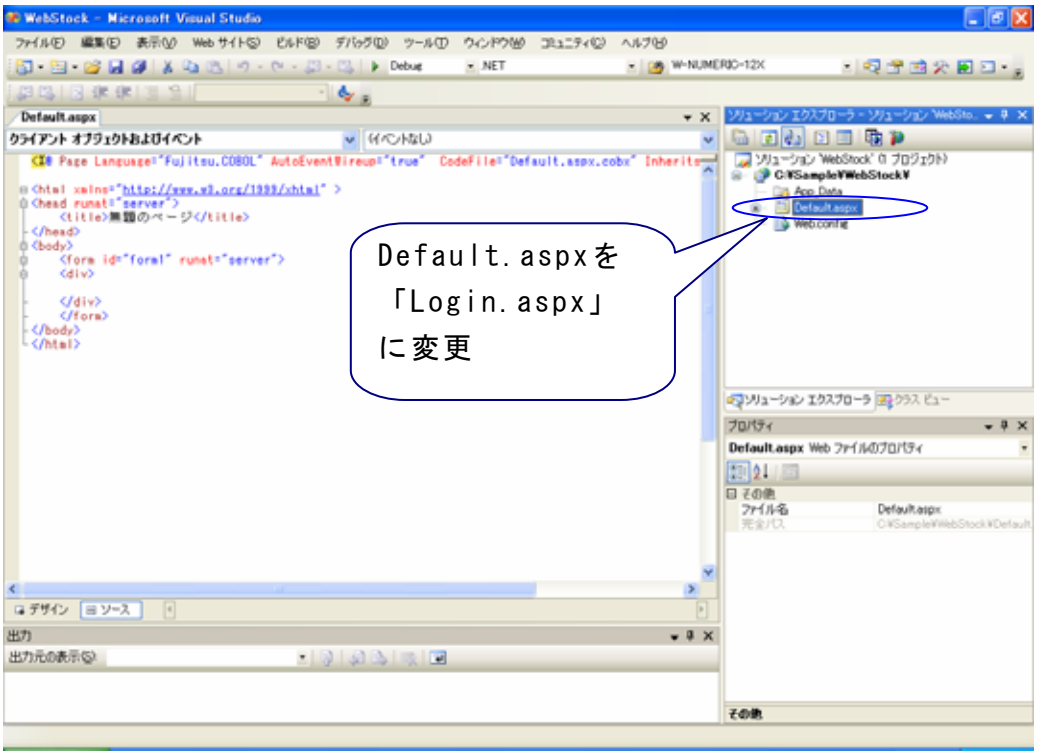
[Visual Studioにインストールされたテンプレート]から「ASP.NET Webサイト」を選択します。[場所]ボックスは「ファイルシステム」を選択し、Webサイトを置くフォルダのパスを設定します（ここでは、C：￥Ｓａｍｐｌｅ￥ＷｅｂＳｔｏｃｋ）。また、[言語]ボックスは「Fujitsu NetCOBOL for .NET」を選択します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



➢ **手順**3

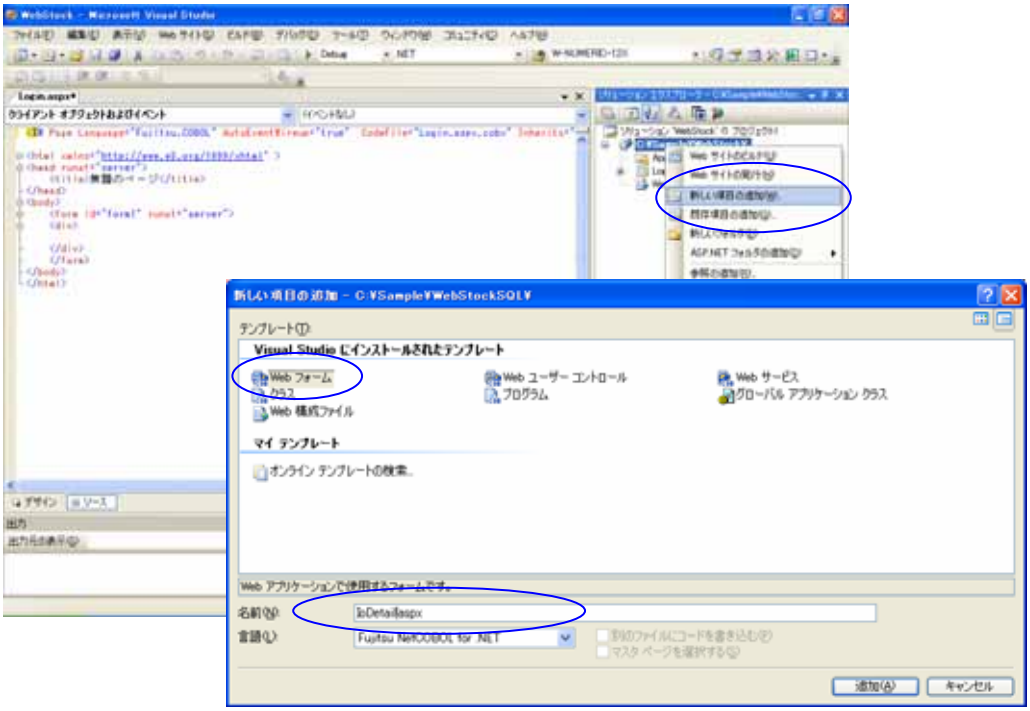
手順2で指定したテンプレートに従って、プロジェクトが自動生成されます(生成直後の画面イメージです)。Webページデザイナウィンドウに未編集のWebフォーム(Default.aspx)が表示されます。ファイル名を「Login.aspx」に変更します。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

➢ **手順**4

今回は「ログイン画面」と「商品入出庫明細照会画面」の2画面必要になるため、Webフォームをプロジェクトに追加します。[ソリューションエクスプローラ]ウィンドウ内のWebサイトプロジェクトを右クリックし、[新しい項目の追加]より、Webフォーム(ファイル名：IoDetail.aspx)を追加します。

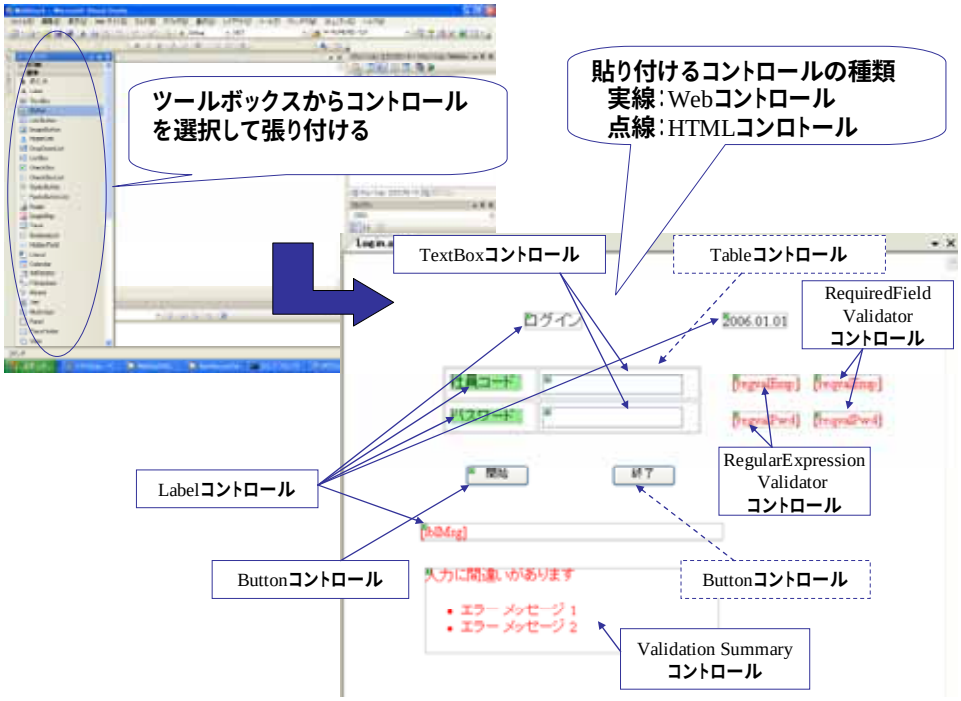


7.5.3 Webフォームの作成

➢ **手順**1

「Login.aspx」にコントロールを配置します。「ツールボックス」からコントロールを選択し、下図のようにレイアウトしてください。

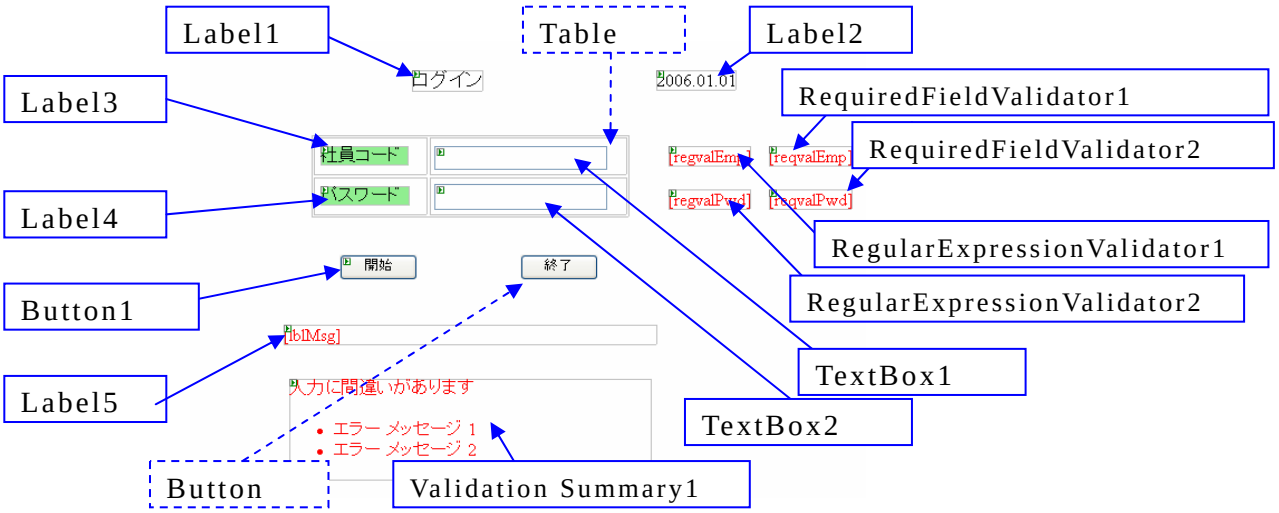
All Rights Reserved, Copyright FUJITSU LIMITED 2006



➢ **手順**2

コントロールの配置が完了したら、各コントロールのプロパティを設定します。プロパティの設定は、コントロールを右クリックしコンテキストメニューの中から「プロパティ」を選択すると表示されるプロパティウィンドウで行ってください。

各コントロールに設定するプロパティは次のとおりです。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

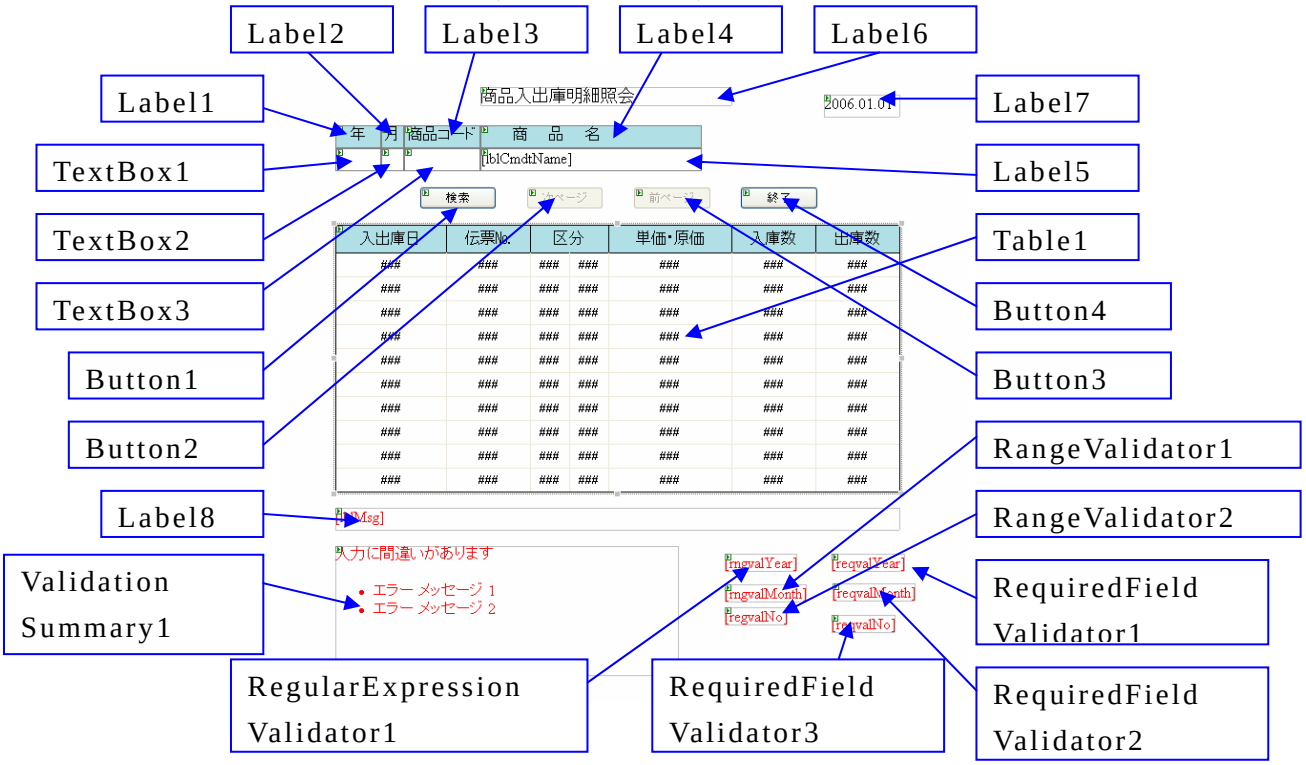
表1 Login.aspxのプロパティ(1)

コントロール	プロパティ	値
DOCUMENT(Webフォーム)	bgcolor	#dcdcdc
Label1	Font(Size)	Larger
	Text	ログイン
Label2	ID	lblDate
	Text	2006.01.01
Label3	BackColor	LightGreen
	Text	社員コード
Label4	BackColor	LightGreen
	Text	パスワード
Label5	ID	lblMsg
	ForeColor	Red
	Text	なし
Table(HTML)	2行2列のTable	
Button1	ID	btnStart
	Text	開始
Button(HTML)	Value	終了
TextBox1	ID	txtEmployeeCode
TextBox2	ID	txtPassword
RegularExpression Validator1	ID	regvalEmp
	ControlToValidate	txtEmployeeCode
	ValidationExpression	¥d{6}
	Display	None
	ErrorMessage	社員コードは数字6桁です
RegularExpression Validator2	ID	regvalPwd
	ControlToValidate	txtPassword
	ValidationExpression	¥w{6}
	Display	None
	ErrorMessage	パスワードは6文字です
RequiredField Validator1	ID	reqvalEmp
	ControlToValidate	txtEmployeeCode
	Display	None
	ErrorMessage	社員コードは必須です
RequiredField Validator2	ID	reqvalPwd
	ControlToValidate	txtPassword
	Display	None
	ErrorMessage	パスワードは必須です
ValidationSummary1	ID	Validationsummary1
	HeaderText	入力に間違いがあります

All Rights Reserved, Copyright FUJITSU LIMITED 2006

➢ **手順**3

次に、「IoDetail.aspx」に下図のようにコントロールを配置し、プロパティの設定を行います(図は完成画面)。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

表2 IoDetail.aspxのプロパティ(1)

コントロール	プロパティ	値
Label1	ID	lblYear
	Text	年
	BackColor	PowderBlue
	BorderColor	Gray
	BorderStyle	Solid
	BorderWidth	1px
	Font(Size)	Small
	CssClass	center
	ForeColor	Black
Label2	ID	lblMonth
	Text	月
	Label1の塗りつぶし範囲と同じ設定	
Label3	ID	lblCmdtNo
	Text	商品コード
	Label1の塗りつぶし範囲と同じ設定	
Label4	ID	lblCmdtTitleName
	Text	商 品 名
	CssClass	left
	Label1の塗りつぶし範囲と同じ設定(CssClassを除く)	
Label5	ID	lblCmdtName
	Text	なし
	BorderStyle	Solid
	BorderWidth	1px
	Font(Size)	Small
	ForeColor	Black
Label6	ID	lblTitle
	Text	商品入出庫明細照会
	Font(Size)	Larger

All Rights Reserved, Copyright FUJITSU LIMITED 2006

表3 IoDetail.aspxのプロパティ(2)

コントロール	プロパティ	値
Label7	ID	lblDate
	Text	2006.01.01
Label8	ID	lblMsg
	ForeColor	Red
	Text	なし
TextBox1	ID	txtYear
	Text	なし
	MaxLength	4
	BorderColor	Gray
	BorderStyle	Solid
	BorderWidth	1px
	CssClass	right
	Font(Size)	Small
	ForeColor	Black
TextBox2	ID	txtMonth
	Text	なし
	MaxLength	2
	TextBox1の塗りつぶし範囲と同じ設定	
TextBox3	ID	txtCmdtNo
	Text	なし
	MaxLength	5
	TextBox1の塗りつぶし範囲と同じ設定	
Button1	ID	btnStart
	Text	開始
Button2	ID	btnFwd
	Enabled	False
	Text	次ページ
Button3	ID	btnBck
	Enabled	False
	Text	前ページ

All Rights Reserved, Copyright FUJITSU LIMITED 2006

表4 IoDetail.aspxのプロパティ(3)

コントロール	プロパティ	値
Button4	ID	btnExit
	CausesValidation	False
	Text	終了
ValidationSummary1	HeaderText	入力に間違いがあります
RangeValidator1	ID	rngvalYear
	ControlToValidate	txtYear
	MaximamValue	9999
	MinimunValue	0000
	Type	Integer
	Display	None
	ErrorMessage	年
RangeValidator2	ID	rngvalMonth
	ControlToValidate	txtMonth
	MaximamValue	12
	MinimunValue	1
	Type	Integer
	Display	None
	ErrorMessage	月
RegularExpression Validator1	ID	regvalNo
	ControlToValidate	txtCmdtNo
	ValidationExpressoin	¥d{5}
	Display	None
	ErrorMessage	商品コード
RequiredField Validator1	ID	reqvalYear
	ControlToValidate	txtYear
	Display	None
	ErrorMessage	年は必須です

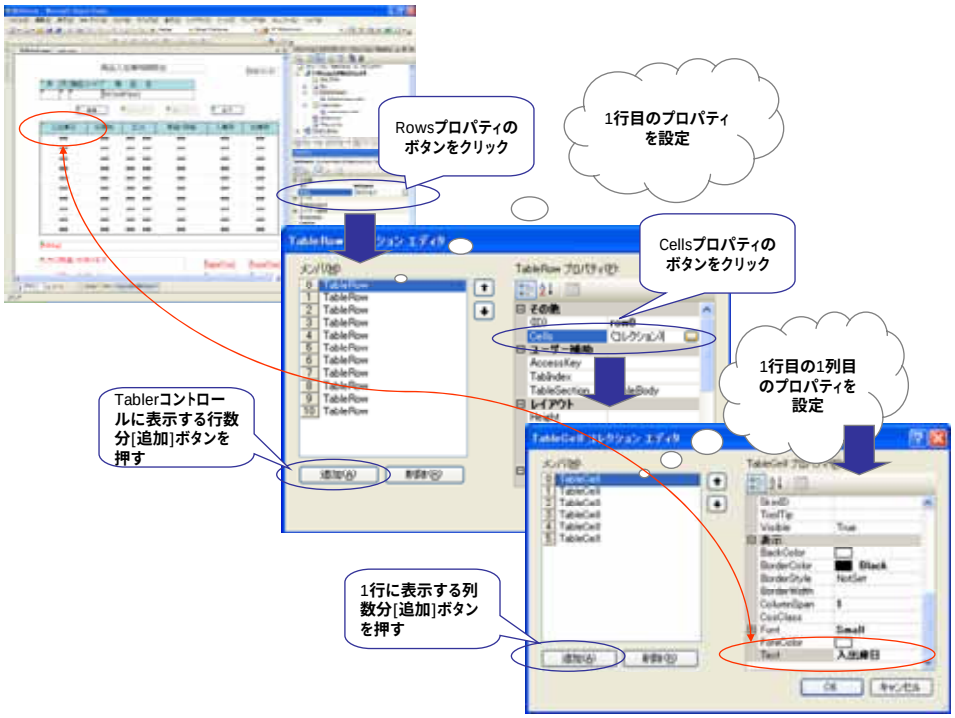
All Rights Reserved, Copyright FUJITSU LIMITED 2006

表5 IoDetail.aspxのプロパティ(4)

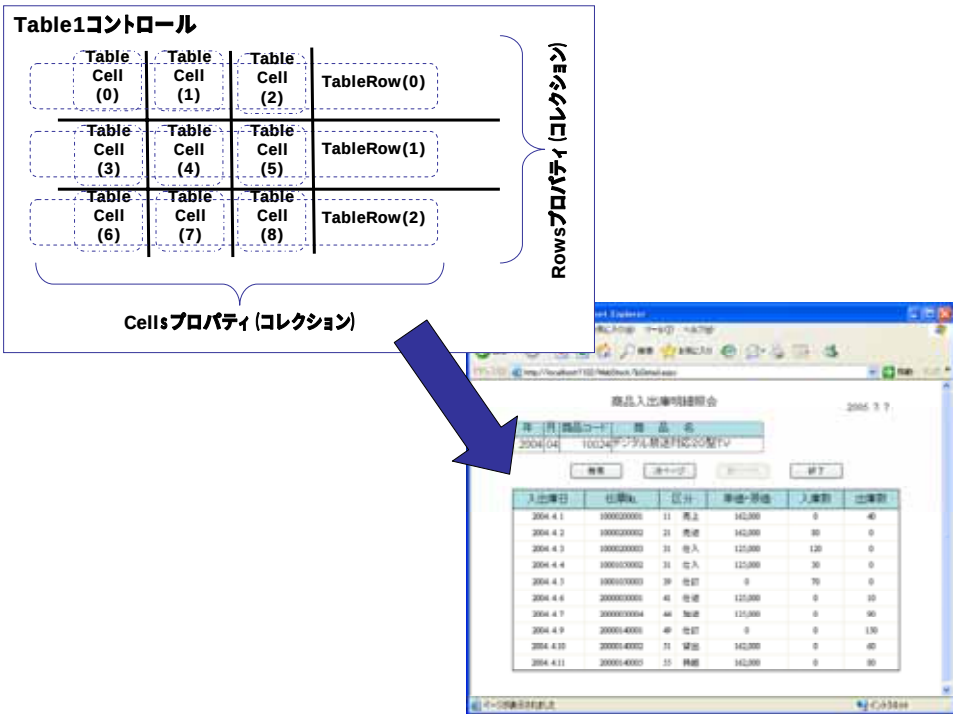
コントロール	プロパティ	値
RequiredField Validator2	ID	reqvalMonth
	ControlToValidate	txtMonth
	Display	None
	ErrorMessage	月は必須です
RequiredField Validator3	ID	reqvalNo
	ControlToValidate	txtCmdtNo
	Display	None
	ErrorMessage	商品コードは必須です
Table1	ID	tblStock
	HorizontalAlign	Center
	BackColor	White
	BorderColor	Black
	BorderStyle	Solid
	BorderWidth	1px
	CellSpacing	0
	Font(Size)	XX-Small
	ForeColor	Black
	GridLines	Both
	Rows	[コレクションエディタ]ダイアログボックスで設定(表 参照

表5のTable1コントロールのRowsプロパティでは、さらに、次の図のようなコレクションエディタを利用して1テーブルコントロールに表示する行(Row)や列(Cell)列名など詳細なテーブルレイアウトを設計します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



上図のコレクションエディタで定義した行(TableRow)や列(TableCell)は下図のようなテーブイルレイアウトをイメージしています。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

表6 IoDetail.aspxのプロパティ(5)

Rows プロパティ	プロパティ	値	Cells プロパティ	プロパティ	値
(0)TableRow	Cells	→	(0)TableCell	Text	入出庫日
	Horizontal Align	Center	(1)TableCell	Text	伝票No.
	Vertical Align	Middle	(2)TableCell	ColumnSpan	2
	BackColor	PowderBlue		Text	区分
	Font(Name)	ＭＳ ゴシック	(3)TableCell	Text	単価・原価
	Font(Size)	Small	(4)TableCell	Text	入庫数
			(5)TableCell	Text	出庫数
(1) TableRow～	Cells		(0)TableCell～ (6)TableCell	設定なし	
	Horizontal Align	Center →			
	VerticalAlign	Middle			
	Font	X-Small			
(2) TableRow～ (10) TableRow	(1)TableRowと同じ設定				

以上でIoDetail.aspxで必要なプロパティの設定は完了です。

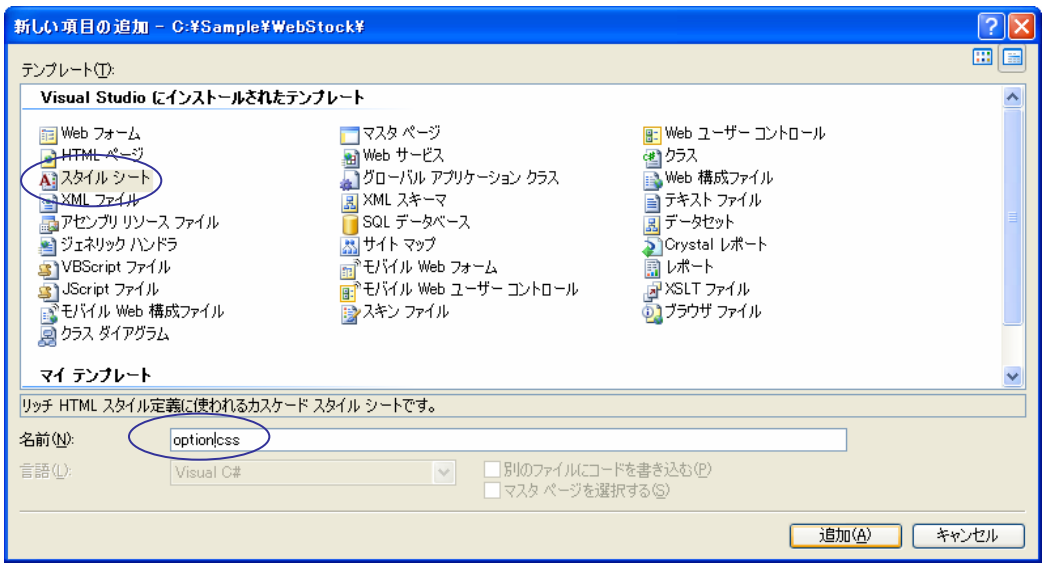
All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.5.4 スタイルシートの作成

7.5.3で設定した各コントロールのプロパティのうち、「CssClass」プロパティではスタイルシートを適用しています。このスタイルシートを作成します。

➢ **手順**1

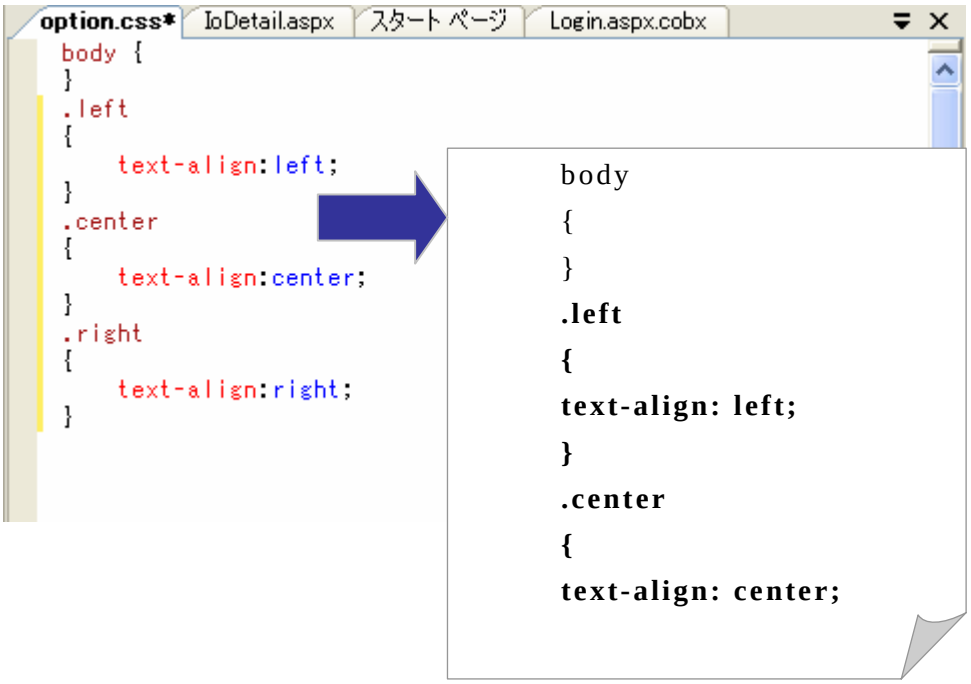
Webサイトプロジェクトを右クリックし[新しい項目の追加]を選択します。[新しい項目の追加]ダイアログボックスで[言語]ボックスから「Visual C#」に切り替え、テンプレートから「スタイルシート」を選択します。名前を「option.css」とします。



➢ **手順**2

スタイルシートの雛形がコードエディタウィンドウに表示されるので次のように編集し保存します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



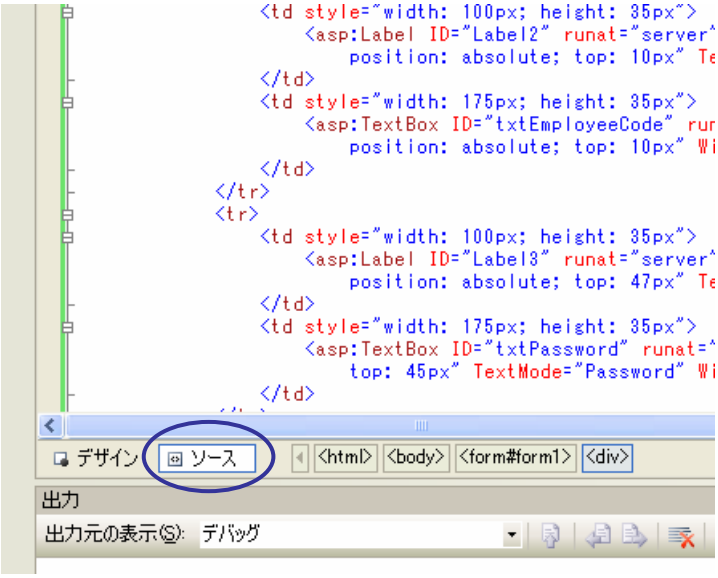
7.5.5 プログラムの作成

➢ **手順**1

次に定義したコントロールに対して処理のコーディングをおこないます。

まず、認証画面(Login.aspx)を開き、「終了」ボタンの処理、および7.5.4で作成したスタイルシートを割り当てる記述をします。Webページデザイナウィンドウの「デザイン」タブを「ソース」タブに切り替え編集を行ってください。なお、スタイルシートを割り当てる記述は商品入出庫画面(IoDetail.aspx)のHTML内にも同様に追加が必要です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



headタグ内にlinkタグを追加します。(Login.aspx、IoDetail.aspx)

```
<head runat="server">
  <link href="option.css" type="text/css" rel="stylesheet" />
.
```

終了ボタンのinputタグを探し、onclick="window.close()"を追加します。

```
<input ...type="button" value="終了"
onclick="window.close()"/>
```

➢ **手順**2

次に、Login.aspx.cobxに定義した各コントロールの処理に必要なコード編集を行います。コードエディタウィンドウを開いてコードを追加してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ REPOSITORY句に次の処理を追加

```
REPOSITORY.
    CLASS CLASS-EVENTARGS AS "System.EventArgs"
    CLASS CLASS-OBJECT AS "System.Object"
    CLASS CLASS-PAGE AS "System.Web.UI.Page"
    CLASS CLASS-SESSION AS
           "System.Web.SessionState.HttpSessionState"
    PROPERTY PROP-TEXT AS "Text"
    PROPERTY PROP-SERVER AS "Server"
    .
```

■ PAGE_LOADイベントに次の処理を追加

```
METHOD-ID. PAGE-LOAD AS "Page_Load".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 W-YYYYMMDD                  PIC X(08).
01 W-Date-R.
   03 W-YYYY                    PIC ZZZ9.
   03 FILLER                    PIC X(01)  VALUE ".".
   03 W-MM                      PIC Z9.
   03 FILLER                    PIC X(01)  VALUE ".".
   03 W-DD                      PIC Z9.
01 W-YMD REDEFINES W-Date-R    PIC X(10).
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    MOVE FUNCTION  CURRENT-DATE  TO W-YYYYMMDD.
    MOVE W-YYYYMMDD(1:4)         TO W-YYYY.
    MOVE W-YYYYMMDD(5:2)         TO W-MM.
    MOVE W-YYYYMMDD(7:2)         TO W-DD.
    SET PROP-TEXT OF lblDate     TO W-YMD.
END METHOD PAGE-LOAD.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnStart_Clickイベントに次の処理を追加(btnStartコントロールをダブルクリックすると雛形のコードが生成されます)

```
METHOD-ID. btnStart_Click PROTECTED.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  WK-EmployeeCode           PIC N(06).
 01  WK-Password               PIC N(06).
 01  WK-Msg                    PIC N(20).
 01  WK-ReturnCode             BINARY-LONG.
 01  W-Websession OBJECT REFERENCE CLASS-SESSION.
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
     SET WK-EmployeeCode TO
         PROP-TEXT OF txtEmployeeCode.
     SET WK-Password TO PROP-TEXT OF txtPassword.

*認証用のSYAINプログラム(ストアドプロシージャ)呼び出し
     CALL "SYAIN2" USING WK-EmployeeCode
                         WK-Password
                         WK-Msg
         RETURNING WK-ReturnCode.
     IF WK-ReturnCode = 0 THEN
       SET W-Websession TO PROP-SESSION OF SELF
*ログインフラグ設定
       INVOKE W-Websession "set_Item"
                    USING BY VALUE N"Login" N"OK"
*紹介画面へ遷移
       INVOKE PROP-SERVER OF SELF "Transfer"
                    USING BY VALUE N"IoDetail.aspx"
     ELSE
       SET PROP-TEXT OF lblMsg TO WK-Msg
     END-IF.
 END METHOD btnStart_Click.
```

以上で、Login.aspx.cobxの編集は完了です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

➢ **手順**3

次に、IoDetail.aspxに定義した各コントロールの処理に必要なコードを追加します。コードエディタウィンドウを開いてコードを追加してください。

■ REPOSITORY句に次のクラスやプロパティを追加

```
REPOSITORY.
     CLASS CLASS-EVENTARGS AS "System.EventArgs"
     CLASS CLASS-OBJECT AS "System.Object"
     CLASS CLASS-PAGE AS "System.Web.UI.Page"
*
     CLASS CLASS-CELL AS
       "System.Web.UI.WebControls.TableCell"
     CLASS CLASS-CELLCOLLECT AS
       "System.Web.UI.WebControls.TableCellCollection"
     CLASS CLASS-ENVIRONMENT AS "System.Environment"
     CLASS CLASS-INT16 AS "System.Int16"
     CLASS CLASS-INT32 AS "System.Int32"
     CLASS CLASS-ROW AS
       "System.Web.UI.WebControls.TableRow"
     CLASS CLASS-SERVERUTILITY AS
       "System.Web.HttpServerUtility"
     CLASS CLASS-SESSION AS
       "System.Web.SessionState.HttpSessionState"
     CLASS CLASS-STRING AS "System.String"
*
     PROPERTY PROP-CELLS AS "Cells"
     PROPERTY PROP-ENABLED AS "Enabled"
     PROPERTY PROP-ID AS "ID"
     PROPERTY PROP-ISPOSTBACK AS "IsPostBack"
     PROPERTY PROP-ROWS AS "Rows"
     PROPERTY PROP-SERVER AS "Server"
     PROPERTY PROP-SESSION AS "Session"
     PROPERTY PROP-TEXT AS "Text"
     .
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ WORKING-STORAGE SECTIONに次の定義を追加

```
WORKING-STORAGE SECTION.

 01 W-Date-R.
   03 W-YYYY        PIC ZZZ9.
   03 FILLER        PIC X(01)  VALUE ".".
   03 W-MM          PIC Z9.
   03 FILLER        PIC X(01)  VALUE ".".
   03 W-DD          PIC Z9.
 01 W-YMD REDEFINES W-Date-R      PIC X(10).
```

■ PAGE_LOADイベントに次の処理を追加

```
METHOD-ID. PAGE-LOAD AS "Page_Load".
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 W-Websession OBJECT REFERENCE CLASS-SESSION.
 01 W-Object OBJECT REFERENCE CLASS-OBJECT.
 01 W-ServerUtility OBJECT REFERENCE CLASS-
SERVERUTILITY.
 01 W-YYYYMMDD                    PIC X(08).
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
*セションから情報取得
     SET W-Websession TO PROP-SESSION OF SELF.
     SET W-Object     TO W-
Websession::"get_Item"(N"Login").
     IF W-Object = NULL THEN
*認証されていない場合は、ログイン画面へ遷移
         SET W-ServerUtility TO PROP-SERVER OF SELF
         INVOKE W-ServerUtility "Transfer" USING
N"Login.aspx"
     END-IF

*ポストバック時以外の処理
     IF PROP-ISPOSTBACK OF SELF = B"0" THEN
*現在の日付をラベルに設定
         MOVE FUNCTION  CURRENT-DATE      TO W-
YYYYMMDD
         MOVE W-YYYYMMDD(1:4)             TO W-YYYY
         MOVE W-YYYYMMDD(5:2)             TO W-MM
         MOVE W-YYYYMMDD(7:2)             TO W-DD
         SET PROP-TEXT OF lblDate         TO W-YMD
*テーブル初期化
         INVOKE SELF "TABLE_CLS"
     END-IF.
 END METHOD PAGE-LOAD.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnStart_Clickイベントに次の処理を追加(btnStartコントロールをダブルクリックすると雛形のコードが生成されます)

```
METHOD-ID. btnStart_Click PROTECTED.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  商品レコード.
     02  商品コード  PIC 9(5).
     02  商品名      PIC N(20).
     02  売価        PIC 9(8).
     02  原価        PIC 9(8).
 COPY IODETAIL.
 01 W-Syocd         OBJECT REFERENCE CLASS-STRING.
 01 W-CmdtNo                  PIC X(05).
 01 W-Year                    PIC X(04).
 01 W-Month                   PIC X(02).
 01 W-Start                   BINARY-SHORT.
 01 W-Msg                     PIC N(20).
 01 W-ReturnCode               BINARY-LONG.
 01 W-Cnt                     PIC 9(03).
 01 W-Next                    PIC 9(01).
 01 W-Websession OBJECT REFERENCE CLASS-SESSION.
 01 W-String OBJECT REFERENCE CLASS-STRING.
 *>01 W-Request OBJECT REFERENCE CLASS-REQUEST.
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
     SET W-Syocd TO PROP-TEXT OF txtCmdtNo.
     SET W-CmdtNo TO PROP-TEXT OF txtCmdtNo.
     MOVE CLASS-INT32::"Parse"(W-Syocd) TO
          商品コード OF 商品レコード.

*商品レコード検索
     CALL "SYOHIN" USING 商品レコード W-Msg
                   RETURNING W-ReturnCode.
*エラー判定(0=正常)
     IF W-ReturnCode = 0 THEN
         SET PROP-TEXT OF lblMsg TO " "

*商品名をラベルに設定
         SET PROP-TEXT OF lblCmdtName TO 商品名

         INITIALIZE M-AREA
         MOVE SPACE TO W-Msg
         MOVE W-CmdtNo TO M-SYOCD
*日付処理
         SET W-Year  TO PROP-TEXT OF txtYear
         SET W-Month TO PROP-TEXT OF txtMonth
         IF  W-Month(2:1) =  SPACE
             MOVE  W-Month(1:1) TO W-Month(2:1)
             MOVE  "0"          TO W-Month(1:1)
         END-IF
         COMPUTE M-Y = FUNCTION NUMVAL(W-Year)
         COMPUTE M-M = FUNCTION NUMVAL(W-Month)
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
        MOVE 売価 TO M-TANKA
        MOVE 原価 TO M-GENKA
        MOVE 1 TO W-Start
        MOVE 0 TO W-Next
*入出庫検索
        CALL "ZAIKO" USING M-AREA W-Start
                           W-Cnt W-Next W-Msg
                       RETURNING W-ReturnCode
        IF W-Msg NOT = SPACE THEN
*メッセージをラベルに設定
            SET PROP-TEXT OF lblMsg TO W-Msg
        END-IF
        IF W-ReturnCode = 0 THEN
            INVOKE SELF "PAGE_EDT" USING M-AREA

            SET W-Websession TO PROP-SESSION OF SELF
*次のレコードがあるかどうか
            IF W-Next = 1 THEN
*ボタンの有効/無効を設定
                SET PROP-ENABLED OF btnFwd TO B"1"
                SET PROP-ENABLED OF btnBck TO B"0"
            ELSE
                SET PROP-ENABLED OF btnFwd TO B"0"
                SET PROP-ENABLED OF btnBck TO B"0"
            END-IF
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"W-Line" 1
*セションに検索条件を保存
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"W-Year"   W-Year
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"W-Month"  W-Month
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"W-CmdtNo" W-CmdtNo
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"M-TANKA"  M-TANKA
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"M-GENKA"  M-GENKA
            SET W-String TO PROP-TEXT OF lblCmdtName
            INVOKE W-Websession "set_Item"
              USING BY VALUE N"lblCmdtName" W-String
        ELSE
            INVOKE SELF "TABLE_CLS"
        END-IF
    ELSE
*商品検索時のエラー処理
        IF W-Msg NOT = SPACE THEN
*エラーメッセージをラベルに設定
            SET PROP-TEXT OF lblMsg TO W-Msg
        END-IF
        INVOKE SELF "TABLE_CLS"
        IF W-ReturnCode = 5 THEN
*商品が見つからない場合は商品名をクリア
            SET PROP-TEXT OF lblCmdtName TO " "
        END-IF
    END-IF
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnFwd_Clickイベントに次の処理を追加(btnFwdコントロールをダブルクリックすると雛形のコードが生成されます)

```
METHOD-ID. btnFwd_Click PROTECTED.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 COPY IODETAIL.
 01 W-Websession OBJECT REFERENCE CLASS-SESSION.
 01 W-Object OBJECT REFERENCE CLASS-OBJECT.
 01 W-Line BINARY-SHORT.
 01 W-Year                    PIC X(04).
 01 W-Month                   PIC X(02).
 01 W-CmdtNo                  PIC X(05).
 01 W-GENKA BINARY-LONG.
 01 W-TANKA BINARY-LONG.
 01 W-Msg                     PIC N(20).
 01 W-ReturnCode              BINARY-LONG.
 01 W-Cnt                     PIC 9(03).
 01 W-Next                    PIC 9(01).
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
*    セションから情報取得
     SET W-Websession TO PROP-SESSION OF SELF.

     SET W-Object TO
         W-Websession::"get_Item"(N"W-Line").
     SET W-Line   TO W-Object AS CLASS-INT16.

     SET W-Object TO
         W-Websession::"get_Item"(N"W-Year").
     SET W-Year   TO W-Object AS CLASS-STRING.

     SET W-Object TO
         W-Websession::"get_Item"(N"W-Month").
     SET W-Month  TO W-Object AS CLASS-STRING.

     SET W-Object TO
         W-Websession::"get_Item"(N"W-CmdtNo").
     SET W-CmdtNo TO W-Object AS CLASS-STRING.

     SET W-Object TO
         W-Websession::"get_Item"(N"M-TANKA").
     SET W-TANKA  TO W-Object AS CLASS-INT32.

     SET W-Object TO
         W-Websession::"get_Item"(N"M-GENKA").
     SET W-GENKA  TO W-Object AS CLASS-INT32.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
      SET W-Object TO
          W-Websession::"get_Item"(N"lblCmdtName").
      SET PROP-TEXT OF lblCmdtName TO
          W-Object AS CLASS-STRING.

      SET PROP-TEXT OF txtCmdtNo TO W-CmdtNo.
      SET PROP-TEXT OF txtYear   TO W-Year.
      SET PROP-TEXT OF txtMonth  TO W-Month.

      SET PROP-TEXT OF lblMsg TO " ".
      INITIALIZE M-AREA.
      MOVE SPACE TO W-Msg.

      MOVE W-CmdtNo    TO M-SYOCD.
      MOVE W-Year      TO M-Y.
      MOVE W-Month     TO M-M.
      MOVE W-TANKA     TO M-TANKA.
      MOVE W-GENKA     TO M-GENKA.
      MOVE 0           TO W-Next.

      ADD 10 TO W-Line.
      CALL "ZAIKO" USING M-AREA W-Line
                         W-Cnt W-Next W-Msg
                 RETURNING W-ReturnCode.
      INVOKE SELF "PAGE_EDT" USING M-AREA.
      IF W-Msg NOT = SPACE THEN
          SET PROP-TEXT OF lblMsg TO W-Msg
      END-IF

      SET W-Websession TO PROP-SESSION OF SELF.
 *次のレコードがあるかどうか
      IF W-Next = 1 THEN
          SET PROP-ENABLED OF btnFwd TO B"1"
      ELSE
          SET PROP-ENABLED OF btnFwd TO B"0"
      END-IF
      SET PROP-ENABLED OF btnBck TO B"1".
      INVOKE W-Websession "set_Item" USING BY VALUE
N"W-Line" W-Line.
  END METHOD btnFwd_Click.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnBck_Clickイベントに次の処理を追加(btnBckコントロールをダブルクリックすると雛形のコードが生成されます)

```
METHOD-ID. btnBck_Click PROTECTED.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY IODETAIL.
01 W-Websession OBJECT REFERENCE CLASS-SESSION.
01 W-Object OBJECT REFERENCE CLASS-OBJECT.
01 W-Line BINARY-SHORT.
01 W-Year                    PIC X(04).
01 W-Month                   PIC X(02).
01 W-CmdtNo                  PIC X(05).
01 W-GENKA BINARY-LONG.
01 W-TANKA BINARY-LONG.
01 W-Msg                     PIC N(20).
01 W-ReturnCode              BINARY-LONG.
01 W-Cnt                     PIC 9(03).
01 W-Next                    PIC 9(01).
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
    SET W-Websession TO PROP-SESSION OF SELF.

    SET W-Object TO
        W-Websession::"get_Item"(N"W-Line").
    SET W-Line   TO W-Object AS CLASS-INT16.

    SET W-Object TO
        W-Websession::"get_Item"(N"W-Year").
    SET W-Year   TO W-Object AS CLASS-STRING.

    SET W-Object TO
        W-Websession::"get_Item"(N"W-Month").
    SET W-Month  TO W-Object AS CLASS-STRING.

    SET W-Object TO
        W-Websession::"get_Item"(N"W-CmdtNo").
    SET W-CmdtNo TO W-Object AS CLASS-STRING.

    SET W-Object TO
        W-Websession::"get_Item"(N"M-TANKA").
    SET W-TANKA  TO W-Object AS CLASS-INT32.

    SET W-Object TO
        W-Websession::"get_Item"(N"M-GENKA").
    SET W-GENKA  TO W-Object AS CLASS-INT32.

    SET W-Object TO
        W-Websession::"get_Item"(N"lblCmdtName").
    SET PROP-TEXT OF lblCmdtName TO
        W-Object AS CLASS-STRING.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
    SET PROP-TEXT OF txtCmdtNo TO W-CmdtNo.
    SET PROP-TEXT OF txtYear   TO W-Year.
    SET PROP-TEXT OF txtMonth  TO W-Month.

    SET PROP-TEXT OF lblMsg TO " ".
    INITIALIZE M-AREA.
    MOVE SPACE TO W-Msg.

    MOVE W-CmdtNo    TO M-SYOCD.
    COMPUTE M-Y = FUNCTION NUMVAL(W-Year).
    COMPUTE M-M = FUNCTION NUMVAL(W-Month).
    MOVE W-TANKA     TO M-TANKA.
    MOVE W-GENKA     TO M-GENKA.
    MOVE 0           TO W-Next.

    SUBTRACT 10 FROM W-Line.
    CALL "ZAIKO" USING M-AREA W-Line
                       W-Cnt W-Next W-Msg
                 RETURNING W-ReturnCode.
    INVOKE SELF "PAGE_EDT" USING M-AREA.
    IF W-Msg NOT = SPACE THEN
        SET PROP-TEXT OF lblMsg TO W-Msg
    END-IF

    SET W-Websession TO PROP-SESSION OF SELF.

    IF W-Next = 1 THEN
        SET PROP-ENABLED OF btnFwd TO B"1"
    ELSE
        SET PROP-ENABLED OF btnFwd TO B"0"
    END-IF

    IF W-Line < 11 THEN
        SET PROP-ENABLED OF btnBck TO B"0"
    ELSE
        SET PROP-ENABLED OF btnBck TO B"1"
    END-IF
    INVOKE W-Websession "set_Item"
           USING BY VALUE N"W-Line" W-Line.
END METHOD btnBck_Click.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ btnExit_Clickイベントの処理を追加(btnExitコントロールをダブルクリックすると雛形のコードが生成されます)

```
METHOD-ID. btnExit_Click PROTECTED.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 W-ServerUtility OBJECT REFERENCE
                    CLASS-SERVERUTILITY.
 LINKAGE SECTION.
 01 sender OBJECT REFERENCE CLASS-OBJECT.
 01 e OBJECT REFERENCE CLASS-EVENTARGS.
 PROCEDURE DIVISION USING BY VALUE sender e.
*ログイン画面へ遷移
     SET W-ServerUtility TO PROP-SERVER OF SELF.
     INVOKE W-ServerUtility "Transfer"
            USING N"Login.aspx".
 END METHOD btnExit_Click.
```

上記のコード以外に、独自定義のTABLE_CLSメソッド(Tableコントロールに空白を設定する)、およびPAGE_EDITメソッド(Tableコントロールに空白を設定する)を新規にコーディングします。

■ TABLE_CLSメソッド

```
METHOD-ID. TABLE_CLS.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 W-Row OBJECT REFERENCE CLASS-ROW.
 01 W-Cell OBJECT REFERENCE CLASS-CELL.
 01 W-Id PIC X(10).
 LINKAGE SECTION.
 PROCEDURE DIVISION.
*row0以外の行を初期化
     PERFORM VARYING W-Row THRU PROP-ROWS OF tblStock
       SET W-Id TO PROP-ID OF W-Row
       IF W-Id NOT = "row0" THEN
         PERFORM VARYING W-Cell THRU PROP-CELLS OF W-
Row
           SET PROP-TEXT OF W-Cell TO N"  "
         END-PERFORM
       END-IF
     END-PERFORM.
 END METHOD TABLE_CLS.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ PAGE_EDTメソッド

```
 METHOD-ID. PAGE_EDT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 W-Row OBJECT REFERENCE CLASS-ROW.
 01 W-Cell OBJECT REFERENCE CLASS-CELL.
 01 W-Id PIC X(10).
 01 W-Idx BINARY-LONG.
 01 W-CellCollect OBJECT REFERENCE CLASS-CELLCOLLECT.
 01 W-Numeric-ZS.
   03 W-Numeric-2ZS                PIC Z9.
   03 W-Numeric-3ZS                PIC ZZ9.
   03 W-Numeric-6ZS                PIC ZZZZZ9.
   03 W-Numeric-8ZS                PIC ZZ,ZZZ,ZZ9.
   03 W-Numeric-12ZS               PIC ZZZZZZZZZZZ9.
   03 W-Numeric-12X                PIC X(12).
 LINKAGE SECTION.
 COPY IODETAIL.
 PROCEDURE DIVISION USING M-AREA.
*row0以外の行設定
     MOVE 0 TO W-Idx.
     PERFORM VARYING W-Row THRU PROP-ROWS OF tblStock
       SET W-Id TO PROP-ID OF W-Row
       IF W-Id NOT = "row0" THEN
         IF M-IODATE(W-Idx) = 0 THEN
           PERFORM VARYING W-Cell
                   THRU PROP-CELLS OF W-Row
             SET PROP-TEXT OF W-Cell TO N"  "
           END-PERFORM
         ELSE
             SET W-CellCollect TO PROP-CELLS OF W-Row
             SET W-Cell TO
                 W-CellCollect::"get_Item" (0)
             MOVE M-IODATE(W-Idx)(1:4) TO W-YYYY
             MOVE M-IODATE(W-Idx)(5:2) TO W-MM
             MOVE M-IODATE(W-Idx)(7:2) TO W-DD
             SET PROP-TEXT OF W-Cell   TO W-YMD

             SET W-Cell TO
                 W-CellCollect::"get_Item" (1)
             MOVE M-DENBAN(W-Idx) TO W-Numeric-12ZS
             SET PROP-TEXT OF W-Cell TO W-Numeric-12ZS

             SET W-Cell TO
                 W-CellCollect::"get_Item" (2)
             MOVE M-KUBUN(W-Idx) TO W-Numeric-2ZS
             SET PROP-TEXT OF W-Cell TO W-Numeric-2ZS

             SET W-Cell TO
                 W-CellCollect::"get_Item" (3)
             SET PROP-TEXT OF W-Cell TO
                 M-KUBUNN(W-Idx)
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
                SET W-Cell TO
                     W-CellCollect::"get_Item" (4)
              MOVE M-KAKAKU(W-Idx) TO W-Numeric-8ZS
              SET PROP-TEXT OF W-Cell TO W-Numeric-8ZS

              SET W-Cell TO
                  W-CellCollect::"get_Item" (5)
              MOVE M-ISU(W-Idx) TO W-Numeric-6ZS
              SET PROP-TEXT OF W-Cell  TO W-Numeric-6ZS

              SET W-Cell TO
                  W-CellCollect::"get_Item" (6)
              MOVE M-OSU(W-Idx) TO W-Numeric-6ZS
              SET  PROP-TEXT OF W-Cell TO W-Numeric-6ZS
          END-IF
        END-IF
        ADD 1 TO W-Idx
      END-PERFORM.
  END METHOD PAGE_EDT.
```

以上で、Login.aspx.cobx、IoDetail.aspx.cobxのコード編集は完了です。

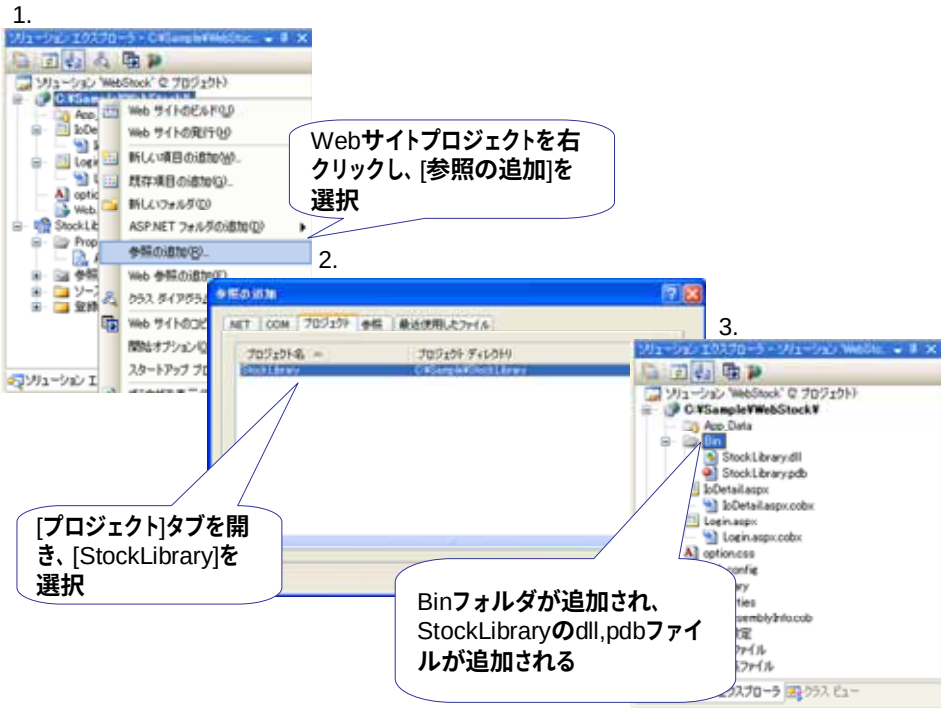
➢ **手順**4

次に、手順2、手順3のイベント処理で呼出している、第5章「5.5 データアクセスプログラムの作成」で作成したクラスライブラリ(StockLibrary)を「WebStock」ソリューションに追加します。
[ソリューションエクスプローラ]より[WebStockソリューション]を右クリックし[追加]-[既存のプロジェクト]で[StockLibrary.cobproj]を選択してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

➢ **手順**5

次に、[WebStock]から呼び出すサブルーチンをまとめたプロジェクト(StockLibrary)を呼び出し元の[参照設定]に追加します([参照設定]の追加方法は下図の手順に従ってください。



また、[StockLibrary]に含まれる登録集のうち、IODETAIL.CBLは[WebStock]でも使用します。ソリューションエクスプローラより[WebStock]のアプリケーション構成ファイル(Web.config)を開きます。<compilers>タグのcompilerOptionsに翻訳オプション/copypathを追加し、IODETAIL.CBLの絶対パス（”～～”の部分）を指定します。

```
<system.codedom>
 :
<compilers>
 <compiler …
 compilerOptions="/copypath: ～～¥StockLibrary
 /wc:SCS(ACP)"/>
</compilers >
</system.codedom>
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

7.5.6 ビルド及びアプリケーション構成ファイルの作成

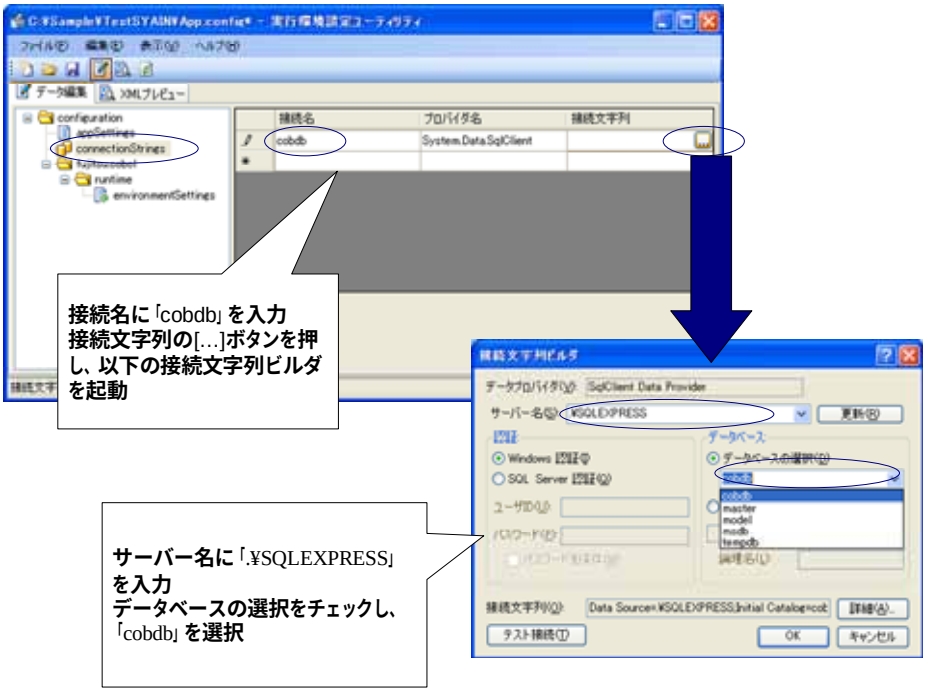
プログラムや登録集の作成が完了したら、ビルドを行ってください。ビルドが正常に終了したら、次に、[ツール]メニューより[Net COBOL for .NETユーティリティ]-[COBOL実行環境設定ユーティリティ]を選択します。このユーティリティを使用してデータベース接続に必要な設定をGUI形式で指定し、アプリケーション構成ファイル(Web.config)に反映させます。

- **手順**1

  [ファイル]メニューより[開く]を選択し、[WebStock]プロジェクトが格納されているフォルダにあるWeb.configファイルを開きます。

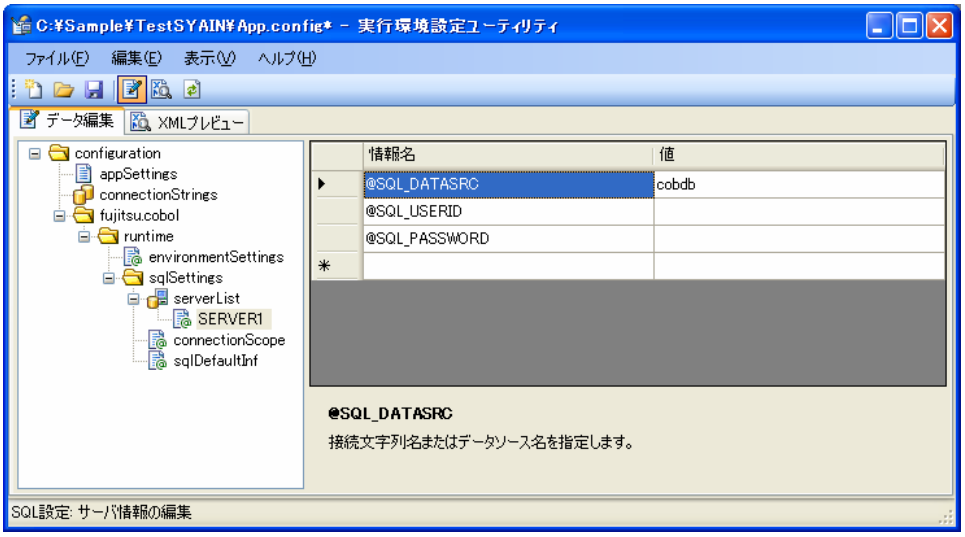
- **手順**2

  データベース接続文字列に該当する「connectionStrings」を変更します。デフォルトで挿入されている「LocalSqlServer」は、接続名を右クリックし[設定の削除]で削除しておきます。次に以下の設定を行います。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

次に、[編集]メニューより[SQL設定の追加]をクリックすると雛形が表示されますので、情報名[@SQL_DATASRC]の値を「cobdb」とします。



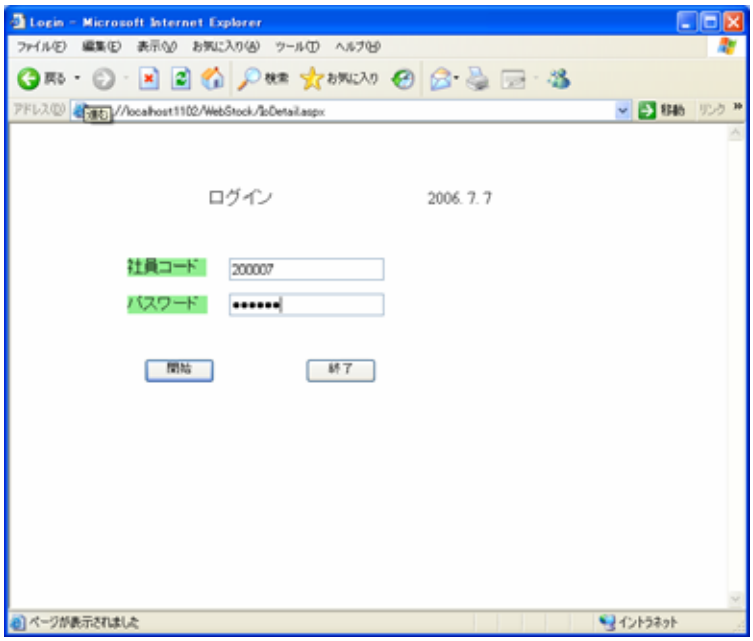
上記の編集が完了したら[ファイル]メニューより[保存]を選択後、[終了]をクリックし、ダイアログボックスを閉じてください。

以上で、Webアプリケーション(WebStock)を実行するために必要なファイルやプログラムは完了です。

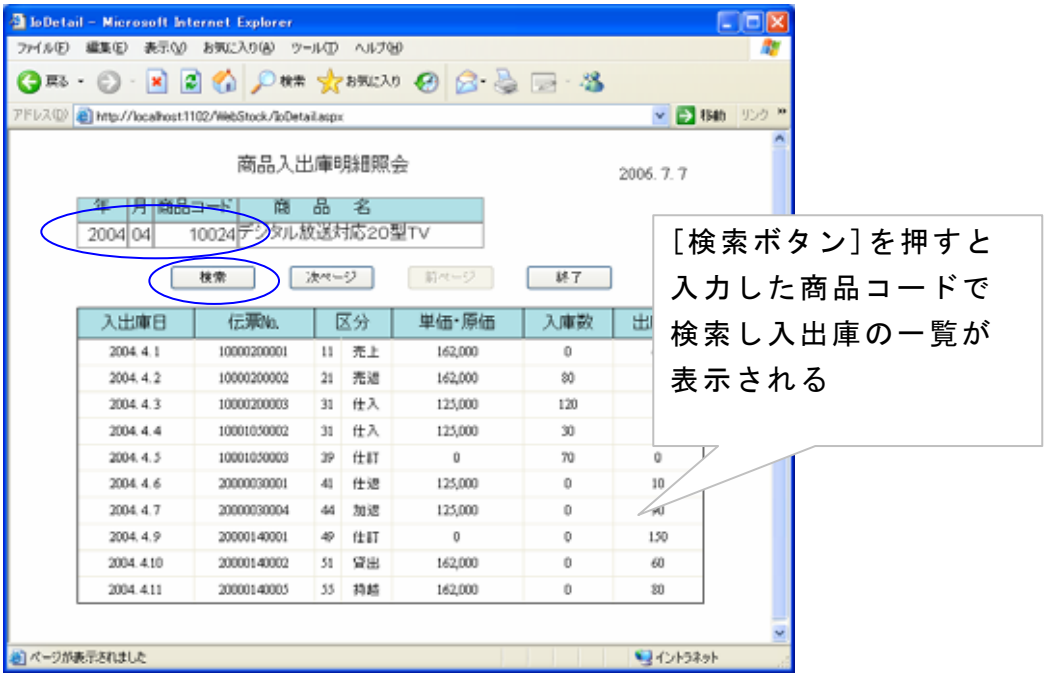
7.5.7 Webアプリケーションの動作確認

Webアプリケーション(WebStock)を実行すると、ログイン画面が表示されます。社員コード、パスワードにはそれぞれ200007と入力し開始ボタンを押し、認証できれば正常に動作しています。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



認証に成功し「商品入出庫明細照会」画面が表示されたら「年 ： 2004　月 ： 04　商品コード ： 10024」と入力して検索ボタンを押します。



以上が、[WebStock]アプリケーションの動作確認手順です。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

付録A 定義体を使用した印刷

A.1 帳票定義体を利用した印刷処理の概要
A.2 帳票定義体を利用した印刷の手順

All Rights Reserved, Copyright FUJITSU LIMITED 2006

A.1 帳票定義体を使用した印刷処理の概要

A.1.1 帳票定義体の設計と運用

帳票定義体の作成と運用には下図のようなFORMとMeFtが必要になります。MeFtを使用せずに帳票定義体の印刷を行うことはできません。

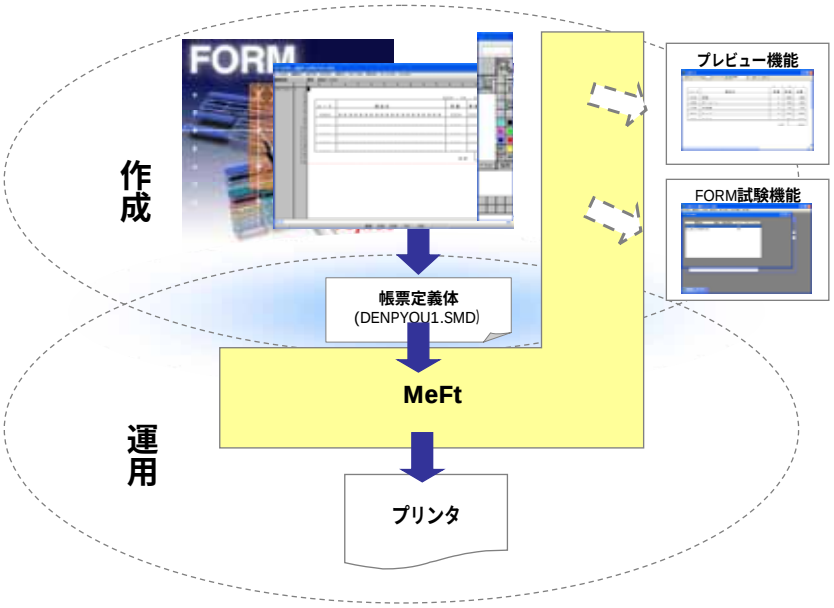


図 A.1.1 帳票定義体の設計と運用

FORMとMeFtの役割

FORMは帳票定義体を作成するためのGUIツールであり、MeFtは帳票定義体の運用時に必要なランタイムの製品です。

FORMには次の特長があります。

- ✓ 罫線、網掛け、出力項目の定義など帳票設計に必要な機能が提供されおり、容易に帳票定義体の作成、更新が可能
- ✓ 汎用機やオフコンの帳票定義体もWindowsに転送後、FORMで編集することが可能

また、設計した帳票について次の保守機能も提供されています(使用

All Rights Reserved, Copyright FUJITSU LIMITED 2006

時はMeFtが必要です)。

✓ プレビュー機能(印刷イメージを事前に確認できる)

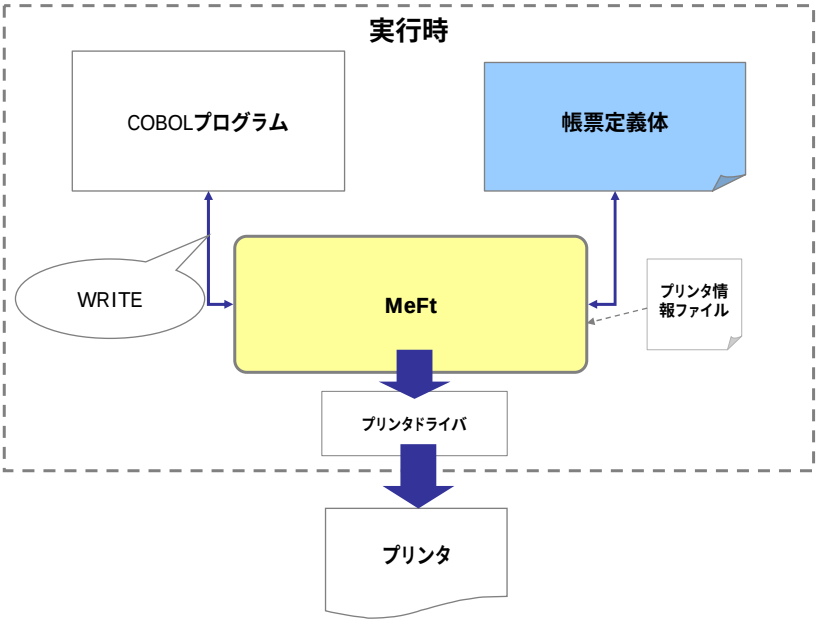
✓ FORM試験機能(アプリケーションを使用せずに帳票定義体を試験的に出力)

なお、FORMにはPowerFORMも含まれており、FORMで作成する帳票定義体より、更に細やかな帳票設計を可能にしています。

また、帳票定義体を利用することで、プログラム側で出力データの位置、罫線などの定義情報を指定する必要がありません。また、プログラムで指定するのは出力項目のみでよいため、帳票定義体を使用するプログラムを容易に開発することが可能です。

A.1.2 COBOLプログラムからの利用イメージ

下図は、帳票定義体をCOBOLプログラムから利用して出力する場合のイメージ図です。



図A.1.2 COBOLプログラムの利用イメージ

COBOLプログラムでは、帳票定義体に定義しているデータ項目をプログラムにレコード形式で取り込み、そのレコード形式のデータ項目に実際のデータを設定して出力することにより、帳票印刷を実現していま

All Rights Reserved, Copyright FUJITSU LIMITED 2006

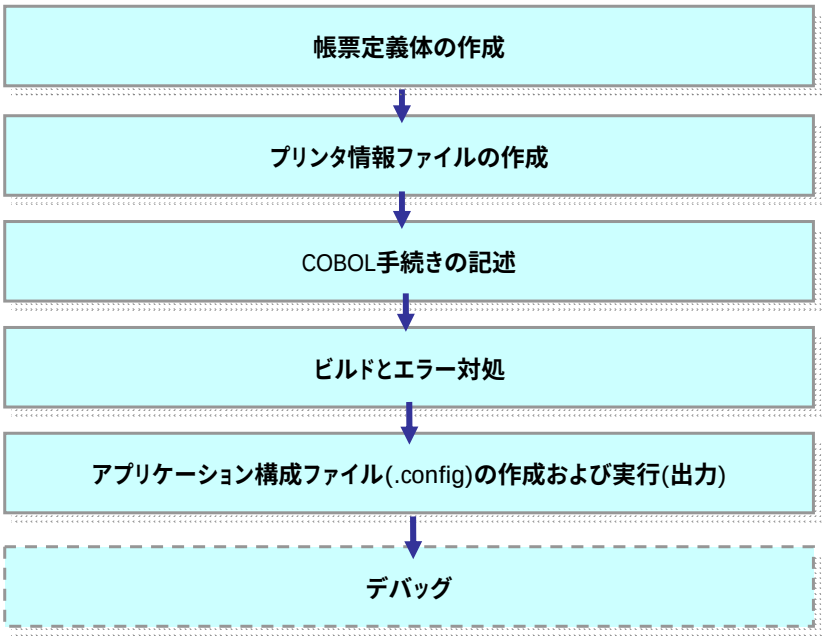
す。帳票定義体の印刷ではMeFtが必須となっていますが、MeFtの機能を使用することにより、プログラム側で、プリンタ固有の制御コードを考慮する必要が全くありません。印刷に必要な情報はMeFtが参照するプリンタ情報ファイルに指定し、COBOLプログラムでは、通常の印刷ファイルを操作するようにWRITE文を実行することで容易に帳票定義体の出力を行うことができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

A.2 帳票定義体を使用した印刷の手順

A.2.1 帳票定義体を使用した印刷処理の流れ

帳票定義体を利用した印刷処理の流れについて説明します。



図A.2.1 帳票定義体を使用した印刷処理の流れ

帳票定義体の作成

FORM(PowerFORM)で帳票定義体を作成します。罫線や網がけ、出力項目、項目群や基本レコードの生成などの定義を行います。

プリンタ情報ファイルの作成

テキストエディタまたは対話形式の「プリンタ情報ファイル設定ツール」を利用してプリンタの操作方法を定義する実行環境ファイルです。

COBOL手続きの記述

プログラムでは、用途に応じてFORMAT句つき印刷ファイルまたは表示ファイルとして定義し、印刷対象の帳票定義体名など必要な情報の転記をおこなったあと、帳票出力にはWRITE文を使用します。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

ビルドとエラー対処

プロジェクトにCOBOLプログラムやクラスを追加するなどして、COBOL言語によるプログラム手続きを記述します。画面のコントロール(部品)に依存する手続きは、画面に貼り付けられたコントロールをダブルクリックし、目的の処理をCOBOL言語で記述します。

アプリケーション構成ファイル(.config)の作成および実行(出力)

プログラムはソリューション単位やプロジェクト単位でビルドすることができます。通常は、ファイルごとにビルドを行い、文法エラーへの対処を行います。また、文法エラーは「タスク一覧」ウィンドウに表示され、エラー表示行をクリックすると、プログラムのエラー箇所にジャンプします。

デバッグ

論理エラーの解消には、デバッグモードで翻訳し、ステップ実行などを行いながらエラーを解消します。統合開発環境に用意されている.NET Framework共通のデバッグ機能を利用することができます。

A.2.2 帳票定義体の作成手順

「FORM」を使用した帳票定義体の作成手順の流れについて定義箇所を交えて説明します。まずはFORMを起動し、「ファイル」メニューから[新規作成]-[帳票定義体]を選択し、帳票定義体の編集画面を表示します(「PowerFORM」とは定義箇所や項目など操作方法が異なります)。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

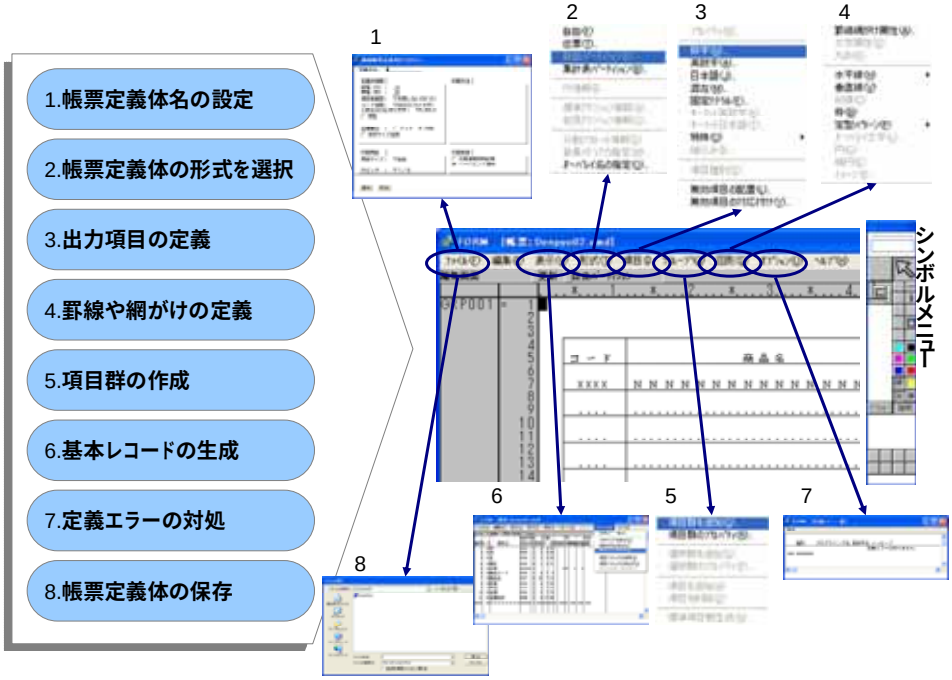


図 A.2.2 帳票定義体の作成の流れ

(1) 帳票定義体名の設定

「ファイル」メニューより[プロパティ]-[画面帳票定義体]で表示される「画面帳票定義体のプロパティ」ダイアログボックス内で定義体名及び、定義体の行桁数や用紙方向、サイズなどの基本情報を設定します。なお、ここで指定した定義体名(日本語のみ、または英数字のみで定義。混在不可)は、プログラムで使用するレコード名となります。なお、帳票定義体名は省略も可能です。

(2) 定義体形式の設定

「形式」メニューより定義体の形式を次の中から選択します。

自由

自由に項目を配置できる形式で、帳票定義体で定義した項目の配置どおりに出力されます。

伝票

伝票をヘッド、ボディ、テイルの3つの領域に分けて設定する形式で、ボディ領域は行単位で下方向に繰り返し数を定義できるため、同じ形

All Rights Reserved, Copyright FUJITSU LIMITED 2006

式のデータを複数出力したい場合に便利です。なお、帳票定義体で定義した項目の配置どおりに出力されます。

自由パーティション

印刷位置は行位置が固定の固定パーティションとプログラムから出力された順番で決まる浮動パーティションで構成される形式で、印刷結果はプログラムからの指定によって異なります。

集計表パーティション

集計表や一覧表の印刷に適した形式で、複数のパーティション(ページ頭書き、制御頭書き、明細、制御脚書き、ページ脚書き)の属性に従い印刷位置はプログラム側から指定する必要があります。

※図の帳票定義体は自由パーティション形式です。

(3) 出力項目の定義

「項目」メニューより、数字項目、英数字項目、日本語項目、混在項目、固定リテラル、組み込みメディア項目から選択し、任意の位置に定義します。特殊項目としてバーコードやID数字、ID英数字、郵便番号等の定義も可能です。項目の桁数や属性(編集形式、書体、色)なども合わせて定義します。

(4) 罫線や網がけの定義

シンボルメニューまたは[図形]メニューで垂直線や平行線、枠や網がけを編集画面内に定義します。

(5) 項目群の作成

複数の項目を一度に出力処理するときに利用するグループ設定です。

項目群の生成は次の手順で行います(集計表パーティション形式の場合は自動的に項目群が設定されています)

1. 「グループ」メニューより[項目群追加]を選択後、項目群名および英数字項目群名を定義(自由パーティションの場合、項目群の種別(固定または浮動)や行数等の指定が必要)
2. 1で作成した項目群に含める項目を編集画面から選択後、右クリックし[項目を追加]をクリック

All Rights Reserved, Copyright FUJITSU LIMITED 2006

3. 必要に応じて2の作業を繰り返し、項目の追加が全て終了したら「表示」メニューより[構成項目表示を解除]を選択

自由形式または伝票形式の場合は、「グループ」メニューより[標準項目群生成]で選択すると表示されるダイアログボックス内で、あらかじめ定義された項目群から生成したい項目群をチェックすることも可能です。なお、組み込みメディア項目は項目群に含めることができません。また、自由形式において、標準項目群生成を実行後に追加した項目は、標準項目群には含まれないため、標準項目群生成の再設定が必要です。

(6) 基本レコードの生成

編集画面で定義した項目を縦横位置の順に基本項目として自動的にレコード定義を行います。「表示」メニューより[レコード定義へ]を選択し表示されたウィンドウの「オプション」メニューより[基本レコードの生成]を選択すると、レコードが自動生成されます。

なお、基本レコード生成を実行後に追加した項目は、レコードに含まれないため、[基本レコード]の生成の再設定が必要です。

(7) 定義エラーの対処

6までの手順で必要な定義は終了しましたが、定義の中で正しくないものについては定義エラーとなり、そのままでは、帳票定義体として保存することができません。定義エラーの内容は編集画面より「オプション」メニューの[定義エラー一覧]で表示されるウィンドウで確認します。「定義エラーはありません」のメッセージがウィンドウ内に出力されるまで対処が必要です。

(8) 帳票定義体の保存

作成した帳票定義体を保存します。「ファイル」メニューより

[名前を付けて保存]-[画面帳票定義体]を選択すると表示される「名前を付けて保存」ダイアログボックスで拡張子(.SMD)の形式で保存します。ファイル名は(1)で指定した帳票定義体名と同じにしておくとプログラムで使いやすくなります。関連性についてはA.2.4で述べます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

A.2.3 プリンタ情報ファイルの作成

帳票定義体の作成が終わったら、プリンタ情報ファイルの作成を行います。プリンタ情報ファイルとは、作成した定義体を実際に出力する際に必要な情報(プリンタドライバの指定、用紙サイズ、給紙口、フォント等)を指定したテキスト形式のファイルです。

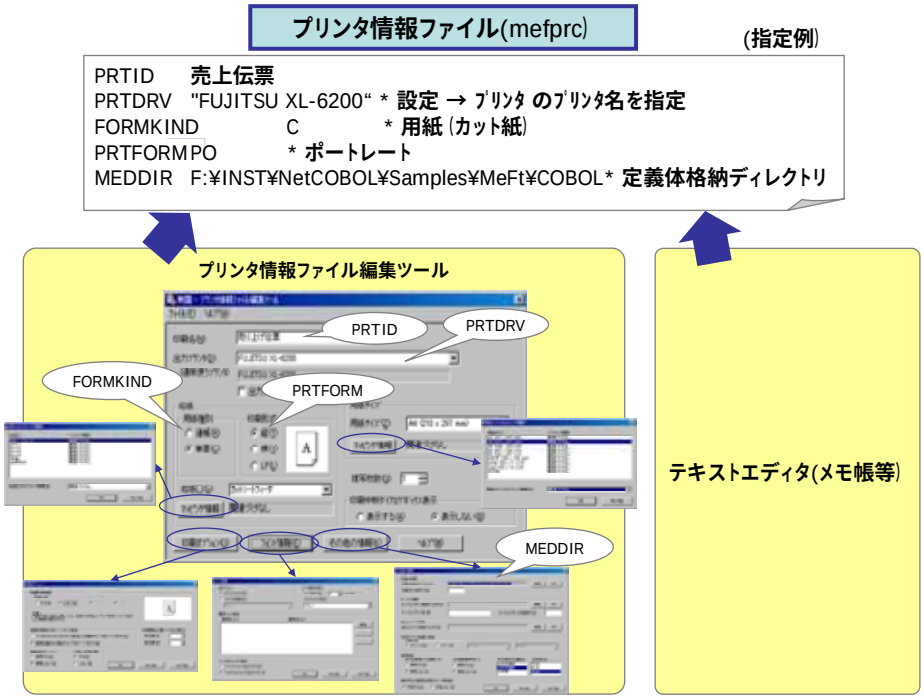


図 A.2.3 プリンタ情報ファイルの指定例

上図は、プリンタ情報ファイルの指定例です。作成方法としてテキストエディタで指定する方法と製品に含まれる「プリンタ情報ファイル編集ツール」で指定する2通りがあります。プリンタ情報ファイルは「キーワード 値」の形式で指定し、コメント行はアスタリスク(*)で表します。定義体格納ディレクトリやフォント名、用紙名、プリンタドライバ名などはスペースや半角/全角に至るまで値を正確に指定しなければ有効にならないため、手動による編集より「プリンタ情報ファイル編集ツール」の使用をお勧めします。なお、指定可能なキーワードおよび値についてはMeFtのマニュアルを参照してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

A.2.4 COBOL手続きの記述

帳票定義体、プリンタ情報ファイルの作成後は、実際に帳票定義体の出力を行うためのCOBOLプログラムを作成します。帳票定義体の出力を行うためのプログラムインタフェースとして、「表示ファイル」と「FORMAT句付き印刷ファイル」が提供されており、帳票定義体の形式によって使用可能なプログラムインタフェースが異なります。

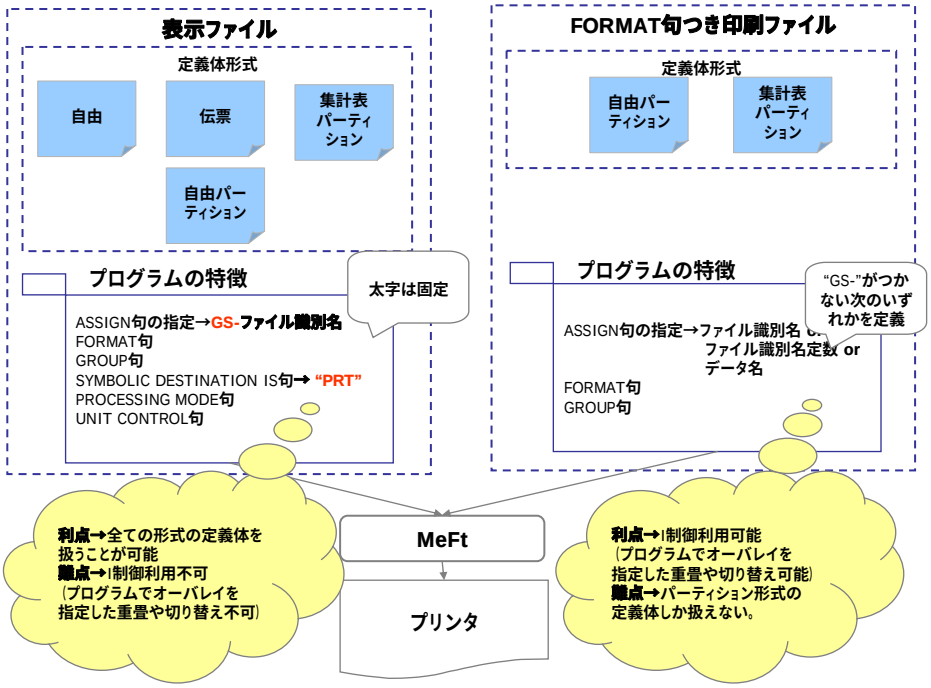


図 A.2.4 定義体形式とプログラムインタフェースの関係

図のように、「表示ファイル」のプログラム記述で代表的な特徴としては、ファイルの割り当てを行うASSIGN句の指定で[GS-ファイル識別名]と"GS-"を付加する点です。FORMTAT句付き印刷ファイルは通常の印刷ファイルの割り当て記述と変わりません。「表示ファイル」では、全ての帳票定義体形式を扱うことができますが、オーバレイ定義体の重畳や切り替えをプログラムで行うI制御レコードを使用することができません。このため、オーバレイを帳票定義体に重畳する場合は帳票定義体側の編集時に指定しておきます。また、「FORMAT句つき印刷ファイル」ではI制御レコードの使用が可能で、プログラムからオーバレイ定義体の重畳や切り替えを行うことができますが、パーティション形式以外の帳票定義体を使用することはできません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

次に「表示ファイル」を使用した帳票印刷プログラムのコーディング例です。このプログラムでは、行順ファイルのデータを読み込み、帳票定義体の項目に転記し、WRITE文を実行し帳票を出力しています。例のプログラムはNetCOBOL for .NETのコンソールアプリケーションとして作成しています。

プログラムでは、帳票定義体のプロパティで指定した帳票定義体名をレコード名として使用する以外に帳票定義体のファイル名を参照する箇所など、異なる名前にした場合、使用箇所を確実に把握する必要があります。特別な理由がない限り同一の名前にしておくとよいでしょう。なお、帳票定義体のプロパティに帳票定義体名を指定しない場合は、プログラム側で参照する帳票定義体は全て帳票定義体のファイル名が使用されます。例では帳票定義体のプロパティで指定した帳票定義体名とファイル名は同一です。

```
@OPTIONS RCS(SJIS)
IDENTIFICATION DIVISION.
PROGRAM-ID. DENPYOU.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT データファイル
      ASSIGN        TO  infile
      ORGANIZATION IS  LINE SEQUENTIAL.
    SELECT 帳票の印刷先
      ASSIGN TO GS-PRTFILE
      SYMBOLIC DESTINATION IS "PRT"
      FORMAT               IS 帳票の名前
      GROUP                IS 項目群の名前
      PROCESSING MODE      IS 処理種別
      UNIT CONTROL         IS 詳細情報.
DATA DIVISION.
FILE SECTION.
 FD  データファイル.
 01  データレコード.
   02  商品レコード.
    03  ＷＫ－商品コード     PIC X(4).
    03                        PIC X.
    03  ＷＫ－商品名         PIC N(20).
    03                        PIC X.
    03  ＷＫ－単価           PIC 9(4).

 FD  帳票の印刷先.
       COPY  DENPY002 OF XMDLIB.
WORKING-STORAGE SECTION.
01  ＷＫ－金額            PIC 9(5).
77  合計                   PIC 9(4)  VALUE 0.
77  商品の数               PIC 9(4)  VALUE 1.
77  帳票の名前             PIC X(8)  VALUE "DENPY002".
77  項目群の名前           PIC X(8)  VALUE "GRP001".
77  処理種別               PIC XX.
77  詳細情報               PIC X(4).
01  年月日                 PIC 9(6).
```

表示ファイルの必須定義

パーティション形式の必須定義

※次のレコード形式として取り込んでいる

01 DENPYO02.
02 伝票 OCCURS 5.
03 商品コード PIC X(4).
03 商品名 PIC N(20).
03 数量 PIC 9(4).
03 単価 PIC 9(4).
03 金額 PIC 9(5).

帳票定義体のファイル名

帳票定義体のファイル名

定義体側の項目群名

All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
PROCEDURE DIVISION.
    OPEN INPUT  データファイル.
    OPEN OUTPUT 帳票の印刷先.
    MOVE SPACE      TO DENPY002.
印刷データを書き込む.
    READ データファイル AT END
    GO TO 帳票出力終了.
    IF 商品の数 > 5 THEN
        GO TO 帳票を印刷する
    END-IF.
    MOVE ＷＫ－商品コード TO
         商品コード OF DENPY002(商品の数).
    MOVE ＷＫ－商品名     TO
         商品名      OF DENPY002(商品の数).
    MOVE ＷＫ－単価       TO
         単価        OF DENPY002(商品の数).
    MOVE 商品の数         TO
         数量        OF DENPY002(商品の数).
    COMPUTE ＷＫ－金額 = ＷＫ－単価    商品の数.
    MOVE ＷＫ－金額       TO
         金額        OF DENPY002(商品の数).
    ADD  1          TO 商品の数.
    ADD  ＷＫ－金額  TO    合計.
    GO TO 印刷データを書き込む.
帳票を印刷する.
    ACCEPT 年月日 FROM DATE.
    MOVE 年月日(1:2)  TO  年   OF DENPY002.
    MOVE 年月日(3:2)  TO  月   OF DENPY002.
    MOVE 年月日(5:2)  TO  日   OF DENPY002.
    MOVE  合計 TO 金額合計 OF DENPY002.

    MOVE "PW"       TO 処理種別.
    MOVE "P001"     TO 詳細情報.
    WRITE DENPY002.

帳票出力終了.
    CLOSE 帳票の印刷先.

END PROGRAM DENPYOU.
```

帳票定義体のプロパティ名（レコード名）

指定した位置にパーティション項目群を出力する指定

帳票定義体のプロパティ名（レコード名）

All Rights Reserved, Copyright FUJITSU LIMITED 2006

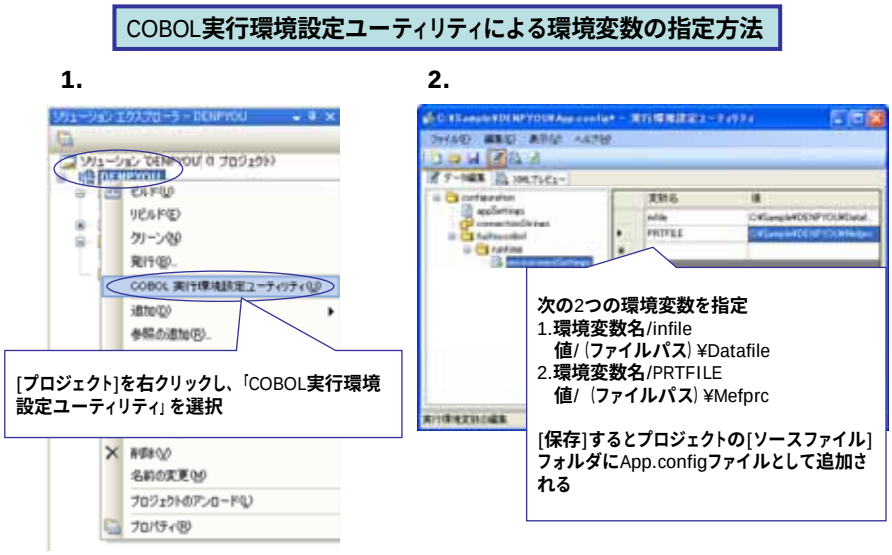
A.2.5 ビルドとエラー対処

プログラムの作成が完了したら、ビルドを行います。

ビルド時にラーが発生した場合、[エラー一覧]ウィンドウに具体的なエラーメッセージが出力されます。出力されたエラーメッセージ上でダブルクリックすると、エラーコードにカーソルが遷移するため、エラー箇所の特定が容易に行えます。エラーの対処後はリビルドを行い、[出力]ウィンドウに「ビルドが成功しました」とメッセージが確認できるまで、繰り返し行います。

A.2.6 アプリケーション構成ファイルの作成および実行

ビルドが終了したら、実行の際に必要な環境変数などが定義されたアプリケーション構成ファイルを作成します。このファイルはCOBOL実行環境設定ユーティリティを使用することで簡単に作成することが可能です。また、従来のCOBOLで使用していたCOBOL85.CBR等のファイルをインポートすることもできます。



A.2.6(1) アプリケーション構成ファイルの作成

All Rights Reserved, Copyright FUJITSU LIMITED 2006

実行環境設定ユーティリティによる環境変数の指定が終了すると、「ソリューションエクスプローラ」ウィンドウのプロジェクト内に拡張子(.config)のアプリケーション構成ファイルが追加されます。このファイルの作成が完了すれば、いよいよ帳票印刷プログラムの実行です。プログラムを実行すると、帳票定義体にテキストファイルのデータを取り込んで下図のように出力されます。

05年05月02日

コード	商品名	数量	単価	金額
0123	定規	1	200	200
0456	ボールペン	2	100	200
0789	分度器	3	50	150
0812	カッター	4	250	1000
0823	はさみ	5	450	2250
			合計	3800

図 A.2.6(2) 帳票出力のイメージ図

以上が、帳票定義体を使用した印刷の手順です。実行時に発生したエラーの対処については、.NET Framework共通のデバッグ機能を利用します。[デバッグ]メニューより「ステップイン」機能等を利用し、プログラムコードのどの部分でエラーとなっているのかを確認し、必要に応じて対処してください。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

◇MEMO◇

All Rights Reserved, Copyright FUJITSU LIMITED 2006

付録B アプリケーションの配置

B.1 プログラムの実行に必要なファイル
B.2 プログラムの配置

All Rights Reserved, Copyright FUJITSU LIMITED 2006

B.1 プログラムの実行に必要なファイル

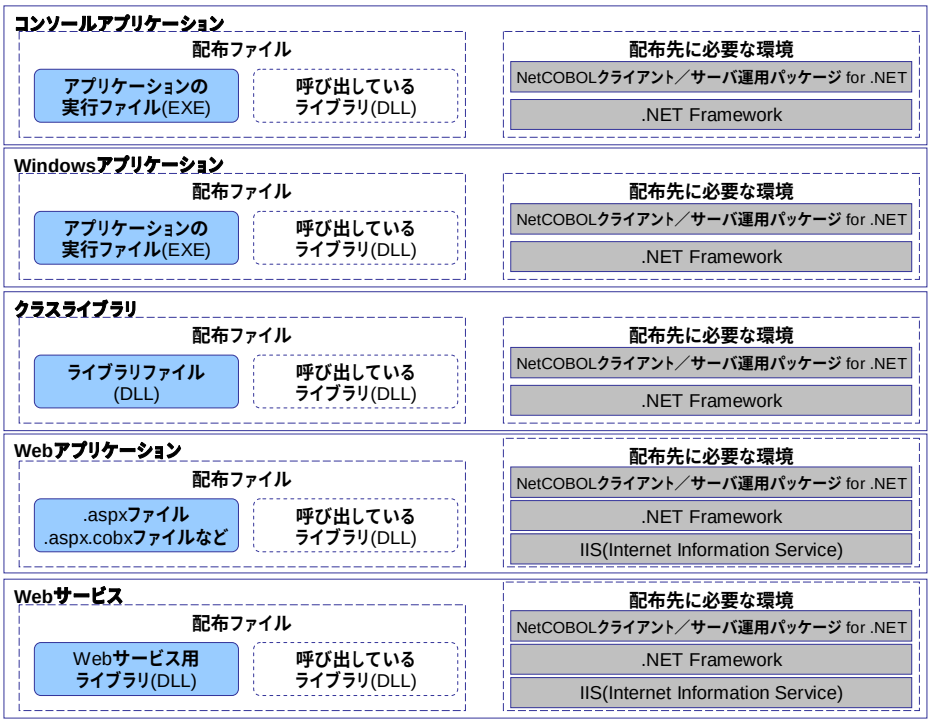
B.1.1 アプリケーションの種類と必要なファイル

NetCOBOL for .NETで作成したアプリケーションの動作環境としては、.NET FrameworkとNetCOBOL for .NETの運用パッケージが必要となります。.NET Frameworkの配布はMicrosoft社により許諾されており、Windows Updateでインストールすることができます。また、再配置可能なパッケージはMicrosoft MSDN Download Centerなどから「Dotnetfx.exe」として提供されています。

NetCOBOL for .NET運用パッケージは、アプリケーションの種類だけではなく、運用するOSの種類(クライアントOS/サーバOS)により異なる場合がありますので注意が必要です。また、WebアプリケーションやWebサービスを運用する場合、IIS(Internet Information Service)が必須となります。IISは、Windows 2000 ServerやWindows Server 2003に添付されています。

NetCOBOL for Windowsで作成したアプリケーションを配置する場合に必要となるファイルや環境は、アプリケーションの種類により異なります。したがって、アプリケーションの種類ごとに、配置方法や設定方法が異なります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



図B.1.1 アプリケーションの種類と配置ファイル

コンソールアプリケーション、Windowsアプリケーション、およびクラスライブラリを配置する場合、ビルド後に作成されたファイルの配置が必要です。Webアプリケーションでは、.aspxファイルやWebアプリケーション用にプリコンパイルされたDLLファイルなどを配置します。また、Webサービスでは.asmxファイルやWebサービス用にプリコンパイルされたDLLファイルなどを配置します。Webサイトは、Visual Studioに組み込まれているVisual Web DeveloperのWebサイトの発行ユーティリティを使って配置することもできます。Webサイトの発行ユーティリティを使う場合は、Visual Studioの[ビルド]メニューから[Webサイトの発行]を選択します。

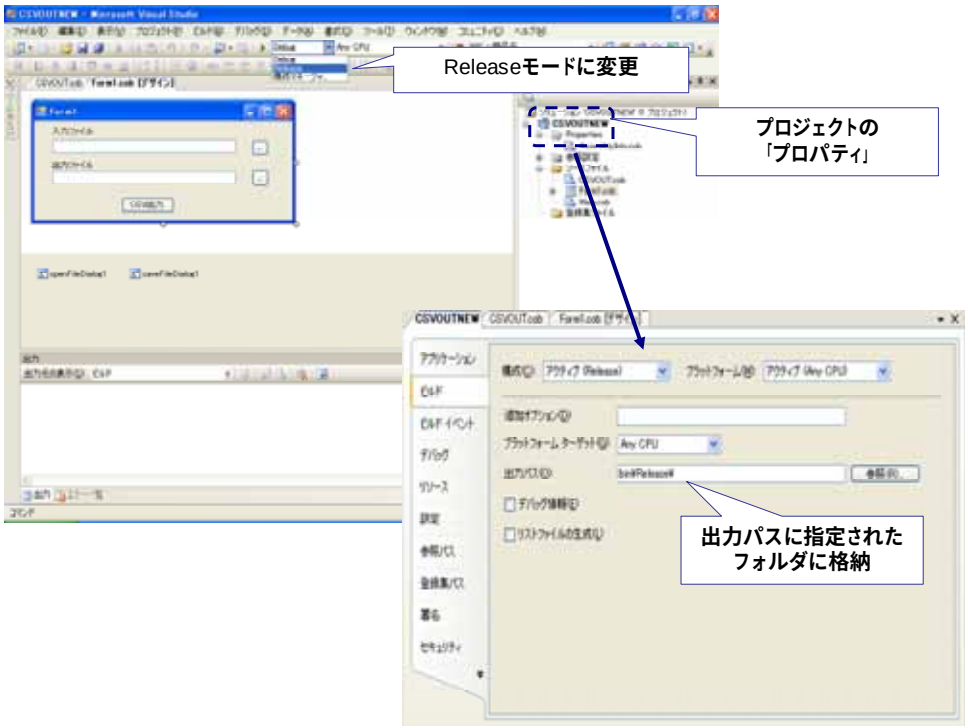
また、WebアプリケーションおよびWebサービスは、ファイル配置後にIISの設定などが必要になります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

B.2 プログラムの配置

B.2.1 Releaseモードでのビルド

コンソールアプリケーションやWindowsアプリケーション、クラスライブラリの場合、通常プログラムは配置して利用します。配置用のモジュールを作成する場合、開発環境でReleaseモードを指定してプログラムをビルドする必要があります。また、ビルドされたプログラムはプロジェクトのプロパティで指定されたフォルダに格納されます。



図B.2.1 Releaseモードと出力先

B.2.2 セットアップ/デプロイメントプロジェクト

Visual Studioには、作成したアプリケーションの配布用のプロジェクトのテンプレートとして、複数のテンプレートが用意されています。これらのセットアップ/デプロイメントプロジェクトを利用すると、簡単にプログラムを配布することができます。

デプロイメントプロジェクトを利用して、配置用のモジュールを作成すると、依存ファイル等の検出や、配置用のモジュールをmsiファイル

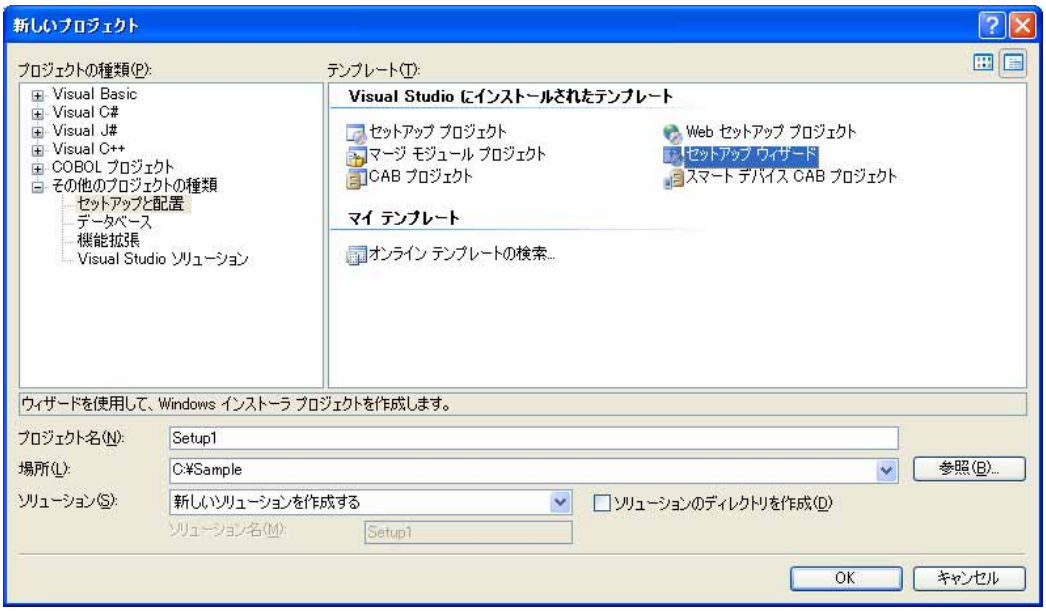
All Rights Reserved, Copyright FUJITSU LIMITED 2006

として生成などアプリケーションの配置に必要な処理を自動的に行うことができます。

次に、第4章で作成したWindowsアプリケーション「CSVWIN」を配置するプロジェクトを作成する例を紹介します(あらかじめCSVWIN.slnをReleaseモードでビルドしておきます)。

- **手順**1

  NetCOBOL for .NETを起動し、[ファイル]-[新規作成]-[プロジェクト]を選択し、「新しいプロジェクト」ダイアログの[プロジェクトの種類]で「その他のプロジェクトの種類」から「セットアップと配置」を選択します。次に[テンプレート]で「セットアップ ウィザード」を選択します。



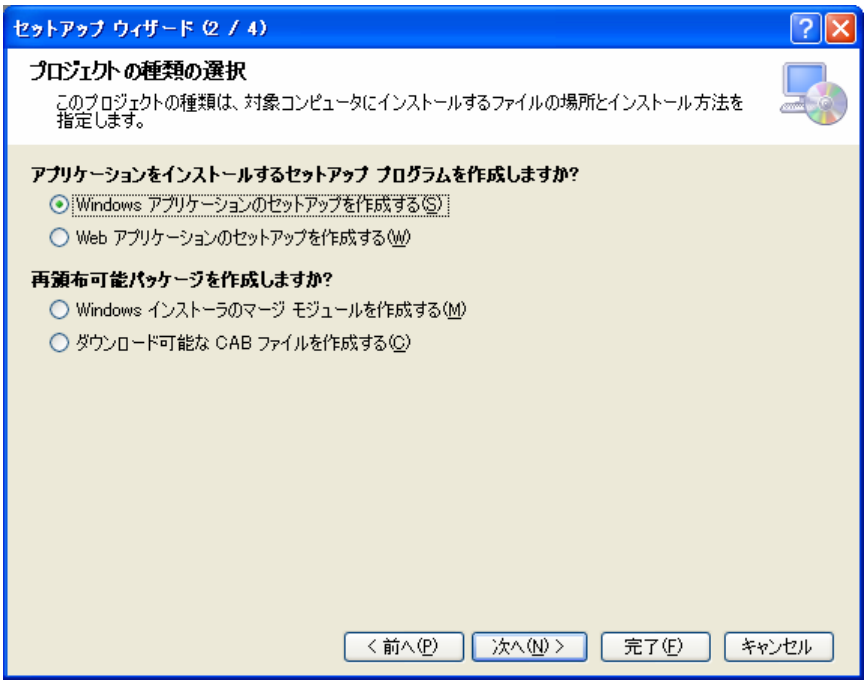
  つづいて、通常のプロジェクトと同じように、[プロジェクト名][場所]に任意プロジェクト名とプロジェクトを格納する場所を指定します。

  なお、プロジェクト名と場所は、配置用モジュールのプロジェクトおよび配置用モジュールの作成先であり、配置するプログラムや配置先とは直接関係ありません。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

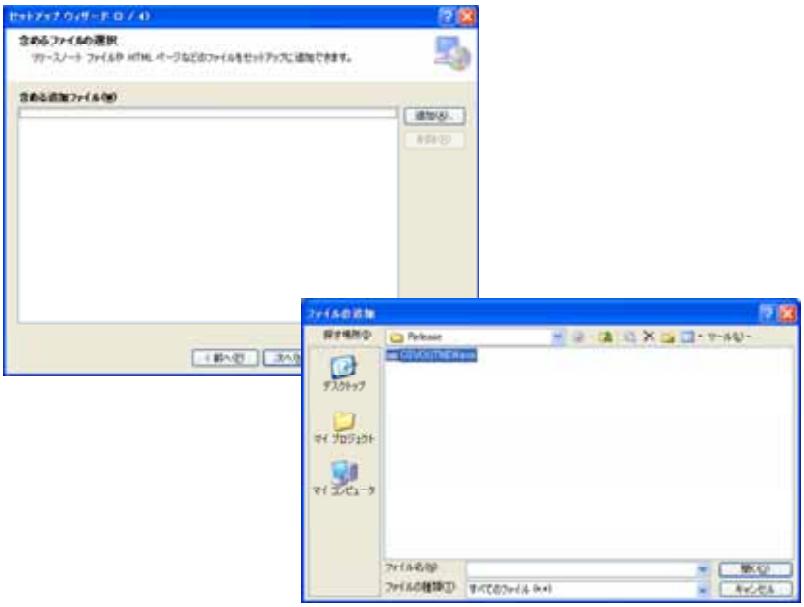
- **手順**2

  「セットアップ ウィザード(1/4)」はそのまま[次へ]をクリックし、「セットアップ ウィザード(2/4)」で[Windowsアプリケーションのセットアップを作成する]を選択し、[次へ]をクリックします。



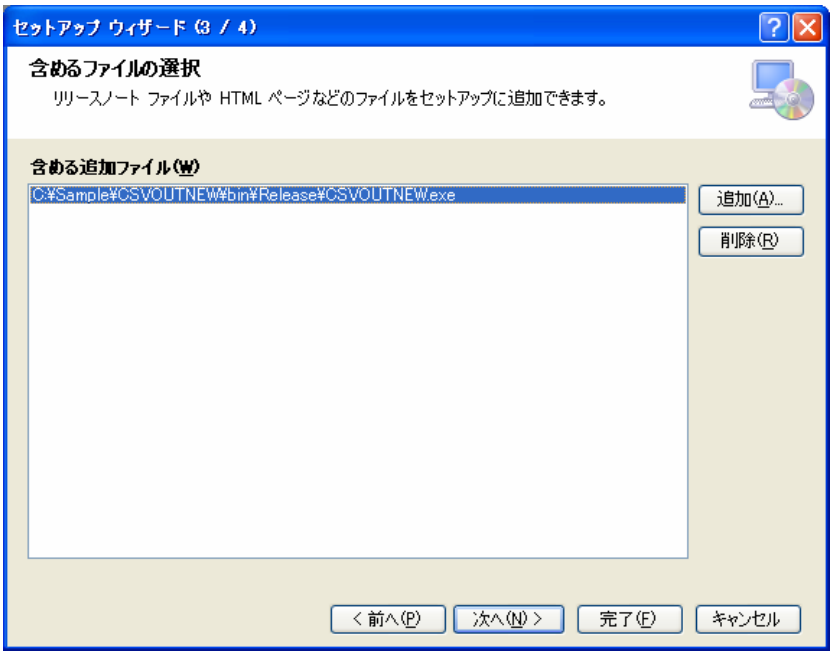
- **手順**3

  「セットアップ ウィザード(3/4)」で[次へ]をクリックし、「ファイルの追加」ウィンドウを表示します。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

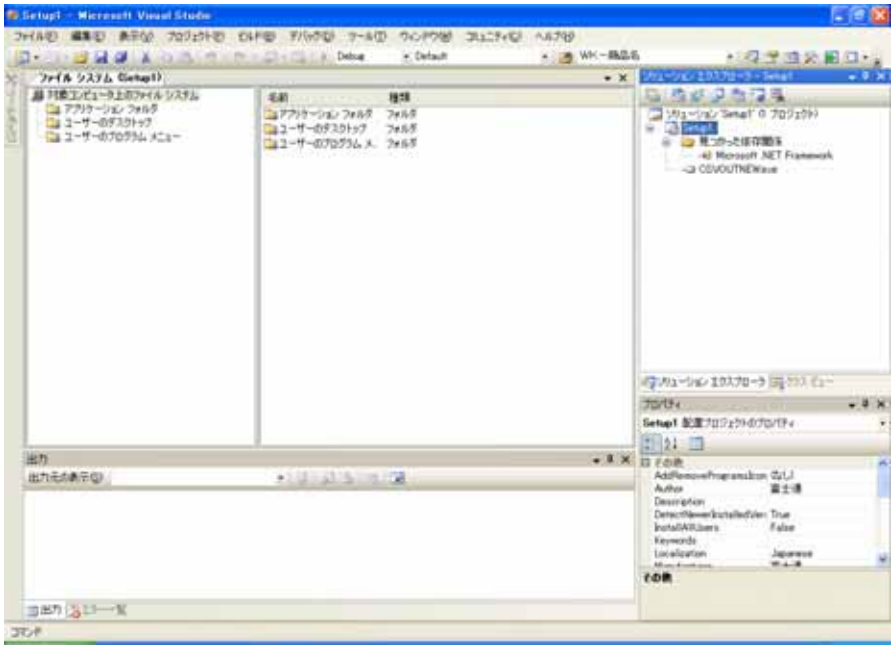
「ファイルの追加」では、Releaseモードでビルドした、配置したい実行形式のファイルを指定します。



目的のファイルが表示されたら[次へ]をクリックします。

- **手順**4

「セットアップ ウィザード(4/4)」で[完了]をクリックすると、セットアップ用のモジュールを作成するためのセットアッププロジェクトが作成されます。



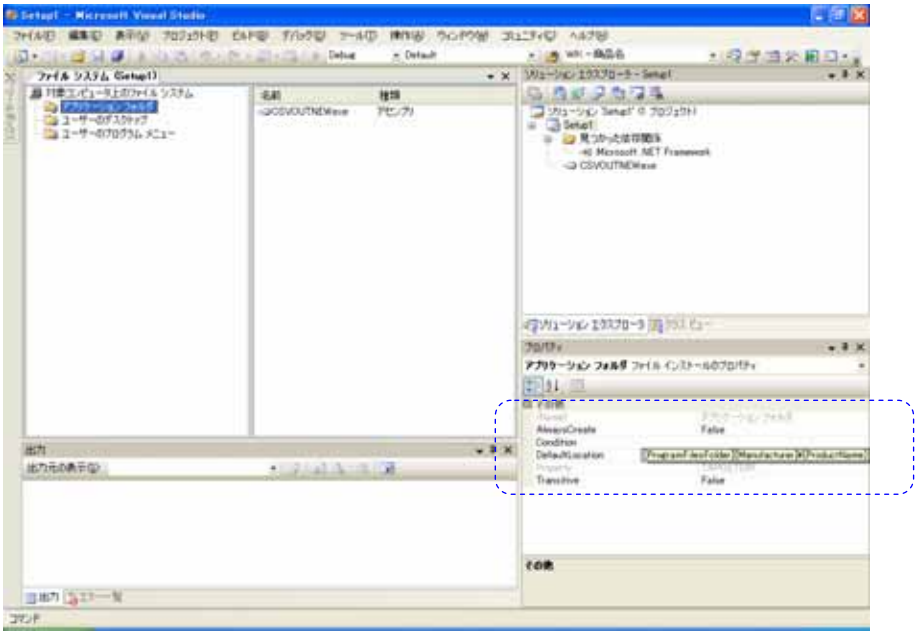
All Rights Reserved, Copyright FUJITSU LIMITED 2006

また、指定した実行ファイルに依存ファイルがある場合、依存ファイルを自動的に検出し、プロジェクトに含めます。

- **手順**5

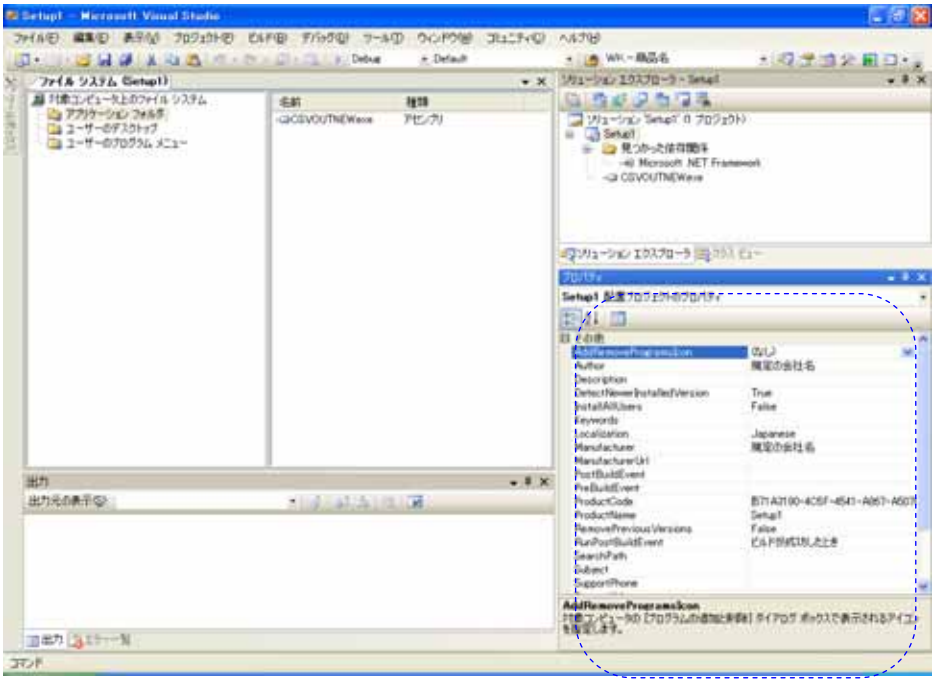
セットアップ プロジェクトでは、配置するファイル(プログラム)やファイルをコピーするフォルダなどの情報を指定する[アプリケーションフォルダ]、デスクトップやプログラムメニューにショートカットやファイルの追加の有無などを指定する[ユーザーのデスクトップ][ユーザーのプログラム メニュー]などのフォルダが表示されます。
特に、[アプリケーションフォルダ]のプロパティでは、Default Locationプロパティに次の指定があります。

[ProgramFilesFolder][Manufacturer]¥[ProductName]



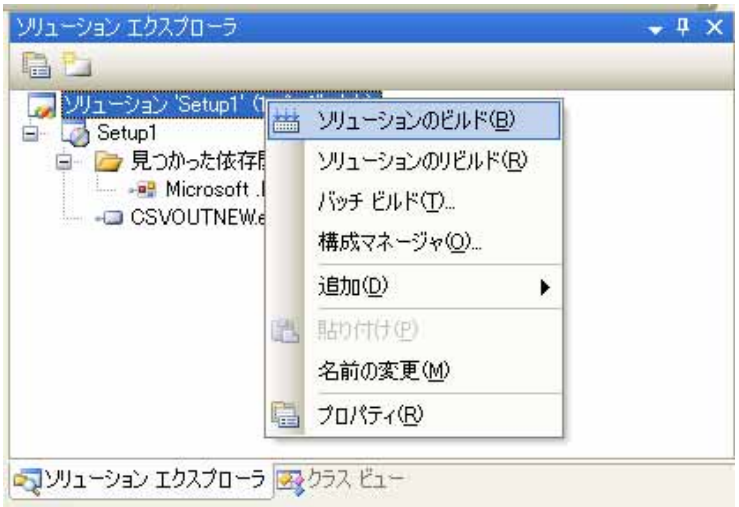
これは、「OS既定のProgram Filesフォルダ」の「作成者」フォルダの「製品名」フォルダに実行ファイルを格納することを意味しています。なお、「作成者」と「製品名」はプロジェクトのプロパティで設定することができます。

All Rights Reserved, Copyright FUJITSU LIMITED 2006



- **手順**6

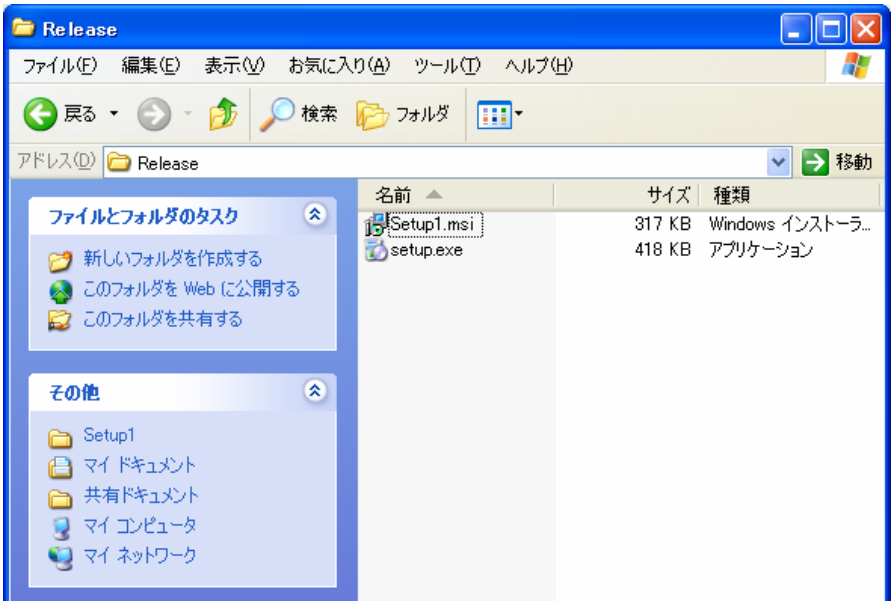
  セットアップ プロジェクトの設定が終了したら、通常のプログラムと同じようにプロジェクト(ソリューション)のビルドを行います。なお、デバッグ情報を埋め込む必要はありませんので、Releaseモードでビルドするようにしてください。



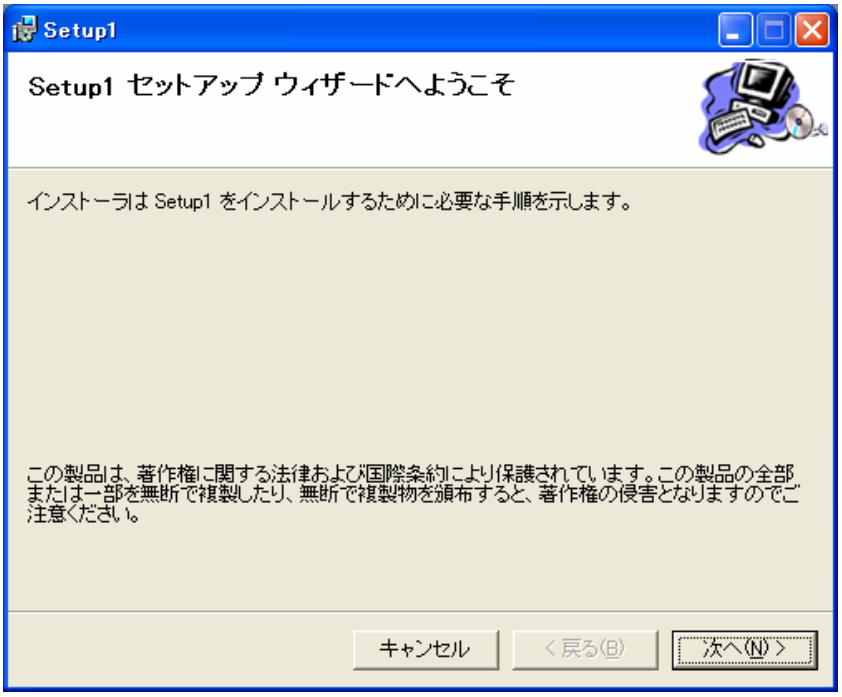
All Rights Reserved, Copyright FUJITSU LIMITED 2006

- **手順7**

  ビルドが終了すると、Releaseフォルダに配置用のセットアッププログラムが作成されます。



プログラムの配置に必要なファイルは全てパッケージされていますので、これらのファイルを配置したいコンピュータにコピーし、Setup.exeを実行すると、アプリケーションのセットアップが開始されます。



All Rights Reserved, Copyright FUJITSU LIMITED 2006

付録C サンプルプログラム概要

C.1 コンソールアプリケーション仕様
C.2 Windowsアプリケーション仕様
C.3 クラスライブラリ仕様
C.4 Webアプリケーション仕様

All Rights Reserved, Copyright FUJITSU LIMITED 2006

C.1 コンソールアプリケーション仕様

ここでは、第2章で説明したサンプルアプリケーションの仕様およびファイルレイアウトを掲載します。

(1) プログラム概要書

資料名	システム名	サブシステム名	作成日	作成者	頁
プログラム概要書	索引ファイル変換システム				
ファイル変換	CSVOUT.COB				

画面レイアウト

＜処理概要＞

・ 索引ファイルをCSVファイルに変換する

1. ファイル変換処理

・ 索引ファイルをCSVファイルに変換する

2. 処理詳細

(1) プログラム起動後、商品ファイル(索引ファイル)を読み込み、商品ファイル(行順ファイル)をCSV形式で出力する

画面項目

入出力項目

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1							
2							
3							
4							

- ： デフォルト

ボタン

No	項目名	処理

ファイル

No	ファイル名	文字コード	処理
1	STOCK.DAT		ファイル変換元
2	STOCK.csv		ファイル変換先

All Rights Reserved, Copyright FUJITSU LIMITED 2006

(2) ファイルレイアウト

商品ファイル(索引)

ファイル名(日本語)	在庫ファイル						レコード名(日本語)	在庫レコード
ファイル編成	COBOL索引ファイル							
項目名	属性	桁数	キー項目				備　　考	
製品番号	9	4	●				キー項目は重複不可	
製品名	X	20						
在庫数量	S9	2						
倉庫番号	9	4						
定価	9	9						
仕切率	S999V99							

商品ファイル(CSV)

ファイル名(日本語)	出力ファイル						レコード名(日本語)	出力レコード
ファイル編成	行順ファイル(CSV)							
項目名	属性	桁数	キー項目				備　　考	
製品番号	X	4						
セパレータ１	X	1					VALUE ","	
製品名	X	20						
セパレータ２	X	1					VALUE ","	
在庫数量	X	3						
セパレータ３	X	1					VALUE ","	
倉庫番号	X	4						
セパレータ４	X	1					VALUE ","	
定価	X	9						
セパレータ５	X	1					VALUE ","	
仕切率	X	7						

All Rights Reserved, Copyright FUJITSU LIMITED 2006

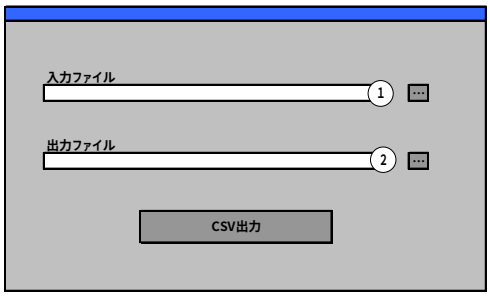
C.2 Windowsアプリケーション仕様

ここでは、第4章で説明したサンプルアプリケーションの仕様を掲載します。なお、ファイルレイアウトは「B.1 コンソールアプリケーション仕様」と同様です

(1) プログラム概要書

資　料　名	システム名	サブシステム名	作成日	作成者	頁
プログラム概要書	索引ファイル変換システム				
ファイル変換	CSVOUTNEW				

画面レイアウト



＜処理概要＞

・ 索引ファイルをCSVファイルに変換する

1. ファイル変換処理

・ 索引ファイルをCSVファイルに変換する

2. 処理詳細

(1) 入力ファイル、出力ファイルをそれぞれテキストボックスに入力するか、参照ボタンより設定する。

(2) 入力ファイル(索引ファイル)を読み込み、出力ファイル(行順ファイル)をCSV形式で出力する。

画面項目

入出力項目

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	入力ファイル	I					
2	出力ファイル	I					
3							
4							

- ： デフォルト

ボタン

No	項目名	処理
1	…(ファイル参照)	入力ファイルを開くダイアログを表示します
2	…(ファイル参照)	出力ファイルを指定するダイアログを表示します
3	CSV出力	CSV変換処理を行います

ファイル

No	ファイル名	文字コード	処理
1	STOCK.DAT	シフトJIS	ファイル変換元
2	STOCK.csv	シフトJIS	ファイル変換先

All Rights Reserved, Copyright FUJITSU LIMITED 2006

C.3 クラスライブラリ仕様

ここでは、第5章で説明したサンプルアプリケーションの仕様および利用するテーブルのレイアウトを掲載します。なお、3つのサブプログラムを1つのクラスライブラリとして生成しています。

(1) プログラム概要書

社員認証処理

資料名	システム名	サブシステム名	作成日	作成者	頁
プログラム概要書	社員認証サブプログラム				
データアクセス	SYAIN.COB				

パラメータと戻り値

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	P-EmployeeCode	I	N	6			従業員コード
2	P-Password	I	N	6			パスワード
3	P-Msg	O	N	20			例外メッセージ
No	項目名	I/O	型	桁数	編集形式		備考
1	R-ReturnCode	O	BINARY-LONG				

テーブル

No	テーブル名	使用内容
1	社員テーブル	認証チェック(従業員コード、パスワード)

1. 社員認証処理

・ 従業員コード/パスワードが正しいか認証する

2. 処理詳細

(1) 従業員コード、パスワードをパラメタとして受け取る

(2) 社員テーブルを参照し、従業員コードに該当するパスワードを読み込む

(3) 受け取ったパスワードと読み込んだパスワードを比較し、同一なら戻り値に「0」を設定する。異なるなら「3」を設定し、例外メッセージにエラーメッセージを格納する(パスワードに、間違いがあります)。

(4) 社員テーブルに該当する従業員コードが存在しない場合、戻り値に「2」を設定し、例外メッセージにエラーメッセージを格納する(社員コードに、間違いがあります)。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

商品レコード参照

資　料　名	システム名	サブシステム名	作成日	作成者	頁
プログラム概要書	商品レコード参照サブプログラム				
データアクセス	SYOHIN.COB				

パラメータと戻り値

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	P-SYOHINREC	I/O	集団				商品レコード
2	P-SYOCD	I/O	9	5			商品コード
3	P-SYONA	O	N	20			商品名
4	P-BAIKA	O	9	8			売価
5	P-GENKA	O	9	8			原価
6	P-Msg	O	N	20			例外メッセージ
No	項目名	I/O	型	桁数	編集形式		備考
1	R-ReturnCode	O	BINARY-LONG				

テーブル

No	テーブル名	使用内容
1	商品テーブル	商品レコード参照

1. 商品レコード参照

・ 商品テーブルを参照し、商品レコードを取得する

2. 処理詳細

(1) 商品コードが格納された商品レコードをパラメタとして受け取る

(2) 商品テーブルを参照し、該当する商品レコードを参照する

(3) 該当する商品レコードが存在する場合、商品レコードを読み込み、戻り値に「0」を設定する。

(4) 該当する商品コレードがない場合、戻り値に「5」を設定する。また
メッセージにエラーメッセージを格納する(商品コードに、間違いがあります)。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

入出庫レコード参照

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	M-AREA	I/O	集団(IODETAIL..CBL				表示用明細
2	P-Start	I	BINARY-SHORT				開始行
3	P-Cnt	O	9	3			件数
4	P-Next	O	9	1			次レコード有無
5	P-Msg	O	N	20			例外メッセージ
No	項目名	I/O	型	桁数	編集形式		備考
1	R-ReturnCode	O	BINARY-LONG				

IODETAIL.CBL

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	M-AREA.		集団				
2	M-Y		9	4			年
3	M-M		9	2			月
4	M-SYOCD		9	5			商品コード
5	M-TANKA		9	8			単価
6	M-GENKA		9	8			原価
7	M-DENPYO		OCCURS			10	明細行
8	M-IODATE		9	8			入出庫日
9	M-DENBAN		9	12			伝票番号
10	M-KUBUN		9	2			区分
11	M-KUBUNN		N	2			区分 (日本語)
12	M-KAKAKU		9	8			価格
13	M-ISU		9	6			入庫数
14	M-OSU		9	6			出庫数

テーブル

No	テーブル名	使用内容
1	入出庫テーブル	商品レコード参照

・ 入出庫テーブルを参照し、ページ行数文の入出庫データを作成する

2. 処理詳細

(1) 明細レコード内の商品コードおよび年、月、開始行をパラメタとして受け取る

(2) 入出庫テーブルを参照し、開始行から10件分の該当するレコードを読み取り
表示用明細に格納し、件数に該当件数、戻り値に「0」を格納する
該当件数が11件以上存在する場合、次レコード有無フラグに「1」を設定する。

(3) 該当する入出庫コレードがない場合、戻り値に「8」を設定する。また
メッセージにエラーメッセージを格納する(該当の入出庫明細がありません)。

(2) テーブルレイアウト(SQL Serverでの例)

社員テーブル

ファイル名(日本語)	SYAPF (社員テーブル)					レコード名(日本語)	社員レコード
テーブルタイプ	SQL Server 実表						
項目名	属性	桁数	キー項目				備　考
SYACD (社員コード)	int		●				キー項目は重複不可/NOT NUL
SYANA (社員名)	nvarchar	20					
PASS (パスワード)	nvarchar	6					
BUSYO (所属部署)	int						
SDATE (取得年月日)	int						
SNAME (取得資格名)	nvarchar	10					

All Rights Reserved, Copyright FUJITSU LIMITED 2006

商品テーブル

ファイル名(日本語)	SYOPF (商品テーブル)						レコード名(日本語)	商品レコード
テーブルタイプ	SQL Server 実表							
項目名	属性	桁数	キー項目				備　考	
SYOCD (商品コード)	int		●				キー項目は重複不可/NOT NUL	
SYONA (商品名)	nvarchar	20						
BAIKA (売価)	int							
GENKA (原価)	int							

入出庫テーブル

ファイル名(日本語)	SYOIOPF (入出庫テーブル)						レコード名(日本語)	入出庫レコード
テーブルタイプ	SQL Server 実表							
項目名	属性	桁数	キー項目				備　考	
IODATE (入出庫日)	int			●			キー項目は重複不可/NOT NUL	
SYOCD (商品コード)	int		●				キー項目は重複不可/NOT NUL	
NODATE (納品日)	int							
SYUBAN (伝票番号主番)	int							
TUBAN (伝票番号通番)	int							
KUBUN (伝票区分コード)	int							
IOSU (入出庫数)	int							

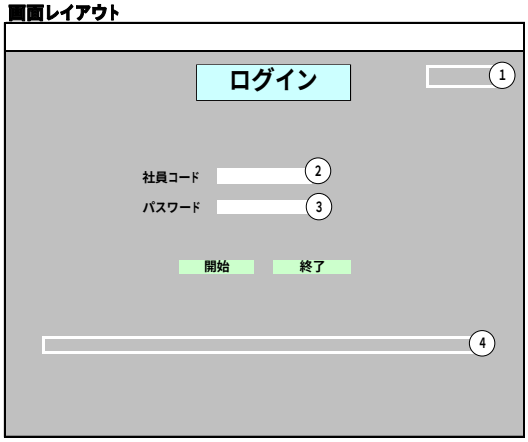
All Rights Reserved, Copyright FUJITSU LIMITED 2006

C.4 Webアプリケーション仕様

ここでは、第7章で説明したサンプルアプリケーションの仕様を掲載します。

認証処理

画面レイアウト



＜処理概要＞

・ ユーザの認証および入出庫明細の一覧表示を行う。

1. ユーザ認証処理

・ ユーザの認証を行う。

2. 処理詳細

(1) 画面起動の際、現在日付（9999.Z9.Z9）を表示する。

(2) 社員コード、パスワードを入力し、[開始]ボタンをクリック後、認証プログラムを呼び出す。

(3) 認証プログラムの戻り値が「0」の場合は、「入出庫紹介」画面に遷移する。

(4) 認証プログラムの戻り値が「0」以外の場合は、認証プログラムから受け取った例外メッセージを表示する。

(5) [終了]ボタンがクリックされた場合、当（ログイン）画面を閉じる。

画面項目

入出力項目

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	現在日	O	X	10			
2	社員コード	I	N	6			
3	パスワード	I	N	6			
4	メッセージ	O	N	30			

- ： デフォルト

ボタン

No	項目名	処理
1	開始	認証処理の実行
2	終了	ログイン画面を閉じる

サブプログラム

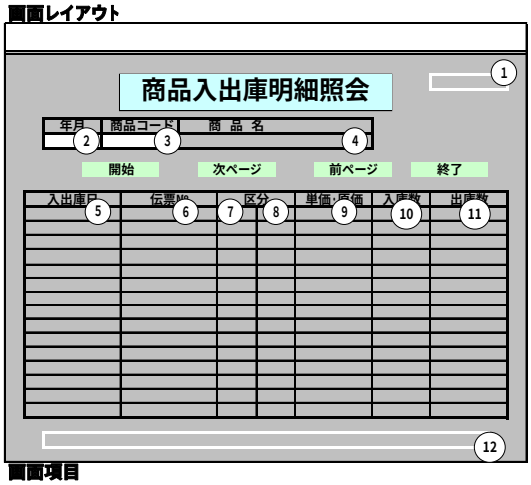
No	ファイル名	処理
1	SYAIN.COB	社員認証処理

All Rights Reserved, Copyright FUJITSU LIMITED 2006

入出庫一覧表示処理

資　料　名	システム名	サブシステム名	作成日	作成者	頁
プログラム概要書	入出庫明細表示システム				
入出庫照会	IoDetail.aspx				

画面レイアウト



画面項目

入出力項目

No	項目名	I/O	型	桁数	編集形式	繰返し数	備考
1	現在日	O	9	10	ZZZ9.Z9.Z9		
2	年月	I	9	6	ZZZ9.Z9		
3	商品コード	I	X	5			
4	商品名	O	N	10			
5	入出庫日	O	9	10	ZZZ9.Z9.Z9	10	
6	伝票№	O	9	12	zzzzzzzzzzz9	10	
7	伝票区分番号	O	9	2	Z9	10	
8	伝票区分名	O	N	2		10	
9	単価･原価	O	9	6	ZZ,ZZZ,ZZ9	10	
10	入庫数	O	9	6	ZZZ,ZZ9	10	
11	出庫数	O	9	6	ZZZ,ZZ9	10	
12	メッセージ	O	N	30			

1. 入出庫明細一覧表示処理

・ 入出庫一覧を表示する。

2. 処理詳細

(1) 画面起動の際、現在日付（9999.Z9.Z9）を表示する。また、次頁、前頁ボタンを無効にする。

(2) 年月、商品コードを入力し、開始ボタンをクリック後、入出庫一覧の表示を行う。

(3) 入力した年月の妥当性チェックを行い、エラーの場合、エラーメッセージを表示する。

(4) 入力した商品コードを商品レコード参照サブプログラムに渡し、商品名を受け取り表示する。

(5) 入力した年月と商品コードで入出庫レコード参照サブプログラムに渡し、
1ページ分の結果を表示する

a なお、表示画面の前ページの明細データが存在する場合は、[前頁]ボタンを有効とし、
次ページの明細データが存在する場合は、[次頁]ボタンを有効にする。

(6) [次頁]ボタンを押下した場合、表示された一覧の次の明細データを編集し画面表示する。
※　aに同じ

(7) [前頁]ボタンを押下した場合、表示された一覧の前の明細データを編集し画面表示する。
※　aに同じ

(8) 呼出したサブプログラムの戻り値が0でない場合は、サブプログラムから返される
例外エラーを表示する。

(9) [終了]ボタンをクリック後、当（入出庫照会）画面を閉じて、「ログイン」画面に遷移する。

ボタン

No	項目名	処理内容
1	前頁	入出庫一覧の前頁を表示
2	次頁	入出庫一覧の次頁を表示
3	終了	入出庫照会画面を閉じる
4		

All Rights Reserved, Copyright FUJITSU LIMITED 2006

付録D ADO.NETサンプル

All Rights Reserved, Copyright FUJITSU LIMITED 2006

D.1 ADO.NETによるデータベースアクセス

ここでは、第5章の「5.5 データアクセスプログラム」で作成した埋め込みSQLでアクセスするプログラムを、ADO.NETのクラスライブラリを使用してアクセスする方法に変更した場合のコーディング例を付録として掲載します。また、このコーディング例では、「5.5 データアクセスプログラム」で作成したプログラムで使用しているアプリケーション構成ファイルをそのまま利用できるように、ADO.NETのファクトリと共通クラスを使用しています。

次の3本のプログラムを修正しています。

- ✓ SYAIN.COB
- ✓ SYOHIN.COB
- ✓ ZAIKO.COB

■ SYAIN.COB

```
*****************************************************************
* プログラム名 : SYAIN
* 概要         : 社員テーブルにアクセスし認証を行う。
*
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYAIN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
     CLASS CLASS-ConfigurationMaager      AS
          "System.Configuration.ConfigurationManager"
     CLASS CLASS-ConnectionStringSettings AS
          "System.Configuration.ConnectionStringSettings"
     CLASS CLASS-String AS "System.String"
     CLASS CLASS-DbProviderFactories      AS
          "System.Data.Common.DbProviderFactories"
     CLASS CLASS-DbProviderFactory        AS
          "System.Data.Common.DbProviderFactory"
     CLASS CLASS-DbConnection             AS
          "System.Data.Common.DbConnection"
     CLASS CLASS-DbCommand                AS
          "System.Data.Common.DbCommand"
     CLASS CLASS-DbParameter              AS
          "System.Data.Common.DbParameter"
     CLASS CALSS-DbDataReader             AS
          "System.Data.Common.DbDataReader"
     CLASS CLASS-Exception                AS
          "System.Exception"
     ENUM ENUM-DbType AS "System.Data.DbType"

     PROPERTY PROP-PARAMETERS             AS "Parameters"
     PROPERTY PROP-VALUE                  AS "Value"
     PROPERTY PROP-MESSAGE                AS "Message"
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
     PROPERTY PROP-CONNECTIONSTRINGS      AS
             "ConnectionStrings"
     PROPERTY PROP-CONNECTIONSTRING       AS
             "ConnectionString"
     PROPERTY PROP-PROVIDERNAME AS "ProviderName"
     PROPERTY PROP-COMMANDTEXT AS "CommandText"
     PROPERTY PROP-INT32 AS "Int32"
     PROPERTY PROP-PARAMETERNAME AS "ParameterName"
     PROPERTY PROP-DBTYPE AS "DbType".

 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.
 01 SYAPF-REC.
   02  SYACD      PIC S9(9) COMP-5.
   02  SYANA      PIC X(20).
   02  PASS       PIC N(6).
   02  BUSYO      PIC S9(9) COMP-5.
   02  SDATE      PIC S9(9) COMP-5.
   02  SNAME      PIC X(20).

 01  W-CONNSTR    OBJECT REFERENCE
                  CLASS-ConnectionStringSettings.
 01  W-PROVIDER   OBJECT REFERENCE CLASS-String.
 01  W-FACTORY    OBJECT REFERENCE CLASS-DbProviderFactory.
 01  W-CONN       OBJECT REFERENCE CLASS-DbConnection.
 01  W-CMD        OBJECT REFERENCE CLASS-DbCommand.
 01  W-DBPARAM    OBJECT REFERENCE CLASS-DbParameter.
 01  W-DBTYPEOBJECT REFERENCE ENUM-DbType.
 01  W-READER     OBJECT REFERENCE CALSS-DbDataReader.
 01  W-EXCEPTION  OBJECT REFERENCE CLASS-Exception.

 COPY MSGTBL.
 LINKAGE                          SECTION.
 01  P-EmployeeCode           PIC N(06).
 01  P-Password               PIC N(06).
 01  P-Msg                    PIC N(20).
 01  R-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION USING
                    P-EmployeeCode   *>社員コード
                    P-Password       *>パスワード
                    P-Msg            *>メッセージ
                    RETURNING
                    R-ReturnCode.    *>戻り値(エラーコード)
     TRY
*        接続文字列取得
         INVOKE PROP-CONNECTIONSTRINGS OF
           CLASS-ConfigurationManager "get_Item"
             USING N"cobdb" RETURNING W-CONNSTR
*        プロバイダファクトリ取得
         SET W-PROVIDER TO PROP-PROVIDERNAME OF W-CONNSTR
         INVOKE CLASS-DbProviderFactories "GetFactory"
           USING W-PROVIDER RETURNING W-FACTORY
*        コネクション生成
         INVOKE W-FACTORY "CreateConnection" RETURNING W-CONN
         SET PROP-CONNECTIONSTRING OF W-CONN TO
             PROP-CONNECTIONSTRING OF W-CONNSTR
*        コマンド生成
         INVOKE W-CONN "CreateCommand" RETURNING W-CMD
         SET PROP-COMMANDTEXT OF W-CMD TO
           N"SELECT SYACD, PASS FROM SYAPF WHERE SYACD = @CD"
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*         パラメータ作成
          INVOKE W-CMD "CreateParameter" RETURNING W-DBPARAM
          SET PROP-PARAMETERNAME OF W-DBPARAM TO N"@CD"
          SET PROP-DBTYPE OF W-DBPARAM TO
                          PROP-INT32 OF ENUM-DbType
          SET PROP-VALUE OF W-DBPARAM TO P-EmployeeCode
          INVOKE PROP-PARAMETERS OF W-CMD "Add" USING W-DBPARAM

*         DBオープン
          INVOKE W-CONN "Open"

*         DbDataReaderの構築
          INVOKE W-CMD "ExecuteReader" RETURNING W-READER
          IF W-READER::"Read" = B"1" THEN
              INVOKE W-READER "GetInt32" USING 0
                              RETURNING SYACD
              INVOKE W-READER "GetString" USING 1
                              RETURNING PASS

*             パスワード比較
              IF PASS = P-Password THEN
                  MOVE 0 TO R-ReturnCode
              ELSE
                  MOVE 3 TO R-ReturnCode
              END-IF
          ELSE
*             見つからなかった場合
              MOVE 2 TO R-ReturnCode
          END-IF

*         DbDataReaderのクローズ
          INVOKE W-READER "Close"

*         DBクローズ
          INVOKE W-CONN "Close"

*         メッセージ設定
          IF R-ReturnCode > 1 THEN
              MOVE MSG-DSP(R-ReturnCode) TO P-Msg
          END-IF
*     例外処理
      CATCH W-EXCEPTION
*         例外メッセージを戻り値に格納
          SET P-Msg TO PROP-MESSAGE OF W-EXCEPTION
          MOVE 99 TO R-ReturnCode
*     終了処理
      FINALLY
*         コマンド終了処理
          IF W-CMD NOT = NULL
              INVOKE W-CMD "Dispose"
          END-IF
*         コネクション終了処理
          IF W-CONN NOT = NULL
            INVOKE W-CONN "Dispose"
          END-IF
      END-TRY.
  END PROGRAM SYAIN.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ SYOHIN.COB

```
*************************************************************
* プログラム名 : SYOHIN
* 概要         : 商品テーブルにアクセスし、商品レコードの取得を行う。
*
*************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYOHIN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
     CLASS CLASS-ConfigurationManager     AS
          "System.Configuration.ConfigurationManager"
     CLASS CLASS-ConnectionStringSettings AS
          "System.Configuration.ConnectionStringSettings"
     CLASS CLASS-String        AS "System.String"
     CLASS CLASS-DbProviderFactories      AS
          "System.Data.Common.DbProviderFactories"
     CLASS CLASS-DbProviderFactory        AS
          "System.Data.Common.DbProviderFactory"
     CLASS CLASS-DbConnection             AS
          "System.Data.Common.DbConnection"
     CLASS CLASS-DbCommand                AS
          "System.Data.Common.DbCommand"
     CLASS CLASS-DbParameter    AS
          "System.Data.Common.DbParameter"
     CLASS CALSS-DbDataReader             AS
          "System.Data.Common.DbDataReader"
     CLASS CLASS-Exception                AS
          "System.Exception"
     ENUM ENUM-DbType         AS "System.Data.DbType"

     PROPERTY PROP-PARAMETERS             AS "Parameters"
     PROPERTY PROP-VALUE                  AS "Value"
     PROPERTY PROP-MESSAGE                AS "Message"
     PROPERTY PROP-CONNECTIONSTRINGS      AS
             "ConnectionStrings"
     PROPERTY PROP-CONNECTIONSTRING       AS
             "ConnectionString"
     PROPERTY PROP-PROVIDERNAME AS "ProviderName"
     PROPERTY PROP-COMMANDTEXT AS "CommandText"
     PROPERTY PROP-INT32 AS "Int32"
     PROPERTY PROP-PARAMETERNAME AS "ParameterName"
     PROPERTY PROP-DBTYPE AS "DbType".

 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SYOPF-REC.
   02  SYOCD      PIC S9(9) COMP-5.
   02  SYONA      PIC N(20).
   02  BAIKA      PIC S9(9) COMP-5.
   02  GENKA      PIC S9(9) COMP-5.

 COPY MSGTBL.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
 01  W-CONNSTR    OBJECT REFERENCE
                  CLASS-ConnectionStringSettings.
 01  W-PROVIDER   OBJECT REFERENCE CLASS-String.
 01  W-FACTORY    OBJECT REFERENCE CLASS-DbProviderFactory.
 01  W-CONN       OBJECT REFERENCE CLASS-DbConnection.
 01  W-CMD        OBJECT REFERENCE CLASS-DbCommand.
 01  W-DBPARAM    OBJECT REFERENCE CLASS-DbParameter.
 01  W-DBTYPEOBJECT REFERENCE ENUM-DbType.
 01  W-READER     OBJECT REFERENCE CALSS-DbDataReader.
 01  W-EXCEPTION  OBJECT REFERENCE CLASS-Exception.
 LINKAGE                          SECTION.
 01  P-SYOHINREC.
     02  P-SYOCD             PIC 9(5).
     02  P-SYONA             PIC N(20).
     02  P-BAIKA             PIC 9(8).
     02  P-GENKA             PIC 9(8).
 01  P-Msg                   PIC N(20).
 01  R-ReturnCode            BINARY-LONG.
 PROCEDURE DIVISION USING
                     P-SYOHINREC      *>商品レコード
                     P-Msg            *>メッセージ
                     RETURNING
                     R-ReturnCode.    *>戻り値(エラーコード)
     TRY
*        接続文字列取得
         INVOKE PROP-CONNECTIONSTRINGS OF
           CLASS-ConfigurationManager "get_Item"
             USING N"cobdb" RETURNING W-CONNSTR
*        プロバイダファクトリ取得
         SET W-PROVIDER TO PROP-PROVIDERNAME OF W-CONNSTR
         INVOKE CLASS-DbProviderFactories "GetFactory"
           USING W-PROVIDER RETURNING W-FACTORY
*        コネクション生成
         INVOKE W-FACTORY "CreateConnection" RETURNING W-CONN
         SET PROP-CONNECTIONSTRING OF W-CONN TO
             PROP-CONNECTIONSTRING OF W-CONNSTR
*        コマンド生成
         INVOKE W-CONN "CreateCommand" RETURNING W-CMD
         SET PROP-COMMANDTEXT OF W-CMD TO
           N"SELECT SYOCD, SYONA, BAIKA, GENKA
             FROM SYOPF WHERE SYOCD = @CD"
*        パラメータ作成
         INVOKE W-CMD "CreateParameter" RETURNING W-DBPARAM
         SET PROP-PARAMETERNAME OF W-DBPARAM TO N"@CD"
         SET PROP-DBTYPE OF W-DBPARAM TO
             PROP-INT32 OF ENUM-DbType
         SET PROP-VALUE OF W-DBPARAM TO P-SYOCD OF P-SYOHINREC
         INVOKE PROP-PARAMETERS OF W-CMD "Add" USING W-DBPARAM

*        DBオープン
         INVOKE W-CONN "Open"

*        DbDataReaderの構築
         INVOKE W-CMD "ExecuteReader" RETURNING W-READER
         IF W-READER::"Read" = B"1" THEN
           INVOKE W-READER "GetInt32" USING 0 RETURNING SYOCD
           INVOKE W-READER "GetString" USING 1
                                       RETURNING P-SYONA
           INVOKE W-READER "GetInt32" USING 2 RETURNING BAIKA
           INVOKE W-READER "GetInt32" USING 3 RETURNING GENKA
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
            MOVE SYOCD TO P-SYOCD
            MOVE BAIKA TO P-BAIKA
            MOVE GENKA TO P-GENKA
          ELSE
*           見つからなかった場合
            MOVE 5 TO R-ReturnCode
          END-IF

*         DbDataReaderのクローズ
          INVOKE W-READER "Close"

*         DBクローズ
          INVOKE W-CONN "Close"

*         メッセージ設定
          IF R-ReturnCode > 1 THEN
              MOVE MSG-DSP(R-ReturnCode) TO P-Msg
          END-IF
*     例外処理
      CATCH W-EXCEPTION
*         例外メッセージを戻り値に格納
          SET P-Msg TO PROP-MESSAGE OF W-EXCEPTION
          MOVE 99 TO R-ReturnCode
*     終了処理
      FINALLY
*         コマンド終了処理
          IF W-CMD NOT = NULL
              INVOKE W-CMD "Dispose"
          END-IF
*         コネクション終了処理
          IF W-CONN NOT = NULL
            INVOKE W-CONN "Dispose"
          END-IF
      END-TRY.
 END PROGRAM SYOHIN.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ ZAIKO.COB

```
*************************************************************
* プログラム名 : ZAIKO
* 概要         : ZaikoClassら入出庫明細を取得する。
*
*************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. ZAIKO.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
     CLASS ZAIKO-CLASS          AS "ZaikoClass"
     CLASS CLASS-Exception      AS "System.Exception"

     PROPERTY PROP-MESSAGE AS "Message".
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 W-ZAIKO OBJECT REFERENCE ZAIKO-CLASS.
 01 W-EXCEPTION  OBJECT REFERENCE CLASS-Exception.
 LINKAGE SECTION.
 COPY IODETAIL.
 01 P-Start                  BINARY-SHORT.
 01 P-Cnt                    PIC 9(03).
 01 P-Next                   PIC 9(01).
 01 P-Msg                    PIC N(20).
 01 R-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION USING
                     M-AREA        *>明細領域
                     P-Start       *>開始行
                     P-Cnt         *>件数
                     P-Next        *>次のレコードがあるかどうか
                                   *>(ある場合は1)
                     P-Msg         *>メッセージ
                     RETURNING
                     R-ReturnCode. *>戻り値(エラーコード)
     TRY
         INVOKE ZAIKO-CLASS "NEW" RETURNING W-ZAIKO
         INVOKE W-ZAIKO "INIT-SEC"
           USING M-AREA P-Cnt
           RETURNING R-ReturnCode
         IF R-ReturnCode = 0 THEN
           INVOKE W-ZAIKO "MAIN-SEC"
             USING M-AREA P-Start P-Cnt P-Next
             RETURNING R-ReturnCode
         END-IF
         INVOKE W-ZAIKO "END-SEC" USING P-Msg R-ReturnCode
*    例外処理
     CATCH
*        例外メッセージを戻り値に格納
         SET P-Msg TO PROP-MESSAGE OF W-EXCEPTION
         MOVE 99 TO R-ReturnCode
*    終了処理
     FINALLY
         INVOKE W-ZAIKO "Dispose"
     END-TRY
 END PROGRAM ZAIKO.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ ZAIKOCLASS.COB

```
*************************************************************
* クラス名      : ZaikoClass
* 概要          : 入出庫テーブルから入出庫明細を取得する。
*
*************************************************************
 IDENTIFICATION DIVISION.
 CLASS-ID. ZAIKO-CLASS AS "ZaikoClass".
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 REPOSITORY.
     CLASS CLASS-ConfigurationManager     AS
          "System.Configuration.ConfigurationManager"
     CLASS CLASS-ConnectionStringSettings AS
          "System.Configuration.ConnectionStringSettings"
     CLASS CLASS-String        AS "System.String"
     CLASS CLASS-DbProviderFactories      AS
          "System.Data.Common.DbProviderFactories"
     CLASS CLASS-DbProviderFactory        AS
          "System.Data.Common.DbProviderFactory"
     CLASS CLASS-DbConnection             AS
          "System.Data.Common.DbConnection"
     CLASS CLASS-DbCommand                AS
          "System.Data.Common.DbCommand"
     CLASS CLASS-DbParameter    AS
          "System.Data.Common.DbParameter"
     CLASS CALSS-DbDataReader             AS
          "System.Data.Common.DbDataReader"
     CLASS CLASS-Exception                AS
          "System.Exception"
     ENUM ENUM-DbType         AS "System.Data.DbType"
     INTERFACE INTRFACE-IDisposable       AS
          "System.IDisposable"

     PROPERTY PROP-PARAMETERS             AS "Parameters"
     PROPERTY PROP-VALUE                  AS "Value"
     PROPERTY PROP-MESSAGE                AS "Message"
     PROPERTY PROP-CONNECTIONSTRINGS      AS
             "ConnectionStrings"
     PROPERTY PROP-CONNECTIONSTRING       AS
             "ConnectionString"
     PROPERTY PROP-PROVIDERNAME AS "ProviderName"
     PROPERTY PROP-COMMANDTEXT AS "CommandText"
     PROPERTY PROP-INT32 AS "Int32"
     PROPERTY PROP-PARAMETERNAME AS "ParameterName"
     PROPERTY PROP-DBTYPE AS "DbType".

 OBJECT. IMPLEMENTS INTRFACE-IDisposable.
 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.
 01 SYOIOPF-REC.
   02  IODATE     PIC S9(9) COMP-5.
   02  SYOCD      PIC S9(9) COMP-5.
   02  NODATE     PIC S9(9) COMP-5.
   02  SYUBAN     PIC S9(9) COMP-5.
   02  TUBAN      PIC S9(9) COMP-5.
   02  KUBUN      PIC S9(9) COMP-5.
   02  IOSU       PIC S9(9) COMP-5.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
 01  S-SIODATE    PIC S9(9) COMP-5.
 01  S-EIODATE    PIC S9(9) COMP-5.
 01  S-SYOCD      PIC S9(9) COMP-5.

 01  W-IODATE     PIC 9(8).

 01  W-DENPYO.
   03  W-SYUBAN   PIC 9(8).
   03  W-TUBAN    PIC 9(4).

*---コードテーブル----------------
   01  S-CODE.
     03  S-TANKA                PIC  X(4)  VALUE "0001".
     03  S-GENKA                PIC  X(4)  VALUE "0002".
     03  S-MUKA                 PIC  X(4)  VALUE "0003".
     03  S-NYUKO                PIC  X(4)  VALUE "0001".
     03  S-SYUKO                PIC  X(4)  VALUE "0002".
*
     03  S-TANGEN               PIC  X(4)  VALUE "0000".
     03  S-NYUSYU               PIC  X(4)  VALUE "0000".

*---伝票区分コードテーブル--------
   01  KUBUN-AREA.
*    01～10 **
     03  FILLER                 PIC  N(20) VALUE
            NC"                                        ".
*    11～20 **
     03  FILLER                 PIC  N(20) VALUE
            NC"売上                          売訂    ".
*    21～30 **
     03  FILLER                 PIC  N(20) VALUE
            NC"売返                          売訂    ".
*    31～40 **
     03  FILLER                 PIC  N(20) VALUE
            NC"仕入送仕状仕加仕浮借浮仕委仕値増仕訂    ".
*    41～50 **
     03  FILLER                 PIC  N(20) VALUE
            NC"仕返送返状返加返浮返浮戻委戻値引仕訂    ".
*    51～60 **
     03  FILLER                 PIC  N(20) VALUE
            NC"貸出持出社出預出持越展出振出他出出訂    ".
*    61～70 **
     03  FILLER                 PIC  N(20) VALUE
            NC"貸戻持戻社戻預戻持戻展戻振入他入入訂    ".
*    70～80 **
     03  FILLER                 PIC  N(20) VALUE
            NC"返出振出                                ".
*    81～90 **
     03  FILLER                 PIC  N(20) VALUE
            NC"返入振入                      他入入訂".
*    91～100**
     03  FILLER                 PIC  N(20) VALUE
            NC"                                        ".
   01 KUBUN-TBL REDEFINES  KUBUN-AREA.
     03  KUBUN-ID          OCCURS 100 TIMES.
       05  FILLER               PIC  N(02).
   01  FILE-STS-AREA.
     03  FILE-STS1              PIC  X(02).
     03  FILE-STS2              PIC  X(06).
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*---フラグ関連--------------------
   01  FLG-AREA.
     03  END-FLG                  PIC  X(03).
     03  MEISAI-FLG               PIC  X(03).

*---プログラム作業領域------
   01  W-AREA.
     03  W-KEY.
       05  W-Y                    PIC  9(04).
       05  W-M                    PIC  9(02).
       05  W-SYOCD                PIC  9(05).
     03  W-IOYMD.
       05  W-IOY                  PIC  9(04).
       05  W-IOM                  PIC  9(02).
       05  W-IOD                  PIC  9(02).
     03  W-RECCNT                 PIC  9(03).

 01  LINEMAX                   PIC 9(03) VALUE 10.

 01  W-CONN       OBJECT REFERENCE CLASS-DbConnection.
 01  W-CMD        OBJECT REFERENCE CLASS-DbCommand .
 01  W-READER     OBJECT REFERENCE CALSS-DbDataReader.

 PROCEDURE DIVISION.

*-------------------------------------------------------------
 METHOD-ID. INIT-SEC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  W-CONNSTR    OBJECT REFERENCE CLASS-
ConnectionStringSettings.
 01  W-PROVIDER   OBJECT REFERENCE CLASS-String.
 01  W-FACTORY    OBJECT REFERENCE CLASS-DbProviderFactory.
 01  W-DBPARAM    OBJECT REFERENCE CLASS-DbParameter.
 01  W-DBTYPEOBJECT REFERENCE ENUM-DbType.
 01  W-EXCEPTION  OBJECT REFERENCE CLASS-Exception.
 LINKAGE SECTION.
 COPY IODETAIL.
 01 P-Cnt                    PIC 9(03).
 01 R-ReturnCode             BINARY-LONG.
 PROCEDURE DIVISION USING M-AREA P-Cnt RETURNING R-ReturnCode.
     MOVE 0 TO R-ReturnCode
     MOVE SPACE TO MEISAI-FLG.
     MOVE SPACE TO END-FLG.

     MOVE 0 TO W-RECCNT.
     MOVE 0 TO P-Cnt.

*---入出庫レコードの検索条件------
     MOVE    M-SYOCD  TO    S-SYOCD.
     MOVE    M-Y      TO    W-IOY.
     MOVE    M-M      TO    W-IOM.
     MOVE    1        TO    W-IOD.
     MOVE    W-IOYMD  TO    W-IODATE.
     MOVE    W-IODATE TO    S-SIODATE.
     MOVE    31       TO    W-IOD.
     MOVE    W-IOYMD  TO    W-IODATE.
     MOVE    W-IODATE TO    S-EIODATE.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
     TRY
*        接続文字列取得
         INVOKE PROP-CONNECTIONSTRINGS OF
           CLASS-ConfigurationManager "get_Item"
             USING N"cobdb" RETURNING W-CONNSTR
*        プロバイダファクトリ取得
         SET W-PROVIDER TO PROP-PROVIDERNAME OF W-CONNSTR
         INVOKE CLASS-DbProviderFactories "GetFactory"
           USING W-PROVIDER RETURNING W-FACTORY
*        コネクション生成
         INVOKE W-FACTORY "CreateConnection" RETURNING W-CONN
         SET PROP-CONNECTIONSTRING OF W-CONN TO
             PROP-CONNECTIONSTRING OF W-CONNSTR
*        コマンド生成
         INVOKE W-CONN "CreateCommand" RETURNING W-CMD
         SET PROP-COMMANDTEXT OF W-CMD TO
           "SELECT * FROM SYOIOPF WHERE (IODATE BETWEEN
            @SDATE AND @EDATE) AND SYOCD = @SYOCD"
*        パラメータ作成
         INVOKE W-CMD "CreateParameter" RETURNING W-DBPARAM
         SET PROP-PARAMETERNAME OF W-DBPARAM TO N"@SDATE"
         SET PROP-DBTYPE OF W-DBPARAM TO
             PROP-INT32 OF ENUM-DbType
         SET PROP-VALUE OF W-DBPARAM TO S-SIODATE
         INVOKE PROP-PARAMETERS OF W-CMD "Add" USING W-DBPARAM

         INVOKE W-CMD "CreateParameter" RETURNING W-DBPARAM
         SET PROP-PARAMETERNAME OF W-DBPARAM TO N"@EDATE"
         SET PROP-DBTYPE OF W-DBPARAM TO
             PROP-INT32 OF ENUM-DbType
         SET PROP-VALUE OF W-DBPARAM TO S-EIODATE
         INVOKE PROP-PARAMETERS OF W-CMD "Add" USING W-DBPARAM

         INVOKE W-CMD "CreateParameter" RETURNING W-DBPARAM
         SET PROP-PARAMETERNAME OF W-DBPARAM TO N"@SYOCD"
         SET PROP-DBTYPE OF W-DBPARAM TO
             PROP-INT32 OF ENUM-DbType
         SET PROP-VALUE OF W-DBPARAM TO S-SYOCD
         INVOKE PROP-PARAMETERS OF W-CMD "Add" USING W-DBPARAM

*        DBオープン
         INVOKE W-CONN "Open"
*   例外を設定
     CATCH W-EXCEPTION
*        データベースにアクセスできない場合
         MOVE 9 TO R-ReturnCode
     END-TRY.
 END METHOD INIT-SEC.
*-------------------------------------------------------------
 METHOD-ID. END-SEC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 COPY MSGTBL.
 LINKAGE SECTION.
 01 P-Msg                      PIC N(20).
 01 P-ReturnCode               BINARY-LONG.
 PROCEDURE DIVISION USING P-Msg P-ReturnCode.
*    DBクローズ
     INVOKE W-CONN "Close".
*    メッセージ設定
     IF P-ReturnCode > 1 THEN
         MOVE MSG-DSP(P-ReturnCode) TO P-Msg
     END-IF.
 END METHOD END-SEC.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*-------------------------------------------------------------
 METHOD-ID. READ-SEC.
 DATA DIVISION.
 PROCEDURE DIVISION.
     IF W-READER::"Read" = B"1" THEN
         COMPUTE W-RECCNT = W-RECCNT + 1
         INVOKE W-READER "GetInt32"   USING 0 RETURNING IODATE
         INVOKE W-READER "GetInt32"   USING 1 RETURNING SYOCD
         INVOKE W-READER "GetInt32"   USING 2 RETURNING NODATE
         INVOKE W-READER "GetInt32"   USING 3 RETURNING SYUBAN
         INVOKE W-READER "GetInt32"   USING 4 RETURNING TUBAN
         INVOKE W-READER "GetInt32"   USING 5 RETURNING KUBUN
         INVOKE W-READER "GetInt32"   USING 6 RETURNING IOSU
     ELSE
         MOVE "END" TO MEISAI-FLG
     END-IF.
 END METHOD READ-SEC.

*-------------------------------------------------------------
 METHOD-ID. MAIN-SEC.
 DATA DIVISION.
 LINKAGE SECTION.
 COPY IODETAIL.
 01 P-Start                  BINARY-SHORT.
 01 P-Cnt                    PIC 9(03).
 01 P-Next                   PIC 9(01).
 01 R-ReturnCode              BINARY-LONG.
 PROCEDURE DIVISION USING M-AREA P-Start P-Cnt P-Next
RETURNING R-ReturnCode.
     INVOKE W-CMD "ExecuteReader" RETURNING W-READER.
     INVOKE SELF "READ-SEC"
*    ループ
     PERFORM UNTIL (END-FLG = "ERR") OR (MEISAI-FLG = "END")
OR (MEISAI-FLG = "MAX")
         IF W-RECCNT >= P-Start THEN
             COMPUTE P-Cnt = P-Cnt + 1
             INVOKE SELF "CODE-SEC" USING M-AREA P-Cnt
         END-IF
*        明細行数を越えたら終了
         IF P-Cnt >= LINEMAX THEN
             MOVE "MAX" TO MEISAI-FLG
*            最後に1件読んで次の行があるか確認する
         END-IF
         INVOKE SELF "READ-SEC"
     END-PERFORM.
     INVOKE W-READER "Close".

     IF MEISAI-FLG = "MAX" THEN
         MOVE 1 TO P-Next
     END-IF
     IF MEISAI-FLG = "END" AND P-Cnt = 0 THEN
         *>商品コードはあったが、該当する年月のレコードなし
         MOVE 8 TO R-ReturnCode
     END-IF

 END METHOD MAIN-SEC.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
*-------------------------------------------------------------
 METHOD-ID. CODE-SEC.
 DATA DIVISION.
 LINKAGE SECTION.
 COPY IODETAIL.
 01 P-Cnt                      PIC 9(03).
 PROCEDURE DIVISION USING M-AREA P-Cnt.
     EVALUATE KUBUN OF SYOIOPF-REC

     WHEN    ZERO
         CONTINUE

*    *** 納品系処理区分（１１～２９）
     WHEN    11  THRU  19
         EVALUATE   KUBUN OF SYOIOPF-REC
             WHEN    11
                 MOVE    S-TANKA  TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
             WHEN    19
                 MOVE    S-MUKA   TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
         END-EVALUATE

     WHEN    21  THRU  29
         EVALUATE    KUBUN OF SYOIOPF-REC
             WHEN    21
                 MOVE    S-TANKA  TO    S-TANGEN
                 MOVE    S-NYUKO  TO    S-NYUSYU
             WHEN    OTHER
                 MOVE    S-MUKA   TO    S-TANGEN
                 MOVE    S-NYUKO  TO    S-NYUSYU
         END-EVALUATE

*    ***仕入系処理区分（３１～４９）
     WHEN    31  THRU  39
         EVALUATE   KUBUN OF SYOIOPF-REC
             WHEN    39
                 MOVE    S-MUKA   TO    S-TANGEN
                 MOVE    S-NYUKO  TO    S-NYUSYU
             WHEN    OTHER
                 MOVE    S-GENKA  TO    S-TANGEN
                 MOVE    S-NYUKO  TO    S-NYUSYU
             END-EVALUATE

     WHEN    41  THRU  49
         EVALUATE    KUBUN OF SYOIOPF-REC
             WHEN    49
                 MOVE    S-MUKA   TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
             WHEN    OTHER
                 MOVE    S-GENKA  TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
         END-EVALUATE

*    *** 社内良品系処理区分（５１～６９）
     WHEN    51  THRU  59
         EVALUATE    KUBUN OF SYOIOPF-REC
             WHEN    51
                 MOVE    S-TANKA  TO    S-TANGEN
                 MOVE    S-SYUKO  TO    S-NYUSYU
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
            WHEN    52
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    59
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE
*
    WHEN    61  THRU  69
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    61
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    62
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    69
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
        END-EVALUATE

*   ***社外不良品系処理区分（７１～８９）
    WHEN    71  THRU  79
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    71  THRU  72
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    79
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-SYUKO  TO    S-NYUSYU
        END-EVALUATE
*
    WHEN    81  THRU  89
        EVALUATE    KUBUN OF SYOIOPF-REC
            WHEN    81  THRU  82
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    89
                MOVE    S-MUKA   TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
            WHEN    OTHER
                MOVE    S-TANKA  TO    S-TANGEN
                MOVE    S-NYUKO  TO    S-NYUSYU
        END-EVALUATE
    END-EVALUATE

    MOVE IODATE OF SYOIOPF-REC           TO M-NODATE(P-Cnt)
    MOVE SYUBAN OF SYOIOPF-REC           TO W-SYUBAN
    MOVE TUBAN OF SYOIOPF-REC            TO W-TUBAN
    MOVE W-DENPYO                        TO M-DENBAN(P-Cnt)
    MOVE KUBUN OF SYOIOPF-REC            TO M-KUBUN(P-Cnt)
    MOVE KUBUN-ID(KUBUN OF SYOIOPF-REC)  TO M-KUBUNN(P-Cnt)
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

```
     EVALUATE S-TANGEN
     WHEN S-TANKA
         MOVE M-TANKA  TO M-KAKAKU(P-Cnt)
     WHEN S-GENKA
         MOVE M-GENKA  TO M-KAKAKU(P-Cnt)
     WHEN S-MUKA
         MOVE 0        TO M-KAKAKU(P-Cnt)
     END-EVALUATE

     IF S-NYUSYU = S-NYUKO THEN
         MOVE IOSU OF SYOIOPF-REC TO M-ISU(P-Cnt)
     ELSE
         MOVE IOSU OF SYOIOPF-REC TO M-OSU(P-Cnt)
     END-IF
 END METHOD CODE-SEC.

*-------------------------------------------------------------
 METHOD-ID. DISPOSE AS "Dispose".
 PROCEDURE DIVISION.
*    コマンド終了処理
     IF W-CMD NOT = NULL
         INVOKE W-CMD "Dispose"
     END-IF
*    コネクション終了処理
     IF W-CONN NOT = NULL
         INVOKE W-CONN "Dispose"
     END-IF
 END METHOD DISPOSE.

 END OBJECT.
 END CLASS ZAIKO-CLASS.
```


All Rights Reserved, Copyright FUJITSU LIMITED 2006

■ **NetCOBOL for .NET の体験版について**

NetCOBOL ホームページ

http://software.fujitsu.com/jp/cobol/

「体験版」より、お申し込みください。

■ **本書に関するお問い合わせ先**

NetCOBOL ホームページ

http://software.fujitsu.com/jp/cobol/

「ご意見・ご質問」より、お願いします。

※　ご質問に関しては、記述箇所のページ、内容を明示してください。

※　ご質問の内容によっては、回答に数日またはそれ以上の期間を要する場合があります。

All Rights Reserved, Copyright FUJITSU LIMITED 2006

NetCOBOL

N e t C O B O L