



B1WW-7011-01Z0(00)

Microsoft® Windows® 98
Microsoft® Windows® Me
Microsoft® Windows NT®

Microsoft® Windows® 2000
Microsoft® Windows® XP
Microsoft® Windows Server™ 2003

NetCOBOL for Windows V8.0

例題プログラム



NetCOBOL

FUJITSU

まえがき

製品の呼び名について

本書に記載されている製品の名称を、以下のように略して表記します。

- 「Microsoft(R) Windows(R) 98 operating system」
→ 「Windows(R) 98」
- 「Microsoft(R) Windows(R) Millennium Edition」
→ 「Windows(R) Me」
- 「Microsoft(R) Windows NT(R) Workstation operating system Version 4.0」
→ 「Windows NT(R)」または、「Windows NT(R) 4.0」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0」
→ 「Windows NT(R)」、「Windows NT(R) 4.0」または、「Windows NT(R) Server」
- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0, Terminal Server Edition」
→ 「Windows NT(R)」、「Windows NT(R) 4.0」または、「Windows NT(R) 4.0 T.S.E」
- 「Microsoft(R) Windows NT(R) Server Network operating system, Enterprise Edition Version 4.0」
→ 「Windows NT(R)」、「Windows NT(R) 4.0」または、「Windows NT(R) Server」
- 「Microsoft(R) Windows(R) 2000 Professional operating system」
→ 「Windows(R) 2000」または、「Windows(R) 2000 Professional」
- 「Microsoft(R) Windows(R) 2000 Server operating system」
→ 「Windows(R) 2000」または、「Windows(R) 2000 Server」
- 「Microsoft(R) Windows(R) 2000 Advanced Server operating system」
→ 「Windows(R) 2000」または、「Windows(R) 2000 Server」
- 「Microsoft(R) Windows(R) XP Professional operating system」
→ 「Windows(R) XP」または、「Windows(R) XP Professional」
- 「Microsoft(R) Windows(R) XP Home Edition operating system」
→ 「Windows(R) XP」または、「Windows(R) XP Home Edition」
- 「Microsoft(R) Windows Server(TM) 2003, Standard Edition」
→ 「Windows Server(TM) 2003」
- 「Microsoft(R) Windows Server(TM) 2003, Enterprise Edition」
→ 「Windows Server(TM) 2003」
- 「Windows(R) 98」、「Windows(R) Me」、「Windows NT(R)」、「Windows(R) 2000」、「Windows(R) XP」および「Windows Server(TM) 2003」
→ 「Windows(R)」
- 「Microsoft(R) Visual C++(R) development system」
→ 「Visual C++(R)」
- 「Microsoft(R) Visual Basic(R) programming system」
→ 「Visual Basic(R)」

本書の目的

本書は、NetCOBOLに添付されたサンプルプログラムの概要および翻訳から実行について説明しています。

本書の読者

本書は、NetCOBOLを使用してCOBOLプログラムを開発する方を対象に書かれています。本書を読むためには、以下の知識が必要です。

- COBOLの文法に関する基本的な知識
- ご使用になるOSに関する基本的な知識

本書の位置付け

本書の関連マニュアルには、以下のマニュアルがあります。

- “COBOL 文法書”
- “NetCOBOL 使用手引書”
- “Web連携ガイド”
- “COBOL Webサブルーチン使用手引書”

注意事項

本書の情報は、プログラミングサービス情報です。COBOLを使用してプログラムを開発する際に使用することができます。

その他の注意事項

- 本書に記載されている画面は、使用されているシステムにより、画面が異なる場合がありますので注意してください。
- 本書では、“COBOL文法書”で“原始プログラム”と記述されている用語を“ソースプログラム”と記述しています。

登録商標について

本書に記載されている登録商標を、以下に示します。

Microsoft、Windows、Windows NT、Windows Server、Visual C++、Visual Basic、ActiveXは、米国Microsoft Corporationの米国およびその他の国における商標または登録商標です。

その他の会社名または製品名は、それぞれ各社の商標または登録商標です。

Microsoft Corporationのガイドラインに従って画面写真を使用しています。

2005年 6月

お願い

- | |
|--|
| <ul style="list-style-type: none">● 本書を無断で他に転載しないようお願いします。● 本書は予告なしに変更されることがあります。 |
|--|

All Rights Reserved, Copyright(C) 富士通株式会社 1992-2005

目次

例題プログラム.....	1
例題1 標準入出力を使ったデータ処理.....	2
例題2 行順ファイルと索引ファイルの操作.....	6
例題3 表示ファイル機能を使ったプログラム.....	10
例題4 スクリーン操作機能を使った画面入出力.....	15
例題5 COBOLプログラム間の呼出し.....	18
例題6 コマンド行引数の受取り方.....	23
例題7 環境変数の操作.....	26
例題8 印刷ファイルを使ったプログラム.....	29
例題9 印刷ファイルを使ったプログラム（応用編）.....	31
例題10 FORMAT句付き印刷ファイルを使ったプログラム.....	34
例題11 データベース機能を使ったプログラム.....	39
例題12 データベース機能を使ったプログラム（応用編）.....	43
例題13 Visual Basic(R)からの呼出し.....	48
例題14 Visual Basic(R)を使った簡易ATM端末処理機能.....	51
例題15 オブジェクト指向プログラム（初級編）.....	57
例題16 コレクションクラス（クラスライブラリ）.....	60
例題17 オブジェクト指向プログラム（中級編）.....	68
例題18 オブジェクト指向プログラム（上級編）.....	78
例題19 オブジェクトの永続化（ファイル）.....	86
例題20 オブジェクトの永続化（データベース）.....	89
例題21 マルチスレッドプログラミング.....	94
例題22 マルチスレッドプログラミング（応用編）.....	103
例題23 COM連携-Excelを操作するプログラム（1）.....	115
例題24 COM連携-Excelを操作するプログラム（2）.....	118
例題25 COM連携-COBOLによるCOMサーバプログラムの作成.....	122
例題26 COM連携-COBOLサーバプログラムの使用（COBOLクライアント）.....	128
例題27 COM連携-COBOLサーバプログラムの使用（ASPクライアント）.....	134
例題28 COM連携-MTSによるトランザクション管理をするプログラム.....	141
例題29 簡易アプリ間通信機能を使ったメッセージ通信.....	147
例題30 Unicodeを使用するプログラム.....	152
例題31 メッセージボックスの出力.....	156
例題32 他のプログラムの起動.....	159
COBOL Webサブルーチンの例題プログラム.....	163
例題 CGIサブルーチンを使ったプログラム.....	164
例題 ISAPIサブルーチンを使ったプログラム.....	166
例題 SAFサブルーチンを使ったプログラム.....	175
例題 セッション管理機能を使ったプログラム.....	184
索引.....	189

例題プログラム

NetCOBOLでは、以下のプログラムをサンプルとして提供しています。

- [例題1 標準入出力を使ったデータ処理](#)
 - [例題2 行順ファイルと索引ファイルの操作](#)
 - [例題3 表示ファイル機能を使ったプログラム](#)
 - [例題4 スクリーン操作機能を使った画面入出力](#)
 - [例題5 COBOLプログラム間の呼出し](#)
 - [例題6 コマンド行引数の受取り方](#)
 - [例題7 環境変数の操作](#)
 - [例題8 印刷ファイルを使ったプログラム](#)
 - [例題9 印刷ファイルを使ったプログラム\(応用編\)](#)
 - [例題10 FORMAT句付き印刷ファイルを使ったプログラム](#)
 - [例題11 データベース機能を使ったプログラム](#)
 - [例題12 データベース機能を使ったプログラム\(応用編\)](#)
 - [例題13 Visual Basic\(R\)からの呼出し](#)
 - [例題14 Visual Basic\(R\)を使った簡易ATM端末処理機能](#)
 - [例題15 オブジェクト指向プログラム\(初級編\)](#)
 - [例題16 コレクションクラス\(クラスライブラリ\)](#)
 - [例題17 オブジェクト指向プログラム\(中級編\)](#)
 - [例題18 オブジェクト指向プログラム\(上級編\)](#)
 - [例題19 オブジェクトの永続化\(ファイル\)](#)
 - [例題20 オブジェクトの永続化\(データベース\)](#)
 - [例題21 マルチスレッドプログラミング](#)
 - [例題22 マルチスレッドプログラミング\(応用編\)](#)
 - [例題23 COM連携-Excelを操作するプログラム\(1\)](#)
 - [例題24 COM連携-Excelを操作するプログラム\(2\)](#)
 - [例題25 COM連携-COBOLによるCOMサーバプログラムの作成](#)
 - [例題26 COM連携-COBOLサーバプログラムの使用\(COBOLクライアント\)](#)
 - [例題27 COM連携-COBOLサーバプログラムの使用\(ASPクライアント\)](#)
 - [例題28 COM連携-MTSによるトランザクション管理をするプログラム](#)
 - [例題29 簡易アプリ間通信機能を使ったメッセージ通信](#)
 - [例題30 Unicodeを使用するプログラム](#)
 - [例題31 メッセージボックスの出力](#)
 - [例題32 他のプログラムの起動](#)
-

例題1 標準入出力を使ったデータ処理

ここでは、本製品で提供するサンプルプログラム-例題1-について説明します。

例題1では、COBOLの小入出力機能を使って、コンソールウィンドウからデータを入力したり、コンソールウィンドウにデータを出力するプログラムの例を示します。小入出力機能の使い方の詳細は、“NetCOBOL 使用手引書”の“11.1 小入出力機能”を参照してください。

概要

コンソールウィンドウからアルファベット1文字(小文字)を入力し、入力したアルファベットで始まる単語をコンソールウィンドウに出力します。

提供プログラム

- SAMPLE1. COB (COBOLソースプログラム)
- SAMPLE1. SVD (デバッグ情報ファイル)
- DBGSAMP1. EXE (デバッガ用実行可能プログラム)
- SAMPLE1. TXT (プログラム説明書)

使用しているCOBOLの機能

- 小入出力機能(コンソールウィンドウ)
- 対話型デバッガ

使用しているCOBOLの文

ACCEPT文、DISPLAY文、EXIT文、IF文、PERFORM文

プログラムの翻訳・リンク・実行

プログラムの翻訳

[参照] “NetCOBOL 使用手引書”の“3.1 サンプルプログラムの翻訳”

プログラムのリンク

[参照] “NetCOBOL 使用手引書”の“4.1 サンプルプログラムのリンク”

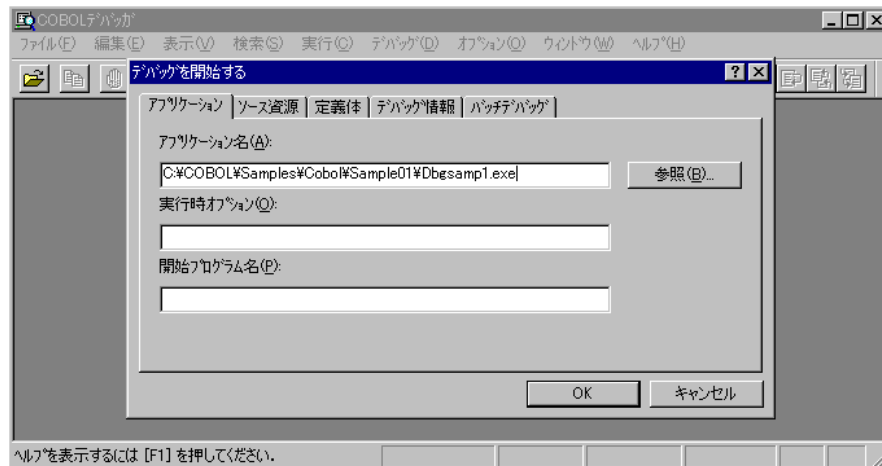
プログラムの実行

[参照] “NetCOBOL 使用手引書”の“5.1 サンプルプログラムの実行”

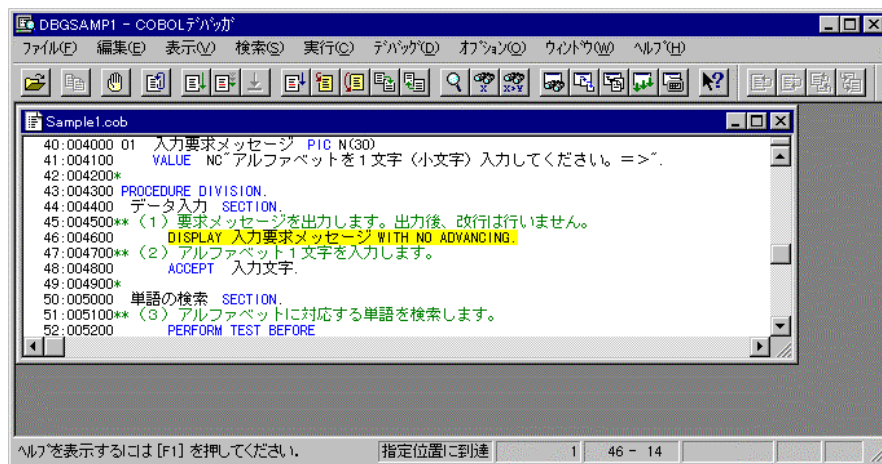
プログラムのデバッグ

例題1では、デバッグ情報ファイルと、デバッグオプションを指定して作成した実行可能プログラム(DBGSAMP1. EXE)を提供しています。ここでは、デバッガを使って例題1を起動してみましよう。

1. プロジェクトマネージャウィンドウの[ツール]メニューから“デバッグ”を選択します。
→ デバッガのメインウィンドウが表示されます。
2. メニューバーの[ファイル]メニューから“デバッグを開始する”を選択します。
→ [デバッグを開始する]ダイアログが表示されます。



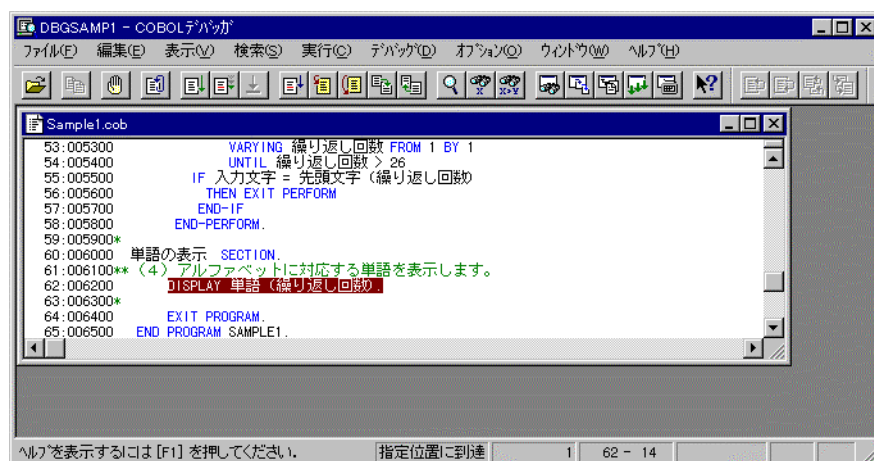
3. アプリケーション名のエディットボックスに実行可能プログラム名 (DBGSAMP1.EXE) を指定し、[OK] ボタンをクリックします。
→ メインウィンドウがアクティブになります。



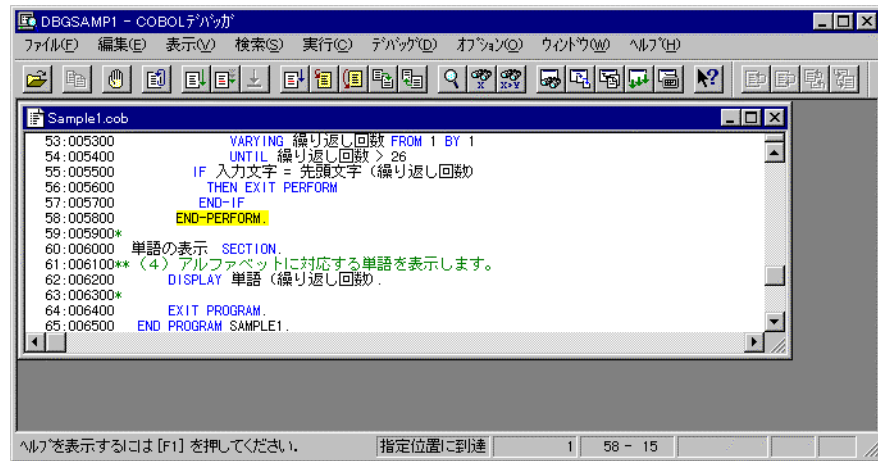
4. データ名 “入力文字” の内容を変更してみましょう。
- データ名 “入力文字” を選択します。
 - メニューバーの [デバッグ] メニューから “データの表示/変更” を選択します。
→ [データの表示/変更] ダイアログが表示されます。



- c) データ値領域でデータの値を変更し、[変更] ボタンをクリックしてください。
[データの表示/変更] ダイアログの詳細については、ヘルプを参照してください。
- d) [閉じる] ボタンをクリックして [データの表示/変更] ダイアログを終了します。
5. 中断点を設定してみましょう。
 - a) ソースファイルウィンドウ内の中断したい文にカーソルを位置付けて、ツールバーの [中断点を設定/解除] ボタンをクリックします。ここでは、62行目のDISPLAY文に中断点を設定します。
→ 中断点を設定し、文に色を付けます。



- b) 中断点を解除する場合は、解除したい中断点が設定されている文にカーソルを位置付けて、ツールバーの [中断点を設定/解除] ボタンをクリックします。
6. 指定した位置までプログラムを実行してみましょう。
 - a) ソースファイルウィンドウ内の中断させたい文にカーソルを位置付けて、マウスの右ボタンをクリックします。
→ ショートカットメニューが表示されます。
 - b) ショートカットメニューから “カーソル位置まで実行” を選択します。
→ プログラムが指定した文まで実行され、文には中断位置を示す色が付きます。



7. デバッグを終了するには、[ファイル] メニューの“デバッガの終了”を選択してください。

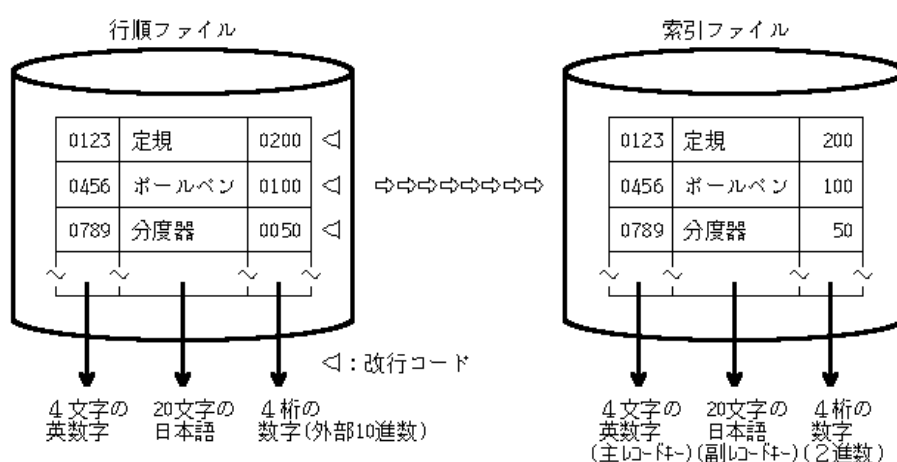
例題2 行順ファイルと索引ファイルの操作

ここでは、本製品で提供するサンプルプログラム-例題2-について説明します。

例題2では、エディタを使って作成したデータファイル(行順ファイル)を読み込み、マスタファイル(索引ファイル)を作成するプログラムの例を示します。行順ファイルおよび索引ファイルの使い方の詳細は、“NetCOBOL 使用手引書”の“第7章 ファイルの処理”を参照してください。例題2で作成したマスタファイル(索引ファイル)は、例題3、例題5、例題7および例題10で入力ファイルとして使用します。

概要

商品コード、商品名および単価が入力されているデータファイル(行順ファイル)を読み込み、商品コードを主レコードキー、商品名を副レコードキーとする索引ファイルを作成します。



提供プログラム

- SAMPLE2. COB (COBOLソースプログラム)
- SYOHINM. CBL (登録集原文)
- DATAFILE (データファイル)
- SAMPLE2. TXT (プログラム説明書)

使用しているCOBOLの機能

- 行順ファイル(参照)
- 索引ファイル(創成)

使用しているCOBOLの文

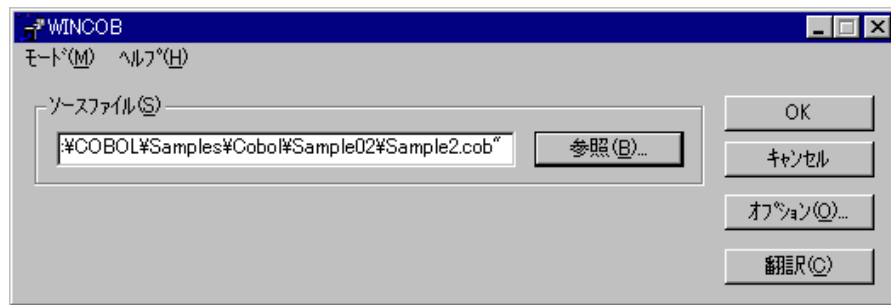
CLOSE文、EXIT文、GO TO文、MOVE文、OPEN文、READ文、WRITE文、COPY文

プログラムの翻訳・リンク・実行

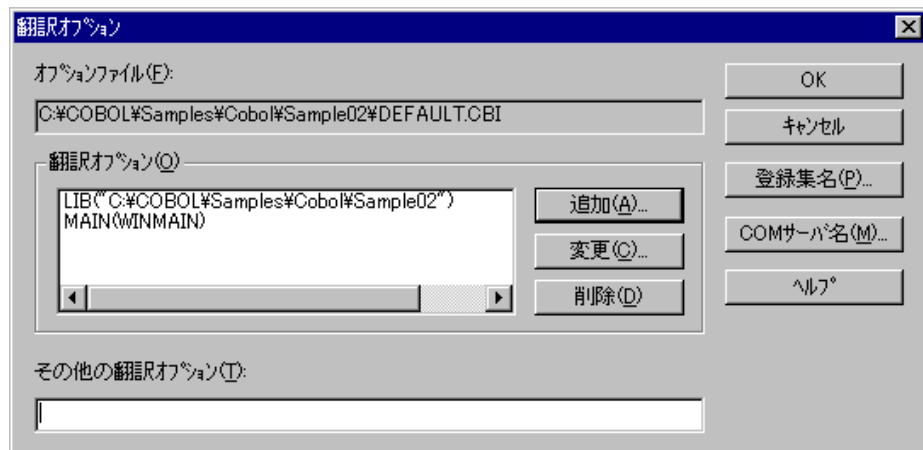
以降では、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLになっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

プログラムの翻訳

1. プロジェクトマネージャの「ツール」メニューから「翻訳」を選択します。
→ WINCOBウインドウが表示されます。



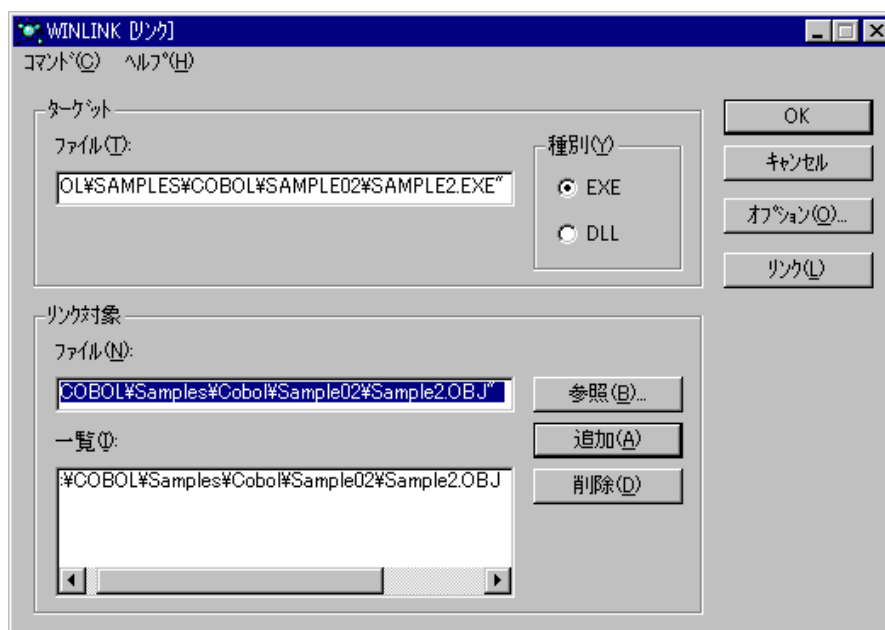
2. ソースファイルのエディットボックスにソースファイル名 (SAMPLE2. COB) を指定します。
3. WINCOBウィンドウの「オプション」ボタンをクリックします。
→ 「翻訳オプション」ダイアログが表示されます。



4. 翻訳オプションMAIN (WINMAIN) を指定します。また、翻訳オプションLIBに登録集ファイル (SYOHINM. CBL) のフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ WINCOBウィンドウに戻ります。
5. WINCOBウィンドウの「翻訳」ボタンをクリックします。
→ 翻訳が開始されます。翻訳終了後、オブジェクトファイル (SAMPLE2. OBJ) が作成されていることを確認してください。

プログラムのリンク

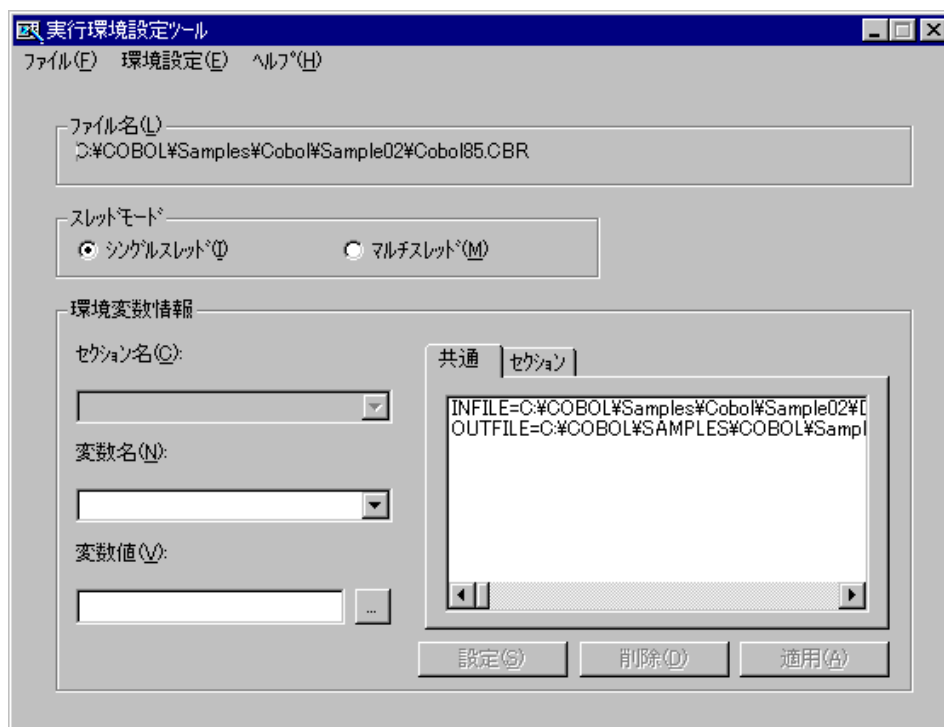
1. プロジェクトマネージャの「ツール」メニューから「リンク」を選択します。
→ WINLINKウィンドウが表示されます。



2. 必要な情報を設定します。
 - 種別に“EXE”を選択します。
 - ファイル(N)のエディットボックスに、オブジェクトファイル(SAMPLE2.OBJ)を指定します。
 - ファイル(T)のエディットボックスに、作成する実行可能プログラム(SAMPLE2.EXE)を指定します。
3. WINLINKウィンドウの〔OK〕ボタンをクリックします。
 - リンクが開始されます。リンク終了後、実行可能プログラム(SAMPLE2.EXE)が作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの〔ツール〕メニューから“実行環境設定ツール”を選択します。
 - 実行環境設定ツールが表示されます。



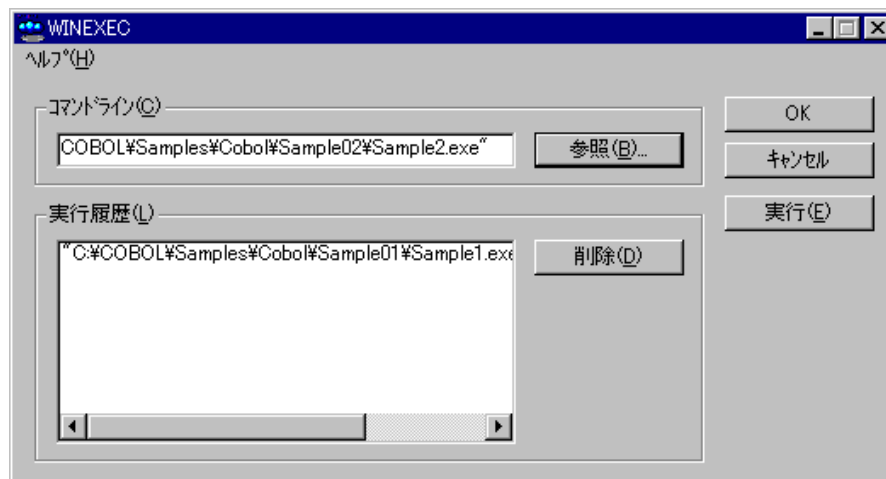
2. [ファイル] メニューの“開く”を選択し、実行可能プログラム(SAMPLE2.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - ファイル識別名INFILEに、データファイル(行順ファイル)のファイル名(DATAFILE)を指定します。
 - ファイル識別名OUTFILEに、マスタファイル(索引ファイル)のファイル名(MASTER)を指定します。
4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

備考

ファイル識別名のINFILEおよびOUTFILEは、COBOLソースプログラムのASSIGN句に指定されているファイル参照子です。ファイル参照子は、COBOLプログラムおよび実際の媒体(ファイル)を対応付けるために使用します。

プログラムの実行

1. プロジェクトマネージャの[ツール] メニューから“実行”を選択します。
→ WINEXECウィンドウが表示されます。



2. コマンドラインのエディットボックスに、実行可能プログラム(SAMPLE2.EXE)を指定し、[実行] ボタンをクリックします。
→ 実行の終了メッセージは、ウィンドウに表示されません。実行終了後、商品コードをキーとする索引ファイル(MASTER)が、例題2のフォルダに作成されます。

備考

実行履歴のリストボックスには、実行履歴が表示されます。例題2の実行には関係ありません。

例題3 表示ファイル機能を使ったプログラム

ここでは、本製品で提供するサンプルプログラム-例題3-について説明します。

例題3では、表示ファイル機能を使って、画面から入力したデータを画面に出力し、さらにデータを印刷装置に出力するプログラムの例を示します。画面入出力を行う場合の、表示ファイル機能の使い方は“NetCOBOL 使用手引書”の“9.2 表示ファイル(画面入出力)の使い方”を、印刷を行う場合の表示ファイル機能の使い方は、“NetCOBOL 使用手引書”の“8.5 表示ファイル(帳票印刷)の使い方”を参照してください。なお、このプログラムを実行するには、MeFtが必要です。

概要

ディスプレイ装置に表示された入力画面に商品コードおよび個数を入力すると、商品コードをキーにマスタファイルを検索し、商品名、単価および金額を求め、さらに合計金額を計算し、ディスプレイ装置に表示します。また、帳票を印刷装置に出力します。

例題3で入力したデータは、売上げファイル(索引ファイル)に格納します。売上げファイルは、例題10で入力ファイルとして使用します。

提供プログラム

- SAMPLE3. COB (COBOLソースプログラム)
- DENPYOU. COB (COBOLソースプログラム)
- URIAGE. CBL (登録集原文)
- DENPYO01. SMD (画面定義体)
- DENPYO02. SMD (帳票定義体)
- MEFWRC (ウィンドウ情報ファイル)
- MEFPRC (プリンタ情報ファイル)
- SAMPLE3. PRJ (プロジェクトファイル)
- SAMPLE3. CBI (翻訳オプションファイル)
- SAMPLE3. TXT (プログラム説明書)



注意

SYOHINM. CBL (登録集原文)は、例題2の提供ファイルを使用します。

使用しているCOBOLの機能

- 表示ファイル機能(画面入出力)
- 表示ファイル機能(帳票印刷)
- 索引ファイル(参照/創成)
- 登録集の取込み
- 小入出力機能(メッセージボックス)
- プロジェクト管理機能

使用しているCOBOLの文

OPEN文、READ文、WRITE文、CLOSE文、PERFORM文、DISPLAY文、IF文、MOVE文、GO TO文、EXIT文、COPY文

プログラムの翻訳・リンク・実行

プログラムを実行する前に

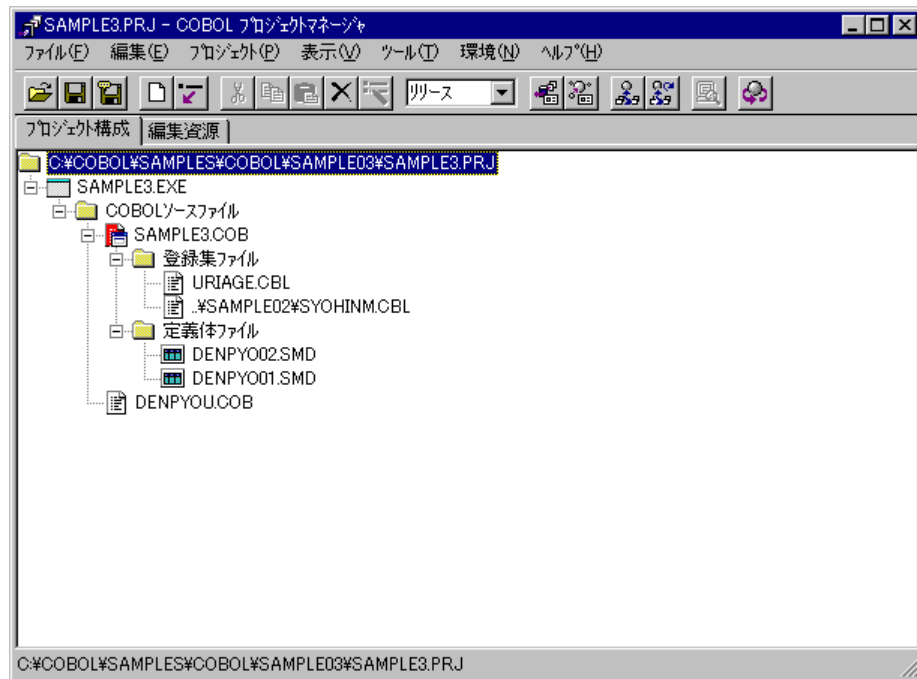
- MeFtのセットアップを行い、使用できる状態にしておいてください。
- 例題2で作成されるマスタファイルを使用するため、“[例題2 行順ファイルと索引ファイルの操作](#)”を実行しておきます。
- 例題3では、例題2で作成されるマスタファイルにある商品コードを入力しないと入力エラーとなるため、“[例題5 COBOLプログラム間の呼出し](#)”を実行し、入力データを印刷しておくと便利です。

ビルド・リビルド

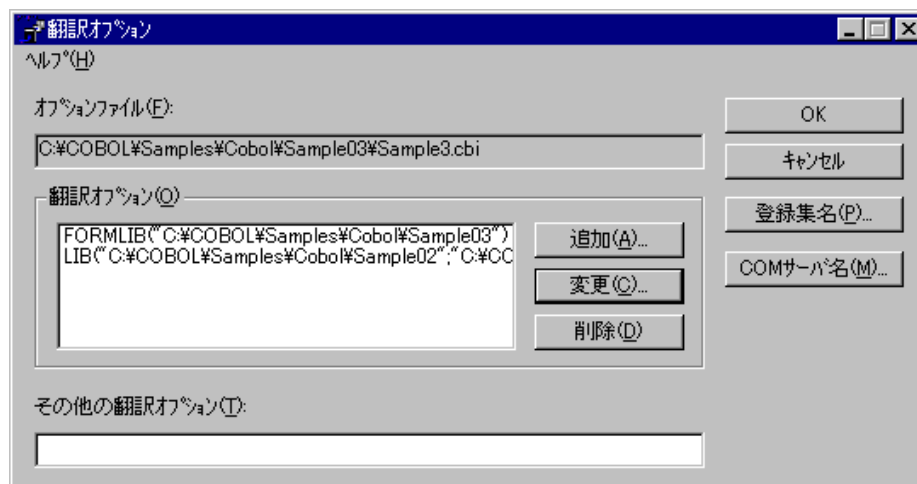
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE3.PRJ”を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションLIBに、SYOHINM.CBLが格納されたフォルダ(例題2のフォルダ)とURIAGE.CBLが格納されたフォルダを指定します。また、翻訳オプションFORMLIBに、DENPY01.SMD、DENPY02.SMDが格納されたフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの[プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE3.EXEが作成されていることを確認してください。

ウィンドウ情報ファイル・プリンタ情報ファイルの設定

表示ファイルを実行するために必要な情報を設定します。

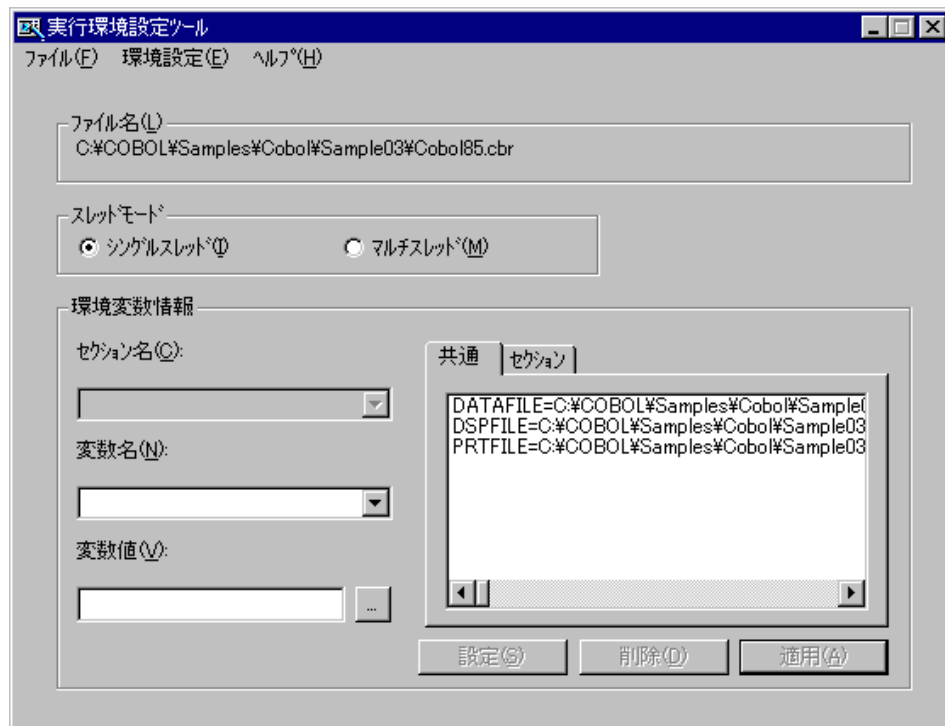
実行を行う前に、ウィンドウ情報ファイル(MEFWRC)およびプリンタ情報ファイル(MEFPRC)の下線部の情報をエディタを使って変更してください。

MEDDIR C:¥COBOL¥SAMPLES¥COBOL¥SAMPLE03

MEDDIR : 画面帳票定義体を格納したフォルダのパス名

実行環境情報の設定

- プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



- [ファイル] メニューの“開く”を選択し、実行可能プログラム(SAMPLE3.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
- 共通タブを選択し、以下を設定します。
 - ファイル識別名DATAFILEに、例題2で作成したマスタファイル(MASTER)を指定します。
 - ファイル識別名DSPFILEに、ウィンドウ情報ファイル(MEFWRC)を指定します。
 - ファイル識別名PRTPFILEに、プリンタ情報ファイル(MEFPRC)を指定します。
- [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
- [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

備考

ファイル識別名のDATAFILE、DSPFILEおよびPRTPFILEは、COBOLソースプログラムのASSIGN句に指定されているファイル参照子です。表示ファイルを使用する場合、ファイル識別名にはウィンドウ情報ファイルまたはプリンタ情報ファイルのパス名を設定します。

プログラムの実行

- プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
→ ディスプレイ装置に以下に示す画面が表示されます。

SAMPLE3

1999 . 4 . 2 (金)

コード	商品名	数量	単価	金額

PF1 : 計算 PF2 : 次の伝票を入力 合計

PF3 : 終了

2. 商品コードおよび数量を入力します。次の項目に移動するには、ENTERキーまたはタブキーを押します。

例題3では、例題2で作成されたマスタファイルにある商品コードを入力しないと入力エラーとなります。商品コードの値は、例題2のDATAFILEまたは例題5を実行し、マスタファイルの内容を参照してから入力してください。

SAMPLE3

1999 . 4 . 2 (金)

コード	商品名	数量	単価	金額
0123		100		
0456		50		
█				

PF1 : 計算 PF2 : 次の伝票を入力 合計

PF3 : 終了

入力を終了し、計算を行うには、F1キーを押します。
 → 商品名、単価、金額、合計金額が表示されます。エラーメッセージが表示された場合、メッセージの内容を確認し、再度入力してください。

SAMPLE3

内容を確認してください

1999 . 4 . 2 (金)

コード	商品名	数量	単価	金額
0123	定規	100	200	20000
0456	ボールペン	50	100	5000

PF1 : 計算 PF2 : 次の伝票を入力 合計 25000
PF3 : 終了 PF4 : 印刷

3. 続けて処理を行う場合は、F2キーを押してください。
4. 帳票の出力を行う場合は、F4キーを押してください。
→ 表示されているデータの内容が標準のプリンタに出力されます。
5. 処理を終了する場合は、F3キーを押してください。
→ 実行終了後、入力データを格納した索引ファイル(SALES)が、例題3のフォルダに作成されます。



注意

画面帳票定義体(DENPY001.SMD、DENPY002.SMD)をテキストエディタなどで開いて保存すると、ファイルの内容が破壊されることがあります。画面帳票定義体を更新する場合には、FORMのエディタ以外は使用しないでください。

例題4 スクリーン操作機能を使った画面入出力

ここでは、本製品で提供するサンプルプログラム-例題4-について説明します。

例題4では、スクリーン操作機能を使って、画面からデータを入力したり、画面にデータを入力するプログラムの例を示します。スクリーン操作機能の使い方の詳細は、“NetCOBOL 使用手引き”の“9.3 スクリーン操作機能の使い方”を参照してください。

概要

ディスプレイ画面から従業員番号および氏名を入力し、従業員番号を主レコードキー、氏名を副レコードキーとする索引ファイルを作成します。

提供プログラム

- SAMPLE4.COB (COBOLソースプログラム)
- SAMPLE4.PRJ (プロジェクトファイル)
- SAMPLE4.KBD (キー定義ファイル)
- SAMPLE4.TXT (プログラム説明書)

使用しているCOBOLの機能

- スクリーン操作機能
- 索引ファイル(参照)
- プロジェクト管理機能

使用しているCOBOLの文

ACCEPT文、CLOSE文、DISPLAY文、EXIT文、GO TO文、IF文、MOVE文、OPEN文、WRITE文

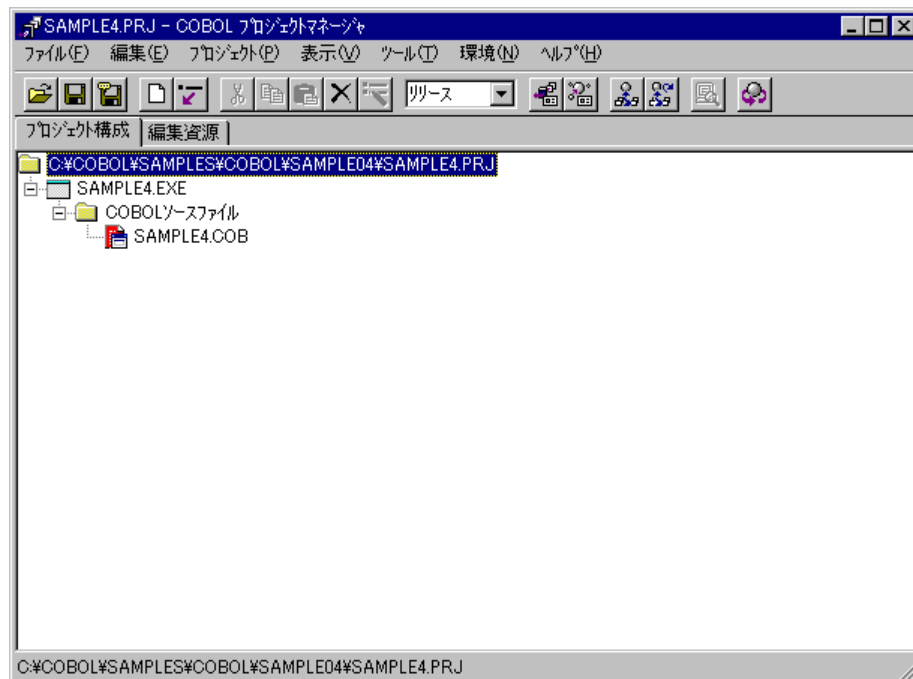
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

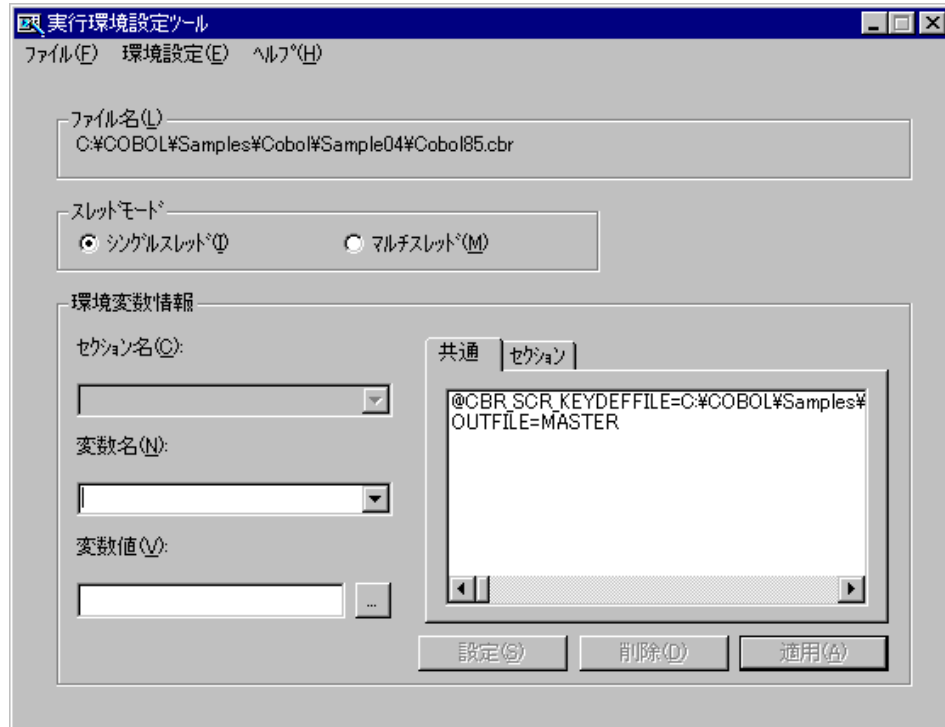
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE4.PRJ”を開きます。



3. プロジェクトマネージャの〔プロジェクト〕メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE4.EXEが作成されていることを確認してください。

実行環境情報の設定

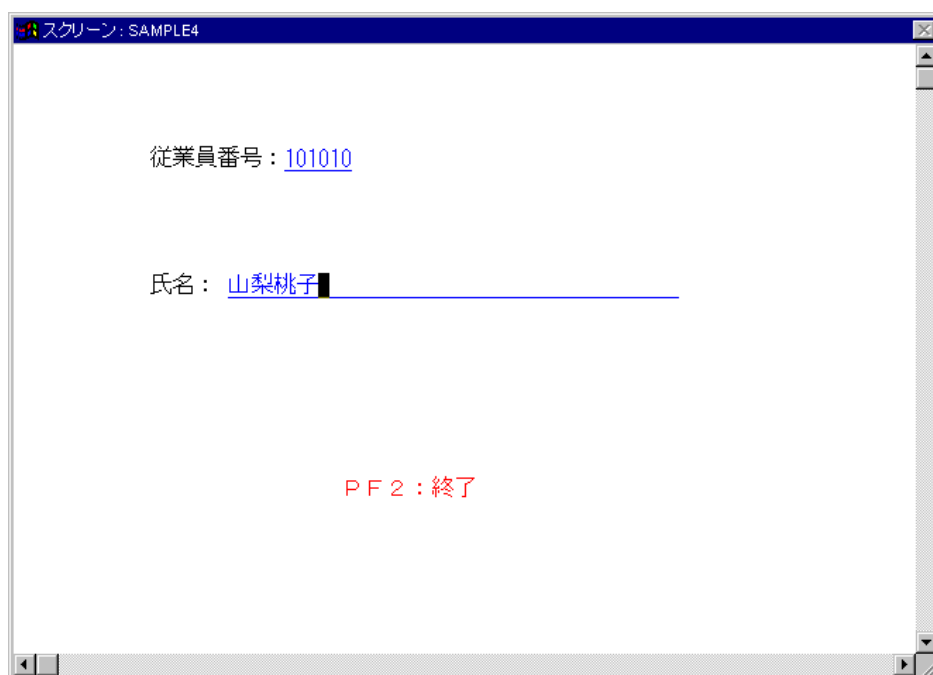
1. プロジェクトマネージャの〔ツール〕メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. 〔ファイル〕メニューの“開く”を選択し、実行可能プログラム(SAMPLE4.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - ファイル識別名OUTFILEに、マスタファイル名(MASTER)を指定します。
 - F2キーの入力だけが有効となるように設定されたキー定義ファイルを、環境変数情報@CBR_SCR_KEYDEFFILEに設定します。
4. 〔適用〕ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. 〔ファイル〕メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

1. プロジェクトマネージャの〔プロジェクト〕メニューから“実行”を選択します。
→ ディスプレイ装置に従業員番号および氏名を入力するための画面が表示されます。



2. 登録する従業員データとして、従業員番号(6桁の数字)および氏名(20文字以内の日本語文字)を入力します。ただし、従業員番号は昇順に入力してください。入力後、ENTERキーを押してください。
→ 設定した内容がマスタファイルに登録され、次の情報を入力するために画面がクリアされます。
3. 処理を終了する場合は、F2キーを押してください。
→ 実行終了後、従業員番号を主レコードキー、氏名を副レコードキーとする索引ファイル(MASTER)が、例題4のフォルダに作成されます。

例題5 COBOL プログラム間の呼出し

ここでは、本製品で提供するサンプルプログラム-例題5-について説明します。

例題5では、主プログラムから、副プログラムを呼び出すプログラムの例を示します。なお、例題5では、正書法の自由形式を使用して記述しています。

概要

商品コード、商品名および単価が格納されているマスタファイル(例題2で作成した索引ファイル)の内容を印刷可能文字に変換して作業用のテキストファイル(*.TMP)に格納し、印刷します。
 なお、作業用のファイルの名前は、プログラム実行時にパラメタで指定します。

提供プログラム

- SAMPLE5. COB (COBOL ソースプログラム)
- INSATSU. COB (COBOL ソースプログラム)
- S_REC. CBL (登録集原文)
- SAMPLE5. PRJ (プロジェクトファイル)
- SAMPLE5. CBI (翻訳オプションファイル)
- SAMPLE5. TXT (プログラム説明書)

使用しているCOBOLの機能

- プロジェクト管理機能
- プログラム間連絡機能
- 登録集の取込み
- 小入出力機能(メッセージボックス)
- 印刷ファイル
- 索引ファイル(参照)
- 行順ファイル(創成)
- 実行時パラメタの受渡し
- 正書法の自由形式

使用しているCOBOLの文

CALL文、DISPLAY文、EXIT文、GO TO文、MOVE文、OPEN文、READ文、WRITE文

プログラムの内容

ソースの記述(自由形式)

正書法の自由形式を使用してCOBOLソースプログラムを記述する例を以下に示します。

カラム

位置

1 -- | -- 10 -- | -- 20 -- | -- 30 -- | -- 40 -- | -- 50 -- | -- 60 -- | -- 70

IDENTIFICATION DIVISION.

PROGRAM-ID. SAMPLE5.

*> このサンプルプログラムは自由形式の正書法で記述されています。

*> 翻訳時には翻訳オプションSRFを使用して、正書法の形式として

*> 自由形式(FREE)を指定してください。

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SPECIAL-NAMES.

SYSERR IS メッセージ出力先.

:

DATA DIVISION.

FILE SECTION.


```

FD マスタファイル.
01 マスタレコード.
   02 商品レコード.
       03 商品コード      PIC X(4).
       03 商品名          PIC N(20).
       03 単価            PIC 9(4) BINARY.
       :
PROCEDURE DIVISION USING パラメタ.
*> (1) 作業ファイル名を決定します。
   IF パラメタ長 = 0
      DISPLAY NG"パラメタが指定されていません。"-
      "パラメタを指定してください。"
      UPON メッセージ出力先
      GO TO 処理終了.
   :
処理終了.
EXIT PROGRAM.
END PROGRAM SAMPLE5.

```

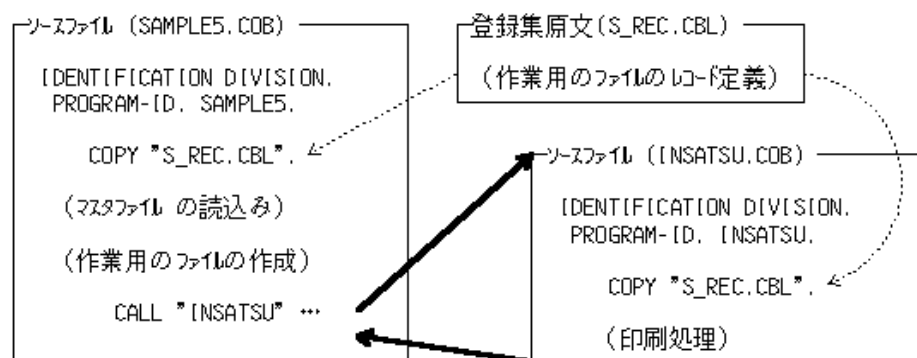
自由形式では、COBOLの文および注記は、行の任意の文字位置から書くことができます。行の最初の空白でない文字の並びが“*”である場合、その行は注記行とみなされます。また、上記の例のような方法で文字定数や日本語文字定数などを複数の行に分けて書くことができます。正書法の自由形式の詳細については、“COBOL文法書”を参照してください。

備考

翻訳時、COBOLソースプログラムおよび登録集の正書法の形式は、翻訳オプションSRFを用いて、それぞれ指定します。このため、1つのCOBOLソースプログラムに正書法の形式の異なる複数の登録集を取り込むことはできません。

また、画面帳票定義体を自由形式のCOBOLソースプログラムに取り込んで使用する場合、登録集の正書法の形式として可変形式を指定してください。[参照] “NetCOBOL 使用手引書”の“A.2.45 SRF（正書法の種類）”

ファイルの依存関係



プログラムの翻訳・リンク・実行

プログラムを実行する前に

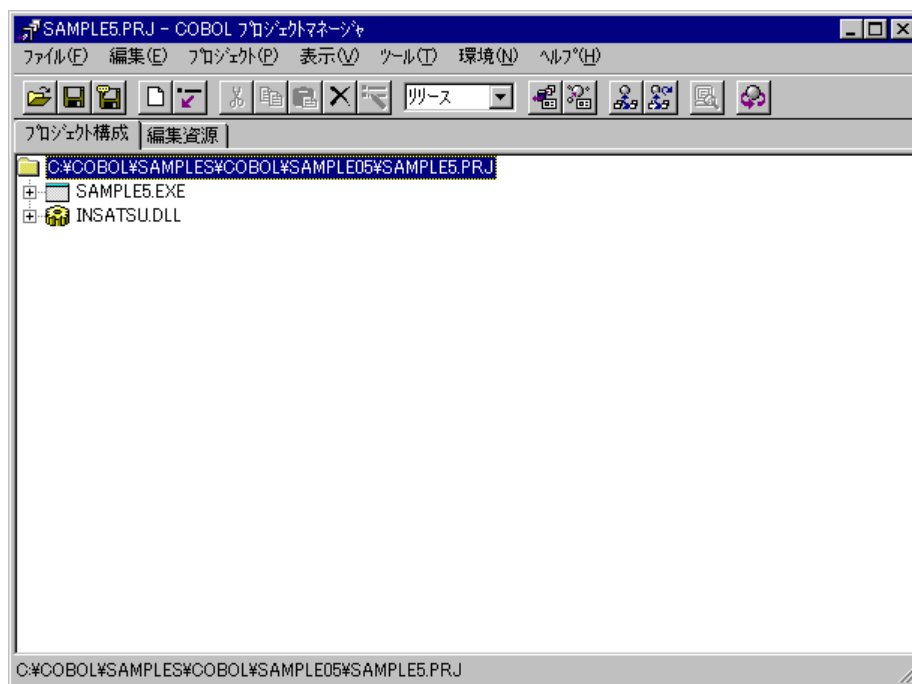
例題2で作成されるマスタファイルを使用するため、“[例題2 行順ファイルと索引ファイルの操作](#)”を実行しておきます。

ビルド・リビルド

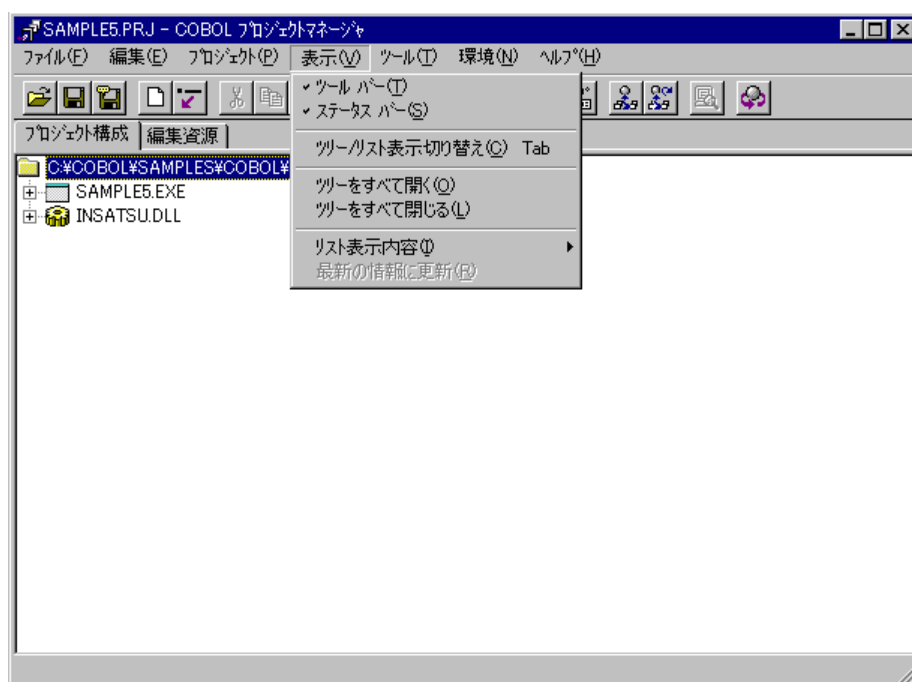
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトファイル(SAMPLE5.PRJ)をオープンします。
2. ターゲットファイルを確認します。このプロジェクトファイルは2つのターゲットファイルを含みます。プロジェクトファイル名の下に、2つのターゲットファイルの名前が表示されます。
 - SAMPLE5.EXE(作成する主プログラムの実行可能プログラム名)
 - INSATSU.DLL(作成する副プログラムのDLL名)

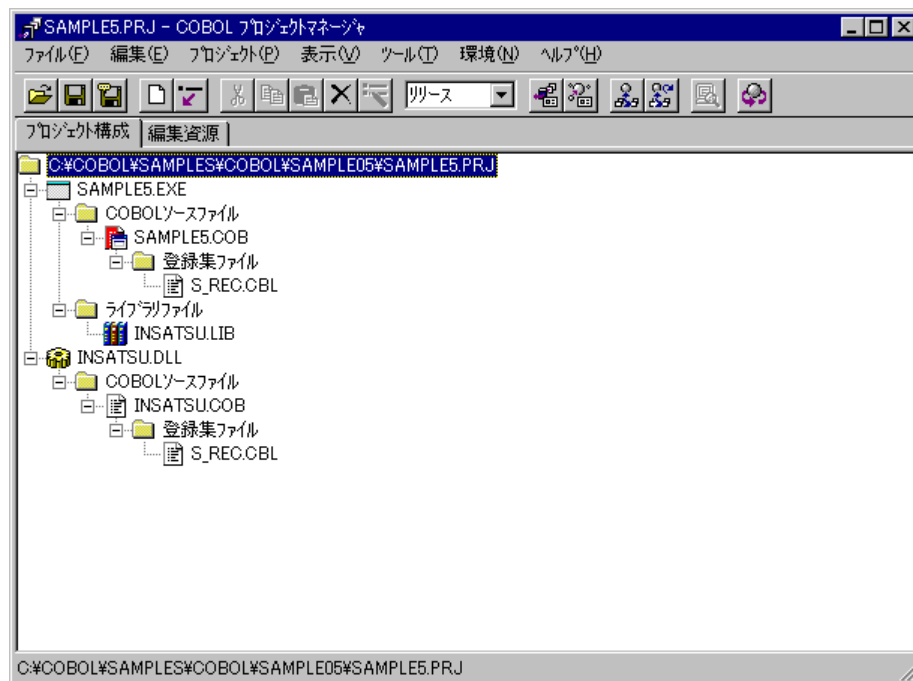


3. 依存ファイルの確認をします。
プロジェクトファイルを選択し、[表示] メニューから“すべて開く”を選択します。
→ 依存ファイルのフォルダが表示され、その下に依存ファイル名が表示されます。

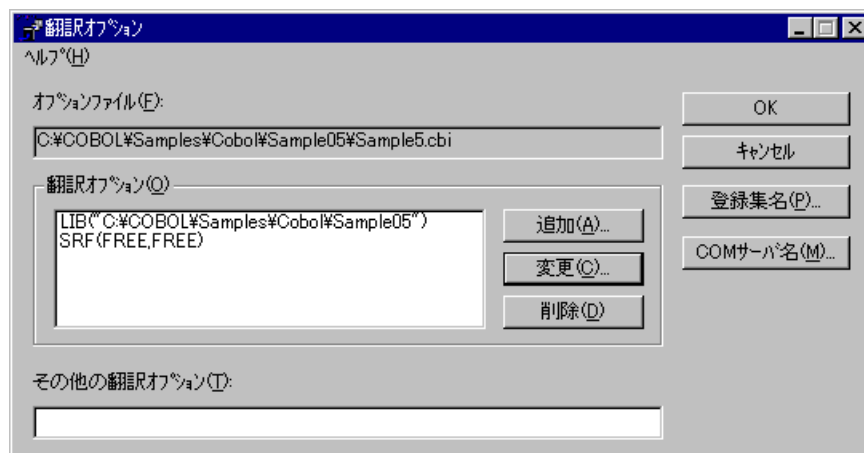


4. 主プログラムの確認をします。

主プログラムとなるCOBOLソース(SAMPLE5.COB)は、ウィンドウメイン型(赤いアイコン)で表示されます。



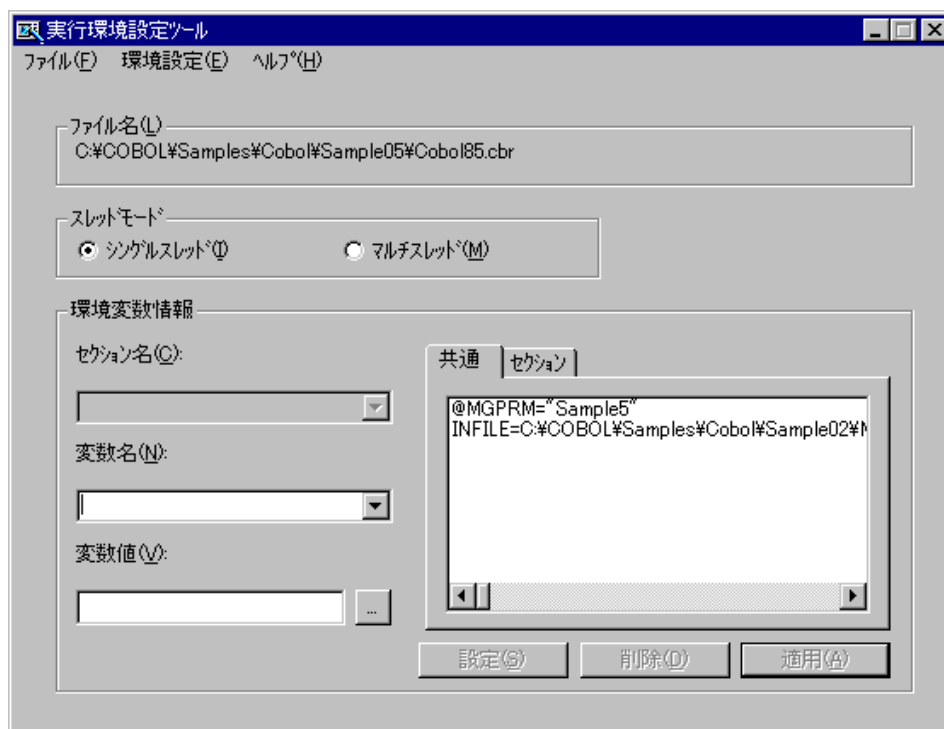
5. 翻訳オプションの確認をします。
プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



6. 翻訳オプションSRF(FREE, FREE)を指定します。また、翻訳オプションLIBに、S_REC.CBLが格納されているフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
7. プロジェクトマネージャの[プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE5.EXEおよびINSATSU.DLLが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. [ファイル] メニューの“開く”を選択し、実行可能プログラム(SAMPLE5. EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85. CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@MGPRM(実行時パラメタの指定)に、作業用ファイル名(sample5)を指定します(英数字8文字以内)。ここで指定した名前に拡張子TMPを付加した名前のファイルが作業用ファイルとして作成されます。
 - ファイル識別名INFILEに、例題2で作成したマスタファイル(MASTER)を指定します。
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
 → “作業ファイル(sample5. TMP)を作成します”というメッセージが表示されます。内容を確認したら、[OK] ボタンをクリックしてメッセージボックスを閉じてください。
 プログラムの実行が終了すると、マスタファイルの内容が“通常使うプリンタ”として設定されている印刷装置に出力されます。

例題6 コマンド行引数の受取り方

ここでは、本製品で提供するサンプルプログラム-例題6-について説明します。

例題6では、コマンド行引数の操作機能を使って、COBOLプログラムの実行時に指定された引数を受け取るプログラムの例を示します。コマンド行引数の操作機能の使い方は、“NetCOBOL 使用手引書”の“11.2 コマンド行引数の取出し”を参照してください。また、例題6では、内部プログラムの呼出しも行います。

概要

開始年月日から終了年月日までの日数を求めます。開始年月日および終了年月日は、コマンドの引数として次の形式で指定されます。

コマンド名 開始年月日 終了年月日

開始年月日および終了年月日は、1900年1月1日～2172年12月31日までの日付をYYYYMMDDで指定します。西暦については4桁で指定します。

提供プログラム

- SAMPLE6. COB (COBOLソースプログラム)
- SAMPLE6. PRJ (プロジェクトファイル)
- SAMPLE6. TXT (プログラム説明書)

使用しているCOBOLの機能

- コマンド行引数の取出し
- コンソール型のアプリケーションの作成
- 内部プログラム
- プロジェクト管理機能

使用しているCOBOLの文

ACCEPT文、CALL文、COMPUTE文、COPY文、DISPLAY文、DIVIDE文、EXIT文、GO TO文、IF文、MOVE文、PERFORM文

プログラムの翻訳・リンク・実行

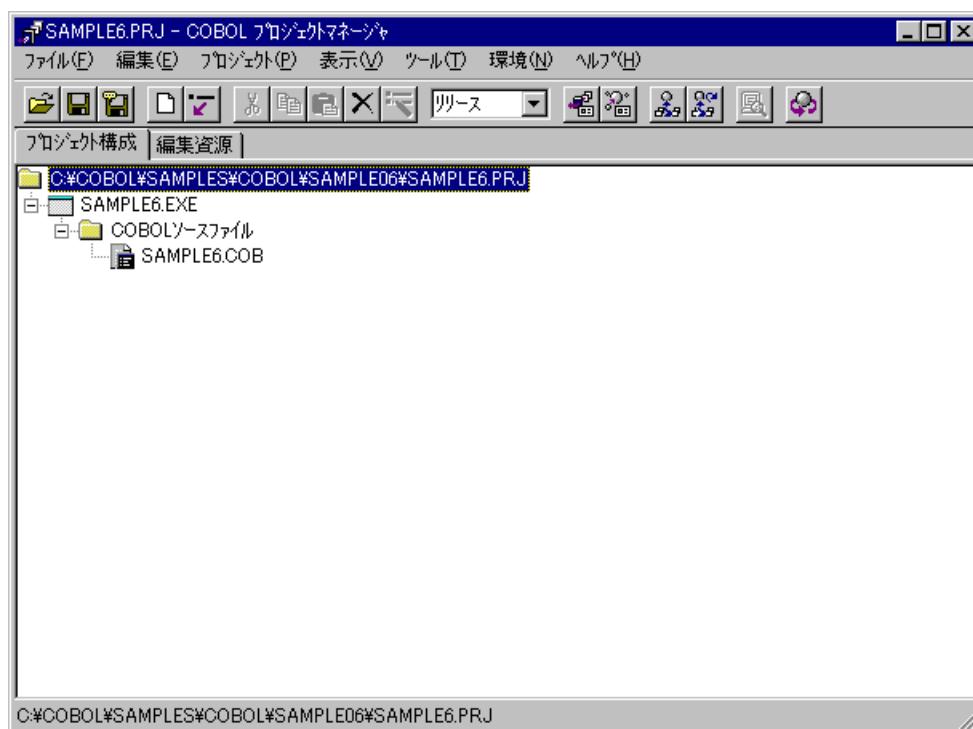
ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE6. PRJ”を開きます。
3. 主プログラムの確認をします。

主プログラムとなるCOBOLソース(SAMPLE6. COB)は、コンソール型(白黒のアイコン)で表示されます。

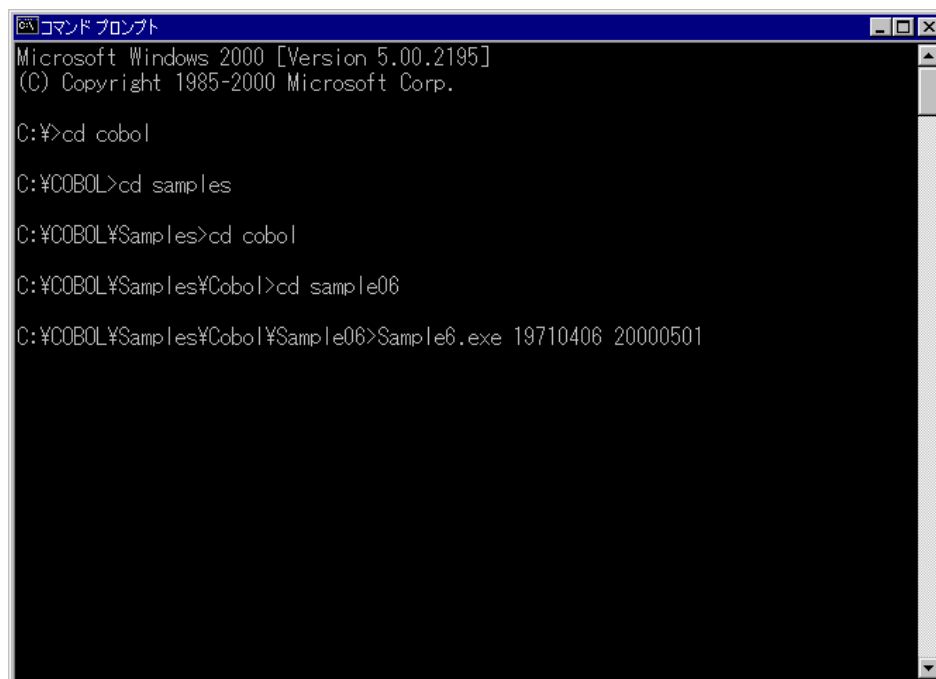


4. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE6.EXEが作成されていることを確認してください。

プログラムの実行

実行可能プログラム (SAMPLE6.EXE) の実行は、コマンドプロンプトから行います。

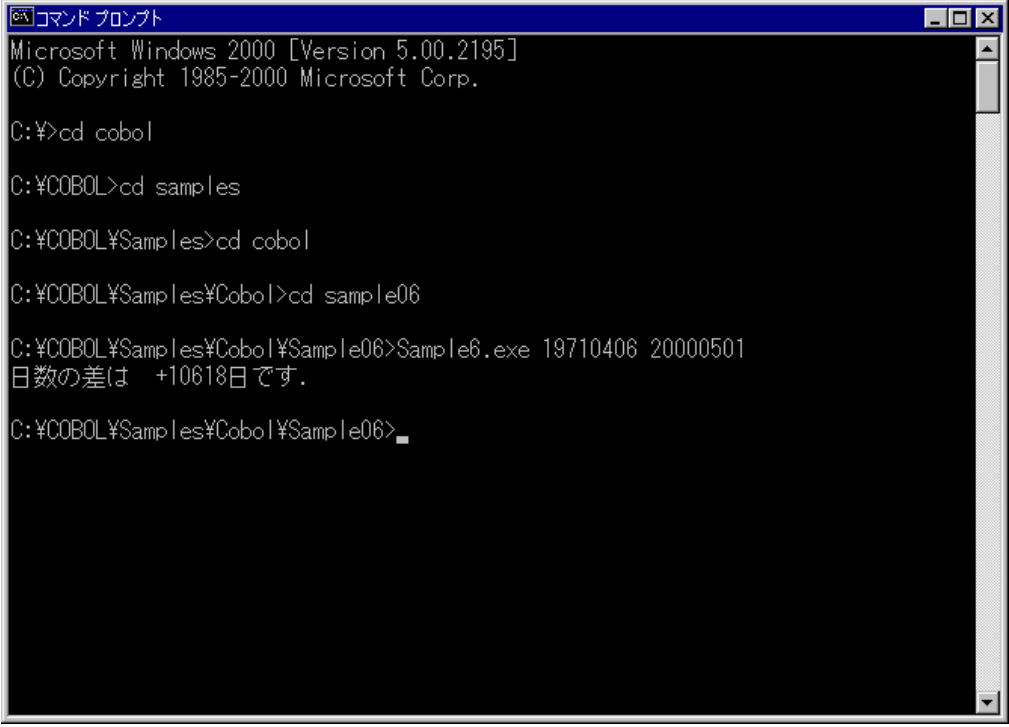
1. コマンドプロンプトを起動し、実行するファイル名 (SAMPLE6.EXE) が存在するフォルダまで移動します。
2. SAMPLE6.EXEを指定し、続けて開始年月日および終了年月日を入力します。



実行結果

コンソール型のアプリケーションでは、DISPLAY文による出力はCOBOLのコンソールウィンドウではなく、システムのコンソール(コマンドプロンプト)に出力されます。

次のように、1971年4月6日から2000年5月1日までの日数が表示されます。



```
コマンド プロンプト
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd cobol
C:\COBOL>cd samples
C:\COBOL\Samples>cd cobol
C:\COBOL\Samples\Cobol>cd sample06
C:\COBOL\Samples\Cobol\Sample06>Sample6.exe 19710406 20000501
日数の差は +10618日です。
C:\COBOL\Samples\Cobol\Sample06>
```

例題7 環境変数の操作

ここでは、本製品で提供するサンプルプログラム-例題7-について説明します。

例題7では、環境変数の操作機能を使って、COBOLプログラム実行中に環境変数の値を変更するプログラムの例を示します。環境変数の操作機能の使い方は、“NetCOBOL 使用手引書”の“11.3 環境変数の操作機能”を参照してください。

概要

商品コード、商品名および単価が格納されているマスタファイル(例題2で作成した索引ファイル)中のデータを、商品コードの値によって2つのマスタファイルに分割します。分割方法および新規に作成する2つのマスタファイルのファイル名を以下に示します。

商品コードの値	ファイル名
先頭が"0"	マスタファイル名.A
先頭が"0"以外	マスタファイル名.B

提供プログラム

- SAMPLE7.COB (COBOLソースプログラム)
- SAMPLE7.PRJ (プロジェクトファイル)
- SAMPLE7.CBI (翻訳オプションファイル)
- SAMPLE7.TXT (プログラム説明書)



注意

SYOHINM.CBL (登録集原文)は、例題2の提供ファイルを使用します。

使用しているCOBOLの機能

- 環境変数の操作機能
- 索引ファイル
- プロジェクト管理機能

使用しているCOBOLの文

ACCEPT文、CLOSE文、DISPLAY文、EXIT文、GO TO文、IF文、OPEN文、READ文、STRING文、WRITE文

プログラムの翻訳・リンク・実行

プログラムを実行する前に

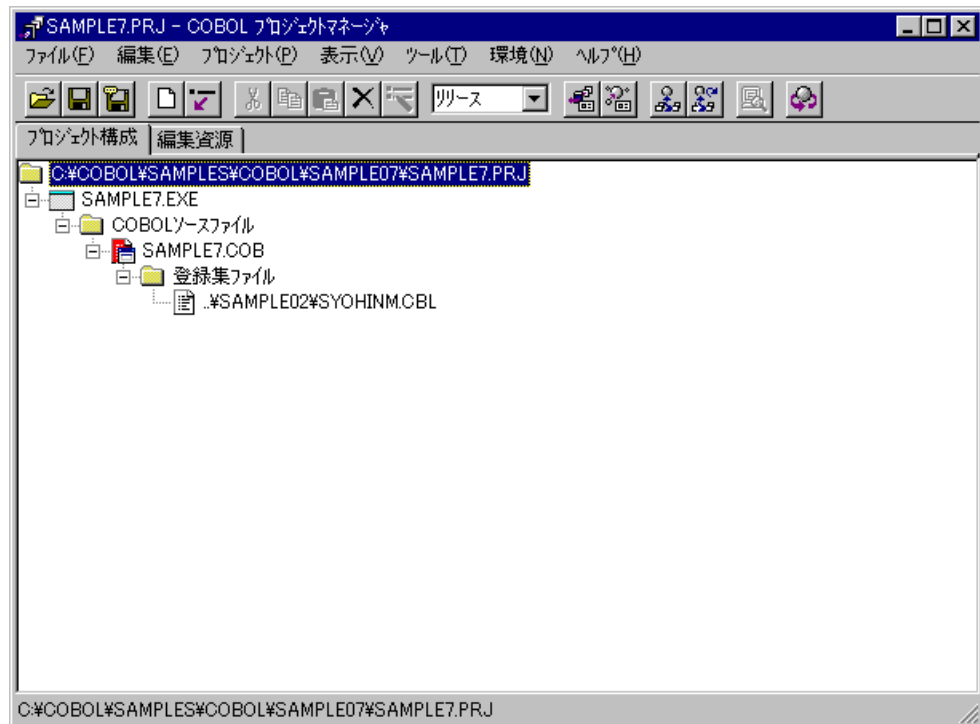
例題2で作成されるマスタファイルを使用するため、“[例題2 行順ファイルと索引ファイルの操作](#)” 実行しておきます。

ビルド・リビルド

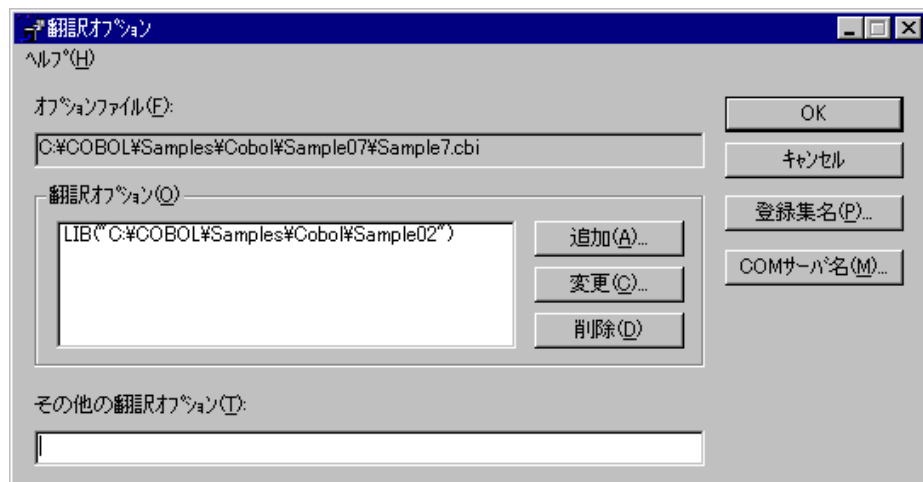
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE7.PRJ”を開きます。



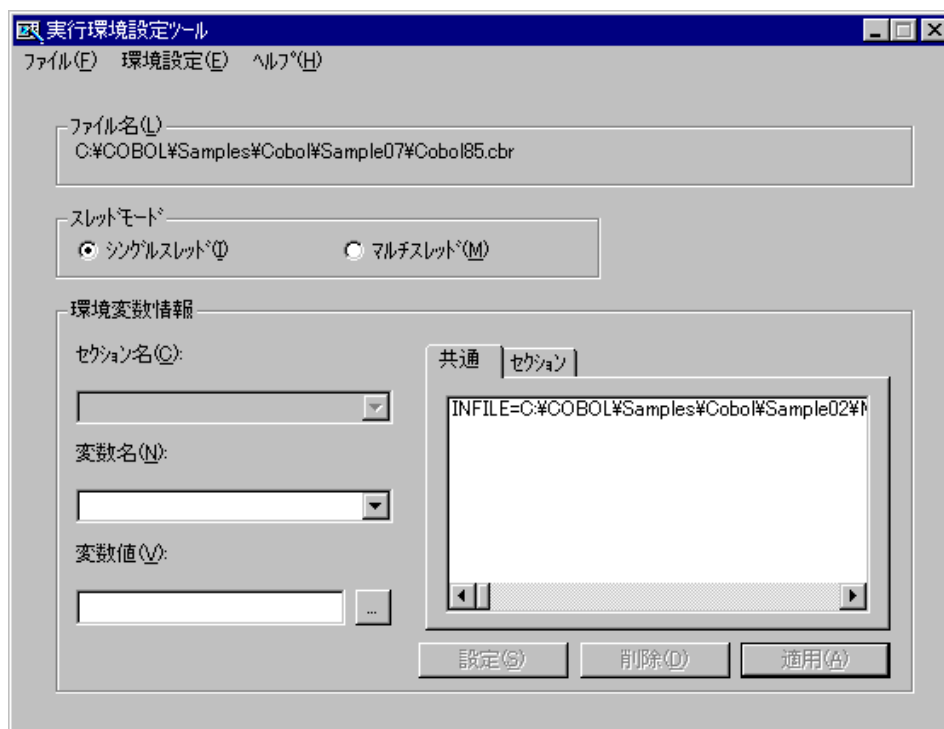
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションLIBに、SYOHINM.CBLが格納されたフォルダ(例題2のフォルダ)を指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE7.EXEが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの [ツール] メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. [ファイル] メニューの“開く”を選択し、実行可能プログラム(SAMPLE7. EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85. CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - ファイル識別名 INFILE に、例題2で作成したマスタファイル(MASTER)を指定します。
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

- プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
- マスタファイルと同じフォルダ(例題2のフォルダ)に次の2つのファイルが作成されます。
- MASTER. A(商品コードの先頭が“0”の商品のデータを格納したマスタファイル)
 - MASTER. B(商品コードの先頭が“0”以外の商品のデータを格納したマスタファイル)

備考

作成したマスタファイル(MASTER. AおよびMASTER. B)の内容は、例題2で作成したマスタファイルと同様に、“[例題5 COBOLプログラム間の呼出し](#)”を使って内容を確認することができます。

例題8 印刷ファイルを使ったプログラム

ここでは、本製品で提供するサンプルプログラム-例題8-について説明します。

例題8では、印刷ファイルを使って、コンソールウィンドウから入力したデータを印刷装置に出力するプログラムの例を示します。印刷ファイルの使い方の詳細は、“NetCOBOL 使用手引書”の“8.2 行単位のデータを印刷する方法”を参照してください。

概要

コンソールウィンドウから英数字のデータ40文字を入力し、印刷装置に出力します。

提供プログラム

- SAMPLE8.COB (COBOLソースプログラム)
- SAMPLE8.PRJ (プロジェクトファイル)
- SAMPLE8.TXT (プログラム説明書)

使用しているCOBOLの機能

- 印刷ファイル
- 小入出力機能(コンソールウィンドウ)
- プロジェクト管理機能

使用しているCOBOLの文

ACCEPT文、CLOSE文、EXIT文、GO TO文、IF文、OPEN文、WRITE文

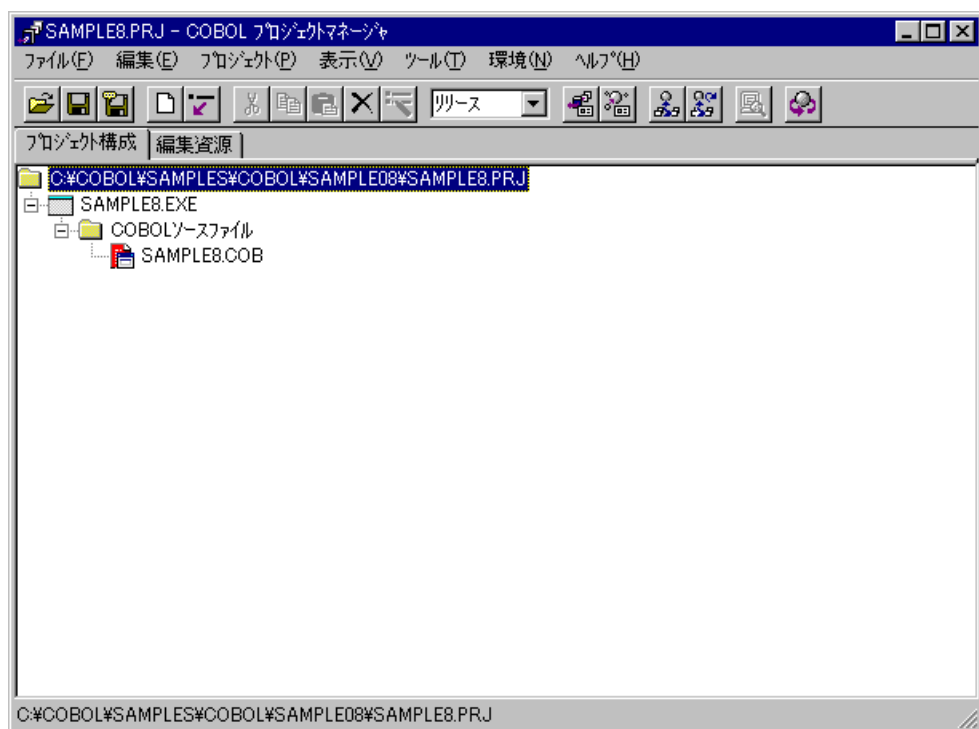
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE8.PRJ”を開きます。



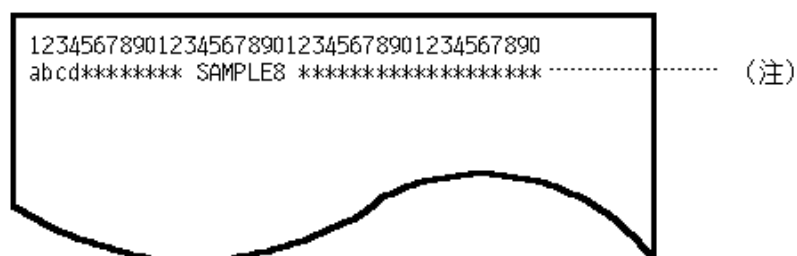
3. プロジェクトマネージャの〔プロジェクト〕メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE8.EXEが作成されていることを確認してください。

プログラムの実行

1. プロジェクトマネージャの〔プロジェクト〕メニューから“実行”を選択します。
→ コンソールウィンドウが表示されます。
2. コンソールウィンドウから、印刷するデータを入力します。1回のデータの入力は40文字以内です。たとえば、以下のようにデータを入力します。



3. データの入力を終了する場合、“/END”に続けて空白を36文字入力し、ENTERキーを押します。
→ プログラムが終了すると、入力したデータがプリンタに印刷されます。



注

コンソールウィンドウでの2回目の入力が40文字未満なので、2回目の入力データと3回目の入力データを合わせたデータが、プログラムでの2回目のACCEPT文の入力データとなります。

例題9 印刷ファイルを使ったプログラム（応用編）

ここでは、本製品で提供するサンプルプログラム-例題9-について説明します。

例題9では、FORMAT句なし印刷ファイルを使って、I制御レコードを使用したページ形式の設定/変更と、CHARACTER TYPE句やPRINTING POSITION句を使用して印字したい文字の修飾および配置（行/桁）を意識して印刷装置に出力するプログラムの例を示します。FORMAT句なし印刷ファイルの使い方の詳細は、“NetCOBOL 使用手引書”の“8.2 行単位のデータを印刷する方法”および“8.3 フォームオーバーレイおよびFCBを使う方法”を参照してください。

概要

FORMAT句なし印刷ファイルを使用して帳票印刷を行う場合、主に利用される機能を想定し、以下の項目について印刷デモを行います。

FCBを使用した6LPI、8LPIでの帳票印刷

FCBを利用した任意の行間隔（6/8LPI）で帳票印刷を行うことを想定し、I制御レコードによるFCB（LPI）の切り替えを行います。ソースプログラムには、CHARACTER TYPE句やPRINTING POSITION句を記述して、行間隔（LPI）や文字間隔（CPI）などの行・桁を意識して帳票の体裁を整えます。

以下の帳票印刷を行います。

- A4用紙を横向きに使用し、1ページすべての行間隔を6LPIとした場合の帳票をイメージし、6LPI/10CPIフォーマットのスペーシングチャート形式のフォームオーバーレイと重畳印刷します。
- A4用紙を横向きに使用し、1ページすべての行間隔を8LPIとした場合の帳票をイメージし、8LPI/10CPIフォーマットのスペーシングチャート形式のフォームオーバーレイと重畳印刷します。

CHARACTER TYPE句で指定する各種文字属性での印刷

I制御レコードを使用し、用紙サイズをA4/横向きからB4/横向きに変更し、これにあわせてFCBもA4/横向き用からB4/横向き用に変更します。

以下の各種文字属性の印字サンプルを印刷装置に出力します。

1. 文字サイズ
1文字ずつ3ポ、7.2ポ、9ポ、12ポ、18ポ、24ポ、36ポ、50ポ、72ポ、100ポ、200ポ、300ポの文字サイズを印字します。
[補足] ここでは、文字ピッチ指定を省略することにより、文字サイズに合わせた最適な文字ピッチをCOBOLランタイムシステムに自動算出させます。
2. 文字ピッチ
文字ピッチ1CPIで1文字、2CPIで2文字、3CPIで3文字、5CPIで5文字、6CPIで6文字、7.5CPIで15文字、20CPIで20文字、24CPIで24文字指定します。
[補足] ここでは、文字サイズ指定を省略することにより、文字ピッチに合わせた最適な文字サイズをCOBOLランタイムシステムに自動算出させます。
3. 文字書体
ゴシック、ゴシック半角（文字形態半角）、明朝、明朝半角（文字形態半角）を10文字ずつ2回繰り返し印字します。
[補足] ここで指定した書体名は、以下の実行環境情報に関連付けられています。

```
@PrinterFontName=(FONT-NAME1, FONT-NAME2)
```

この指定により、“MINCHOU”、“MINCHOU-HANKAKU”を指定したデータ項目は、“FONT-NAME1”に指定されたフォントで印字され、“GOTHIC”、“GOTHIC-HANKAKU”を指定したデータ項目は、“FONT-NAME2”に指定されたフォントで印字されます。
なお、この例題プログラムでは、実行用の初期化ファイル（COBOL85.CBR）に“@PrinterFontName=(MS 明朝, MS ゴシック)”を指定しています。

4. 文字回転
縦書き(反時計回りに90度回転)、横書きを10文字ずつ繰り返し印字します。
5. 文字形態
全角、半角、全角平体、半角平体、全角長体、半角長体、全角倍角、半角倍角の文字形態指定を9文字ずつ印刷します。
6. 上記5つの文字属性を組み合わせた印刷を行います。

提供プログラム

- SAMPLE9. COB (COBOLソースプログラム)
- KOL5A4L6. OVD (フォームオーバーレイパターン)
- KOL5A4L8. OVD (フォームオーバーレイパターン)
- KOL5B4OV. OVD (フォームオーバーレイパターン)
- SAMPLE9. PRJ (プロジェクトファイル)
- COBOL85. CBR (実行用の初期化ファイル)
- SAMPLE9. TXT (プログラム説明書)

使用しているCOBOLの機能

- 印刷ファイル
- 小入出力機能(コンソールウィンドウ)
- プロジェクト管理機能

使用しているCOBOLの文

ADD文、CLOSE文、DISPLAY文、IF文、MOVE文、OPEN文、PERFORM文、STOP文、WRITE文

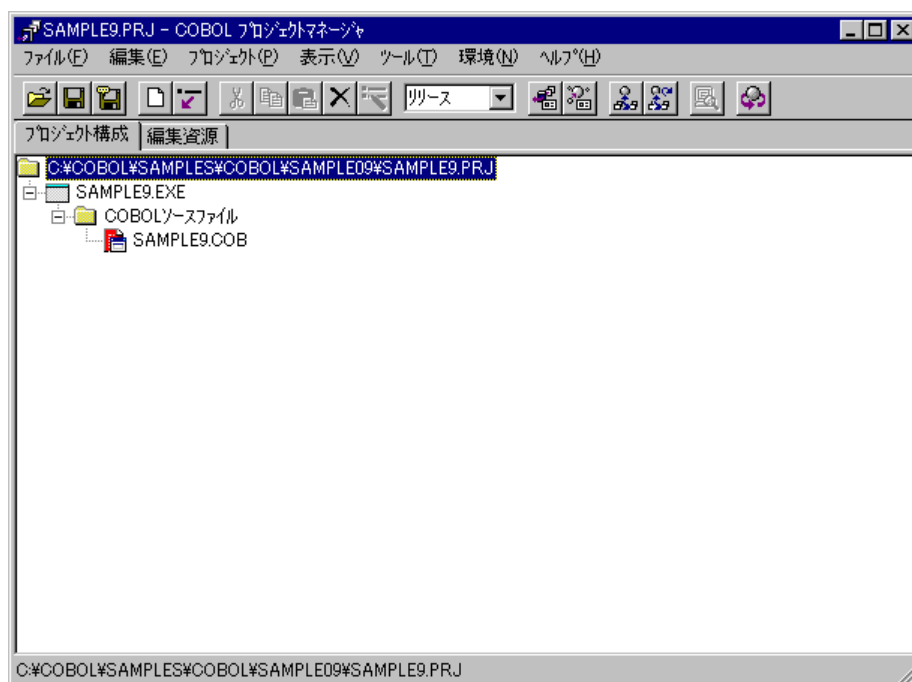
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

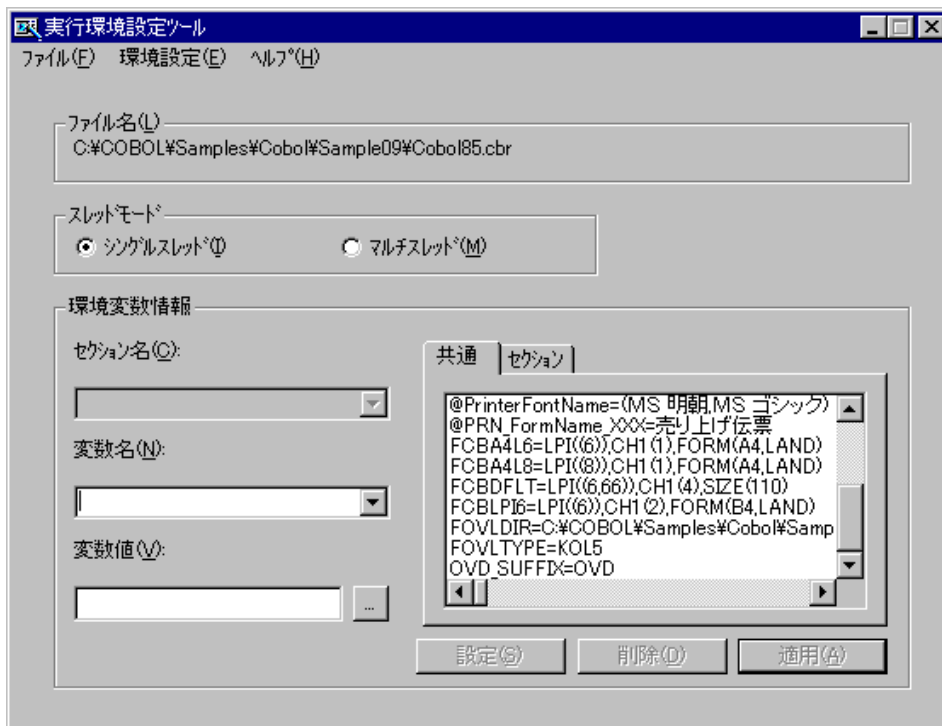
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE9. PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから「ビルド」を選択します。
→ ビルド終了後、SAMPLE9. EXEが作成されていることを確認してください。

実行環境情報の設定

- プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



- [ファイル]メニューの“開く”を選択し、実行可能プログラム(SAMPLE9.EXE)が存在するフォルダにある実行用の初期化ファイル(COBOL85.CBR)を開きます。
- 共通タブを選択します。例題9で必要な実行環境情報は、あらかじめCOBOL85.CBRに設定されています。以下の実行環境情報だけを追加してください。
 - 環境変数情報FOVLDIR(フォームオーバレイパターンのフォルダの指定)に、実行可能プログラム(SAMPLE9.EXE)が存在するフォルダを指定します。

```
FOVLDIR=C:\COBOL\SAMPLES\COBOL\SAMPLE09
```

- [適用]ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
- [ファイル]メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
→ 実行は、特に応答・操作する必要はなく自動的に終了します。実行が終了すると、サンプル帳票が、“通常使うプリンタ”として設定されている印刷装置に出力されます。

例題10 FORMAT句付き印刷ファイルを使ったプログラム

ここでは、本製品で提供するサンプルプログラム-例題10-について説明します。

例題10では、FORMAT句付き印刷ファイルを使って、集計表を印刷装置に出力するプログラムの例を示します。FORMAT句付き印刷ファイルの使い方は、“NetCOBOL 使用手引書”の“8.4 帳票定義体を使う印刷ファイルの使い方”を参照してください。なお、このプログラムを実行するには、MeFtが必要です。

概要

商品コード、商品名および単価が格納されているマスタファイル(例題2で作成した索引ファイル)と受注日、数量および売上げ金額が格納されている売上げファイル(例題3で作成した索引ファイル)を入力して、売上集計表を印刷装置に出力します。

提供プログラム

- SAMPLE10. COB (COBOL ソースプログラム)
- SYUUKKEI. PMD (帳票定義体)
- MEFPRC (プリンタ情報ファイル)
- SAMPLE10. CBI (翻訳オプションファイル)
- SAMPLE10. PRJ (プロジェクトファイル)
- SAMPLE10. TXT (プログラム説明書)



注意

SYOHINM. CBL (登録集原文) は、例題2で提供されたものを使用します。URIAGE. CBL (登録集原文) は、例題3で提供されたものを使用します。

使用しているCOBOLの機能

- FORMAT句付き印刷ファイル
- 索引ファイル(参照)
- 登録集の取込み
- 小入出力機能(メッセージボックス)
- プロジェクト管理機能

使用しているCOBOLの文

OPEN文、READ文、WRITE文、START文、CLOSE文、PERFORM文、DISPLAY文、IF文、MOVE文、SET文、GO TO文、EXIT文、COPY文、ADD文

使用している帳票定義体

売上集計表 (SYUKEI.PMD)

The screenshot displays the PowerFORM software window titled 'PowerFORM - [SYUKEI.pmd]'. The main canvas shows a form layout with a title '売上集計表' (Sales Summary Table) and a date field 'YYYY/MM/DD'. Below the title is a table with columns: '商品コード' (Item Code), '商品名' (Item Name), '単価' (Unit Price), '受注日' (Order Date), '数量' (Quantity), and '売上げ' (Sales Amount). The table contains sample data for item 'XXXX'. A summary row shows '合計' (Total) with values for quantity and sales amount. A footer row shows '小計' (Subtotal) and 'Page: ZZZ'. The left-hand tree view shows the form's structure with sections PH, CH1, DE, CF2, CF1, and PF. The bottom status bar indicates '強制' (Mandatory), 'インチ' (Inch), '位置' (Position), '5.64 In. 0.56 In.', and '項目長' (Item Length).

形式:

集計表形式

用紙サイズ:

A4

用紙方向:

縦

行ピッチ:

1/6インチ

パーティション:

PH(ページ頭書き)

[固定パーティション、印刷開始位置:0インチ(1行目)、縦幅:1インチ(6行)]

CH1(制御頭書き)

[浮動パーティション、縦幅:0.83インチ(5行)]

DE(明細)

[浮動パーティション、縦幅:0.33インチ(2行)]

CF1(制御脚書き)

[浮動パーティション、縦幅:0.83インチ(5行)]

CF2(制御脚書き)

[浮動パーティション、縦幅:0.67インチ(4行)]

PF(ページ脚書き)

[固定パーティション、印刷開始位置:10.48インチ(63行目)、縦幅:0.49インチ(3行)]

プログラムを作成する上でのポイント

- PHおよびPFは、固定パーティション(固定の印刷位置情報を持っている)なので、パーティションを出力すると、必ず、パーティションに定義されている印刷開始位置に出力されます。
- CH1、DE、CF1およびCF2は、浮動パーティション(固定の印刷位置情報を持たない)なので、

自由な位置に出力することができる反面、パーティション出力時に印刷位置を制御する必要があります。

- 各パーティションに定義された出力項目は、翻訳時にCOPY文で帳票定義体からレコードに展開されます。このとき、定義した出力項目の項目名がデータ名になります。

プログラムの翻訳・リンク・実行

プログラムを実行する前に

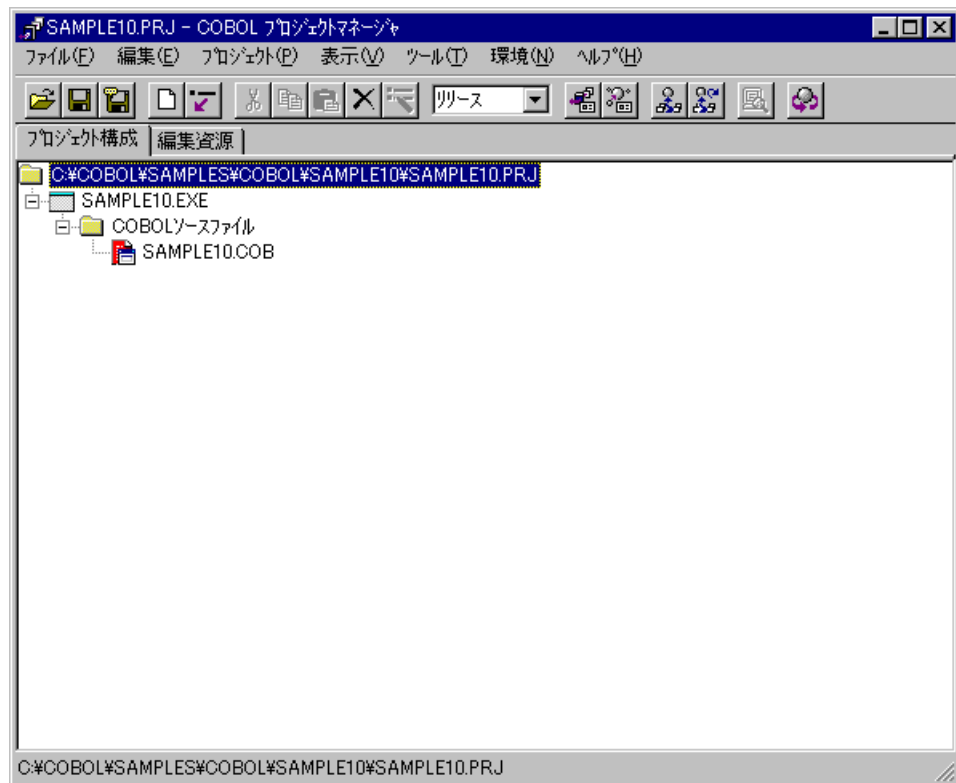
- MeFtのセットアップを行い、使用できる状態にしておいてください。
- 例題2で作成されるマスタファイルを使用するため、“[例題2 行順ファイルと索引ファイルの操作](#)”を実行しておきます。
- 例題3で作成される売上げファイルを使用するため、“[例題3 表示ファイル機能を使ったプログラム](#)”を実行しておきます。

ビルド・リビルド

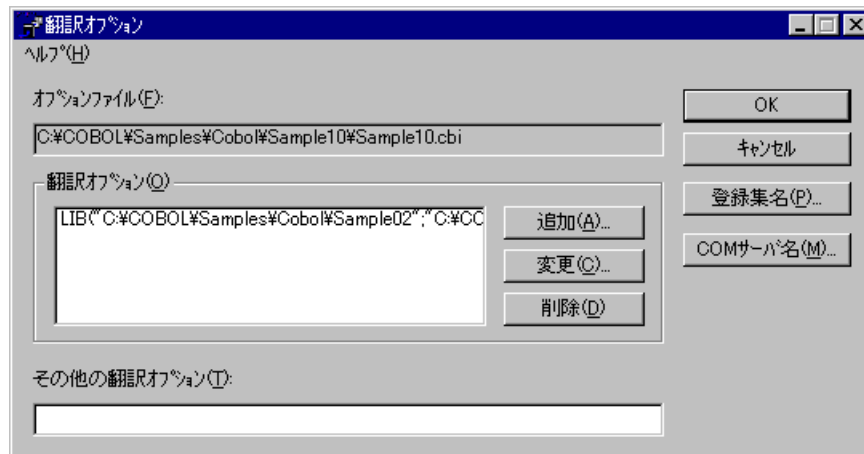
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE10.PRJ”を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションLIBに、SYOHINM.CBLが格納されたフォルダ(例題2のフォルダ)と URIAGE.CBLが格納されたフォルダ(例題3のフォルダ)を指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの[プロジェクト]メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE10.EXEが作成されていることを確認してください。

プリンタ情報ファイルの設定

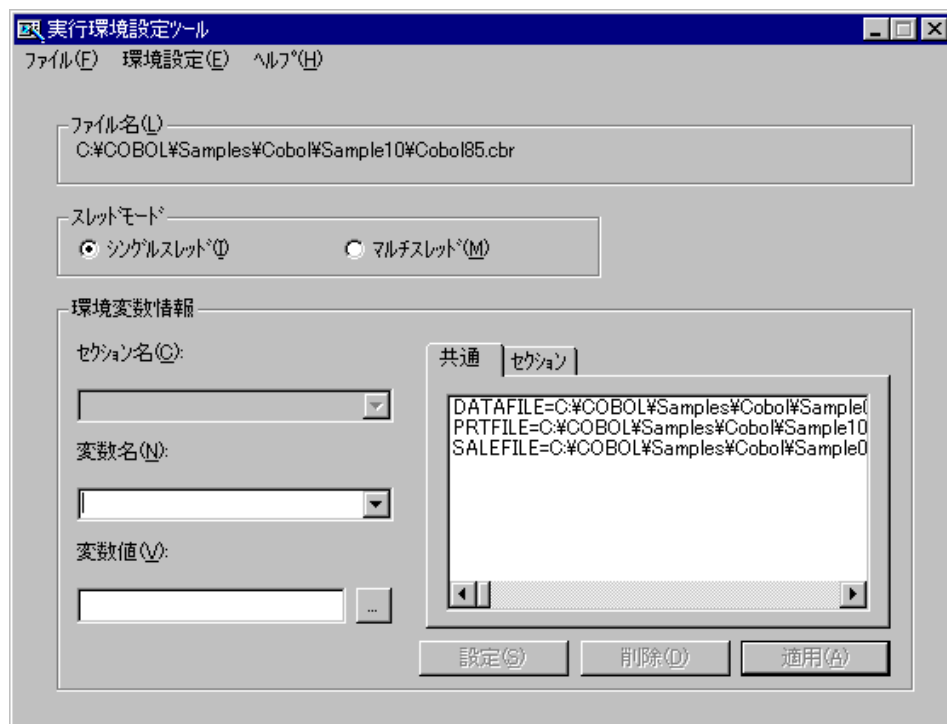
実行する前にプリンタ情報ファイル(MEFPRC)の下線部の情報をエディタを使用して変更しておきます。

MEDDIR C:\COBOL\SAMPLES\COBOL\SAMPLE10

MEDDIR : 帳票定義体(SYUKEI.PMD)を格納したフォルダのパス名

実行環境情報の設定

1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. [ファイル]メニューの“開く”を選択し、実行可能プログラム(SAMPLE10.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - ファイル識別名DATAFILEに、例題2で作成したマスタファイル(MASTER)を指定します。
 - ファイル識別名SALEFILEに、例題3で作成した売上げファイル(SALES)を指定します。
 - ファイル識別名PRTFILEに、プリンタ情報ファイル(MEFPRC)を指定します。
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル]メニューの“終了”を選択し、実行環境設定ツールを終了します。

備考

ファイル識別名のDATAFILE、SALEFILEおよびPRTFILEは、COBOLソースプログラムのASSIGN句に指定されているファイル参照子です。FORMAT句付き印刷ファイルを使用する場合、ファイル識別名にはプリンタ情報ファイルのパス名を指定します。

プログラムの実行

- プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
- 通常使うプリンタに設定されている印刷装置に集計表が出力されます。

例題11 データベース機能を使ったプログラム

ここでは、本製品で提供するサンプルプログラム-例題11-について説明します。

例題11では、データベース(SQL)機能を使ってデータベースからデータを取り出しホスト変数に格納する例を示します。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。

データベースのアクセスは、ODBCドライバを経由して行います。ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL 使用手引書”の“第21章 リモートデータベースアクセス(ODBC)”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側

- ODBCドライバマネージャ
- ODBCドライバ
- ODBCドライバの必要とする製品

サーバ側

- データベース
- データベースにODBCでアクセスするために必要な製品

概要

サーバのデータベースにアクセスし、データベース上の表“STOCK”に格納されている全データをディスプレイ装置に表示します。データをすべて参照し終わると、データベースとの接続を切断します。

提供プログラム

- SAMPLE11. PRJ(プロジェクトファイル)
- SAMPLE11. COB(COBOLソースプログラム)
- SAMPLE11. TXT(プログラム説明書)

使用しているCOBOLの機能

- リモートデータベースアクセス
- プロジェクト管理機能

使用しているCOBOLの文

DISPLAY文、GO TO文、IF文、PERFORM文

埋込みSQL文(埋込み例外宣言、CONNECT文、カーソル宣言、OPEN文、FETCH文、CLOSE文、ROLLBACK文、DISCONNECT文)

プログラムの翻訳・リンク・実行

プログラムを実行する前に

- ODBCドライバを経由してサーバのデータベースへアクセスできる環境を構築しておいてください。
- デフォルトで接続するサーバを設定し、そのサーバのデータベース上に“STOCK”という名前で表を作成しておいてください。
STOCK表は、以下の形式で作成してください。

GNO	GOODS	QOH	WHNO	←列の名前
2進整数 4桁	固定長文字 20バイト	2進整数 9桁	2進整数 4桁	←列の属性

STOCK表に格納しておくデータは任意です。以下に例を示します。

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
110	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	04	1
201	AIR CONDITIONER	15	1
212	TELEVISION	10	2
215	VIDEO	05	2
226	REFRIGERATOR	08	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	2
390	DRIER	540	2

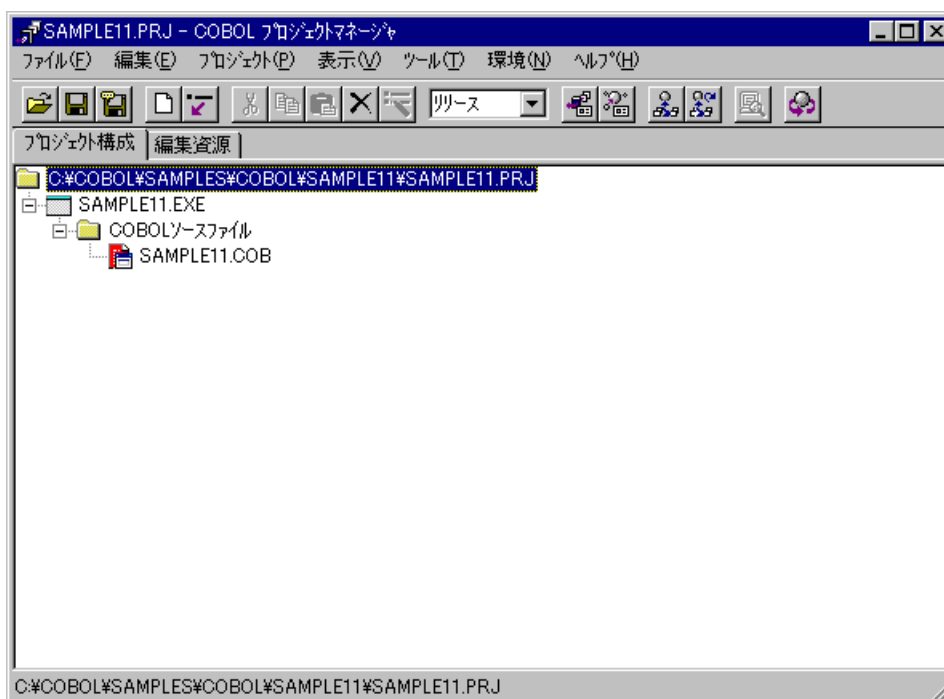
- ODBC情報ファイル設定ツール(SQLODBCS.EXE)を使用して、ODBC情報ファイル(ここではC:\¥DBMSACS.INFとします)を作成してください。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

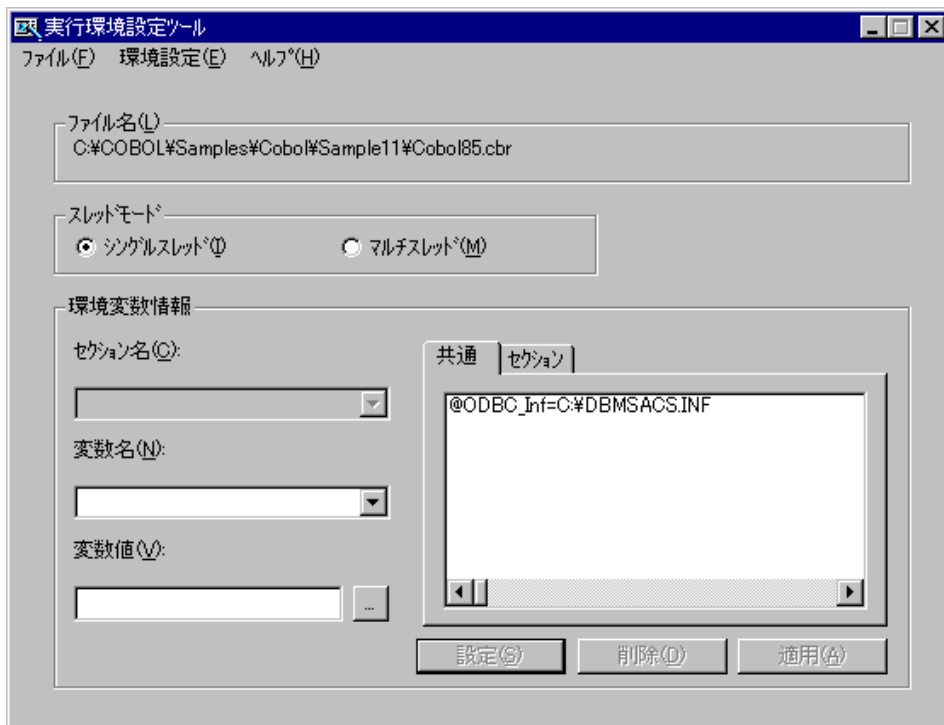
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE11.PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE11.EXEが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの「ツール」メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



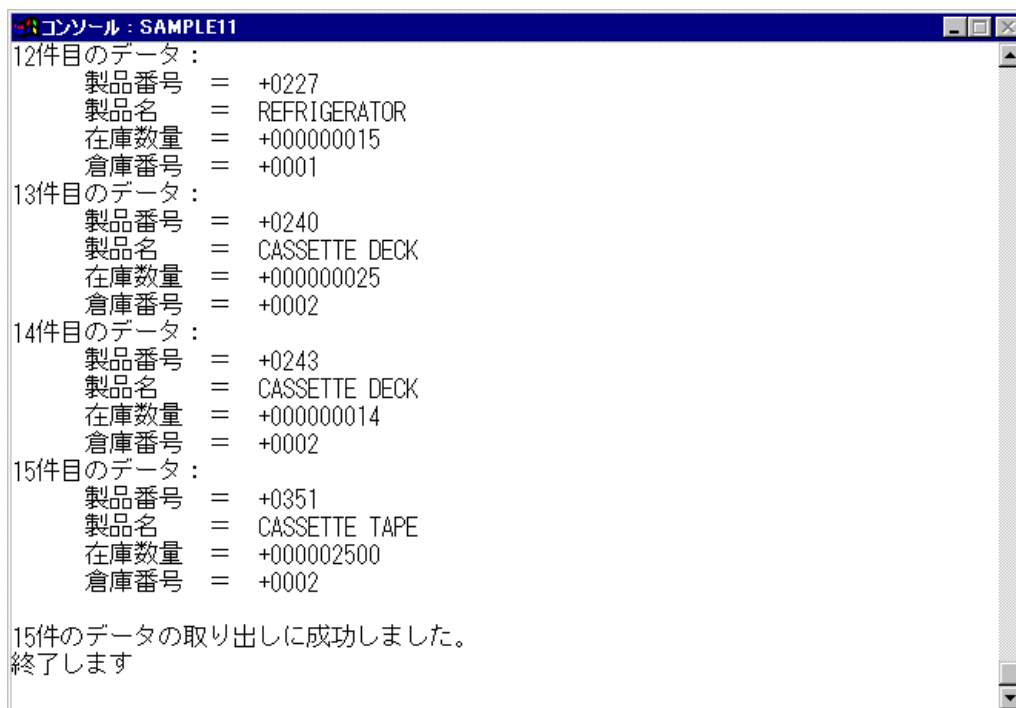
2. 「ファイル」メニューの“開く”を選択し、実行格納プログラム (SAMPLE11.EXE) の存在するフォルダに、実行用の初期化ファイル (COBOL85.CBR) を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@ODBC_Inf (ODBC情報ファイルの指定) に、ODBC情報ファイル名を指定します。

4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。

→ COBOLのコンソールウィンドウに、以下が表示されます。



```
CONSOLE: SAMPLE11
12件目のデータ：
  製品番号 = +0227
  製品名   = REFRIGERATOR
  在庫数量 = +000000015
  倉庫番号 = +0001
13件目のデータ：
  製品番号 = +0240
  製品名   = CASSETTE DECK
  在庫数量 = +000000025
  倉庫番号 = +0002
14件目のデータ：
  製品番号 = +0243
  製品名   = CASSETTE DECK
  在庫数量 = +000000014
  倉庫番号 = +0002
15件目のデータ：
  製品番号 = +0351
  製品名   = CASSETTE TAPE
  在庫数量 = +000002500
  倉庫番号 = +0002

15件のデータの取り出しに成功しました。
終了します
```


例題12 データベース機能を使ったプログラム （応用編）

ここでは、本製品で提供するサンプルプログラム-例題12-について説明します。

例題12では、データベース (SQL) 機能のより進んだ使い方として、データベースの複数の行を1つの埋込みSQL文で操作する例を示します。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。

データベースのアクセスは、ODBCドライバを経由して行います。ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL 使用手引書”の“第21章 リモートデータベースアクセス (ODBC)”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側

- ODBCドライバマネージャ
- ODBCドライバ
- ODBCドライバの必要とする製品

サーバ側

- データベース
- データベースにODBCでアクセスするために必要な製品

概要

例題11と同じデータベースにアクセスし、次の操作を行ってからデータベースとの接続を切断します。

- データベース全データの表示
- GOODSの値が“TELEVISION”である行のGNOの値の取り出し
- 取り出したGNOを持つ行のQOHの更新
- GOODSの値が“RADIO”、“SHAVER”、“DRIVER”の行の削除
- データベースの全データの再表示

なお、出力結果は翻訳オプションSSOUTを使用して、一部をファイルに出力します。

提供プログラム

- SAMPLE12. PRJ (プロジェクトファイル)
- SAMPLE12. COB (COBOLソースプログラム)
- SAMPLE12. TXT (プログラム説明書)

使用しているCOBOLの機能

- リモートデータベースアクセス
- プロジェクト管理機能

使用しているCOBOLの文

CALL文、DISPLAY文、GO TO文、IF文、PERFORM文

埋込みSQL文 (複数行指定ホスト変数、表指定ホスト変数、埋込み例外宣言、CONNECT文、カーソル宣言、OPEN文、FETCH文、SELECT文、DELETE文、UPDATE文、CLOSE文、COMMIT文、ROLLBACK文、DISCONNECT文)

プログラムの翻訳・リンク・実行

プログラムを実行する前に

- ODBCドライバを経由してサーバのデータベースへアクセスできる環境を構築しておいてください。
- デフォルトで接続するサーバを設定し、そのサーバのデータベース上に“STOCK”という名前で表を作成しておいてください。
STOCK表は、以下の形式で作成してください。

GNO	GOODS	QOH	WHNO	←列の名前
2進整数 4桁	固定長文字 20バイト	2進整数 9桁	2進整数 4桁	←列の属性

STOCK表に次のデータを格納しておいてください。

GNO	GOODS	QOH	WHNO
110	TELEVISION	85	2
110	TELEVISION	90	2
123	REFRIGERATOR	60	1
124	REFRIGERATOR	75	1
137	RADIO	150	2
138	RADIO	200	2
140	CASSETTE DECK	120	2
141	CASSETTE DECK	80	2
200	AIR CONDITIONER	04	1
201	AIR CONDITIONER	15	1
212	TELEVISION	10	2
215	VIDEO	05	2
226	REFRIGERATOR	08	1
227	REFRIGERATOR	15	1
240	CASSETTE DECK	25	2
243	CASSETTE DECK	14	2
351	CASSETTE TAPE	2500	2
380	SHAVER	870	3
390	DRIER	540	3

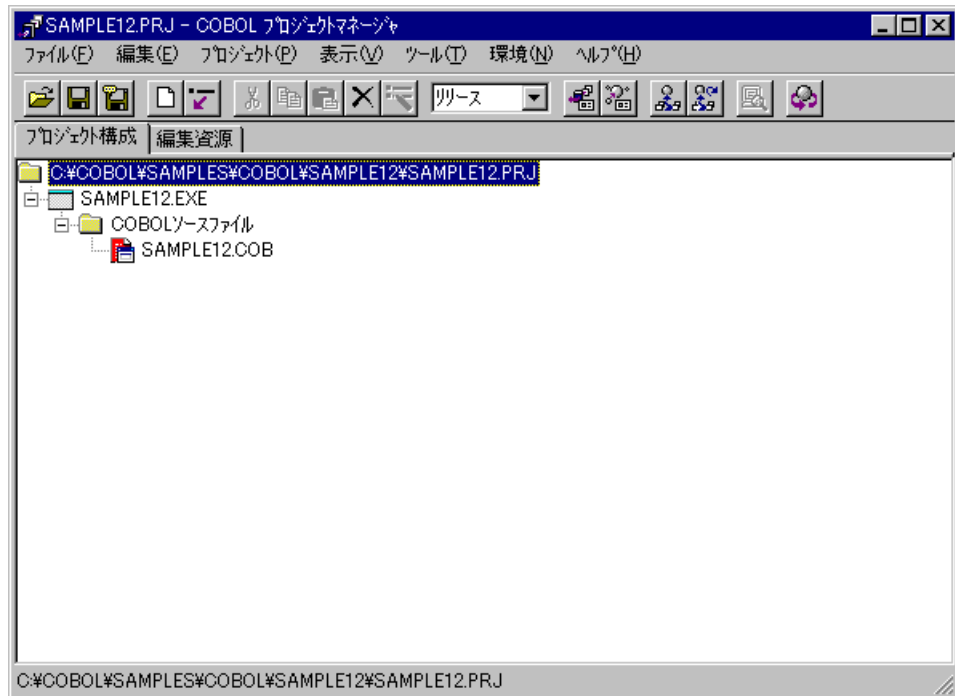
- ODBC情報ファイル設定ツール(SQLODBCS.EXE)を使用して、ODBC情報ファイル(ここではC:\YDBMSACS.INFとします)を作成してください。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\YCOBOLとして説明しています。フォルダ名がC:\YCOBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

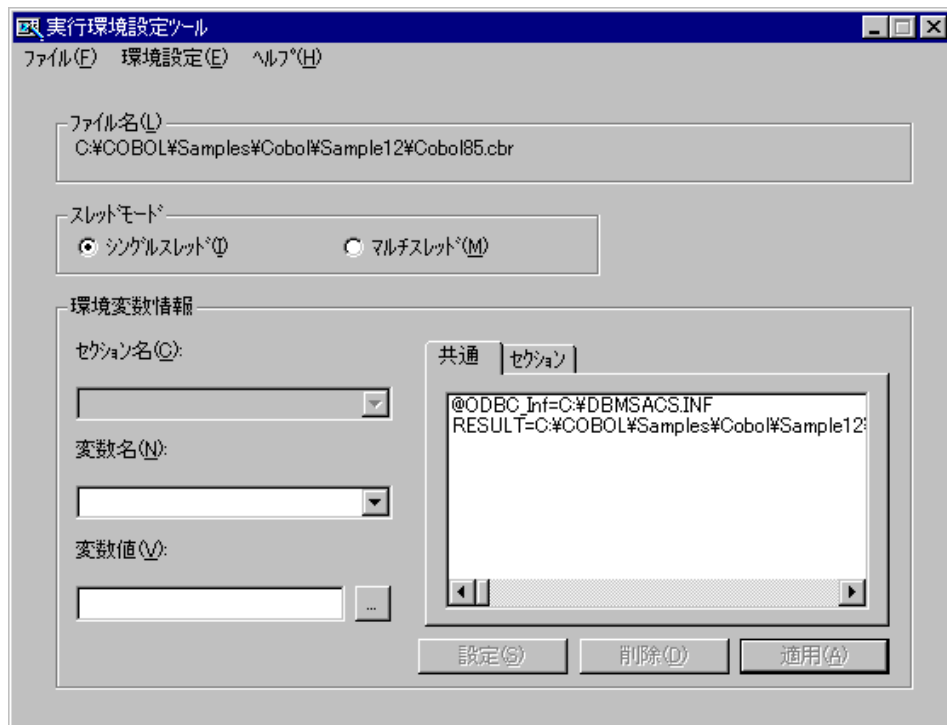
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE12.PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE12.EXEが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの「ツール」メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。

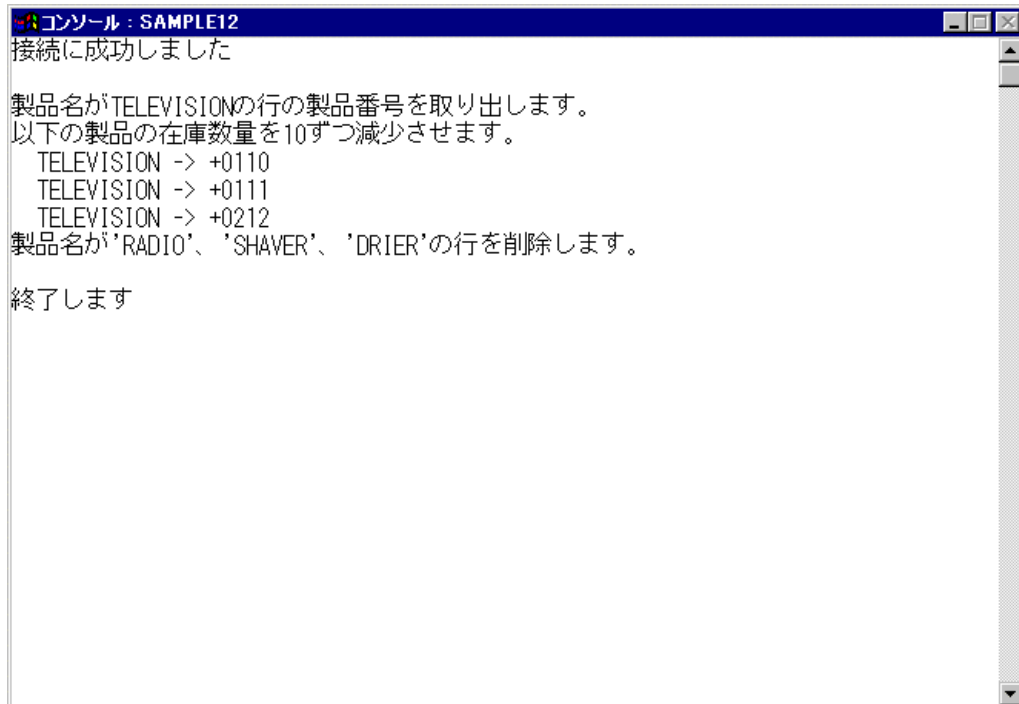


2. 「ファイル」メニューの“開く”を選択し、実行可能プログラム (SAMPLE12.EXE) が存在するフォルダに、実行用の初期化ファイル (COBOL85.CBR) を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@ODBC_Inf (ODBC情報ファイルの指定) に、ODBC情報ファイル名を指定します。

- 環境変数RESULTに、DISPLAY文の出力結果を保存するファイルを指定します。
- 4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
- 5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
→ COBOLのコンソールウィンドウに、以下が表示されます。



環境変数RESULTに割り当てたファイルには、次の形式で操作の前後でのSTOCKテーブルの内容が出力されます。

処理前のテーブルの内容

01件目のデータ :

```
製品番号 = +0110
製品名   = TELEVISION
在庫数量 = +000000085
倉庫番号 = +0002
.
```

19件目のデータ :

```
製品番号 = +0390
製品名   = DRIER
在庫数量 = +000000540
倉庫番号 = +0003
```

全データ件数は19件です

処理後のテーブルの内容

01件目のデータ :

製品番号 = +0110
製品名 = TELEVISION
在庫数量 = +000000075
倉庫番号 = +0002

・
・

15件目のデータ：

製品番号 = +0351
製品名 = CASSETTE TAPE
在庫数量 = +000002500
倉庫番号 = +0002

全データ件数は15件です

例題13 Visual Basic(R)からの呼出し

ここでは、本製品で提供するサンプルプログラム-例題13-について説明します。

例題13では、Visual Basic(R)アプリケーションから、NetCOBOLで作成したDLLを呼び出すプログラムの例を示します。

なお、このプログラムを動作させるためには、Visual Basic(R) 5.0、6.0 または7.0 が必要です。Visual Basic(R) 5.0 を使用する場合は、Visual Basic(R) のモジュールを再翻訳してください。また、Microsoft(R) Visual Studio(R) .NET(TM) 環境では、Visual Basic(R) 7.0 でSAMPLE13.VBP(Visual Basic(R)プロジェクトファイル)を読み込み、「アップグレードウィザード」にてファイルの自動変換を行った上で再翻訳、実行を行ってください。

概要

Visual Basic(R)アプリケーションを起動し、フォームがロードされたときにCOBOLプログラムの初期化手続きを行うサブルーチンJMPCINT2を呼び出します。

4桁以下の2つの数値をVisual Basic(R)アプリケーションのテキストボックスから入力し、[=](イコール)ボタンを押すと、その2つの数値がCOBOLアプリケーションに渡されます。

COBOLアプリケーションは、2つの数値を乗算し、結果を文字に編集してVisual Basic(R)アプリケーションに返却します。Visual Basic(R)アプリケーションでは、返却された編集結果の文字をテキストボックスに出力します。

Visual Basic(R)アプリケーションを終了し、フォームがアンロードされたときに、COBOLプログラムの終了手続きを行うサブルーチンJMPCINT3を呼び出します。

提供プログラム

- SAMPLE13.EXE(Visual Basic(R)実行可能ファイル)
- SAMPLE13.VBP(Visual Basic(R)プロジェクトファイル)
- SAMPLE13.FRM(Visual Basic(R)フォームモジュール)
- SAMPLE13.COB(COBOLソースプログラム)
- SAMPLE13.PRJ(プロジェクトファイル)
- SAMPLE13.LNI(リンクオプションファイル)
- SAMPLE13.TXT(プログラム説明書)

使用しているCOBOLの機能

- Visual Basic(R)からの呼出し
- プロジェクト管理機能
- COBOLランタイムシステムのサブルーチン(JMPCINT2、JMPCINT3)の使用

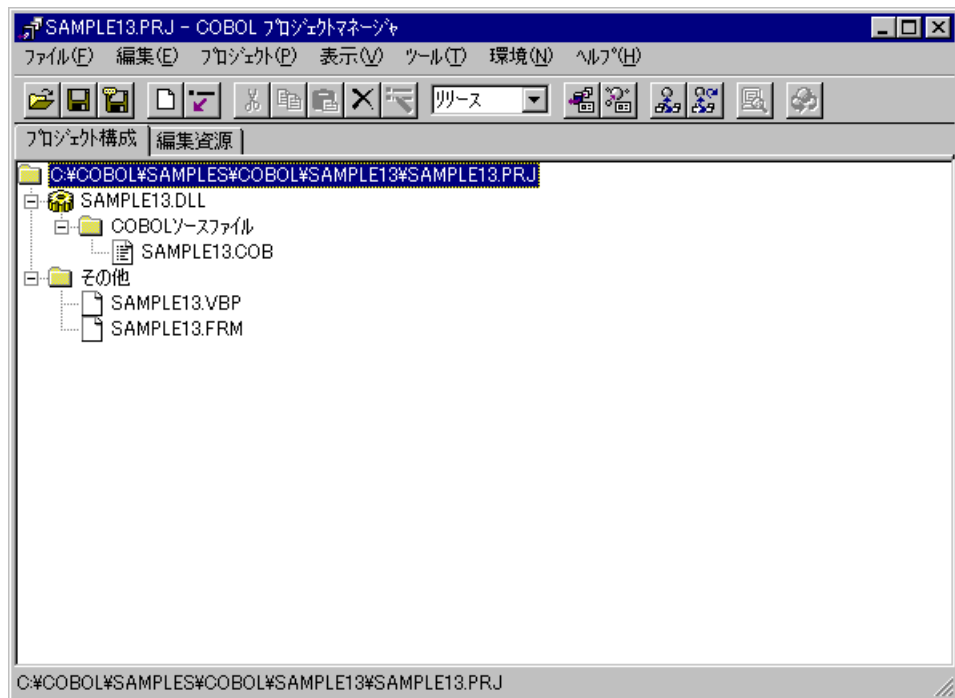
プログラムの翻訳・リンク・実行

ビルド・リビルド

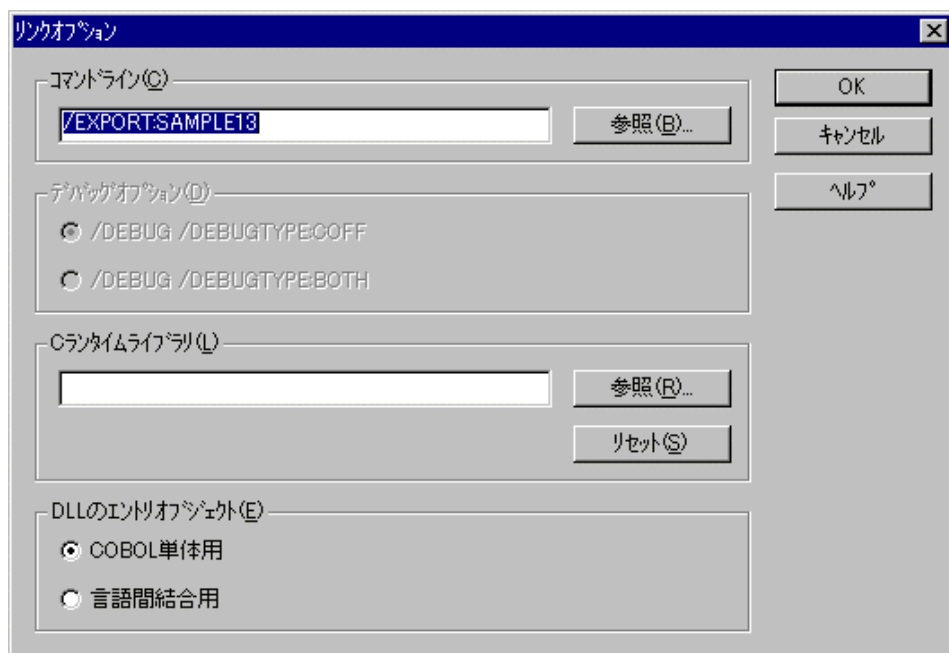
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルはNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。以降の説明で、フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE13.PRJ”を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“リンクオプション”を選択します。
→ [リンクオプション] ダイアログが表示されます。



4. [コマンドライン]エディットボックスに、“/EXPORT:SAMPLE13”を指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの[プロジェクト]メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE13.DLLが作成されていることを確認してください。
6. Visual Basic (R) のモジュールを再翻訳する場合、プロジェクトの“その他”のフォルダに含まれるSAMPLE13.VBP (Visual Basic (R) プロジェクトファイル)を選択し、[プロジェクト]メニューから“編集”を選択します。
→ Visual Basic (R) が起動しますので、[ファイル]メニューから“SAMPLE13.EXEの作成”

を選択します。



この操作はVisual Basic (R)がインストールされており、PowerGEM Plusの設定で“Windowsに登録されている拡張子に関連づけられたコマンドをツールとして組み込む”が有効になっている場合のみ可能です。

プログラムの実行

SAMPLE13. DLLファイルが、カレントフォルダまたは環境変数PATHに設定したフォルダにあることを確認してください。

Visual Basic (R)を使用する場合

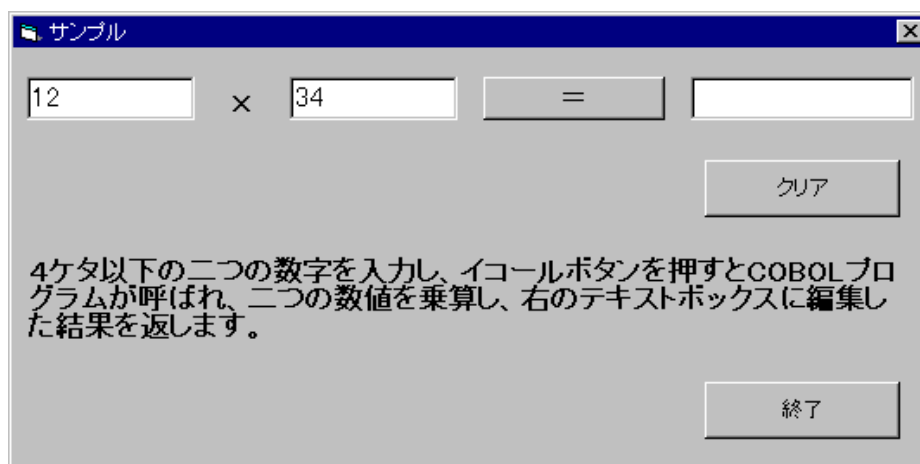
〔ファイル〕メニューから“プロジェクトを開く”を選択し、プロジェクトファイル(SAMPLE13. VBP)を指定します。

その後、〔実行〕メニューから“開始”を選択します。

Visual Basic (R)で作成した実行可能ファイルの場合

実行可能ファイルを実行してください。

1. プログラムを実行すると、Visual Basic (R)で作成したフォームが表示されます。フォームには、左辺を示す2つの入力用のテキストボックスおよび右辺を示す1つの出力用テキストボックスがあります。



2. 左辺を示すテキストボックスに数値を入力します。
3. [=] (イコール) ボタンを押すと、COBOLで作成したアプリケーションに左辺で指定した数値が渡されます。
COBOLアプリケーションでは、2つの数値を乗算し、編集項目に格納し、これをVisual Basic (R)アプリケーションに返却します。
4. Visual Basic (R)アプリケーションは、返却された文字を右辺に示すテキストボックスに表示します。
5. 各テキストボックスの値をクリアするには、〔クリア〕ボタンを押します。

例題14 Visual Basic(R)を使った簡易ATM端末処理機能

ここでは、本製品で提供するサンプルプログラム-例題14-について説明します。

例題14では、Visual Basic(R)アプリケーションから、COBOLで作成したDLLの呼出し方法を簡易ATM端末処理機能のプログラム例で示します。

なお、このプログラムを動作させるためには、Visual Basic(R) 5.0、6.0 または7.0 が必要です。Visual Basic(R) 5.0 を使用する場合は、Visual Basic(R) のモジュールを再翻訳してください。また、Microsoft(R) Visual Studio(R) .NET(TM) 環境では、Visual Basic(R) 7.0 でSAMPLE13.VBP(Visual Basic(R)プロジェクトファイル)を読み込み、「アップグレードウィザード」にてファイルの自動変換を行った上で再翻訳、実行を行ってください。

概要

サンプルプログラムは、以下のATM端末の機能を実現しています。

- 新規口座の作成
- 入金処理
- 出金処理

口座のデータ(口座番号、暗証番号、氏名および貯金額)は、索引ファイルに保存します。索引ファイルの構造は、以下に示すとおりです。

口座番号	9(5)	主レコードキー
暗証番号	9(4)	
氏名	N(6)	
貯金額	9(9)	

ATM端末から要求された機能により、索引ファイル内の任意の口座のレコードのデータを更新します。

提供プログラム

- SAMPLE14.EXE(Visual Basic(R)実行可能ファイル)
- SAMPLE14.VBP(Visual Basic(R)プロジェクトファイル)
- SAMPLE14.BAS(Visual Basic(R)標準モジュール)
- SAMPLE14.FRM(Visual Basic(R)フォームモジュール):簡易ATM端末処理のメイン処理を行う
- SINKI.FRM(Visual Basic(R)フォームモジュール):新規口座を開く
- SINKCHK.FRM(Visual Basic(R)フォームモジュール):新規口座について口座番号を確認する
- SELE.FRM(Visual Basic(R)フォームモジュール):入金/出金選択を行う
- NYUKIN.FRM(Visual Basic(R)フォームモジュール):入金処理を行う
- SYUKIN.FRM(Visual Basic(R)フォームモジュール):出金処理を行う
- ERROR_H.FRM(Visual Basic(R)フォームモジュール):エラーメッセージを表示する
- K_KEN.COB(COBOLソースプログラム):口座番号を検索する
- K_SIN.COB(COBOLソースプログラム):新規口座を作成する
- K_NYU.COB(COBOLソースプログラム):口座に入金を行う
- K_SYU.COB(COBOLソースプログラム):口座に出金を行う
- SAMPLE14.PRJ(プロジェクトファイル)
- SAMPLE14.LNI(リンクオプションファイル)
- SAMPLE14.TXT(プログラム説明書)

使用しているCOBOLの機能

- Visual Basic(R)からの呼出し
- プロジェクト管理機能
- COBOLランタイムシステムのサブルーチン(JMPCINT2、JMPCINT3)の使用

使用しているCOBOLの文

MOVE文、IF文、PERFORM文、COMPUTE文、OPEN文、READ文、WRITE文、REWRITE文、CLOSE文、EXIT文

プログラムの内容

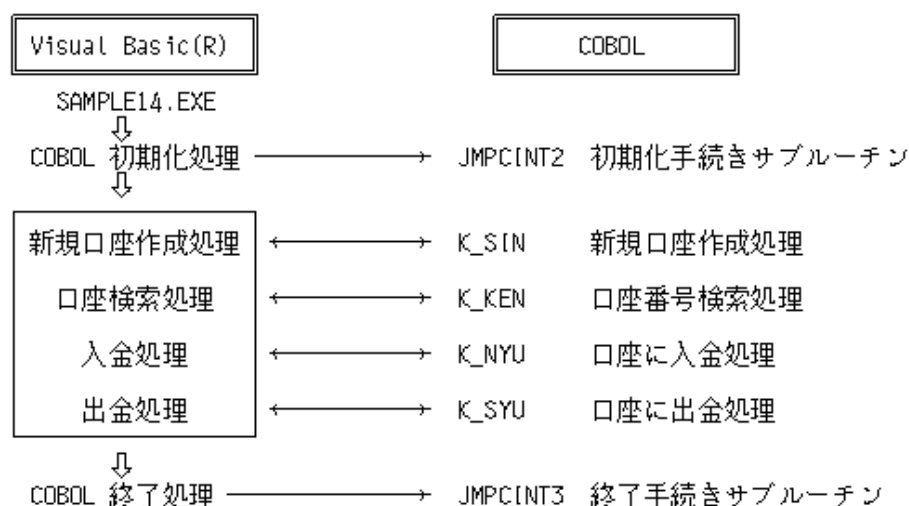
Visual Basic(R)アプリケーションを起動し、フォームがロードされたときにCOBOLプログラムの初期化手続きを行うサブルーチンJMPCINT2を呼び出します。

〔新規口座〕ボタンを押すと、入力用フォームが開き、各項目を入力して〔OK〕ボタンを押すと、入力した内容がCOBOLアプリケーションに渡されます。

COBOLアプリケーションは、入力した内容をレコードに書き込み、数値をVisual Basic(R)アプリケーションに返却します。

Visual Basic(R)アプリケーションでは、返却された数値をテキストボックスに出力します。

Visual Basic(R)アプリケーションを終了し、フォームがアンロードされたときに、COBOLプログラムの終了手続きを行うサブルーチンJMPCINT3を呼び出します。



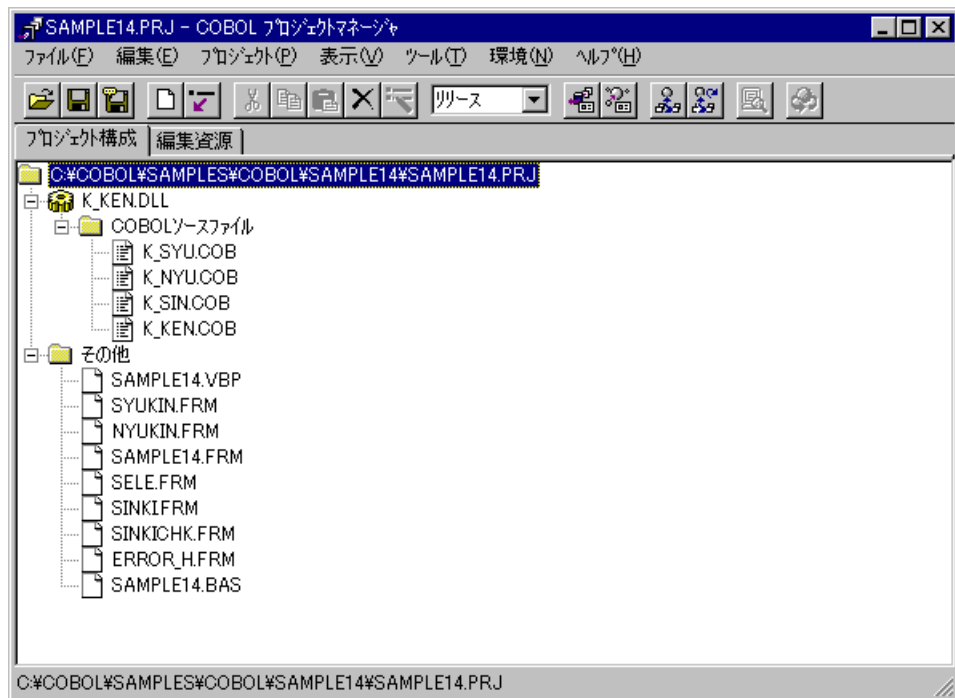
プログラムの翻訳・リンク・実行

ビルド・リビルド

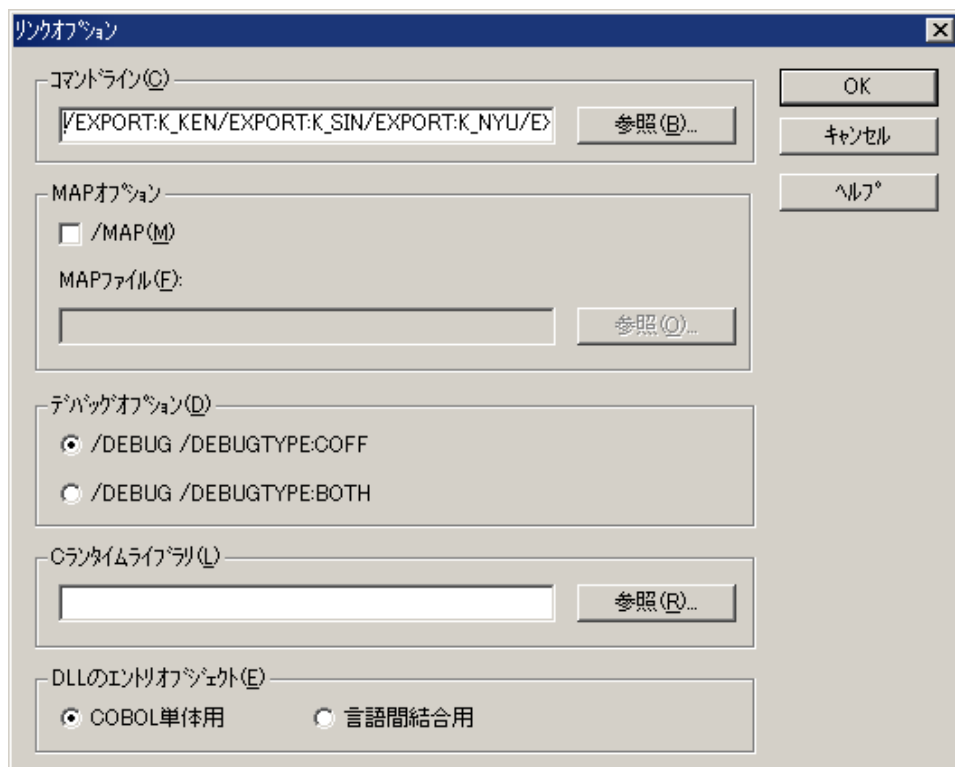
翻訳および実行は、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルはNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。以降の説明で、フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE14.PRJ”を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“リンクオプション”を選択します。
→ [リンクオプション] ダイアログが表示されます。



4. [コマンドライン]エディットボックスに、“/EXPORT:K_KEN /EXPORT:K_SIN /EXPORT:K_NYU /EXPORT:K_SYU”を指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャに戻ります。
5. プロジェクトマネージャの[プロジェクト]メニューから“ビルド”を選択します。
→ ビルド終了後、K_KEN.DLLが作成されていることを確認してください。
6. Visual Basic (R) のモジュールを再翻訳する場合、プロジェクトの“その他”のフォルダ

に含まれるSAMPLE14.VBP (Visual Basic (R) プロジェクトファイル) を選択し、[プロジェクト] メニューから“編集”を選択します。

→ Visual Basic (R) が起動しますので、[ファイル] メニューから“SAMPLE14.EXEの作成”を選択します。



この操作はVisual Basic (R) がインストールされており、PowerGEM Plusの設定で“Windowsに登録されている拡張子に関連づけられたコマンドをツールとして組み込む”が有効になっている場合のみ可能です。

プログラムの実行

作成したDLLファイルは、カレントフォルダまたは環境変数PATHに設定したフォルダにあることを確認してください。

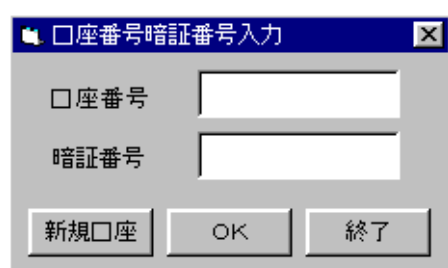
Visual Basic (R) を使用して実行する場合

1. Visual Basic (R) を起動して、[ファイル] メニューから“プロジェクトを開く”を選択し、ファイル一覧の中から“SAMPLE14.VBP”を指定します。または、エクスプローラから“SAMPLE14.VBP”をダブルクリックします。
2. [実行] メニューから“開始”を選択します。

Visual Basic (R) で作成したプログラムを実行する場合

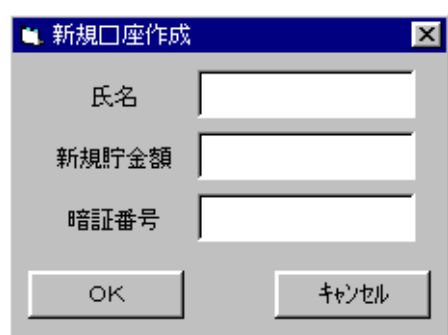
1. “SAMPLE14.EXE”を実行します。
→ 実行が終了すると、カレントフォルダに索引ファイル(TYOKIN.DAT)が作成されます。

〔口座番号暗証番号入力〕ダイアログ

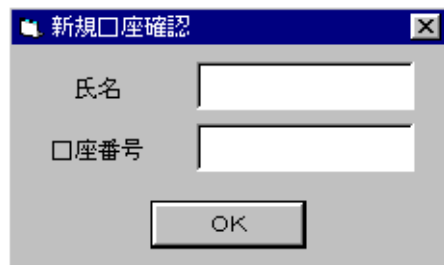


1. [新規口座] ボタンをクリックします。
→ [新規口座作成] ダイアログが表示されます。ここで、新規口座を作成し、口座番号と暗証番号を取得します。
2. 取得した口座番号および暗証番号を入力し、[OK] ボタンをクリックします。
→ 該当する口座番号の〔口座番号、氏名、貯金額表示〕ダイアログが表示されます。エラーが発生した場合は、[エラー画面] ダイアログが表示されます。
3. プログラムを終了するときには、[終了] ボタンをクリックします。

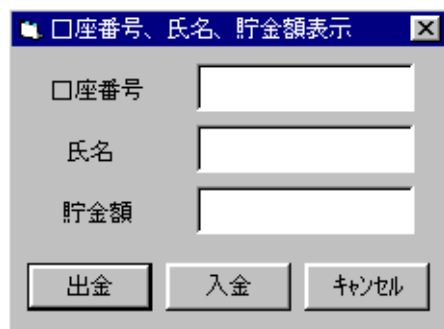
〔新規口座作成〕ダイアログ



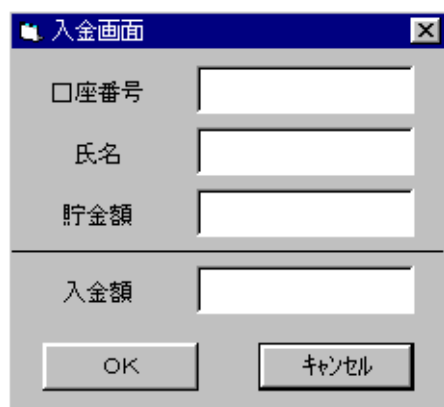
1. 氏名、貯金額、暗証番号を入力し、〔OK〕 ボタンをクリックします。
→ 新規口座作成処理が行われ、〔新規口座確認〕 ダイアログが表示されます。エラーが発生した場合は、〔エラー画面〕 ダイアログが表示されます。
2. 新規口座作成を取りやめる場合は、〔キャンセル〕 ボタンをクリックします。
→ 〔口座番号暗証番号入力〕 ダイアログに戻ります。

〔新規口座確認〕 ダイアログA dialog box titled "新規口座確認" (New Account Confirmation). It contains two input fields: "氏名" (Name) and "口座番号" (Account Number). Below the fields is an "OK" button.

1. 口座番号を確認して〔OK〕 ボタンをクリックします。
→ 〔口座番号暗証番号入力〕 ダイアログに戻ります。

〔口座番号、氏名、貯金額表示〕 ダイアログA dialog box titled "口座番号、氏名、貯金額表示" (Account Number, Name, Balance Display). It contains three input fields: "口座番号" (Account Number), "氏名" (Name), and "貯金額" (Balance). Below the fields are three buttons: "出金" (Withdrawal), "入金" (Deposit), and "キャンセル" (Cancel).

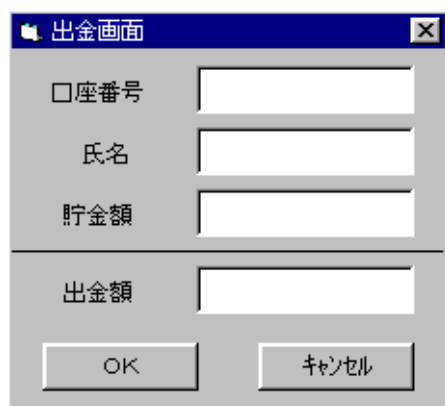
1. 出金の場合は、〔出金〕 ボタンをクリックします。
→ 〔出金〕 ダイアログが表示されます。
2. 入金の場合は、〔入金〕 ボタンをクリックします。
→ 〔入金〕 ダイアログが表示されます。
3. 処理を中断する場合は、〔キャンセル〕 ボタンをクリックします。
→ 〔口座番号暗証番号入力〕 ダイアログに戻ります。

〔入金画面〕 ダイアログA dialog box titled "入金画面" (Deposit Screen). It contains four input fields: "口座番号" (Account Number), "氏名" (Name), "貯金額" (Balance), and "入金額" (Deposit Amount). Below the fields are two buttons: "OK" and "キャンセル" (Cancel).

1. 入金額を入力し、〔OK〕 ボタンをクリックします。
→ 入金処理が行われ、〔口座番号、氏名、貯金額表示〕 ダイアログが表示されます。エラーが発生した場合は、〔エラー画面〕 ダイアログが表示されます。

2. 処理を中断する場合は、〔キャンセル〕ボタンをクリックします。
→ 〔口座番号、氏名、貯金額表示〕ダイアログに戻ります。

〔出金画面〕ダイアログ



1. 出金額を入力し、〔OK〕ボタンをクリックします。
→ 出金処理が行われ、〔口座番号、氏名、貯金額表示〕ダイアログが表示されます。エラーが発生した場合は、〔エラー画面〕ダイアログが表示されます。
2. 処理を中断する場合は、〔キャンセル〕ボタンをクリックします。
→ 〔口座番号、氏名、貯金額表示〕ダイアログに戻ります。

〔エラー画面〕ダイアログ



1. エラーメッセージを確認し、〔OK〕ボタンをクリックします。
→ エラー元の画面に戻ります。

例題15 オブジェクト指向プログラム（初級編）

ここでは、本製品で提供するサンプルプログラム-例題15-について説明します。

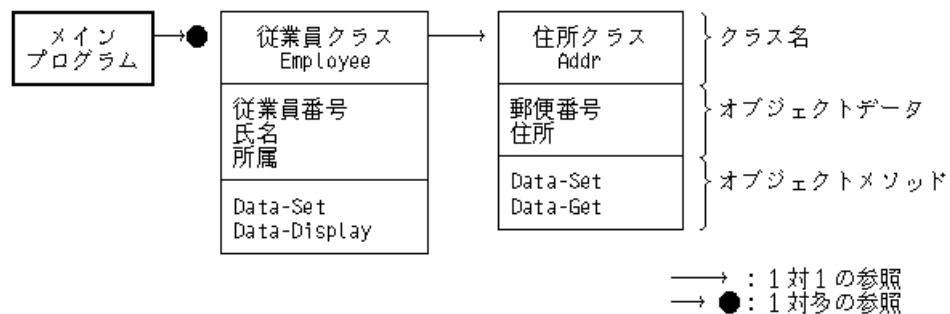
例題15では、オブジェクト指向プログラミング機能を使ったプログラムの例を示します。このプログラムでは、カプセル化、オブジェクトの生成、メソッド呼出しといった、オブジェクト指向の基本的な機能だけを使用しています。

概要

最初に従業員オブジェクトを3つ生成しています。“NEW”メソッドでオブジェクトを生成した後、“Data-Set”を呼び出してデータを設定しています。それぞれの従業員オブジェクトはすべて同じ形をしていますが、持っているデータ（従業員番号、氏名、所属、住所情報）は異なります。また、住所情報もオブジェクトであり、郵便番号と住所を持っています。

画面から従業員番号を入力すると、該当する従業員オブジェクトに対して“Data-Display”メソッドを呼び出し、そのオブジェクトが持っている従業員情報を画面に表示します。このとき、従業員オブジェクトは、住所の情報を得るために、従業員オブジェクトが指している住所オブジェクトに対し、“Data-Get”メソッドを呼び出します。

従業員オブジェクトは、3つのデータと住所オブジェクトから構成されています。しかし、これを使う側（この場合はメインプログラム）はオブジェクトの構造を知っている必要はありません。“Data-Set”と“Data-Display”の2つのメソッドだけを知っていれば十分です。つまり、データとアクセス手段を1つにまとめる（カプセル化）ことにより、オブジェクト内部の情報を完全に隠蔽しているわけです。



提供プログラム

- MAIN. COB (COBOLソースプログラム)
- MEMBER. COB (COBOLソースプログラム)
- ADDRESS. COB (COBOLソースプログラム)
- SAMPLE15. PRJ (プロジェクトファイル)
- SAMPLE15. CBI (翻訳オプションファイル)
- SAMPLE15. TXT (プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
 - クラスの定義(カプセル化)
 - オブジェクトの生成
 - メソッド呼出し
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- リポジトリ段落

- クラス定義、オブジェクト定義、メソッド定義

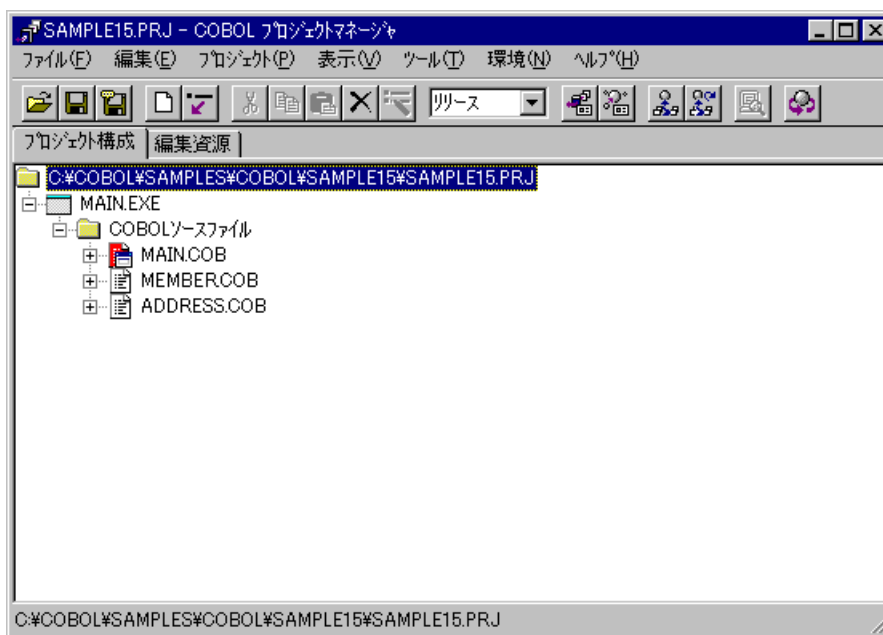
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE15.PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、MAIN.EXEが作成されることを確認してください。

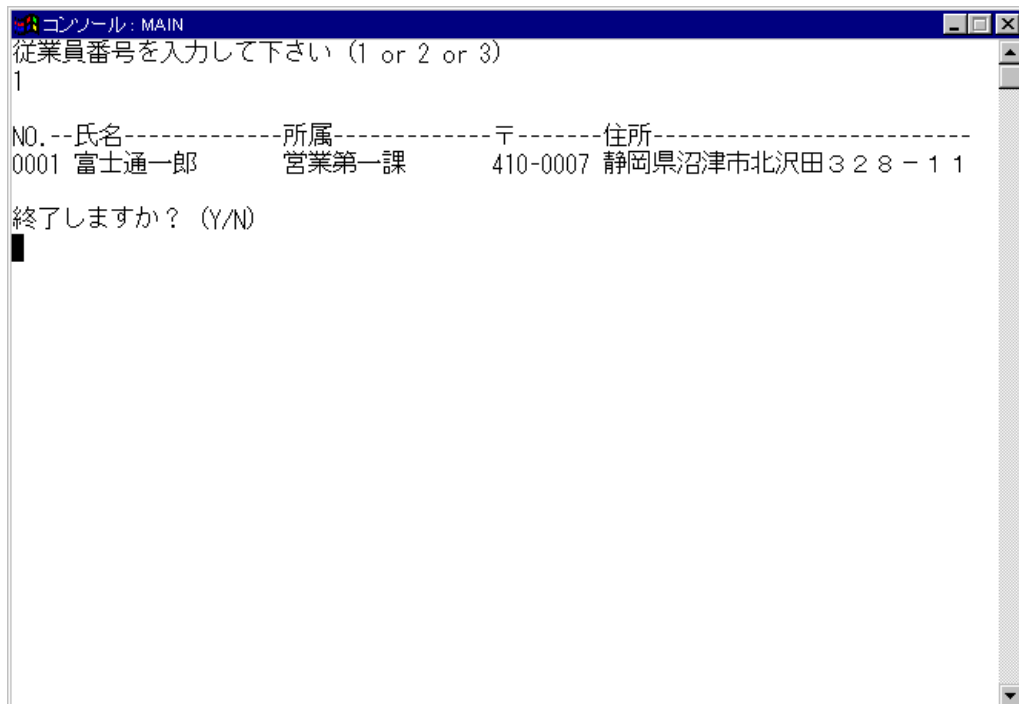
プログラムの実行

プロジェクトマネージャの「プロジェクト」メニューから“実行”を選択します。

→ ディスプレイに従業員番号を入力するためのCOBOLコンソールが表示されます。



従業員番号(1～3)を入力すると、従業員情報が表示されます。



例題16 コレクションクラス（クラスライブラリ）

ここでは、本製品で提供するサンプルプログラム-例題16-について説明します。

例題16では、クラスライブラリの作成方法の例として、コレクションクラスの例を示します。このサンプルは、クラスライブラリの作成方法の例として使用するだけでなく、実際にプログラムに組み込んで使用することができます。また、このサンプルには基本的な操作しか含まれていませんが、改造・変更することにより、さらに使いやすいクラスライブラリにすることができます。

概要

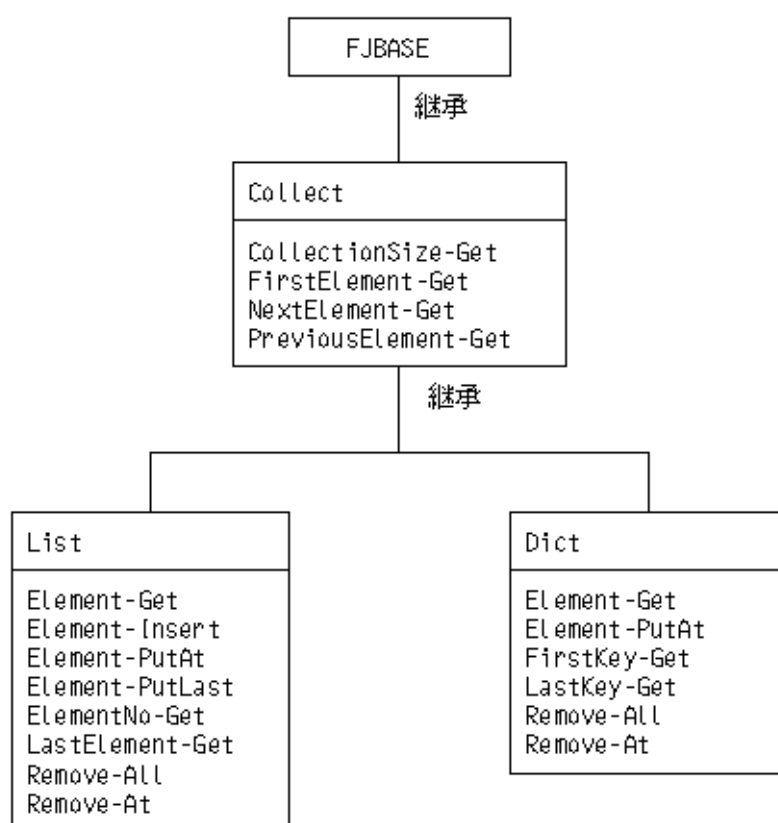
“コレクションクラス”とは、集合を扱うクラスの総称です。つまり、たくさんのオブジェクトをまとめて扱ったり、格納したりするためのクラスです。サンプルには、以下のクラスが含まれています。

- Collect(Collection:収集)
- Dict(Dictionary:辞書)
- List(リスト)

プログラムの内容

クラス階層

例題16のクラスの階層関係を下図に示します。



備考

サンプルクラスには、上記の他にBinaryTree-Class、DictionaryNode-ClassおよびListNode-Classが含まれています。これらのクラスは、ListクラスおよびDictクラス内部

で使用しているクラスで、コレクションクラス使用者からは見えなくなっています。そのため、ここではこれらのクラスの説明は省略します。

Collectクラス

コレクションクラスの最上位のクラスです。すべてのコレクションクラスはこのクラスを継承しています。Collectは抽象クラスであり、オブジェクトを作成しません。

このクラスは、FJBASEクラスを継承しているので、FJBASEクラスで定義されているメソッドもすべて使用することができます。

定義

```

CLASS-ID. Collect INHERITS FJBASE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS FJBASE.
OBJECT.
PROCEDURE DIVISION.
METHOD-ID. CollectionSize-Get.
METHOD-ID. FirstElement-Get.
METHOD-ID. NextElement-Get.
METHOD-ID. PreviousElement-Get.
END OBJECT.
END CLASS Collect.

```

CollectionSize-Getメソッド

集合の要素数を調べます。

[パラメタ]

なし

[復帰値]

PIC 9(8) BINARY:

集合の要素数を返します。

FirstElement-Getメソッド

集合の先頭の要素を取り出します。

[パラメタ]

なし

[復帰値]

USAGE OBJECT REFERENCE:

集合の先頭の要素を返します。要素がない場合は、NULLを返します。

NextElement-Getメソッド

現在指している要素の次の要素を取り出します。

[パラメタ]

なし

[復帰値]

USAGE OBJECT REFERENCE:

現在指している要素の次の要素を返します。次の要素がない場合は、NULLを返します。

PreviousElement-Getメソッド

現在指している要素の直前の要素を取り出します。

[パラメタ]

なし

[復帰値]

USAGE OBJECT REFERENCE:

直前の要素を返します。直前の要素がない場合は、NULLを返します。

Dictクラス

以下の特徴を持つ集合クラスです。

- 各要素はキーを持っています。
- キーの値は一意です。
- キーによる検索ができます。
- キーにより順序付けされています。

このクラスは、Collectクラスを継承しているので、Collectクラスで定義されているメソッドもすべて使用することができます。

定義

```
-----
CLASS-ID. Dict INHERITS Collect.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
    CLASS Collect.
OBJECT.
PROCEDURE DIVISION.
METHOD-ID. Element-Get.
METHOD-ID. Element-PutAt.
METHOD-ID. FirstKey-Get.
METHOD-ID. LastKey-Get.
METHOD-ID. Remove-All.
METHOD-ID. Remove-At.
END OBJECT.
END CLASS Dict.
-----
```

Element-Getメソッド

指定されたキーの要素を取り出します。

[パラメタ]

PIC X ANY LENGTH:

取り出す要素のキー値を指定します。

[復帰値]

USAGE OBJECT REFERENCE:

指定されたキーの要素が見つかった場合はその要素を、見つからなかった場合はNULLを返します。

Element-PutAtメソッド

指定されたキーの要素を追加します。すでに同じキーを持つ要素が存在している場合は、新しい要素で置き換えます。

[パラメタ]

PIC X ANY LENGTH:

追加・置換する要素のキー値を指定します。

USAGE OBJECT REFERENCE:

追加・置換する要素を指定します。

[復帰値]

なし

FirstKey-Getメソッド

先頭の要素のキー値を求めます。

[パラメタ]

なし

[復帰値]

PIC X ANY LENGTH:

先頭の要素のキー値を返します。要素数が0の場合または先頭の要素のキーが空白の場合、空白を返します。

LastKey-Getメソッド

最後の要素のキー値を求めます。

[パラメタ]

なし

[復帰値]

PIC X ANY LENGTH:

最後の要素のキー値を返します。要素数が0の場合または最後の要素のキーが空白の場合、空白を返します。

Remove-Allメソッド

集合に含まれるすべての要素を削除します。

[パラメタ]

なし

[復帰値]

なし

Remove-Atメソッド

指定されたキーの要素を削除します。

[パラメタ]

PIC X ANY LENGTH:

削除する要素のキー値を指定します。

[復帰値]

なし

Listクラス

以下の特徴を持つ集合クラスです。

- 要素間に順序があります。
- 同一の要素を複数含むことができます。

このクラスは、Collectクラスを継承しているので、Collectクラスで定義されているメソッドもすべて使用することができます。

定義

```
CLASS-ID. List INHERITS Collect.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
```

```

REPOSITORY.
  CLASS Collect.
OBJECT.
PROCEDURE DIVISION.
METHOD-ID. Element-Get.
METHOD-ID. Element-Insert.
METHOD-ID. Element-PutAt.
METHOD-ID. Element-PutLast.
METHOD-ID. ElementNo-Get.
METHOD-ID. LastElement-Get.
METHOD-ID. Remove-All.
METHOD-ID. Remove-At.
END OBJECT.
END CLASS List.

```

Element-Getメソッド

指定された位置(インデックス)の要素を取り出します。

[パラメタ]

PIC 9(8) BINARY:

取り出す要素の位置を、先頭を1とした整数で指定します。

[復帰値]

USAGE OBJECT REFERENCE:

取り出した要素を返します。指定した位置の要素がなかった場合は、NULLを返します。

Element-Insertメソッド

指定された位置(インデックス)に要素を追加します。

[パラメタ]

PIC 9(8) BINARY:

要素を追加する位置を、先頭を1とした整数で指定します。なお、要素数+1より大きい数を指定した場合は、要素は追加されません。

USAGE OBJECT REFERENCE:

追加する要素を指定します。

[復帰値]

PIC 9(8) BINARY:

要素を追加した位置を、先頭を1とした整数で返します。要素が追加されなかった場合は、0を返します。

Element-PutAtメソッド

指定された位置(インデックス)の要素を置き換えます。

[パラメタ]

PIC 9(8) BINARY:

置き換える要素の位置を、先頭を1とした整数で指定します。なお、要素数より大きい数を指定した場合は、置き換えられません。

USAGE OBJECT REFERENCE:

置き換える要素を指定します。

[復帰値]

PIC 9(4) BINARY:

要素を置き換えた位置を、先頭を1とした整数で返します。要素が置き換えられなかった場合は、0を返します。

Element-PutLastメソッド

最後の要素の後に、要素を追加します。

【パラメタ】

USAGE OBJECT REFERENCE:

追加する要素を指定します。

【復帰値】

なし

ElementNo-Getメソッド

指定した要素の位置(インデックス)を調べます。

【パラメタ】

USAGE OBJECT REFERENCE:

位置を調べる要素を指定します。

【復帰値】

PIC 9(8) BINARY:

要素の位置を、先頭を1とした整数で返します。指定した要素が見つからなかった場合は0を返します。同じ要素が複数存在する場合は、最初に見つかった位置を返します。

LastElement-Getメソッド

最後の要素を取り出します。

【パラメタ】

なし

【復帰値】

USAGE OBJECT REFERENCE:

最後の要素を返します。要素数が0の場合は、NULLを返します。

Remove-Allメソッド

集合に含まれるすべての要素を削除します。

【パラメタ】

なし

【復帰値】

なし

Remove-Atメソッド

指定された位置(インデックス)の要素を削除します。

【パラメタ】

PIC 9(8) BINARY:

削除する要素の位置を、先頭を1とした整数で指定します。なお、要素数より大きい数を指定した場合は、削除されません。

【復帰値】

PIC 9(4) BINARY:

要素を削除した位置を、先頭を1とした整数で返します。削除されなかった場合は、0を返します。

提供プログラム

- COLLECT. COB (COBOLソースプログラム)
- DICT. COB (COBOLソースプログラム)
- LIST. COB (COBOLソースプログラム)
- BIN_TREE. COB (COBOLソースプログラム)
- D_NODE. COB (COBOLソースプログラム)

- L_NODE.COB (COBOLソースプログラム)
- SAMPLE16.PRJ (プロジェクトファイル)
- SAMPLE16.CBI (翻訳オプションファイル)
- SAMPLE16.TXT (プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
 - クラスの定義 (カプセル化)
 - 継承
 - オブジェクトの生成
 - メソッド呼出し
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- オブジェクトプロパティ
- メソッドの行内呼出し
- リポジトリ段落
- クラス定義、オブジェクト定義、メソッド定義

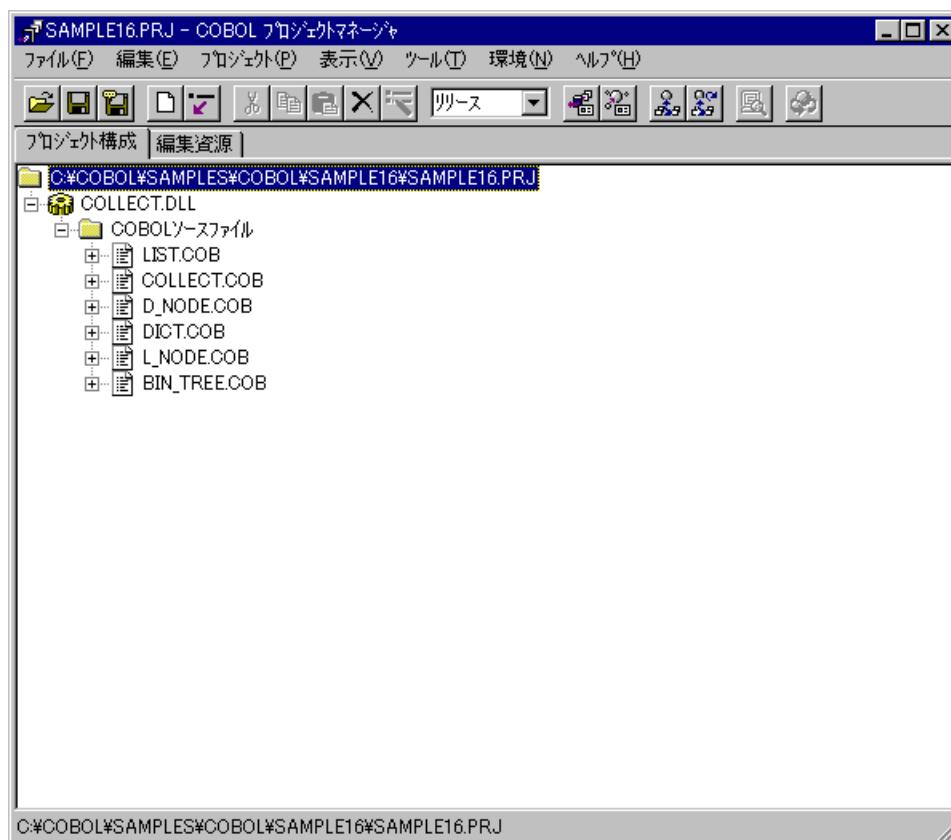
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用していきます。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル “SAMPLE16.PRJ” を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから「ビルド」を選択します。
→ ビルド終了後、以下のファイルが作成されることを確認してください。

- COLLECT. DLL
- COLLECT. LIB
- COLLECT. REP
- DICT. REP
- LIST. REP

備考

上記以外にも作成されるファイルがありますが、それらはクラスライブラリ使用时には必要ありません。

クラスライブラリの利用

サンプルクラスライブラリをプログラムに組み込んで使用する場合、以下のファイルが必要です。

翻訳時およびリンク時

- COLLECT. LIB (インポートライブラリ)
- COLLECT. REP (リポジトリファイル)
- DICT. REP (リポジトリファイル)
- LIST. REP (リポジトリファイル)

これらのファイルを、クラスライブラリを使用するプロジェクトに組み込んで使用します。[参照] “[例題17 オブジェクト指向プログラム\(中級編\)](#)” “[例題18 オブジェクト指向プログラム\(上級編\)](#)”

実行時

- COLLECT. DLL (ダイナミックリンクライブラリ)

例題17 オブジェクト指向プログラム（中級編）

ここでは、本製品で提供するサンプルプログラム-例題17-について説明します。

例題17では、集約、シングルトン、イテレータといったオブジェクト指向の一般的なデザインパターンを使用したプログラムの例を示します。

なお、このプログラムでは、“[例題16 コレクションクラス\(クラスライブラリ\)](#)”で作成したDictクラスとListクラスを使用しています。

また、このプログラムを動作させるためには、以下の製品が必要です。

- Microsoft (R) Excel (以降、Excelと略します。)

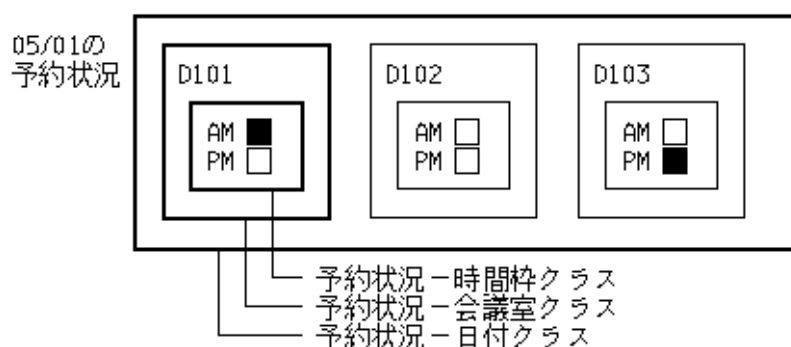
概要

会議室の予約処理(予約、予約取消、予約参照)および会議室管理処理(会議室情報の一覧表示、追加、更新、削除)を行います。これらの処理において、以下のデザインパターンを使用しています。

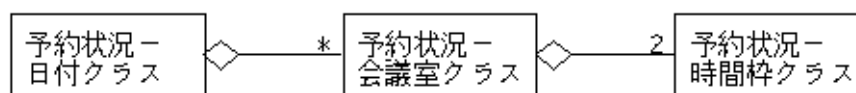
- 集約

集約関係とは、クラスの間に「全体-部分」の関係があることをいいます。

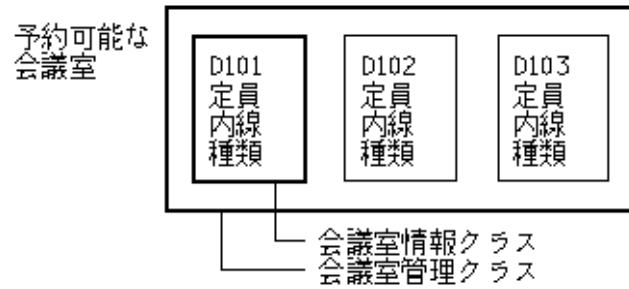
この例題では、日付ごとの予約状況を管理するために、以下のようなクラス関係を持っています。



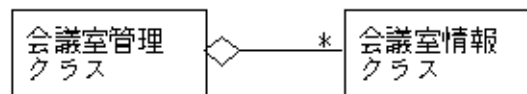
“予約状況-日付”クラスは複数の“予約状況-会議室”クラスを含み、“予約状況-会議室”クラスは複数(この場合は2つの)“予約状況-時間枠”クラスを含んでいます。そのため、以下の集約関係があるといえます。



また、予約できる会議室の情報を管理するために、以下のようなクラス関係も持っています。



“会議室管理”クラスは“会議室情報”クラスを含んでいます。そのため、これらのクラスの間にも集約関係があるといえます。



- シングルトン

シングルトンは、あるクラスのインスタンスが1つしか存在しないことを保証するための機構を提供します。この例題では、シングルトンクラスでこの機構を実装しています。シングルトンクラスはインスタンスを1つしか持たない以下のクラスで継承し、使用しています。

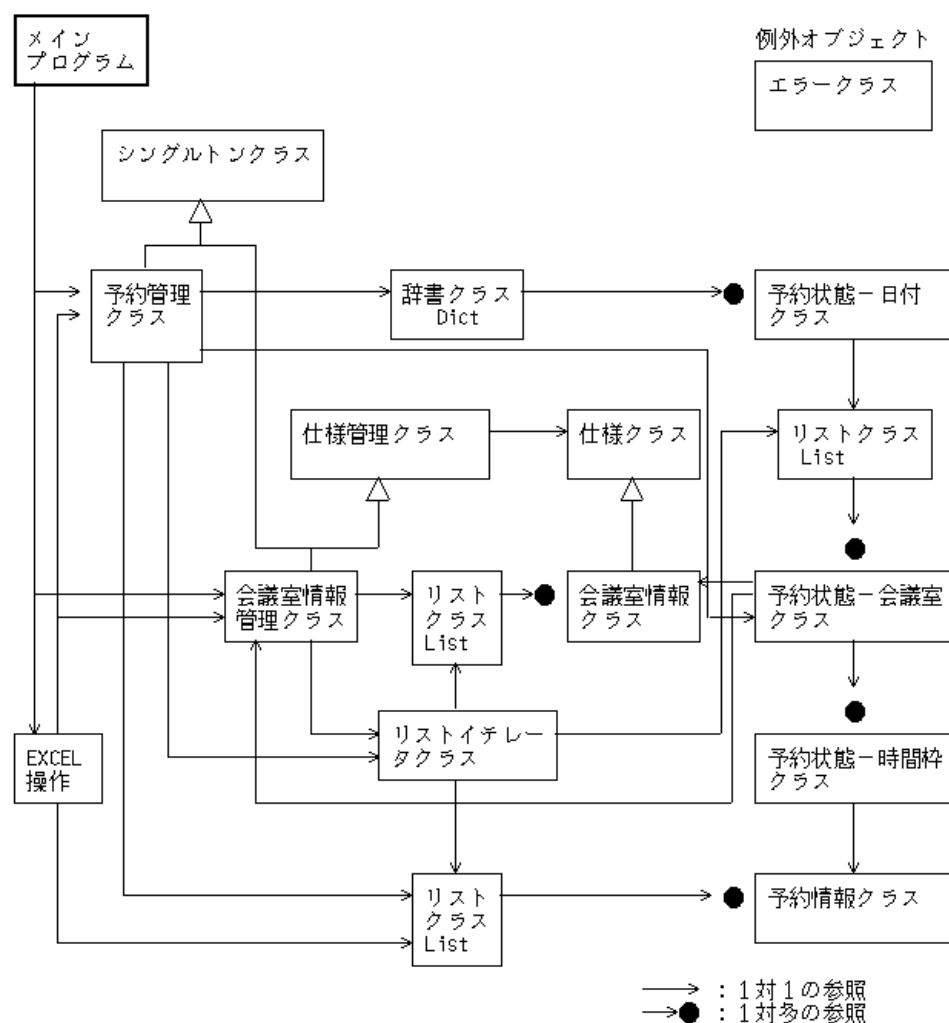
- 予約管理クラス
- 会議室情報管理クラス

- イテレータ

イテレータは、集約オブジェクトの内部構造を意識せずにその要素に順にアクセスする方法を提供します。この例題では、リストイテレータクラスでこれを実装しています。リストイテレータオブジェクトは1つのリストオブジェクトに対して複数作成することができます。この例題では、以下の処理でイテレータを使用しています。

- 予約状況の表示
- 予約状態オブジェクト(日付、会議室、時間枠)の削除
- 会議室情報、予約情報の検索

この例題では、これらの機能に加えて、会議室情報オブジェクトおよび予約情報オブジェクトをExcelファイルに格納し、永続化する機能を追加しています。



提供プログラム

- MAIN. COB (COBOL ソースプログラム)
- EXCELEDT. COB (COBOL ソースプログラム)
- RSVCTRL. COB (COBOL ソースプログラム)
- ROOMCTRL. COB (COBOL ソースプログラム)
- DATESTA. COB (COBOL ソースプログラム)
- ROOMSTA. COB (COBOL ソースプログラム)
- TIMESTA. COB (COBOL ソースプログラム)
- RESERVE. COB (COBOL ソースプログラム)
- SPECCTRL. COB (COBOL ソースプログラム)
- SPEC. COB (COBOL ソースプログラム)
- SINGLETN. COB (COBOL ソースプログラム)
- LISTITER. COB (COBOL ソースプログラム)
- ERRORPUT. COB (COBOL ソースプログラム)
- RSVINFO. CBL (登録集ファイル)
- ROOMINFO. CBL (登録集ファイル)
- R_CONST. CBL (登録集ファイル)
- SPECINFO. CBL (登録集ファイル)
- ROOMLIST. XLS (Excel ファイル)

- RSVLIST. XLS (Excelファイル)
- DICT. REP (リポジトリファイル)
- LIST. REP (リポジトリファイル)
- SAMPLE17. PRJ (プロジェクトファイル)
- COLLECT. DLL (DLLファイル)
- COLLECT. LIB (インポートライブラリ)
- SAMPLE17. TXT (プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
 - クラスの定義(カプセル化)
 - 継承
 - 多重継承
 - オブジェクトの生成
 - メソッド呼出し
 - 例外処理
 - COM連携(Excel連携)
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- オブジェクトプロパティ
- メソッドの行内呼出し
- リポジトリ段落
- クラス定義、ファクトリ定義、オブジェクト定義、メソッド定義
- 型定義

プログラムの翻訳・リンク・実行

プログラムを実行する前に

Excelファイルのファイル名の下線部分を、NetCOBOLをインストールしたフォルダの名前に書き換えてください。

<R_CONST. CBL>

会議室ファイル名	IS	"C:¥COBOL¥SAMPLES¥COBOL¥SAMPLE17¥RoomList. XLS"
予約ファイル名	IS	"C:¥COBOL¥SAMPLES¥COBOL¥SAMPLE17¥RsvList. XLS"

なお、RoomList. XLSおよびRsvList. XLSは、本プログラムの動作時には随時書き換えが行われます。必要に応じてバックアップをお取りください。



注意

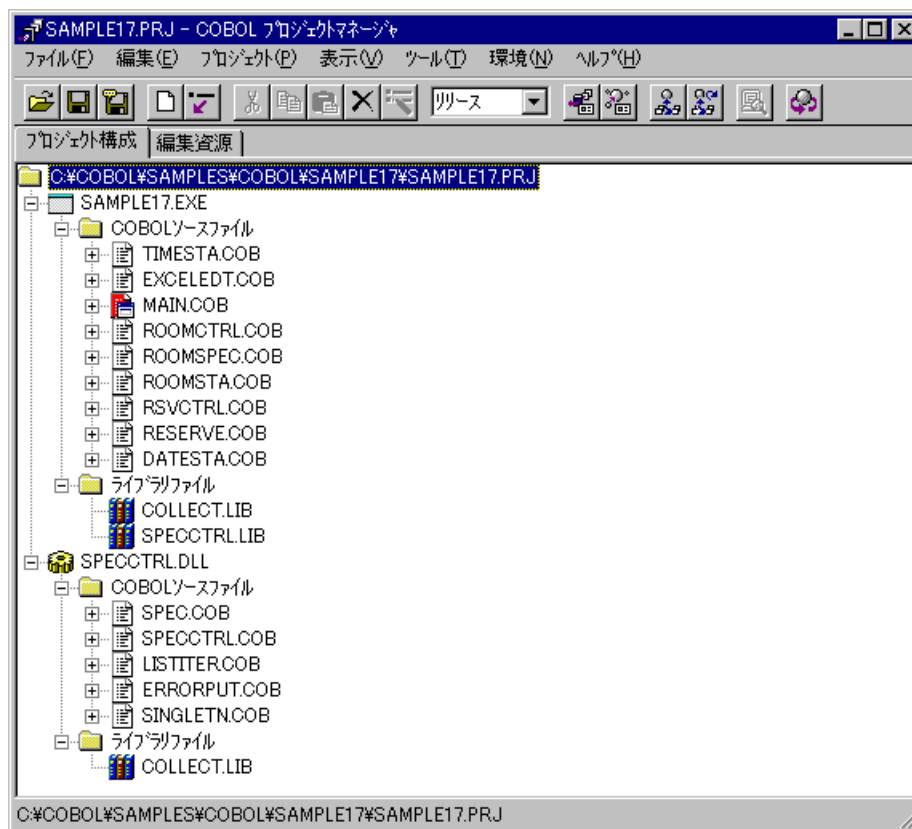
プログラムの仕様から、システムの現在日付より古い日付のデータはExcelファイルからの復元時に自動的に破棄されます。プログラム終了時に保存されるExcelファイルには破棄されたデータは反映されませんので注意してください。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:¥COBOLとして説明しています。フォルダ名がC:¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

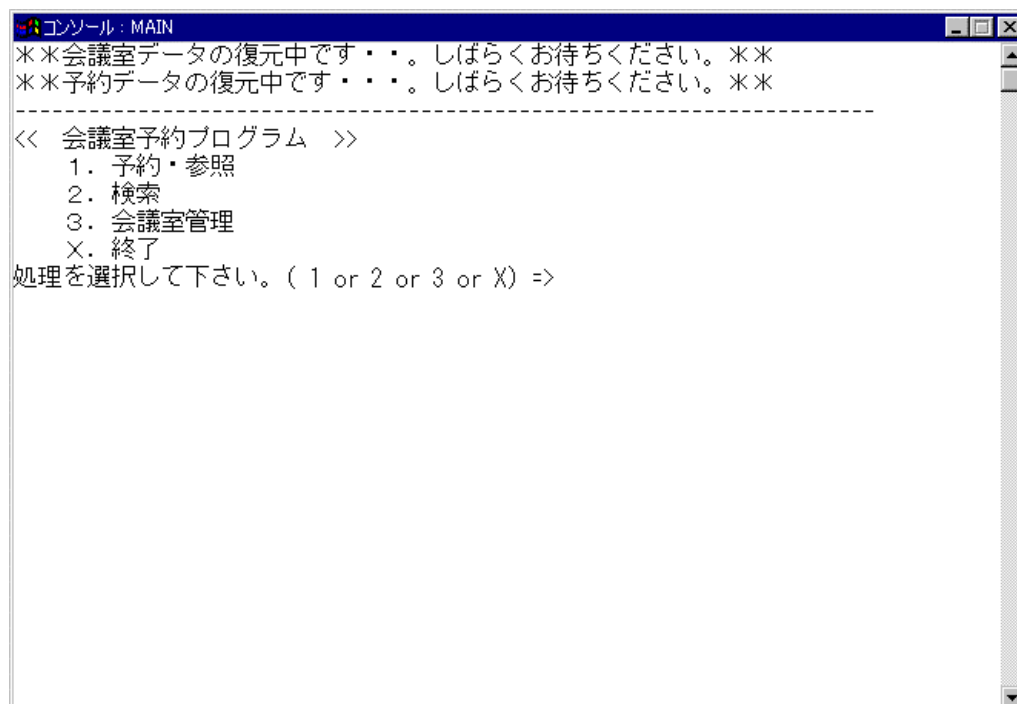
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE17. PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE17.EXEと各DLL(ダイナミックリンクライブラリ)が作成されていることを確認してください。

プログラムの実行

プロジェクトマネージャの「プロジェクト」メニューから“実行”を選択します。
→ Excelファイルに格納された会議室情報と予約情報が復元され、以下のメニューが表示されます。



実行する処理の番号を入力し、ENTERキーを押してください。

予約・参照・取り消し

会議室の予約、参照および取り消し処理を行います。

1. 予約・参照する日付(8桁の数字)を入力し、ENTERキーを押してください。
→ 指定された日付の会議室予約状況を表示します。ただし、システムの現在日付より古い日付は、入力エラーとなります。

```

コンソール: MAIN
<< 会議室予約プログラム >>
  1. 予約・参照
  2. 検索
  3. 会議室管理
  X. 終了
処理を選択して下さい。( 1 or 2 or 3 or X) => 1

-----
予約・参照する日付を入力してください。
(8桁数字で入力: 例.1999年3月18日 => 19990318と入力)
  予約日      : 19991202

-----
日付      : 19991202
会議室    : D101
  時間枠   : AM   : 空
  時間枠   : PM   : 予約
会議室    : D102
  時間枠   : AM   : 空
  時間枠   : PM   : 予約
会議室    : D103
  時間枠   : AM   : 空
  時間枠   : PM   : 空

-----
予約・参照する会議室名と時間枠を入力して下さい。
会議室名 (4桁英数字: 例. D101) : 
  
```

2. 予約する場合は、“空”状態の会議室名(4桁の英数字)および時間枠(AMまたはPM)を入力し、ENTERキーを押してください。
→ 予約情報の入力処理に移ります。
3. 予約者名(10文字以内の日本語文字または英数字)、内線(9桁の数字)、所属(10文字以内の日本語または英数字)を入力し、ENTERキーを押してください。
→ 予約情報が登録され、予約番号が通知されます。予約番号は、予約取消時に必要となります。

```

コンソール: MAIN
  時間枠   : AM   : 空
  時間枠   : PM   : 予約
会議室    : D103
  時間枠   : AM   : 空
  時間枠   : PM   : 空

-----
予約・参照する会議室名と時間枠を入力して下さい。
会議室名 (4桁英数字: 例. D101) : D103
時間枠   ('AM' or 'PM')       : AM

-----
<<予約情報の入力>>
予約者名 (10文字まで: 例.富士通太郎) : 富士通太郎
内線     (数字9桁まで: 例.1234-5678) : 4321-9876
所属     (10文字まで: 例.勤労課)     : サポート部

-----
予約できました。予約番号はキャンセル時に必要になりますので控えてお
ください。
*** 予約番号 *** : +0004
=====
-日付-----会議室---時間枠---予約者名-----内線-----所属-----
19991202  D103    AM      富士通太郎          4321-9876  サポート部
=====
予約・参照を終了するなら'X'を、続けるならEnterを入力して下さい。=>
  
```

4. 予約済の会議室の情報を参照する場合または予約を取り消す場合、“予約”状態の会議室名(4桁の英数字)および時間枠(AMまたはPM)を入力し、ENTERキーを押してください。
→ 予約情報が表示されます。
5. 予約を取り消す場合、“C”を入力後、予約時に通知された予約番号を入力し、ENTERキーを押してください。

```

コンソール: MAIN
時間枠 = PM : 予約
会議室 : D102
時間枠 = AM : 空き
時間枠 = PM : 予約
会議室 : D103
時間枠 = AM : 予約
時間枠 = PM : 空き

-----
予約・参照する会議室名と時間枠を入力して下さい。
会議室名 (4桁英数字: 例. D101) : D103
時間枠 ('AM' or 'PM') : AM

=====
-日付-----会議室---時間枠---予約者名-----内線-----所属-----
19991202 D103 AM 富士通一郎 4321-9876 サポート部

=====
-会議室名---定員---内線-----種類-----
D103 10 1234-1103 テレビ会議室

=====
キャンセルするなら'C'を、終了するならEnterを入力して下さい。=> C

キャンセルには予約時の予約番号が必要です。予約番号を入力して下さい。
予約番号 (数字4桁: 例.0001) : 0004
  
```

検索

予約者名から予約情報を検索し、予約情報を表示します。

1. 予約者名(10文字以内の日本語文字または英数字)を入力し、ENTERキーを押してください。
→ 予約情報の中から予約者名が一致した情報の一覧を表示します。

```

コンソール: MAIN
**会議室データの復元中です・・。しばらくお待ちください。**
**予約データの復元中です・・。しばらくお待ちください。**

<< 会議室予約プログラム >>
  1. 予約・参照
  2. 検索
  3. 会議室管理
  X. 終了
処理を選択して下さい。( 1 or 2 or 3 or X) => 2

<< 予約者情報検索 >>
予約した名前で予約情報を検索します。予約者名を入力して下さい。
予約者名 (10文字まで: 例.富士通太郎) : 富士通太郎

=====
-日付-----会議室---時間枠---予約者名-----内線-----所属-----
19991201 D101 AM 富士通太郎 1122-3344 勤労課

=====
-日付-----会議室---時間枠---予約者名-----内線-----所属-----
19991202 D102 PM 富士通太郎 1122-3344 勤労課

=====
検索処理を終了するなら'X'を、続けるならEnterを入力して下さい。=>
  
```




検索結果は5個までしか表示できません。表示個数を増やしたい場合には、以下を修正してください。

<R_CONST.CBL>

RSV-MAX IS 5

会議室管理

会議室情報の一覧表示、追加・更新・削除を行います。

```

=====
-日付-----会議室---時間枠---予約者名-----内線-----所属-----
19991202  D102    PM      富士通太郎      1122-3344  勤労課
=====
検索処理を終了するなら'X'を、続けるならEnterを入力して下さい。=> X
=====
<< 会議室予約プログラム >>
  1. 予約・参照
  2. 検索
  3. 会議室管理
  X. 終了
処理を選択して下さい。( 1 or 2 or 3 or X) => 3
=====
-会議室名---定員---内線-----種類-----
D101        10    1234-1101  一般会議室
D102        24    1234-1102  一般会議室
D103        10    1234-1103  テレビ会議室
=====
<< 会議室管理 >>
指定された会議室の追加・更新・削除を行います。会議室を入力してください。
会議室名（英数字4桁：例.D101）：
  
```

1. 会議室名(4桁の英数字)を入力し、ENTERキーを押してください。
→ 一覧に表示されなかった会議室名を入力すると、新規に登録するかどうかの確認後、会議室情報の登録処理に移ります。一覧に表示された会議室名を入力した場合、会議室情報の更新または削除処理に移ります。
2. 新規に登録する会議室の定員(2桁の数字)、内線(9桁の数字)、会議室の種類(一般会議室なら'N'、テレビ会議室なら'T')を入力し、ENTERキーを押してください。
→ 登録するかどうかの確認後、会議室情報が登録されます。

```

コンソール: MAIN
処理を選択して下さい。( 1 or 2 or 3 or X) => 3
=====
-会議室名---定員---内線-----種類-----
D101      10      1234-1101   一般会議室
D102      24      1234-1102   一般会議室
D103      10      1234-1103   テレビ会議室
=====
<< 会議室管理 >>
指定された会議室の追加・更新・削除を行います。会議室を入力してください。
  会議室名(英数字4桁: 例.D101) :   D104
=====
この会議室は未登録です。
新規に登録するなら'Y'を、終了するならEnterを入力して下さい。=> Y
=====
< 会議室情報の追加 >
  定員数   (数字2桁: 例.10 )       :   24
  内線番号 (数字9桁まで: 例.1234-5678) : 1234-1104
  種別     (テレビ会議室なら'T'、一般会議室なら'N') :   T
=====
-会議室名---定員---内線-----種類-----
D104      24      1234-1104   テレビ会議室
=====
上記の情報を登録するなら'Y'を、中止するならEnterを入力して下さい。=> Y
会議室管理処理を終了するなら'X'を、続けるならEnterを入力して下さい。=>

```

3. 会議室情報を更新する場合、“R”を入力後、更新する会議室の定員(2桁の数字)、内線(9桁の数字)、会議室の種類(一般会議室なら'N'、テレビ会議室なら'T')を入力し、ENTERキーを押してください。
→ 更新した情報に会議室情報が置き換えられます。
4. 会議室情報を削除する場合、“D”を入力し、ENTERキーを押してください。
→ 会議室情報は削除されます。

```

コンソール: MAIN
D104      24      1234-1104   テレビ会議室
=====
<< 会議室管理 >>
指定された会議室の追加・更新・削除を行います。会議室を入力してください。
  会議室名(英数字4桁: 例.D101) :   D104
=====
この会議室は既に登録されています。
データを更新するなら'R'を、削除するなら'D'を、中止するならEnterを入力
して下さい。=> R
=====
-会議室名---定員---内線-----種類-----
D104      24      1234-1104   テレビ会議室
=====
変更後のデータを入力して下さい。
  定員数   (数字2桁: 例.10 )       :   10
  内線番号 (数字9桁まで: 例.1234-5678) : 1234-1104
  種別     (テレビ会議室なら'T'、一般会議室なら'N') :   T
< 変更後の会議室情報 >
=====
-会議室名---定員---内線-----種類-----
D104      10      1234-1104   テレビ会議室
=====
会議室管理処理を終了するなら'X'を、続けるならEnterを入力して下さい。=>

```

会議室管理処理を終了すると、修正された会議室情報を反映するために以下の処理が行われます。

- 会議室情報オブジェクトと予約情報オブジェクトをExcelファイルに保存
- 予約状態オブジェクトの削除、および最新の会議室情報に合わせた予約状態オブジェクトの復元

終了

処理を終了します。予約情報を反映するために以下の処理が行われます。

- 予約情報オブジェクトをExcelファイルに保存

コンソール: MAIN

```
=====
-会議室名---定員---内線-----種類-----
D104      24    1234-1104  テレビ会議室
=====
変更後のデータを入力して下さい。
  定員数   (数字2桁: 例.10)      :   10
  内線番号 (数字9桁まで: 例.1234-5678) : 1234-1104
  種別     (テレビ会議室なら'T'、一般会議室なら'N') :   T
< 変更後の会議室情報 >
=====
-会議室名---定員---内線-----種類-----
D104      10    1234-1104  テレビ会議室
=====
会議室管理処理を終了するなら'X'を、続けるならEnterを入力して下さい。=> X
**会議室データの保存中・反映中です。しばらくお待ちください。**
=====
<< 会議室予約プログラム >>
  1. 予約・参照
  2. 検索
  3. 会議室管理
  X. 終了
処理を選択して下さい。( 1 or 2 or 3 or X) => X
**予約データを保存しています。**
```

例題18 オブジェクト指向プログラム（上級編）

ここでは、本製品で提供するサンプルプログラム-例題18-について説明します。

例題18では、カプセル化、継承、多態といったオブジェクト指向の特徴的な機能をすべて使用したプログラムの例を示します。

なお、このプログラムでは、複数の従業員オブジェクトを扱うために、“[例題16 コレクションクラス\(クラスライブラリ\)](#)”で作成したDictクラスを使用しています。

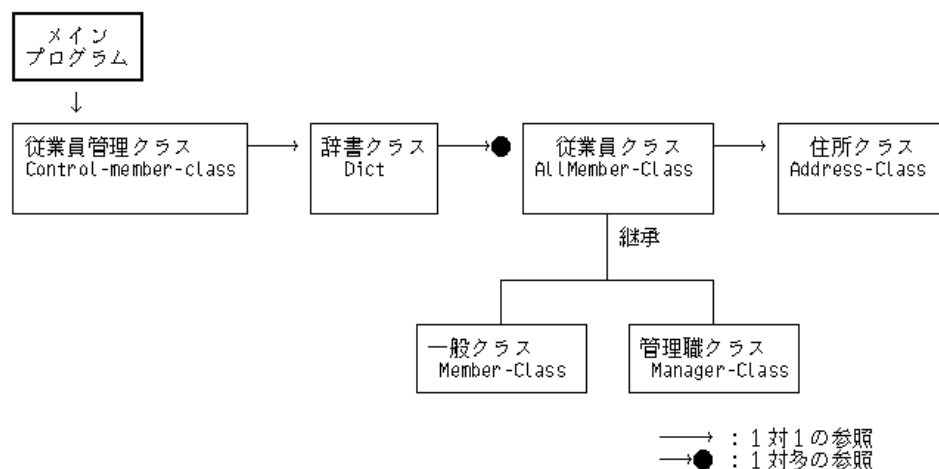
概要

従業員情報の管理(登録、削除、修正)、給与計算および住所録の印刷を行います。

従業員には、一般従業員と管理職があり、それぞれ保持するデータや給与計算方法が異なります。そのため、全従業員共通の属性を定義した従業員クラス(AllMember-class)、一般従業員固有の属性を定義した一般クラス(Member-Class)および管理職固有の属性を定義した管理職クラス(Manager-Class)の3つのクラスを用意しています。なお、一般クラスおよび管理職クラスは、従業員クラスを継承しています。

従業員情報のうち、住所については独立のクラス(住所クラス:Address-Class)で管理しています。そのため、従業員(およびその子クラス)のオブジェクトから住所オブジェクトを参照するようになっています。

管理対象となる従業員の数が多いと、その分の従業員オブジェクトを管理しなければなりません。そのために、“[例題16 コレクションクラス\(クラスライブラリ\)](#)”で作成した辞書クラス(Dict)を使用しています。従業員番号をキーとして従業員オブジェクトを辞書に登録しておきます。辞書クラスを使用すると、複数の従業員オブジェクトに対する繰り返し処理や、従業員オブジェクトの検索を簡単に行うことができます。



提供プログラム

- MAIN. COB (COBOL ソースプログラム)
- CTL_MEMB. COB (COBOL ソースプログラム)
- ALLMEM. COB (COBOL ソースプログラム)
- MEMBER. COB (COBOL ソースプログラム)
- MANAGER. COB (COBOL ソースプログラム)
- ADDRESS. COB (COBOL ソースプログラム)
- BONU_MAN. COB (COBOL ソースプログラム)
- SALA_MAN. COB (COBOL ソースプログラム)
- SALA_MEM. COB (COBOL ソースプログラム)
- DICT. REP (リポジトリファイル)

- LIST.REP(リポジトリファイル)
- SAMPLE18.PRJ(プロジェクトファイル)
- SAMPLE18.CBI(翻訳オプションファイル)
- COBOL85.CBR(実行用の初期化ファイル)
- COLLECT.DLL(DLLファイル)
- COLLECT.LIB(インポートライブラリ)
- COLLECT.REP(リポジトリファイル)
- SAMPLE18.KBD(キー定義ファイル)
- SAMPLE18.TXT(プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
 - クラスの定義(カプセル化)
 - 継承
 - オブジェクトの生成
 - メソッド呼出し
 - 多態
- スクリーン操作機能
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- オブジェクトプロパティ
- メソッドの行内呼出し
- リポジトリ段落
- クラス定義、オブジェクト定義、メソッド定義

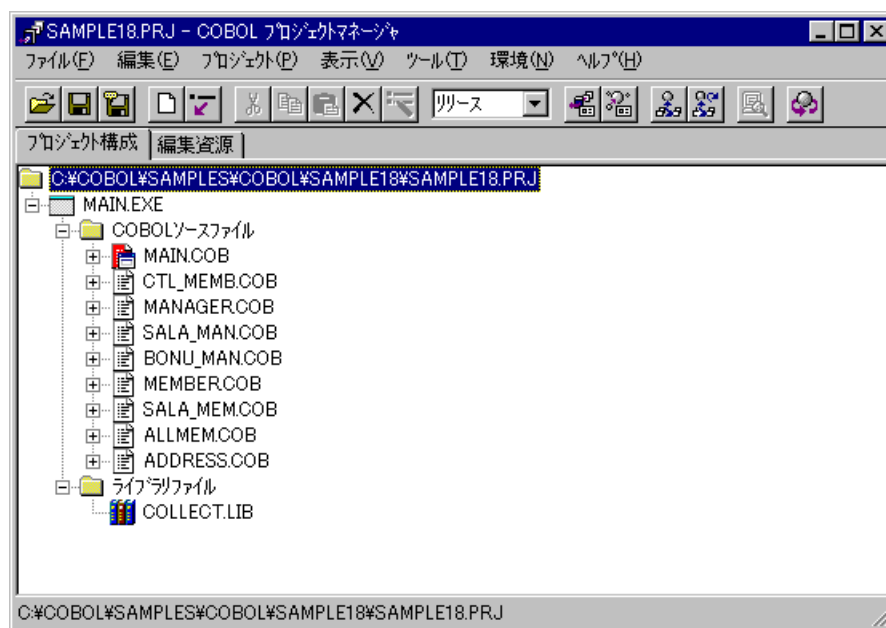
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE18.PRJ”を開きます。

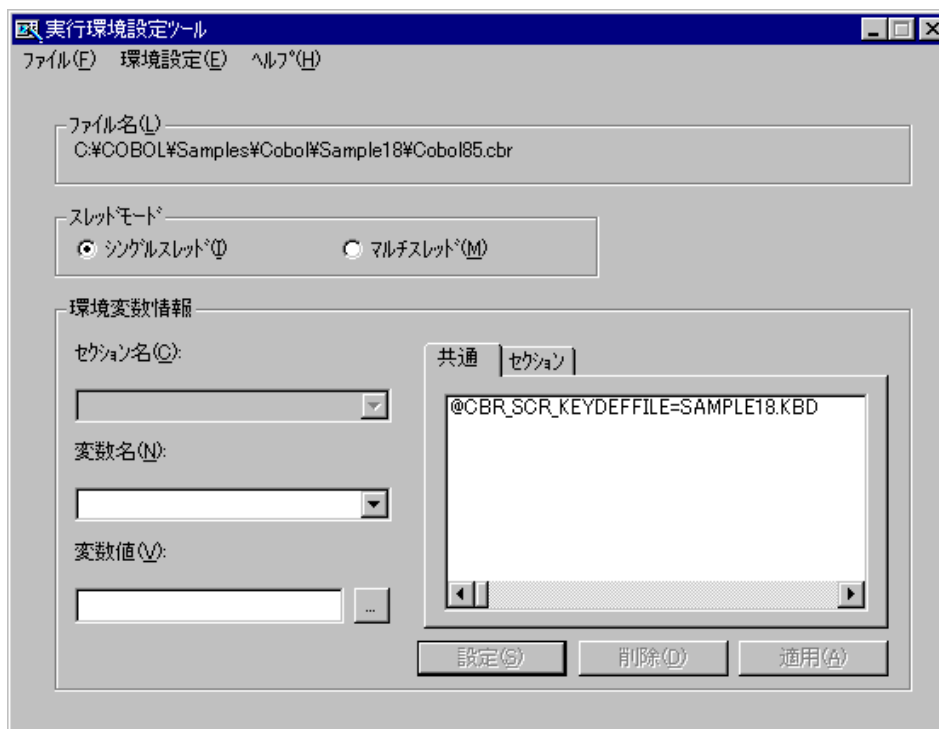


3. プロジェクトマネージャの「プロジェクト」メニューから「ビルド」を選択します。

→ ビルド終了後、MAIN.EXEが作成されていることを確認してください。

実行環境情報の設定

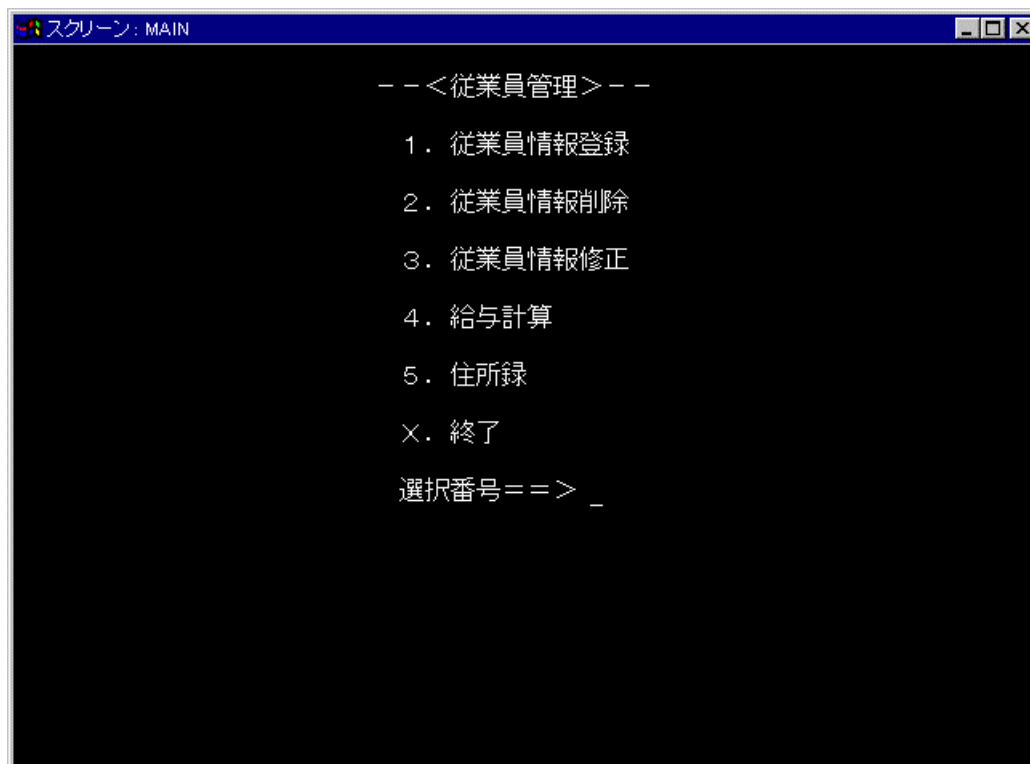
あらかじめ、以下の情報が設定されています。設定する必要はありません。



プログラムの実行

プロジェクトマネージャの「プロジェクト」メニューから“実行”を選択します。

→ 以下のメニューが表示されます。



“選択番号” に、実行する処理の番号を入力し、ENTERキーを押してください。

従業員情報登録

従業員情報を入力します。従業員番号(4桁の数字)、氏名(8文字以内の日本語文字)、住所(郵便番号と20文字以内の日本語文字)、入社年月日(YYYYMMDD形式)、職級(管理者の場合は1/一般社員の場合は2)および基本給(8桁以内の数字)を入力します。また、管理者の場合は特別手当(6桁以内の数字)を、一般社員の場合は時間外手当(5桁以内の数字)を入力します。最後にF3キーを押すと、データが登録されます。

```
スクリーン: MAIN
--<従業員登録>--

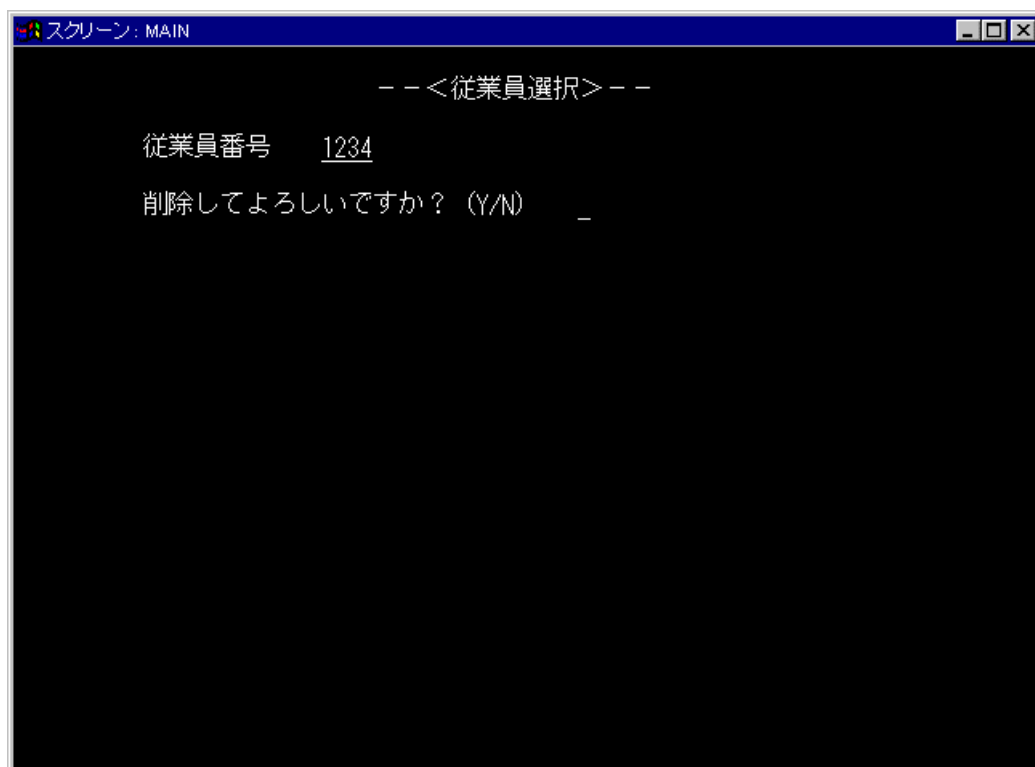
従業員番号  1234
氏名        富士通太郎
住所        410-0399 沼津市愛鷹69-1
入社年月日  19900401
職級        2
基本給      00225000
特別手当    000000
時間外手当  02150

F2: 終了  F3: 登録
```

従業員情報登録を終了する場合は、F2キーを押してください。

従業員情報削除

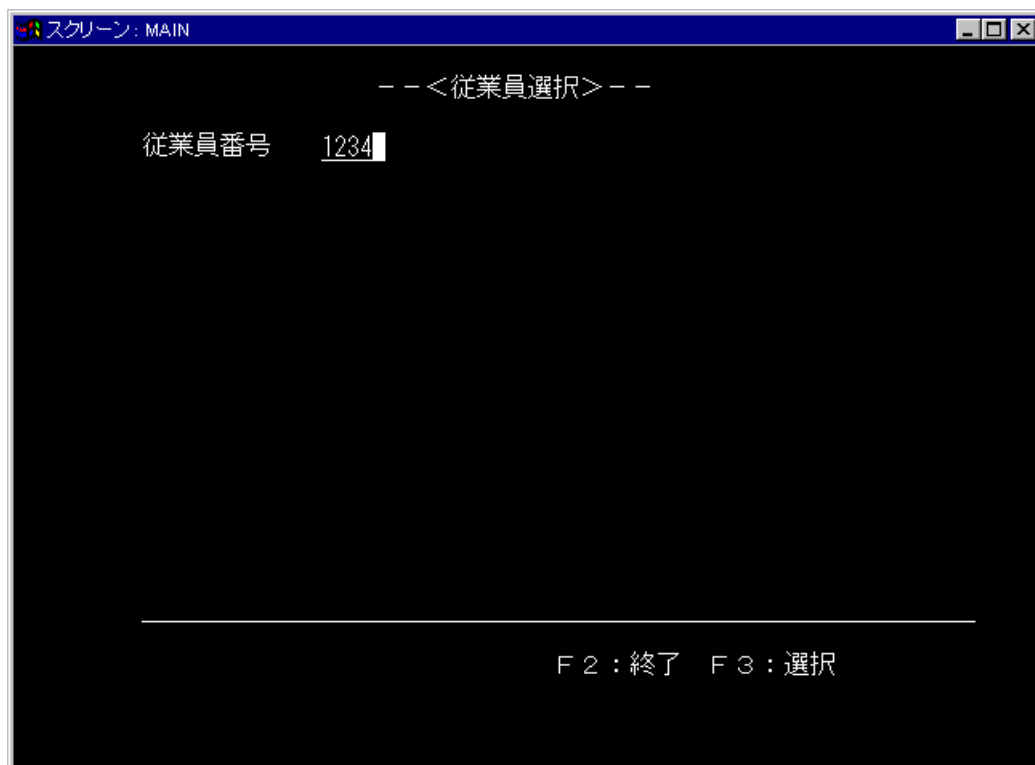
登録されている従業員情報を削除します。従業員番号(4桁の数字)を入力し、F3キーを押してください。



従業員情報削除を終了する場合は、F2キーを押してください。

従業員情報修正

登録済みの従業員情報を修正します。従業員番号(4桁の数字)を入力し、F3キーを押してください。



表示されたデータを修正します。この画面では、登録時の入力したデータ以外に、残業時間(整数部3桁、小数部1桁)を入力できます。F3キーを押すと、修正が反映されます。

スクリーン: MAIN

--<従業員修正>--

氏名	富士通太郎
住所	410-0399 沼津市愛鷹69-1
入社年月日	19900401
退社年月日	
職級	2
基本給	00225000
特別手当	000000
時間外手当	02150
残業時間	025.5

F2: 取消 F3: 修正

従業員情報修正を終了する場合は、F2キーを押してください。

給与計算

全社員の給与を計算します。

スクリーン: MAIN

--<給与計算結果の参照>--

給与計算処理が完了しました。

各従業員の給与情報を参照したい場合は、以下に従業員番号を入力して下さい。

従業員番号 1234

F2: 終了 F3: 選択

従業員番号(4桁の数字)を入力してF3キーを押すと、従業員ごとの給与情報を表示します。

スクリーン: MAIN

--<給与計算結果>--

従業員番号 1234

氏名 富士通太郎

職級 2

基本給 00225000

特別手当 000000

時間外手当 02150 × 残業時間 025.5

総支給 00279825

F 2 : 終了

給与計算を終了する場合は、F2キーを押してください。

住所録

従業員の住所録を印刷します。管理者(1)か、一般従業員(2)かを選択し、F3 キーを押してください。

スクリーン: MAIN

--<住所録印刷>--

1. 管理者

2. 一般社員

選択番号 ==> 2

F 2 : 終了 F 3 : 印刷

< 一般社員住所録 >			
従業員番号	氏 名	住 所	
1234	富士通太郎	410-0399	静岡県沼津市愛鷹6-9-1
:	:	:	:

終了

処理を終了します。

例題19 オブジェクトの永続化（ファイル）

ここでは、本製品で提供するサンプルプログラム-例題19-について説明します。

例題19では、“[例題18 オブジェクト指向プログラム\(上級編\)](#)”で作成したプログラムを基に、オブジェクトを永続化する例を示します。例題18では、オブジェクトはすべてメモリ上に作成されていました。そのため、プログラム終了時に、オブジェクトはすべて消えてしまいます。しかし、実際のシステムでは、プログラム終了後もデータは残っていなければなりません。つまり、一部のオブジェクトはプログラムの実行をまたがって存在し続けなければなりません。このようなオブジェクトを、“永続オブジェクト”と呼びます。

従来のプログラムでは、このようなデータはファイルやデータベースに格納されていました。永続オブジェクトの一般的な実現方法は、永続化するオブジェクトをファイルやデータベースのデータに対応付けることです。この例では、それぞれのオブジェクトを索引ファイルのレコードに対応付けることにより、永続オブジェクトを実現しています。

オブジェクトの永続化の詳細については、“NetCOBOL 使用手引書”の“17.5 オブジェクトの永続化”を参照してください。

概要

例題18と同様に、従業員情報の管理(登録、削除、修正)、給与計算および住所録の印刷を行います。

この例題では、これらの機能に加えて、従業員オブジェクトおよび住所オブジェクトを索引ファイルに格納し、永続化する機能を追加しています。これらの機能は従業員オブジェクトおよび住所オブジェクトに、以下のメソッドを追加することで実現しています。

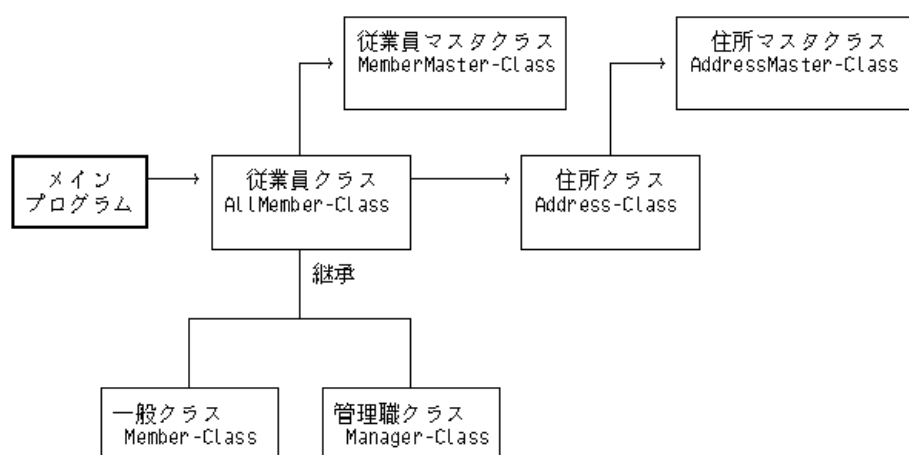
ファクトリメソッド

- RetAt-Methodメソッド(従業員番号をキーに、オブジェクトをファイルから読み込む)
- RemoveAt-Methodメソッド(従業員番号をキーに、オブジェクトをファイルから削除する)

オブジェクトメソッド

- Store-Methodメソッド(オブジェクトをファイルに格納する)

実際には、従業員マスタクラス、住所マスタクラスと連携しながら永続化を行っていますが、オブジェクトの使用者(この例の場合は、メインプログラム)からは従業員オブジェクトがすべて行っているように見えます。



→ : 1対1の参照

提供プログラム

- MAIN.COB (COBOL ソースプログラム)

- ALLMEM. COB (COBOLソースプログラム)
- MEMBER. COB (COBOLソースプログラム)
- MANAGER. COB (COBOLソースプログラム)
- ADDRESS. COB (COBOLソースプログラム)
- SET. COB (COBOLソースプログラム)
- STORE. COB (COBOLソースプログラム)
- ALLMEM_M. COB (COBOLソースプログラム)
- ALLMEMMF (データファイル)
- BONU_MAN. COB (COBOLソースプログラム)
- BONU_MEM. COB (COBOLソースプログラム)
- MEM_SET. COB (COBOLソースプログラム)
- MEM_STOR. COB (COBOLソースプログラム)
- MAN_SET. COB (COBOLソースプログラム)
- MAN_STOR. COB (COBOLソースプログラム)
- ADDR_M. COB (COBOLソースプログラム)
- ADDR_MF (データファイル)
- SALA_MAN. COB (COBOLソースプログラム)
- SALA_MEM. COB (COBOLソースプログラム)
- SAMPLE19. PRJ (プロジェクトファイル)
- SAMPLE19. CBI (翻訳オプションファイル)
- COBOL85. CBR (実行用の初期化ファイル)
- SAMPLE19. KBD (キー定義ファイル)
- SAMPLE19. TXT (プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
- 索引ファイル機能
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE文、SET文
- オブジェクトプロパティ
- メソッドの行内呼出し
- リポジトリ段落
- クラス定義、オブジェクト定義、ファクトリ定義、メソッド定義

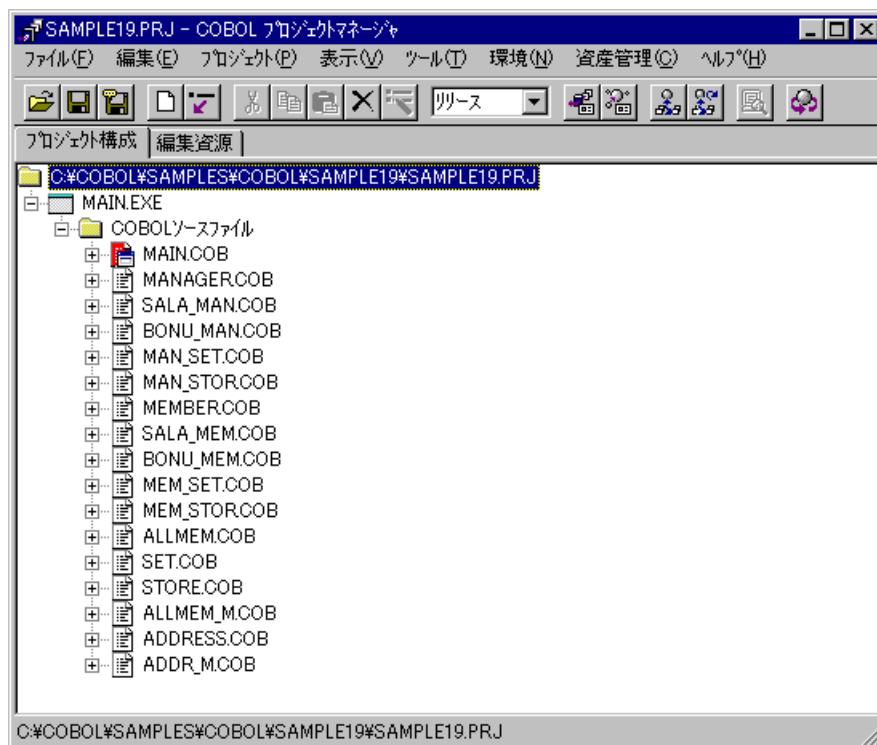
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

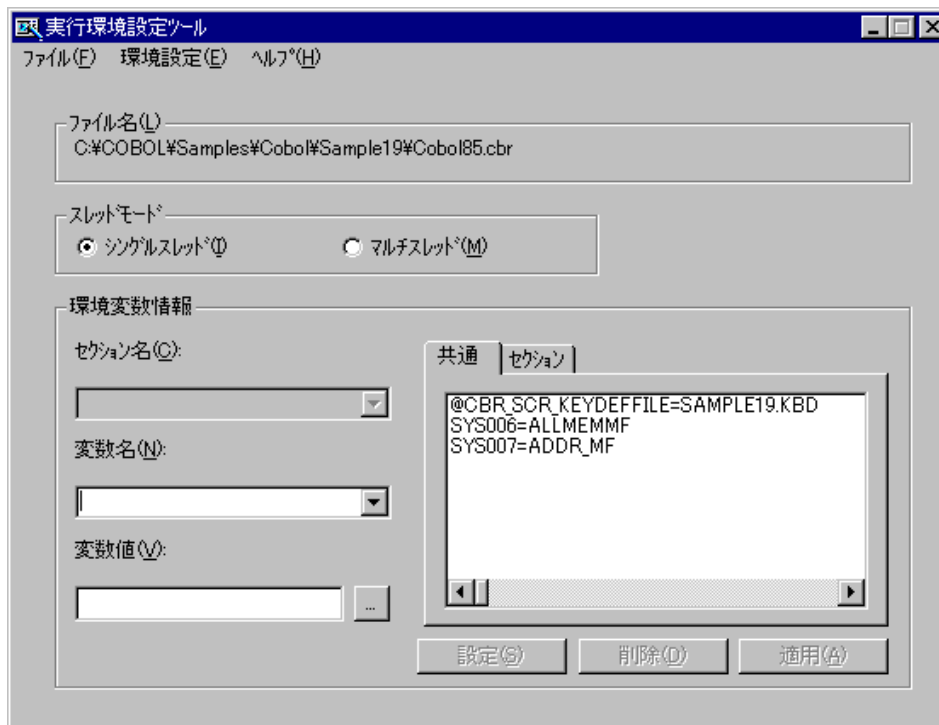
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE19. PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、MAIN.EXEが作成されていることを確認してください。

実行環境情報の設定

あらかじめ、以下の情報が設定されています。設定する必要はありません。



プログラムの実行

プロジェクトマネージャの「プロジェクト」メニューから“実行”を選択します。
実行手順については、“[例題18 オブジェクト指向プログラム\(上級編\)](#)”を参照してください。

例題20 オブジェクトの永続化（データベース）

ここでは、本製品で提供するサンプルプログラム-例題20-について説明します。

例題20では、“[例題18 オブジェクト指向プログラム\(上級編\)](#)”で作成したプログラムを基に、オブジェクトを永続化する例を示します。例題18では、オブジェクトはすべてメモリ上に作成されていました。そのため、プログラム終了時に、オブジェクトはすべて消えてしまいます。しかし、実際のシステムでは、プログラム終了後もデータは残っていなければなりません。つまり、一部のオブジェクトはプログラムの実行をまたがって存在し続けなければなりません。このようなオブジェクトを、“永続オブジェクト”と呼びます。

従来のプログラムでは、このようなデータはファイルやデータベースに格納されていました。永続オブジェクトの一般的な実現方法は、永続化するオブジェクトをファイルやデータベースのデータに対応付けることです。この例では、それぞれのオブジェクトをデータベース表の行に対応付けることにより、永続オブジェクトを実現しています。

オブジェクトの永続化の詳細については、“NetCOBOL 使用手引書”の“17.5 オブジェクトの永続化”を参照してください。

データベースはサーバ上に存在し、クライアント側からこれにアクセスします。

データベースのアクセスは、ODBCドライバを経由して行います。ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL 使用手引書”の“第21章 リモートデータベースアクセス(ODBC)”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

クライアント側

- ODBCドライバマネージャ
- ODBCドライバ
- ODBCドライバの必要とする製品

サーバ側

- データベース
- データベースにODBCでアクセスするために必要な製品

概要

例題18と同様に、従業員情報の管理(登録、削除、修正)、給与計算および住所録の印刷を行います。

この例題では、これらの機能に加えて、従業員オブジェクトおよび住所オブジェクトをデータベースに格納し、永続化する機能を追加しています。これらの機能は従業員オブジェクトおよび住所オブジェクトに、以下のメソッドを追加することにより実現しています。

オブジェクトメソッド

- Store-Methodメソッド(オブジェクトをデータベースに格納する)
- RetAt-Methodメソッド(従業員番号をキーに、オブジェクトをデータベースから読み込む)
- RemoveAt-Methodメソッド(従業員番号をキーに、オブジェクトをデータベースから削除する)
- Update-Methodメソッド(データベースに格納されるオブジェクトを更新する)

提供プログラム

- MAIN. COB (COBOL ソースプログラム)
- ALLMEM. COB (COBOL ソースプログラム)
- MEMBER. COB (COBOL ソースプログラム)
- MANAGER. COB (COBOL ソースプログラム)
- ADDRESS. COB (COBOL ソースプログラム)
- SET. COB (COBOL ソースプログラム)
- ALLMEM_M. COB (COBOL ソースプログラム)

- BONU_MAN. COB (COBOL ソースプログラム)
- BONU_MEM. COB (COBOL ソースプログラム)
- MEM_SET. COB (COBOL ソースプログラム)
- MEM_STOR. COB (COBOL ソースプログラム)
- MAN_SET. COB (COBOL ソースプログラム)
- MAN_STOR. COB (COBOL ソースプログラム)
- ADDR_M. COB (COBOL ソースプログラム)
- SALA_MAN. COB (COBOL ソースプログラム)
- SALA_MEM. COB (COBOL ソースプログラム)
- SAMPLE20. PRJ (プロジェクトファイル)
- SAMPLE20. CBI (翻訳オプションファイル)
- COBOL85. CBR (実行用の初期化ファイル)
- SAMPLE20. KBD (キー定義ファイル)
- SAMPLE20. TXT (プログラム説明書)

使用しているCOBOLの機能

- オブジェクト指向プログラミング機能
- リモートデータベースアクセス (ODBC) 機能
- プロジェクト管理機能

使用しているオブジェクト指向の文/段落/定義

- INVOKE 文、SET 文
- オブジェクトプロパティ
- メソッドの行内呼出し
- リポジトリ段落
- クラス定義、オブジェクト定義、ファクトリ定義、メソッド定義

プログラムの翻訳・リンク・実行

プログラムを実行する前に

ODBC ドライバを経由してサーバのデータベースへアクセスできる環境を構築しておいてください。

デフォルトで接続するサーバを設定し、そのサーバのデータベース上に“住所表”および“従業員表”という名前の2つの表を作成しておいてください。

各表は、以下の形式で作成してください。

住所表

住所識別	郵便番号	住所	←列の名前
10進整数 4桁	固定長文字 7バイト	固定長文字 40バイト	←列の属性

従業員表

従業員番号	氏名	入社年月日	退社年月日	職級	基本給
10進整数 4桁	固定長文字 16バイト	固定長文字 8バイト	固定長文字 8バイト	10進整数 1桁	10進整数 8桁

以下に続く→

→続き

総支給	時間外手当	残業時間	特別手当	←列の名前
10進整数 8桁	10進整数 5桁	10進整数 5桁 小数部 1 桁	10進整数 6桁	←列の属性

ODBC情報ファイル設定ツール(SQLODBCS.EXE)を使用して、ODBC情報ファイルを作成してください。
Microsoft(R) SQLServer(TM)を使用する場合は、@SQL_CONCURRENCY(カーソルの同時実行)に、LOCK、
ROWVERまたはVALUESを指定してください。

**注意**

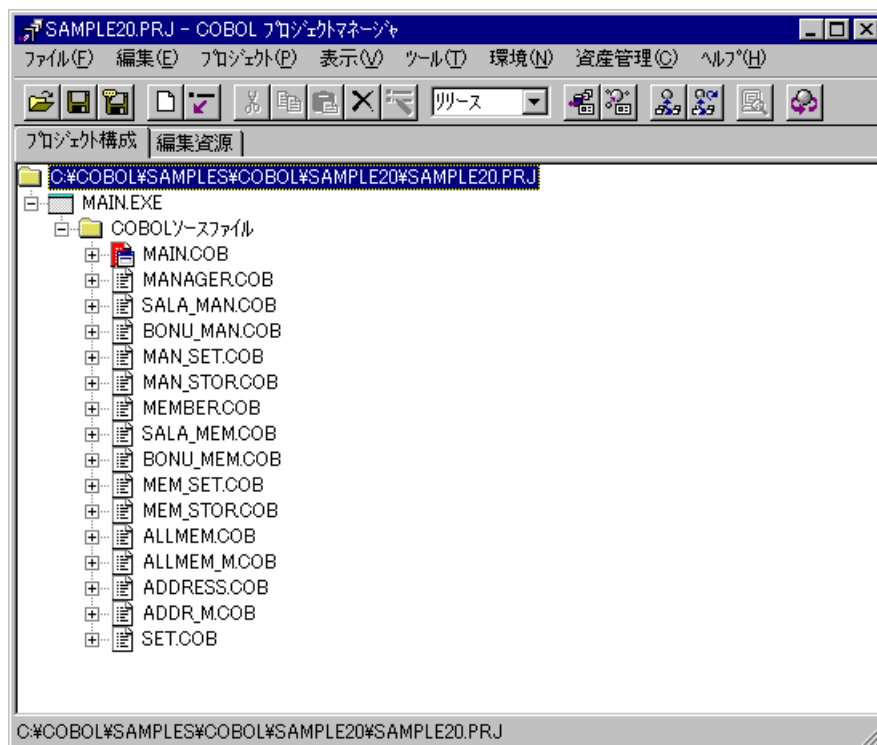
ODBC情報設定ツールでは、サーバ情報の拡張オプション“カーソルライブラリを使用する”を有効にする必要があります。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

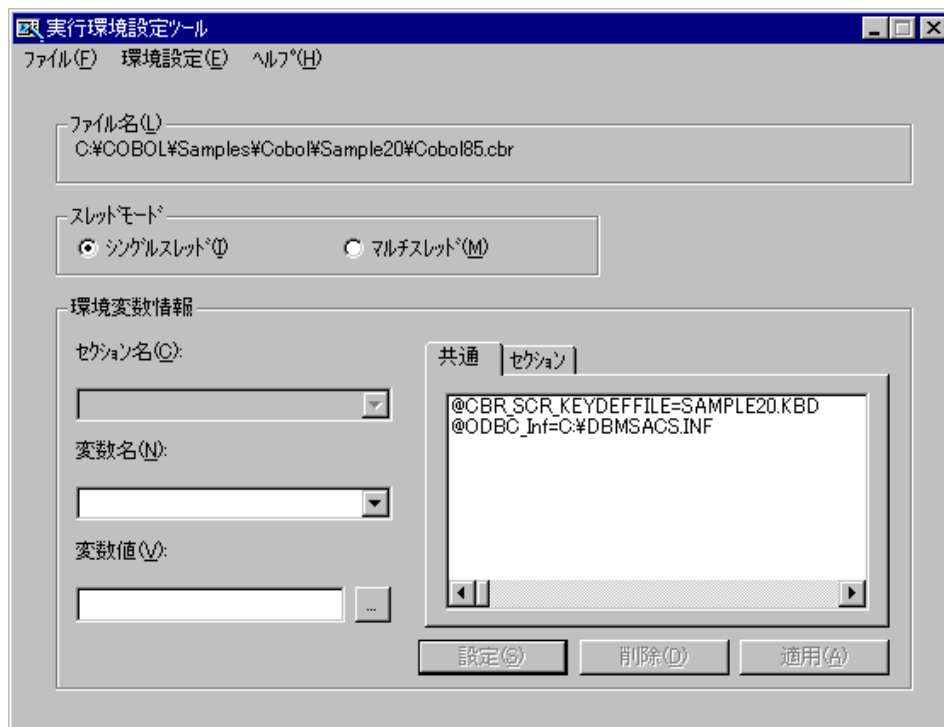
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE20.PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、MAIN.EXEが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの「ツール」メニューから“実行可能設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. 「ファイル」メニューの“開く”を選択し、実行可能プログラム (MAIN.EXE) が存在するフォルダの、実行用の初期化ファイル (COBOL85.CBR) を開きます。(
3. 共通タブを選択し、以下を設定します。
— 環境変数情報@ODBC_Inf (ODBC情報ファイルの指定) に、ODBC情報ファイル名を指定

- します。
- 環境変数情報@CBR_SCR_KEYDEFFILE(スクリーン操作のキー定義ファイルの指定)に、キー定義ファイル名を指定します。(SAMPLE20. KBD)
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
 5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

プロジェクトマネージャの[プロジェクト] メニューから“実行”を選択します。

実行手順については、“[例題18 オブジェクト指向プログラム\(上級編\)](#)”を参照してください。

ただし、以下の点に注意してください。

- 氏名および住所の入力の際、日本語文字は使用できません。英大文字、英小文字または数字を使用してください。
- データベースへのアクセス(接続、取り出しなど)の際、エラーが発生した場合にはエラー内容が出力され、実行が終了します。



注意

例題プログラムでは、カーソルを使用したUPDATE文(位置付け)が記述されています。したがって、UPDATE文(位置付け)が使用できない環境では、プログラムを修正する必要があります。

修正箇所は、提供プログラム(MAIN. COB)の給与計算処理SECTIONです。修正内容については、提供プログラム(MAIN. COB)の給与計算処理SECTIONを参照してください。

例題21 マルチスレッドプログラミング

ここでは、本製品で提供するサンプルプログラム-例題21-について説明します。

例題21では、NetCOBOLのマルチスレッドプログラミング機能を使って、スレッド間でリソース(ファイル・データ)の共有を行ったり、スレッド間の同期制御を行うプログラムの例を示します。NetCOBOLのマルチスレッドプログラミング機能の詳細については“NetCOBOL 使用手引書”の“第23章 マルチスレッド”を参照してください。

また、例題21のプログラムは、Webアプリケーションでもあります。これはCOBOLアプリケーションでマルチスレッドプログラミングが必要となる典型的な例が、COBOLでWebアプリケーションを構築する場合であるからです。Web連携機能の詳細については、“NetCOBOL Web連携ガイド”、“COBOL Webサブルーチン使用手引書”を参照してください。

なお、このプログラムを動作させるためには、クライアント側・サーバ側で以下の製品が必要となります。

クライアント側

- WWWブラウザ
 - Microsoft(R) Internet Explorer 4.0以上
 - または
 - Netscape Navigator(TM) 4.0以上

サーバ側

- 以下のいずれかの製品
 - Windows NT(R) Server
 - Windows(R) 2000 Server
 - Windows Server(TM) 2003
- Microsoft(R) Internet Information Server 4.0以上
(Windows NT(R) Serverの場合には、Microsoft(R) Windows NT(R) OptionPack 4.0が必要です。)

概要

サンプルプログラムは、次の3つの部分からなります。

- 開始処理
スレッド間でのリソース(ファイル・データ)を獲得し、初期設定をします。
- 認証処理
スレッド間でのリソース(ファイル・データ)を参照して、認証処理を実現します。
- 終了処理
スレッド間でのリソース(ファイル・データ)を開放します。

それぞれ、Web連携機能を使用するプログラムから、スレッド間でリソース(ファイル・データ)の共有を行ったり、スレッド間の同期制御を行うプログラムを呼び出します。

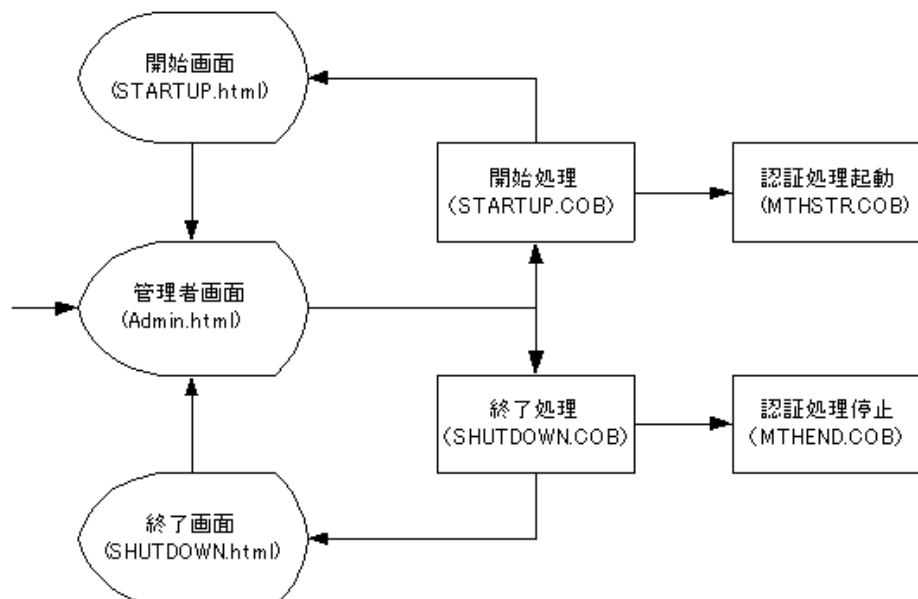
提供ファイル一覧

- プロジェクトファイル
 - ISAPIAPL.PRJ
 - MTHAPL.PRJ
- オプションファイル
 - ISAPIAPL.CBI
 - MTHAPL.CBI
- COBOLソースファイル
 - AUTH.COB
 - ISAINIT.COB
 - ISATERM.COB
 - MTHEND.COB

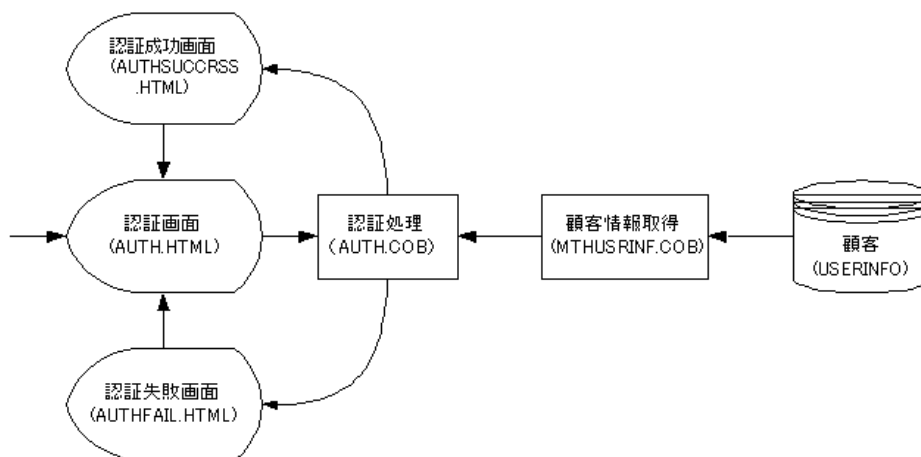
- MTHSTR.COB
- MTHUSRINF.COB
- SHUTDOWN.COB
- STARTUP.COB
- STUPINIT.COB
- 登録集原文
 - USER-INFO.CBL
 - USER-LOCK.CBL
- モジュール定義ファイル
 - AUTH.DEF
 - SHUTDOWN.DEF
 - STARTUP.DEF
- データファイル
 - USERINFO
- 実行用の初期化ファイル
 - COBOL85.CBR
- HTMLファイル
 - ADMIN.HTML
 - AUTH.HTML
 - AUTHFAIL.HTML
 - AUTHSUCCESS.HTML
 - NOTOPENED.HTML
 - OPENED.HTML
 - SHUTDOWN.HTML
 - STARTUP.HTML
 - SYSERROR.HTML
 - SYSTEMERROR.HTML
- プログラム説明書
 - SAMPLE21.TXT

プログラムの呼出し関係

業務開始・終了



認証サービス



使用しているCOBOLの機能

- 索引ファイル(参照)
- 外部データ
- 外部ファイル
- データロックサブルーチン
- COBOL ISAPIサブルーチン

使用しているCOBOLの文

CALL文、CLOSE文、EXIT文、GO TO文、IF文、MOVE文、OPEN文、PERFORM文、READ文、SET文

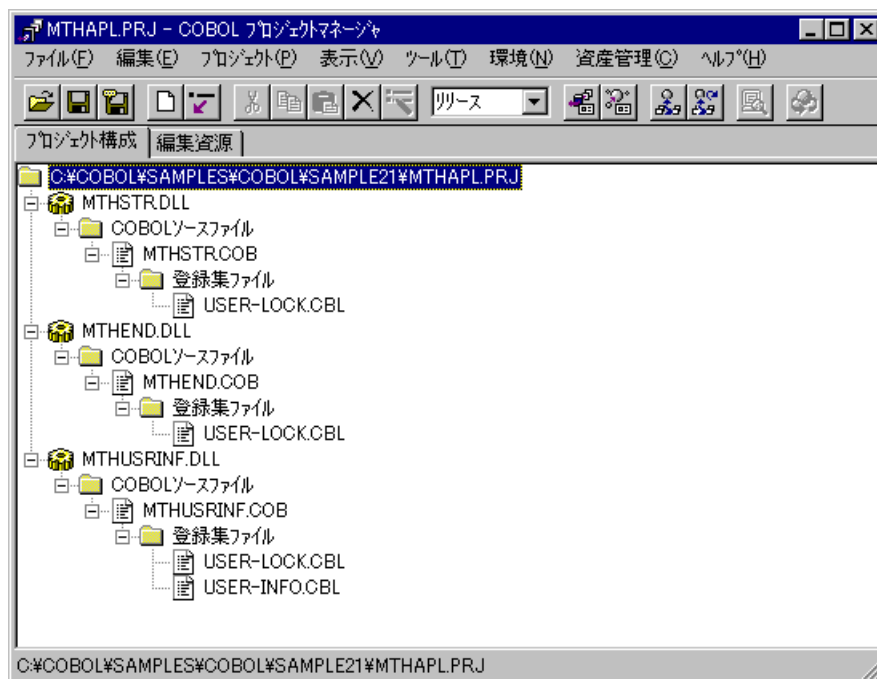
プログラムの翻訳・リンク・実行

ビルド・リビルド

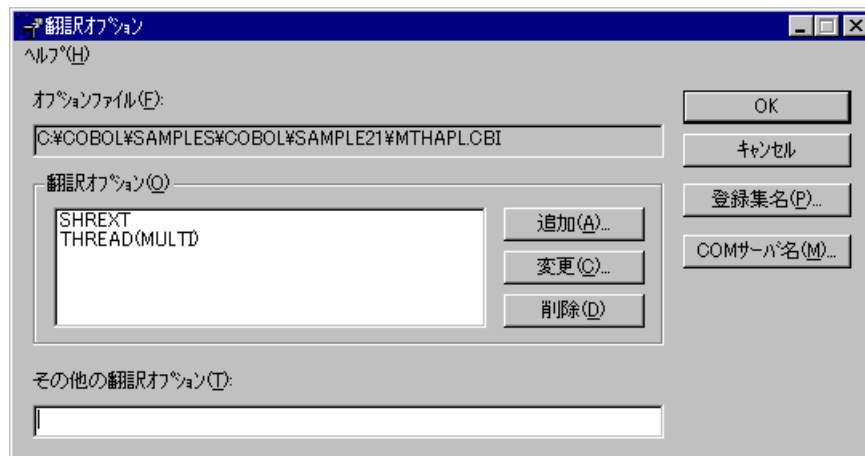
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

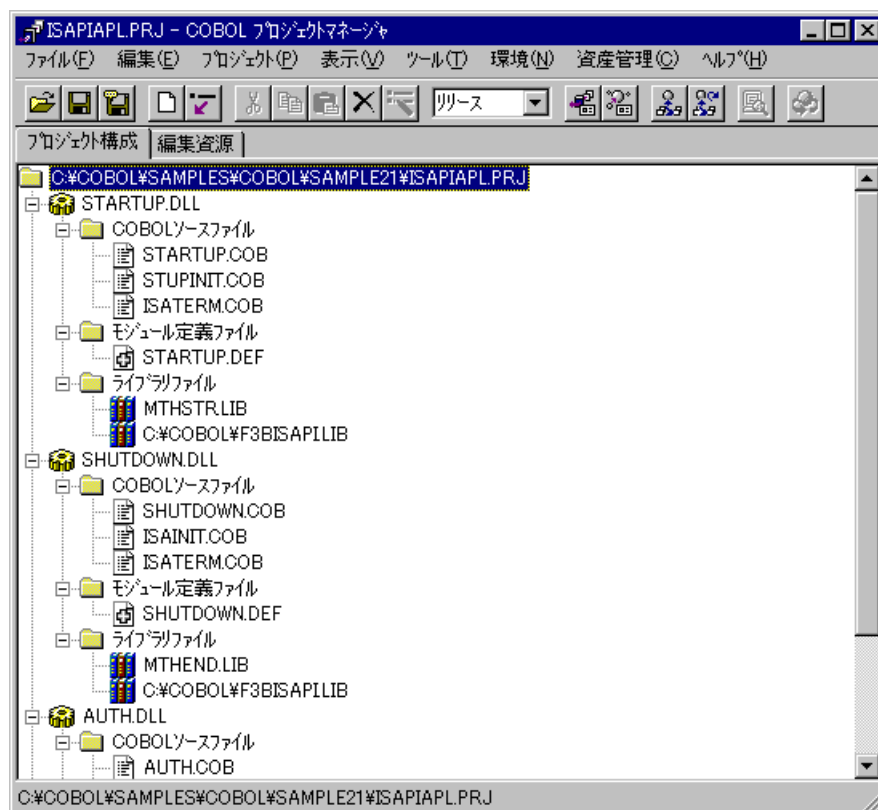
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“MTHAPL.PRJ”を開きます。



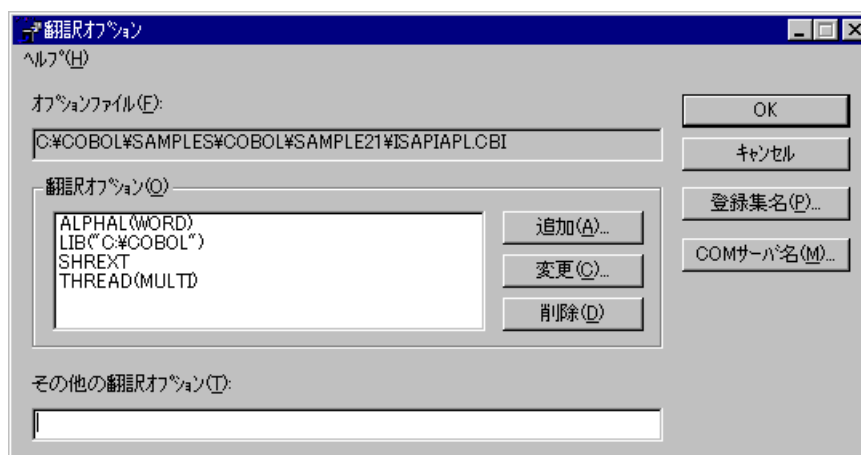
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションTHREAD (MULTI)、SHREXTを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。
→ プロジェクトに登録した各DLL(ダイナミックリンクライブラリ)が作成されていることを確認してください。
6. プロジェクトファイル“ISAPIAPL.PRJ”を開きます。



7. 3. と同じ手順で [翻訳オプション] ダイアログを表示し、翻訳オプションTHREAD (MULTI)、SHREXT、ALPHAL (WORD)を指定します。また、翻訳オプションLIBに登録集ファイルのフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。



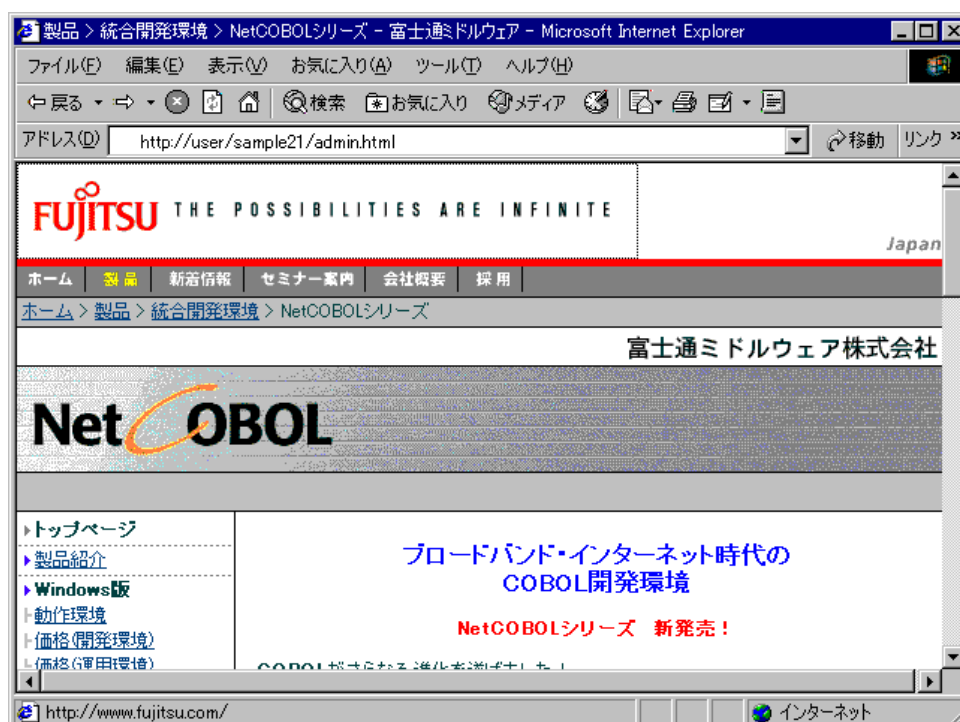
8. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ プロジェクトに登録した各DLL(ダイナミックリンクライブラリ)が作成されていることを確認してください。

プログラムの実行

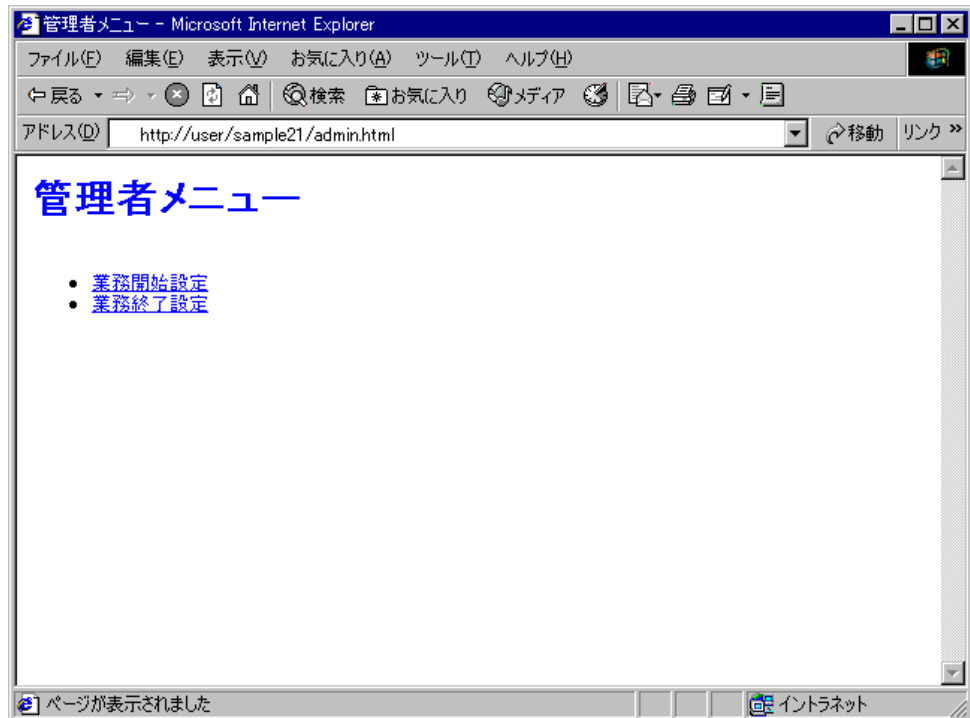
ここでは、ドメイン名を“user”、仮想ディレクトリ名を“sample21”としてIIS(Internet Information Server)に登録しています。

WWWブラウザは、Microsoft(R) Internet Explorerを使用しています。

1. 認証サービスを開始します。
URLに以下の情報を設定します。



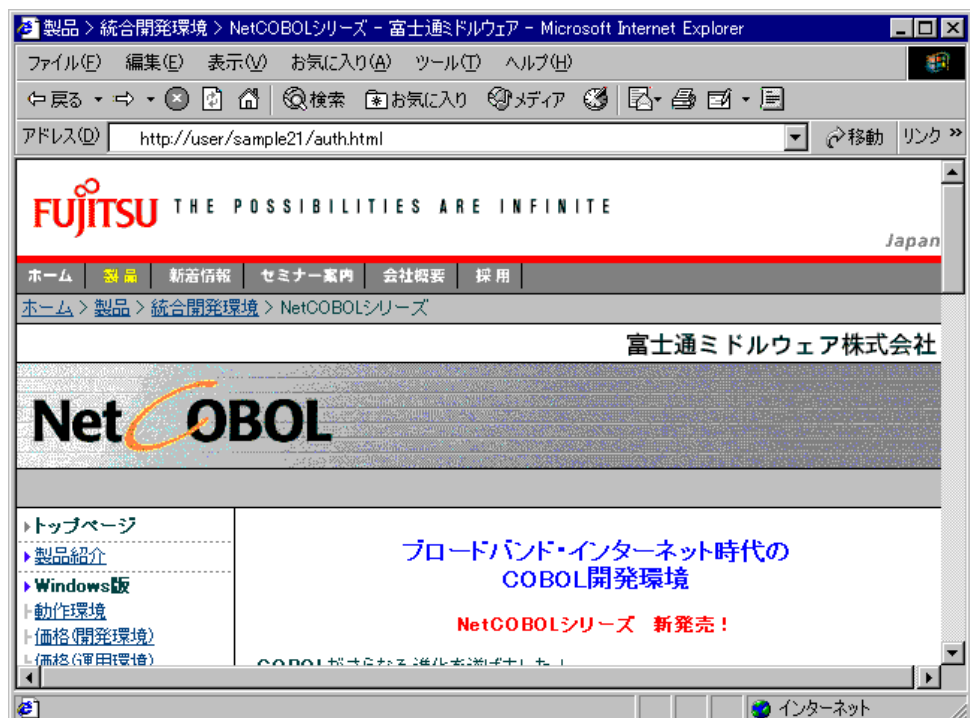
管理者メニューの画面が表示されるので、“業務開始設定”をクリックします。



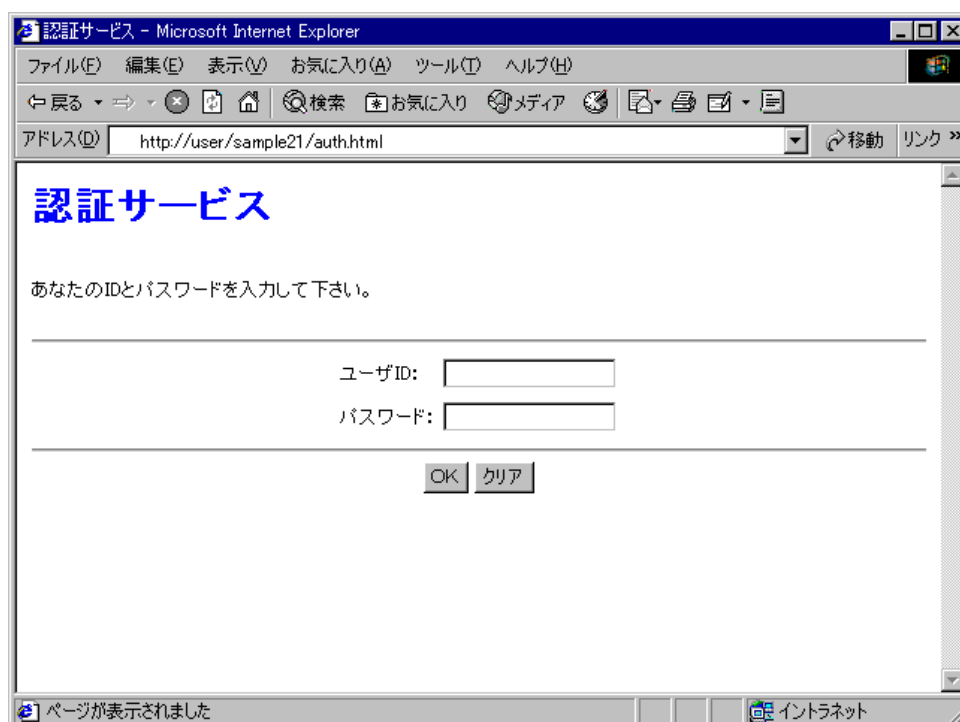
“業務開始設定”をクリックすると、認証サービスが開始されます。認証サービス起動する前に、必ず業務開始設定を行ってください。

2. 認証サービスを起動します。

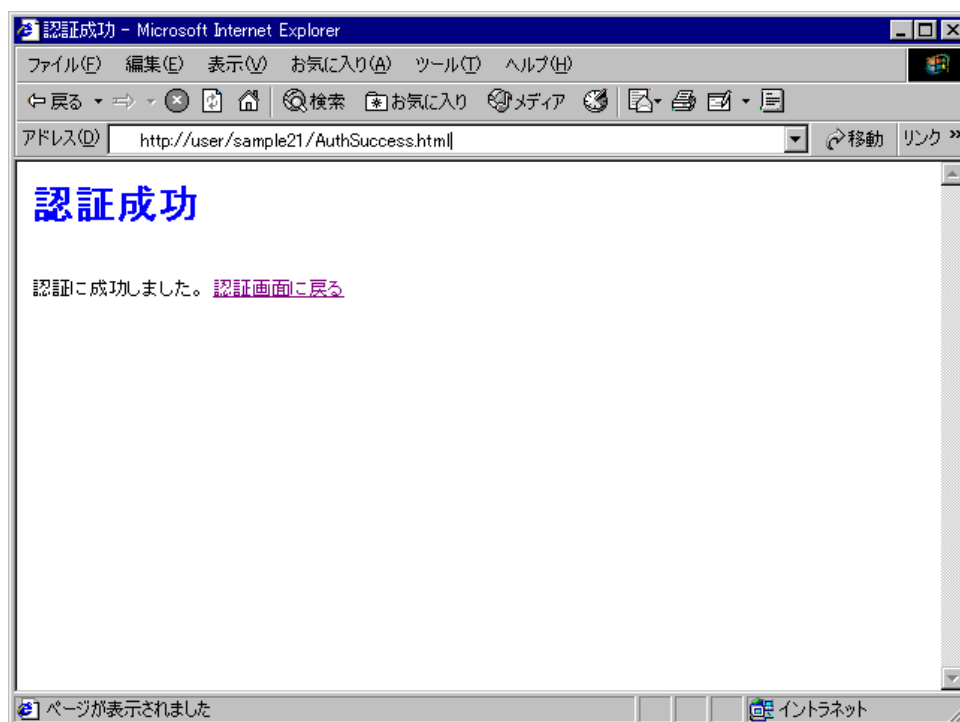
URLに以下の情報を設定して実行キーを押します。



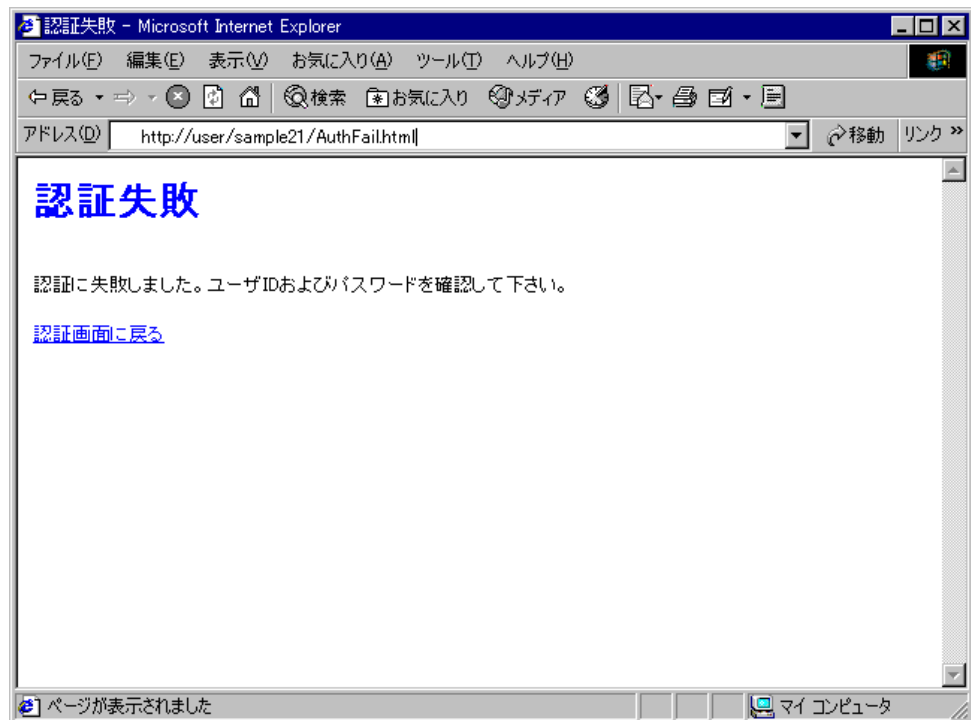
認証サービス画面が表示されます。画面が表示されたら、ユーザIDとパスワードを入力して〔OK〕ボタンをクリックします。ここで、入力できるユーザIDはUSER0001からUSER0030までです。パスワードはユーザIDと同じです。



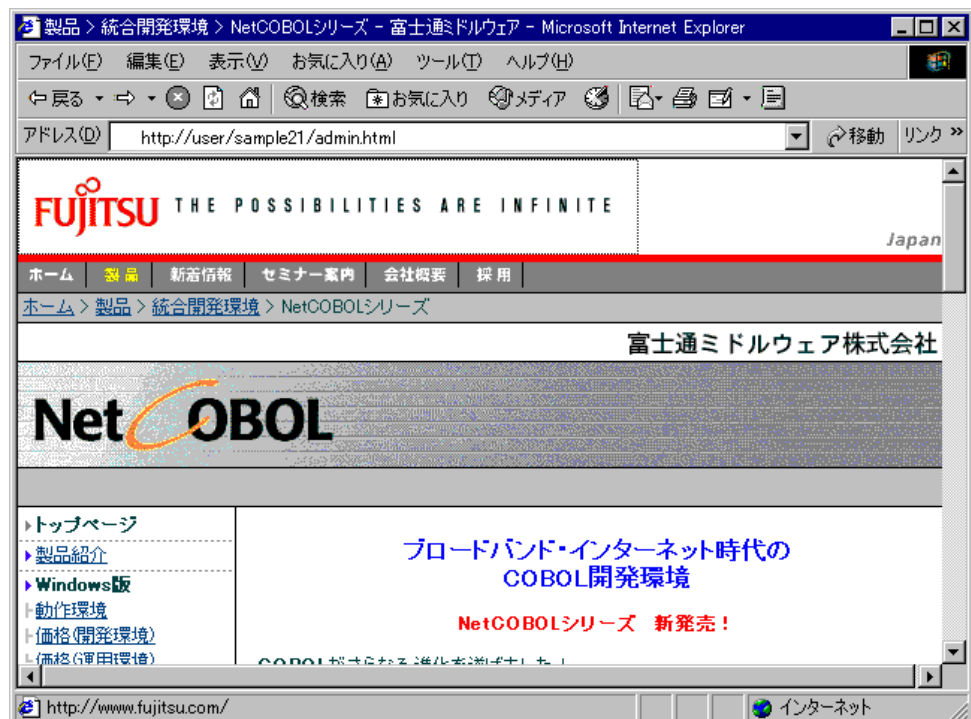
〔OK〕 ボタンをクリックすると認証成功画面が表示されます。



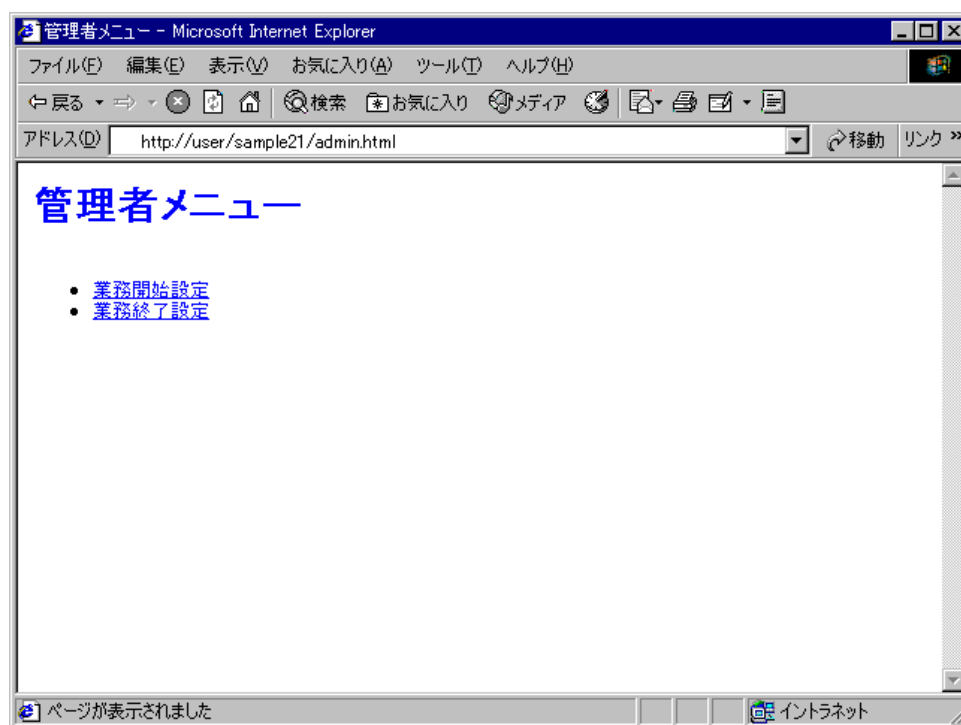
ユーザIDあるいはパスワードに不適切な値を入力して〔OK〕 ボタンをクリックすると認証失敗の画面が表示されます。



3. 認証サービスを終了します。
URLに以下の情報を設定して実行キーを押します。



管理者メニューの画面が表示されるので、“業務終了設定”をクリックします。



例題22 マルチスレッドプログラミング（応用編）

ここでは、本製品で提供するサンプルプログラム-例題22-について説明します。

例題22では、例題21で作成したプログラムを拡張して、オンラインストアの業務を実現するアプリケーションの例を示します。

ここで示すプログラムは、NetCOBOLのマルチスレッドプログラミング機能とWeb連携機能を使用したものです。マルチスレッドプログラミング機能の詳細については“NetCOBOL 使用手引書”の“第23章 マルチスレッド”を、Web連携機能の詳細については、“NetCOBOL Web連携ガイド”、“COBOL Webサブルーチン使用手引書”を参照してください。

また、サーバアプリケーションの運用に有効な機能であるイベントログ出力サブルーチンの使用方法も合わせて示します。イベントログ出力サブルーチンの詳細については“NetCOBOL 使用手引書”の“付録H COBOLが提供するサブルーチン”を参照してください。

なお、このプログラムを動作させるためには、クライアント側・サーバ側で以下の製品が必要となります。

クライアント側

- Microsoft(R) Internet Explorer 4.0以上
または
- Netscape Navigator(TM) 4.0以上

サーバ側

- 以下のいずれかの製品
 - Windows NT(R) Server
 - Windows(R) 2000 Server
 - Windows Server(TM) 2003
- Microsoft(R) Internet Information Server 4.0以上
(Windows NT(R) Serverの場合には、Microsoft(R) Windows NT(R) OptionPack 4.0が必要です。)

概要

サンプルプログラムは、次の5つの部分からなります。

- 開始処理
スレッド間でのリソース(ファイル・データ)を獲得し、初期設定をします。
- 認証処理
スレッド間でのリソース(ファイル・データ)を参照して、認証処理を実現します。
- オーダー確認処理
スレッド間でのリソース(ファイル・データ)を参照して、オーダー確認処理を実現します。
- オーダー発行処理
スレッド間でのリソース(ファイル・データ)を参照して、オーダー発行処理を実現します。
共有するファイルの更新処理も行います。
- 終了処理
スレッド間でのリソース(ファイル・データ)を開放します。

それぞれ、Web連携機能を使用するプログラムから、スレッド間でリソース(ファイル・データ)の共有を行ったり、スレッド間の同期制御を行うプログラムを呼び出します。

提供ファイル一覧

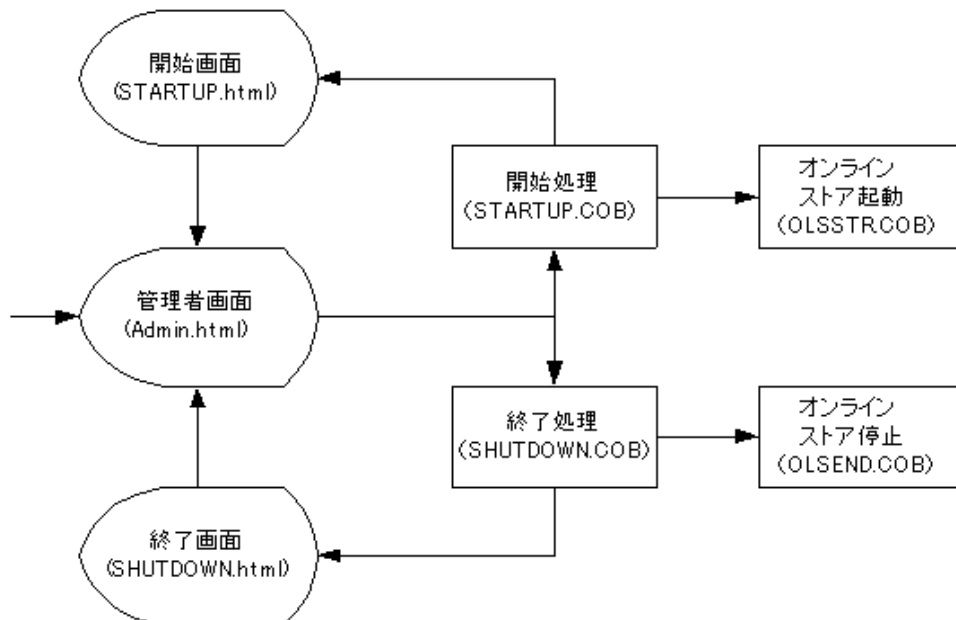
- プロジェクトファイル
 - ISAPIAPL.PRJ
 - OLSAPL.PRJ
- オプションファイル
 - ISAPIAPL.CBI
 - OLSAPL.CBI

- COBOLソースファイル
 - AUTH. COB
 - CONFIRM. COB
 - ENTRY. COB
 - ISAINIT. COB
 - ISATERM. COB
 - OLSEND. COB
 - OLSSTCGT. COB
 - OLSPRDGT. COB
 - OLSSTCODR. COB
 - OLSSTR. COB
 - OLSUSRINF. COB
 - SHUTDOWN. COB
 - STARTUP. COB
 - STUPINIT. COB
- 登録集原文
 - ORDER-INFO. CBL
 - PRODUCT-INFO. CBL
 - STOCK-INFO. CBL
 - USER-INFO. CBL
 - USER-LOCK. CBL
 - USER-LOG. CBL
- モジュール定義ファイル
 - AUTH. DEF
 - CONFIRM. DEF
 - ENTRY. DEF
 - SHUTDOWN. DEF
 - STARTUP. DEF
- データファイル
 - STOCKINFO
 - PRODUCTINFO
 - USERINFO
- 実行用の初期化ファイル
 - COBOL85. CBR
- HTMLファイル
 - ADMIN. HTML
 - AUTH. HTML
 - AUTHFAIL. HTML
 - CATALOG. HTML
 - CONFIRM. HTML
 - CONFIRMDetailPARTS. HTML
 - CONFIRMHEAD. HTML
 - CONFIRMTAIL. HTML
 - ILLIGALACCESS. HTML
 - ILLIGALSYSTEM. HTML
 - NOTOPENED. HTML
 - OPENED. HTML
 - ORDERDETAILPARTS. HTML
 - ORDERRESULTHEAD. HTML
 - ORDERRESULTTAIL. HTML
 - SHOPPINGMENU. HTML
 - SHORTAGESTOCK. HTML

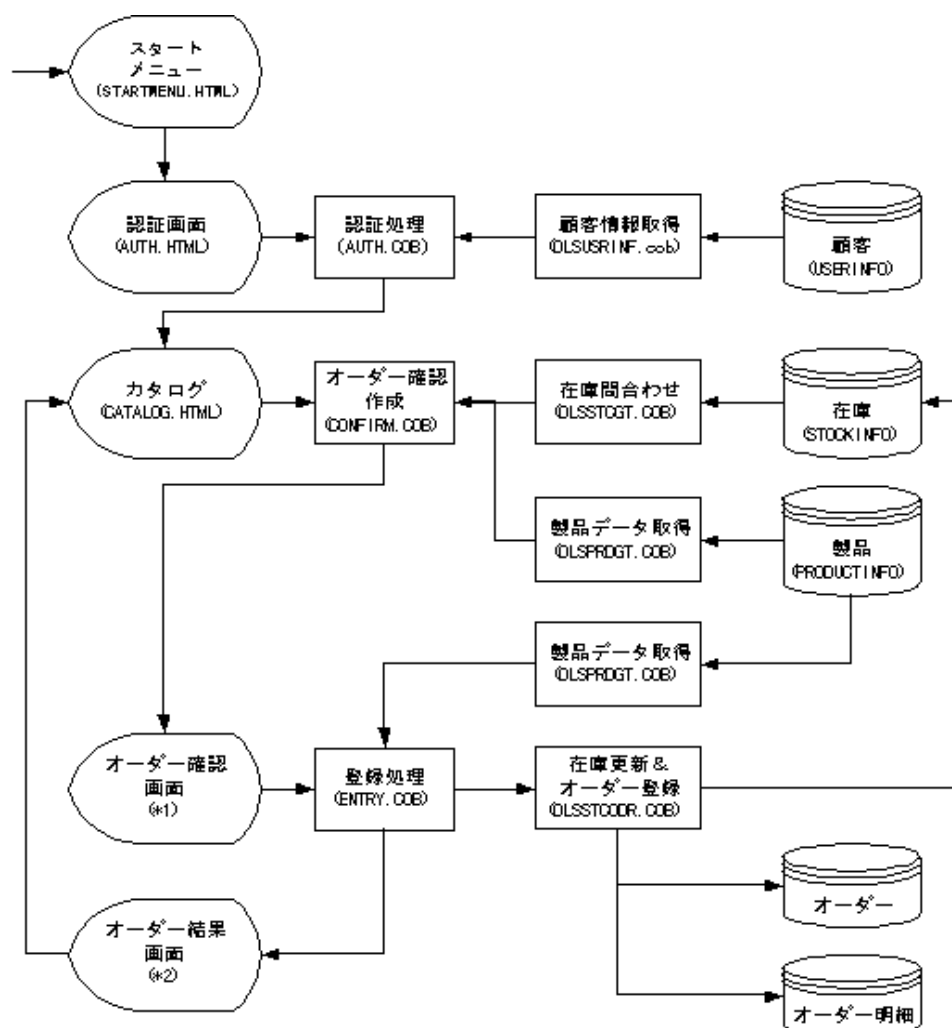
- SHUTDOWN. HTML
- STARTUP. HTML
- SYSERROR. HTML
- SYSTEMERROR. HTML
- STARTMENU. HTML
- UNDERCONSTRUCTION. HTML
- GIFファイル
 - CATALOGTITLE. GIF
 - FJLOGO. GIF
 - TITLE. GIF
- JPEGファイル
 - FMV-6450DX2. JPG
 - FMV-6450TX2. JPG
- プログラム説明書
 - SAMPLE22. TXT

プログラムの呼出し関係

業務開始・終了



オンラインストア



*1:CONFIRMHEAD.HTML、CONFIRMDTAILPARTS.HTML、CONFIRMTAIL.HTMLから作成
 *2:ORDERRESULTHEAD.HTML、ORDERDETAILPARTS.HTML、CONFIRMRESULTTAIL.HTMLから作成

使用しているCOBOLの機能

- 索引ファイル(創成・参照・更新・書換え)
- 外部データ
- 外部ファイル
- データロックサブルーチン
- COBOL ISAPIサブルーチン
- 外部ファイルイベントログ(利用者定義情報の出力)

使用しているCOBOLの文

CALL文、CLOSE文、EXIT文、GO TO文、IF文、MOVE文、OPEN文、PERFORM文、READ文、REWRITE文、SET文、START文、WRITE文

プログラムの翻訳・リンク・実行

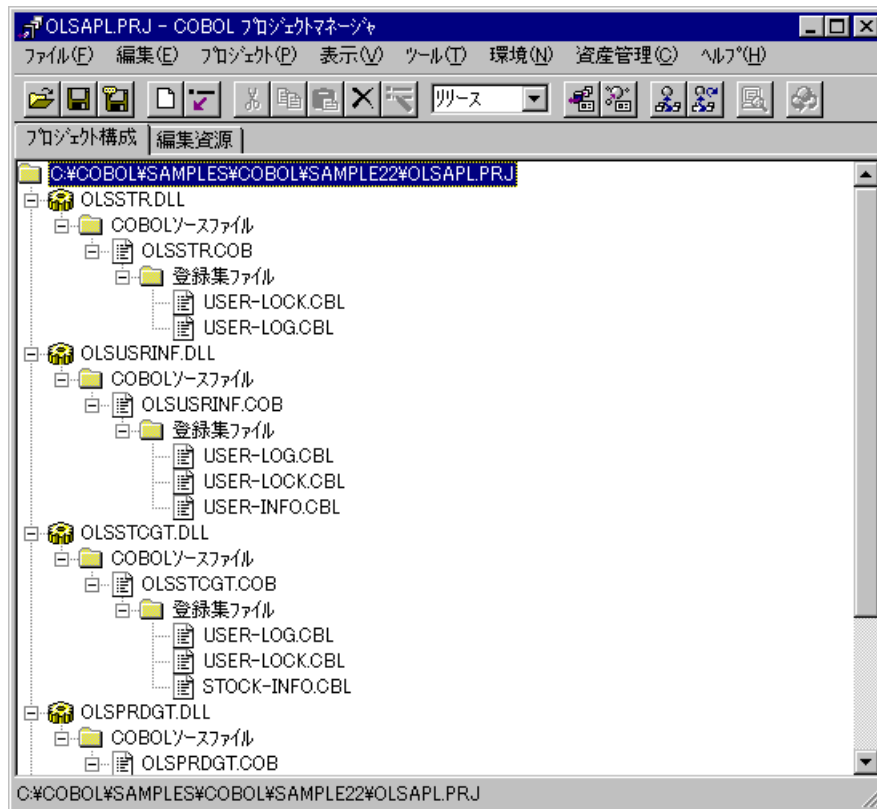
ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用していきます。

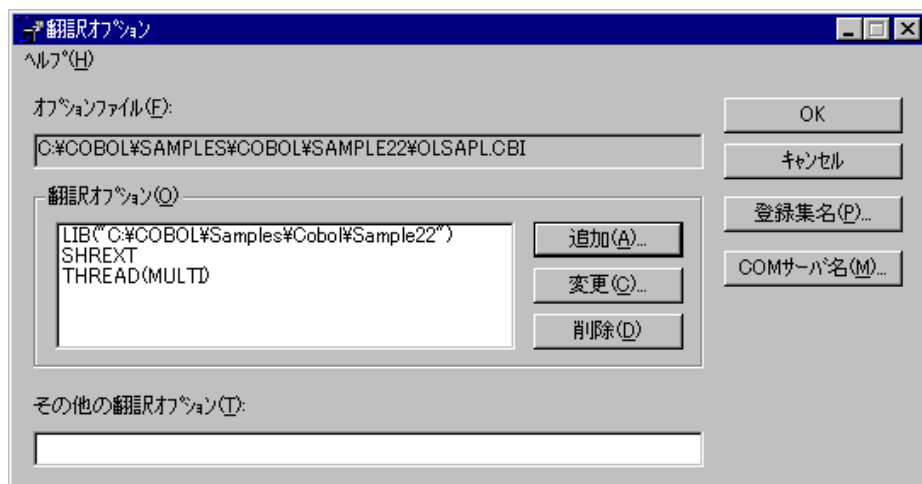
なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。以降の説明で、フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。

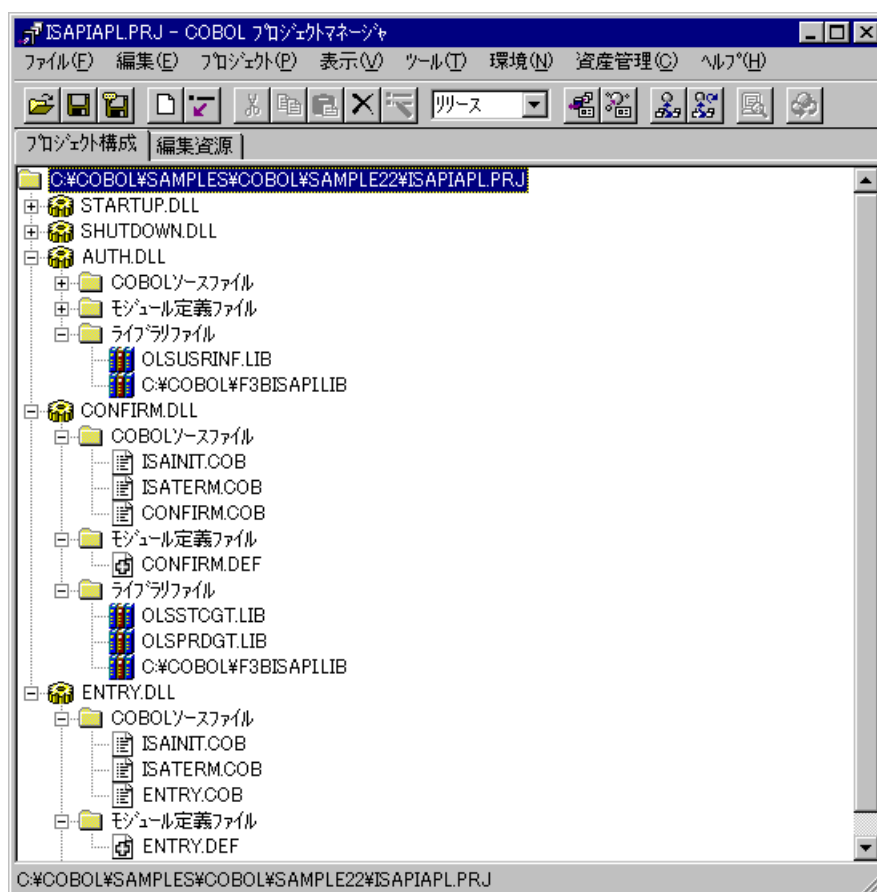
2. プロジェクトファイル“OLSAPL.PRJ”を開きます。



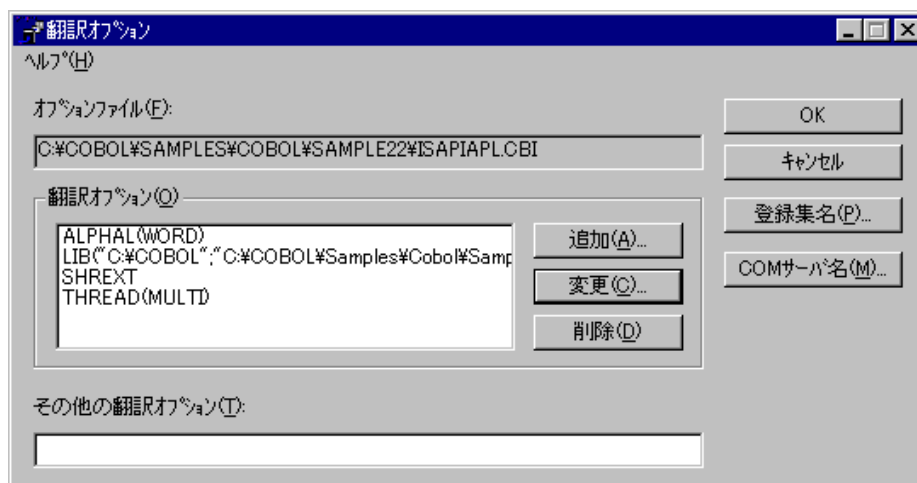
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションTHREAD(MULTI)、SHREXTを指定します。また、翻訳オプションLIBに、登録集ファイルが格納されているフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。
→ プロジェクトに登録した各DLL(ダイナミックリンクライブラリ)が作成されていることを確認してください。
6. 続いて、プロジェクトファイル“ISAPIAPL.PRJ”を開きます。



7. 3. と同じ手順で「翻訳オプション」ダイアログを表示し、翻訳オプションTHREAD (MULTI)、SHREXT、ALPHAL (WORD)を指定します。また、翻訳オプションLIBに登録集ファイルのフォルダを指定します。確認後、「OK」ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。



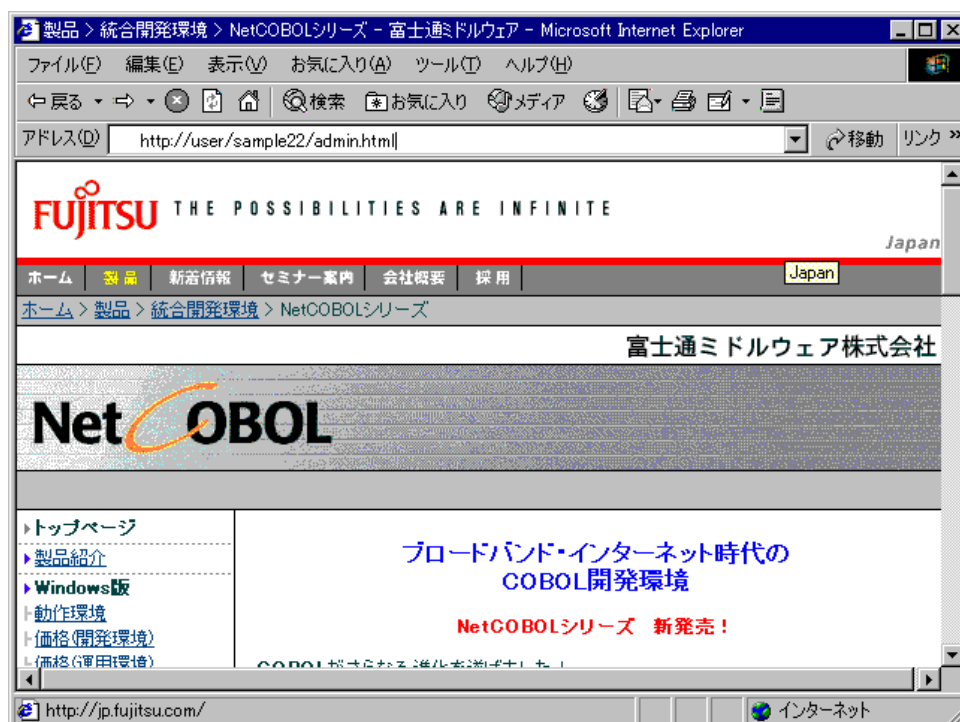
8. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ プロジェクトに登録した各DLL(ダイナミックリンクライブラリ)が作成されていることを確認してください。

プログラムの実行

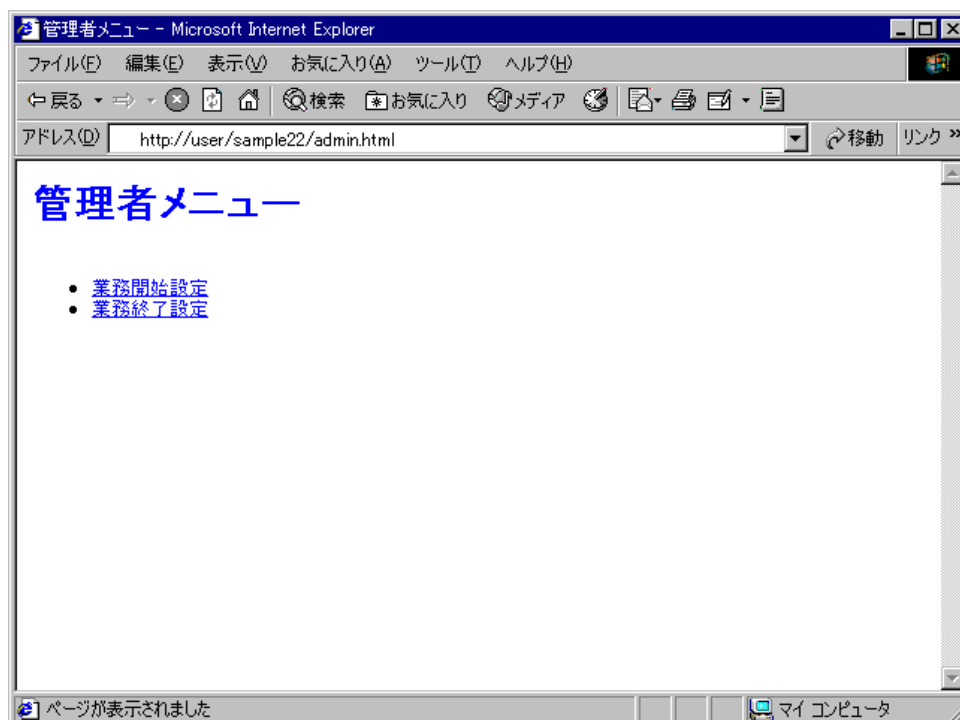
ここでは、ドメイン名を“user”、仮想ディレクトリ名を“sample22”としてIIS(Internet Information Server)に登録しています。WWWブラウザは、Microsoft(R) Internet Explorerを使用しています。

1. オンラインストアを開始します。

URLに以下の情報を設定します。



管理者メニューの画面が表示されるので、“業務開始設定”をクリックします。



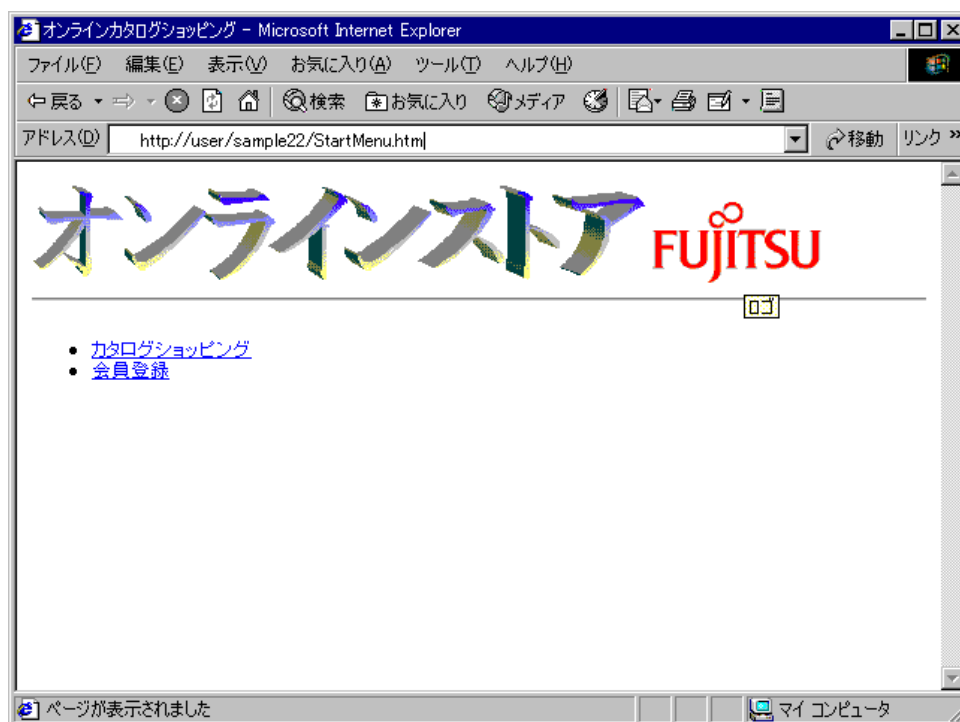
“業務開始設定”をクリックすると、オンラインストアが開始されます。オンラインストアを起動する前に、必ず業務開始設定を行ってください。

2. オンラインストアを起動します。

URLに以下の情報を設定して実行キーを押します。



オンラインストアの画面が表示されるので、“カタログショッピング”をクリックします。



“カタログショッピング”をクリックすると会員認証画面が表示されます。画面が表示されたら、ユーザIDとパスワードを入力して〔OK〕ボタンをクリックします。ここで、入力できるユーザIDはUSER0001からUSER0030までです。パスワードはユーザIDと同じです。

会員認証 - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

戻る 進む 検索 お気に入り メディア

アドレス(D) http://user/sample22/Auth.html 移動 リンク >>

会員認証

あなたのIDとパスワードを入力して下さい。会員登録がまだの方は、[会員登録](#)を行って下さい。

ユーザID:

パスワード:

OK クリア

ページが表示されました イン트라ネット



[OK] ボタンをクリックするとカタログ画面が表示されます。ここで、注文するパソコンのモデルと数量を指定し [オーダー] ボタンをクリックします。

カタログ - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

戻る 進む 検索 お気に入り メディア

アドレス(D) http://user/sample22/auth.dll?%83%86%81%5B%83UID=USER0001&%83p%83X%83%8F% 移動 リンク >>

製品名	スペック	モデル	単価(円)	数量
FMV-6450TX2	Pentium-III 450MHz, 64MB, 4.3GB, Viper V550, 100BASE-TX	WindowsNTモデル	428000	<input type="text" value="0"/>
FMV-6400TX2	Pentium-III 400MHz, 64MB, 4.3GB, Viper V550, 100BASE-TX	WindowsNTモデル	368000	<input type="text" value="2"/>
FMV-6450DX2	Pentium-III 450MHz, 32MB, 4.3GB, RAGE PRO TURBO, サウンド, 100BASE-TX	WindowsNTモデル	398000	<input type="text" value="3"/>
FMV-6400DX2	Pentium-III 400MHz, 32MB, 4.3GB, RAGE PRO TURBO, サウンド, 100BASE-TX	WindowsNTモデル	338000	<input type="text" value="2"/>
FMV-6350DX2	Pentium-III 350MHz, 32MB, 4.3GB, RAGE PRO TURBO, サウンド, 100BASE-TX	WindowsNTモデル	278000	<input type="text" value="0"/>
FMV-6366DX2c	Celeron 366MHz, 32MB, 4.3GB, RAGE PRO TURBO, サウンド, 100BASE-TX	WindowsNTモデル	238000	<input type="text" value="0"/>

オーダー クリア メニューに戻る

ページが表示されました イン트라ネット

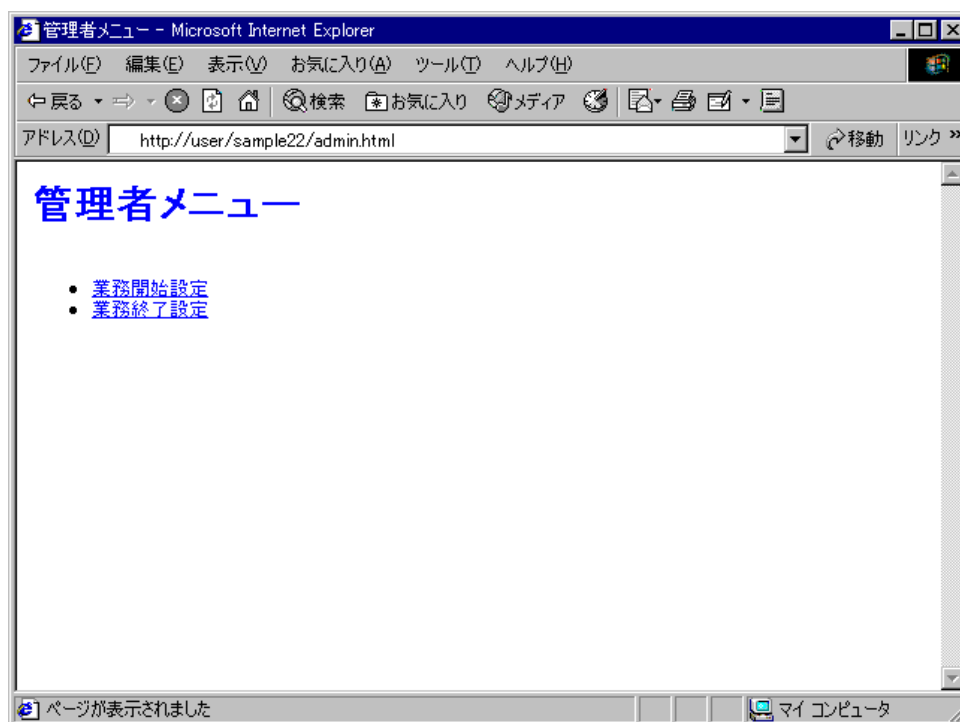
「オーダー」ボタンをクリックすると、オーダー確認画面が表示されます。オーダーの内容を確認して「オーダー発行」ボタンをクリックします。

「オーダー発行」ボタンをクリックすると、オーダー結果画面が表示されます。

3. オンラインストアを終了します。
URLに以下の情報を設定して実行キーを押します。



管理者メニューの画面が表示されるので、“業務終了設定”をクリックします。



イベントログ出力サブルーチンの出力例

この例題プログラムでは、プログラムで検出したエラーの詳細情報をイベントログ出力サブルーチンを使用して、イベントログに出力します。

以下に、イベントログの確認する方法を示します。

1. 管理ツールのイベントビューアを起動し、アプリケーションのログを選択します。

例題23 COM連携-Excelを操作するプログラム (1)

ここでは、本製品で提供するサンプルプログラム-例題23-について説明します。

例題23では、NetCOBOLのCOMクライアント機能を使って、Excelを操作するプログラムの例を示します。

NetCOBOLのCOMクライアント機能では、COMオブジェクトを作成し、それを使ってCOMサーバの提供する機能をCOBOLのメソッドと同様に使用することができます。

Excelは独立したアプリケーションであるだけでなく、COMサーバとしての側面を持つため、Excelを操作するアプリケーションをCOBOLでも記述することができます。

このプログラムを動作させるためには、以下の製品が必要です。

- Microsoft (R) Excel (以降、Excelと略します。)

なお、NetCOBOLのCOM機能の詳細については、“NetCOBOL 使用手引書”の“第25章 COM機能”を参照してください。

概要

COBOLアプリケーションから、Excelに対して次の操作を行います。

- Excelの起動と終了
- Excelシートのオープン
- ワークシートの選択とデータの挿入
- グラフの作成
- 選択したシートの印刷

NetCOBOLのCOMクライアント機能では、COMサーバのメソッドやインタフェースを認識する方法として、アーライバインドとレイトバインドの2つを持ちます。

この例題ではレイトバインドを使用します。

レイトバインドを使用する場合、使用できるCOBOLの書き方が制限されることや実行性能の面でアーライバインドの場合に劣るなどの短所はありますが、COMサーバの変更の影響を受けづらい点では優れています。このため、COMサーバとしてはExcel197とExcel2000はまったく別の存在ですが、レイトバインドを使用するこのプログラムはそのどちらも操作することが可能です。

アーライバインドの場合の例については“[例題24 COM連携-Excelを操作するプログラム\(2\)](#)”を参照してください。

提供プログラム

- SAMPLE23. PRJ (プロジェクトファイル)
- SAMPLE23. COB (COBOLソースファイル)
- SAMPLE23. TXT (プログラム説明書)
- GRAPHDATA. XLS (テスト用Excelファイル)

使用しているCOBOLの機能

- COMクライアント機能

使用しているCOBOLの文

DISPLAY文、IF文、INVOKE文、PERFORM文、SET文

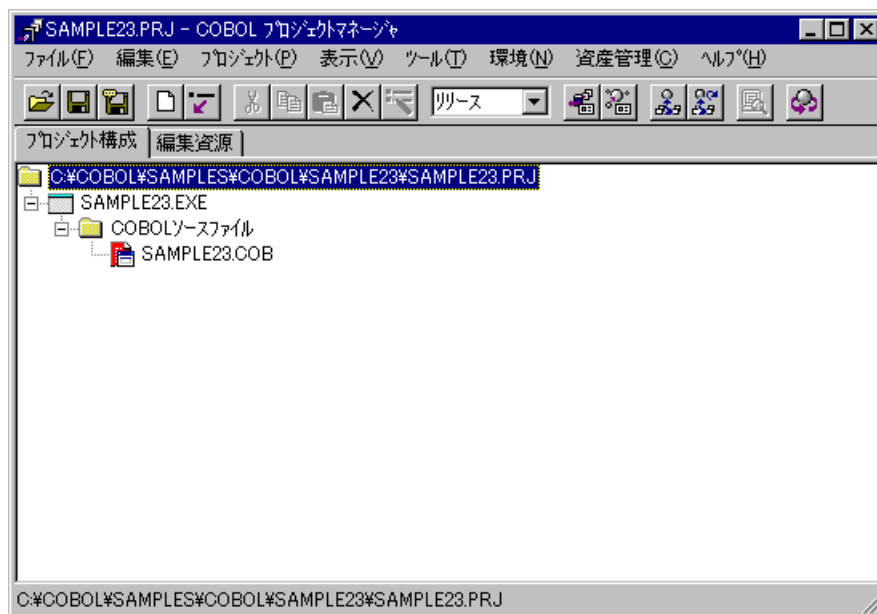
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

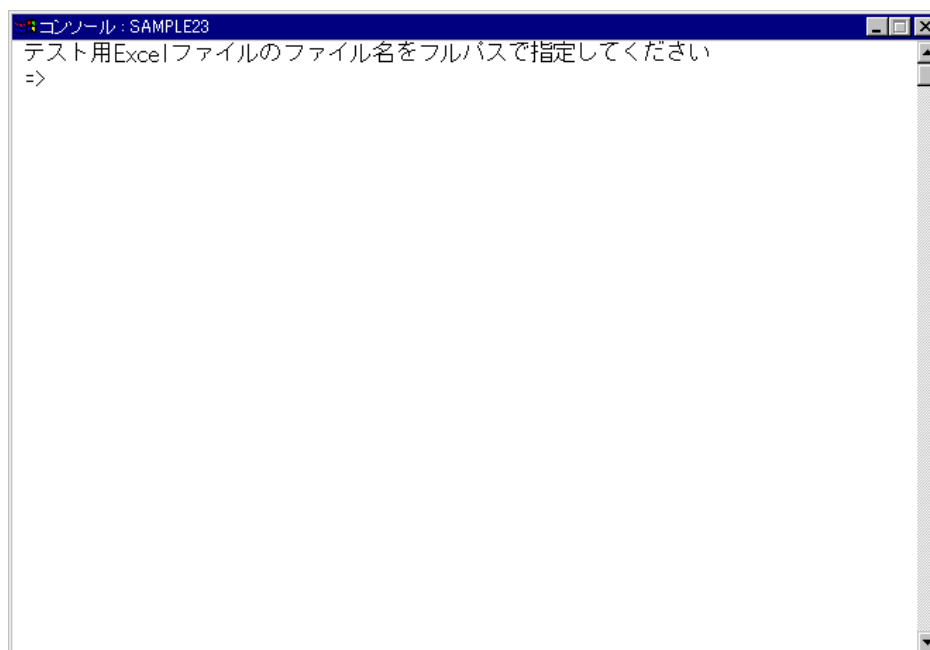
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE23. PRJ”を開きます。



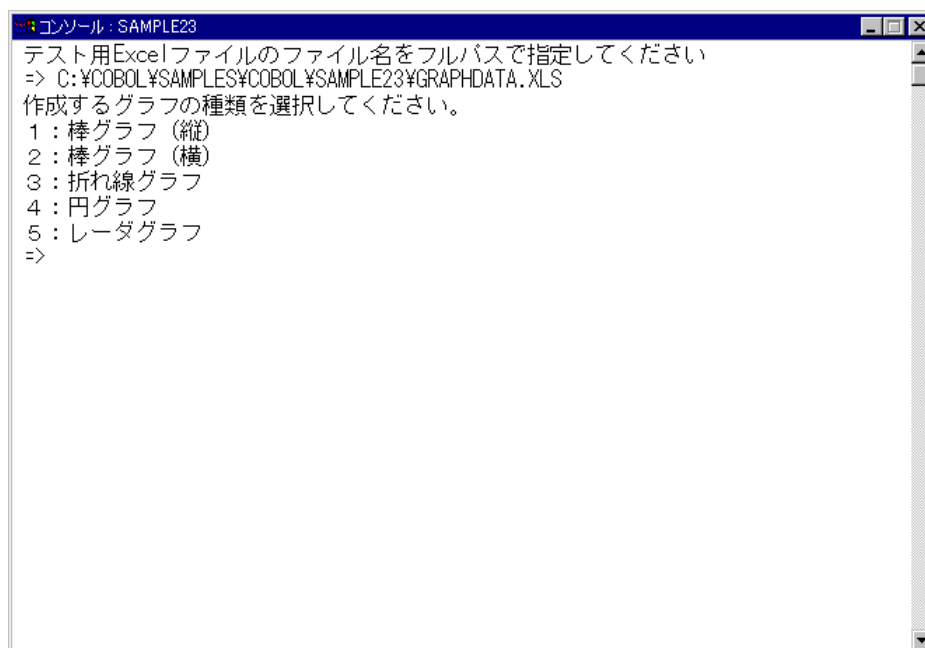
3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE23.EXEが作成されていることを確認してください。

プログラムの実行

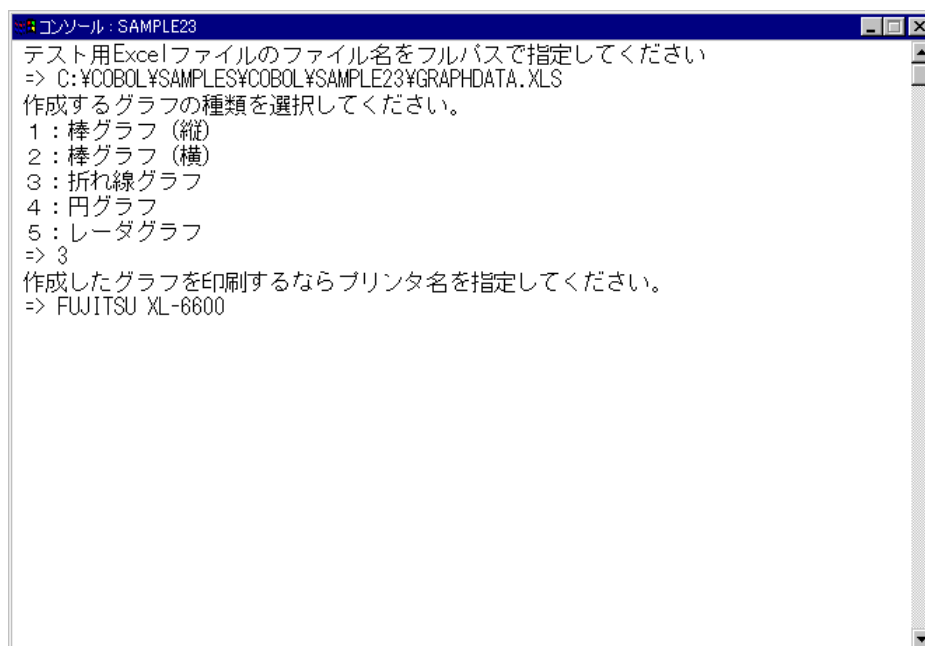
1. プロジェクトマネージャの「プロジェクト」メニューから“実行”を選択します。
→ Excelが起動します。また、COBOLのコンソールに次のメッセージが表示され、入力待ちの状態になります。



2. テスト用のExcelファイルのファイル名をフルパスで指定します。
→ 指定したテスト用のExcelファイルが開かれ、プログラムによるExcelの操作が行われます。その後で、COBOLのコンソールに次のメッセージが表示され、再度入力待ちの状態になります。



3. 作成したいグラフの種類を入力します。
→ グラフが描画され、描画が終了すると、COBOLのコンソールに次のメッセージが表示され、再度入力待ちになります。



4. グラフを印刷する場合、プリンタ名を指定してします。印刷の必要がなければ、何も入力せずにENTERキーを押します。
→ Excelを終了させて、実行が終了します。プリンタ名を指定した場合は、Excelを終了する前にグラフが印刷されます。

例題24 COM連携-Excelを操作するプログラム (2)

ここでは、本製品で提供するサンプルプログラム-例題24-について説明します。

例題24では、NetCOBOLのCOMクライアント機能を使って、Excelを操作する例を示します。

NetCOBOLのCOMクライアント機能では、COMオブジェクトを作成し、それを使ってCOMサーバの提供する機能をCOBOLのメソッドと同様に使用することができます。

Excelは独立したアプリケーションであるだけでなく、COMサーバとしての側面を持つため、Excelを操作するアプリケーションをCOBOLでも記述することができます。

このプログラムを動作させるためには、以下の製品が必要です。

- Microsoft (R) Excel (以降、Excelと略します。)

なお、NetCOBOLのCOM機能の詳細については、“NetCOBOL 使用手引書”の“第25章 COM機能”を参照してください。

概要

COBOLアプリケーションから、Excelに対して次の操作を行います。

- Excelの起動と終了
- Excelシートのオープン
- ワークシートの選択とデータの挿入
- グラフの作成
- 選択したシートの印刷

NetCOBOLのCOMクライアント機能では、COMサーバのメソッドやインタフェースを認識する方法として、アーリバインドとレイトバインドの2つを持ちます。

この例題ではアーリバインドを使用します。

アーリバインドを使用する場合、オブジェクトプロパティやメソッドの行内呼出しが記述できます。また、性能の面でレイトバインドの場合より優れています。反面、開発時に型ライブラリが必須となる点やCOMサーバの変更の影響を受けやすいなどの短所もあります。このプログラムではExcelの機能を使用するためCOMサーバ名EXCELを使用していますが、プログラムの翻訳に先立って、このCOMサーバ名に対して、Excelの型ライブラリを指定しておく必要があります。また、Excel2000の型ライブラリを参照してビルドした実行可能ファイルは、Excel97を操作することができませんし、逆にExcel97の型ライブラリを参照してビルドした実行可能ファイルは、Excel2000を操作することができません。

レイトバインドの場合の例については“[例題23 COM連携-Excelを操作するプログラム\(1\)](#)”を参照してください。

提供プログラム

- SAMPLE24. PRJ (プロジェクトファイル)
- SAMPLE24. COB (COBOLソースファイル)
- SAMPLE24. TXT (プログラム説明書)
- GRAPHDATA. XLS (テスト用Excelファイル)

使用しているCOBOLの機能

- COMクライアント機能
- プロジェクト管理機能

使用しているCOBOLの文

DISPLAY文、IF文、INVOKE文、PERFORM文、SET文

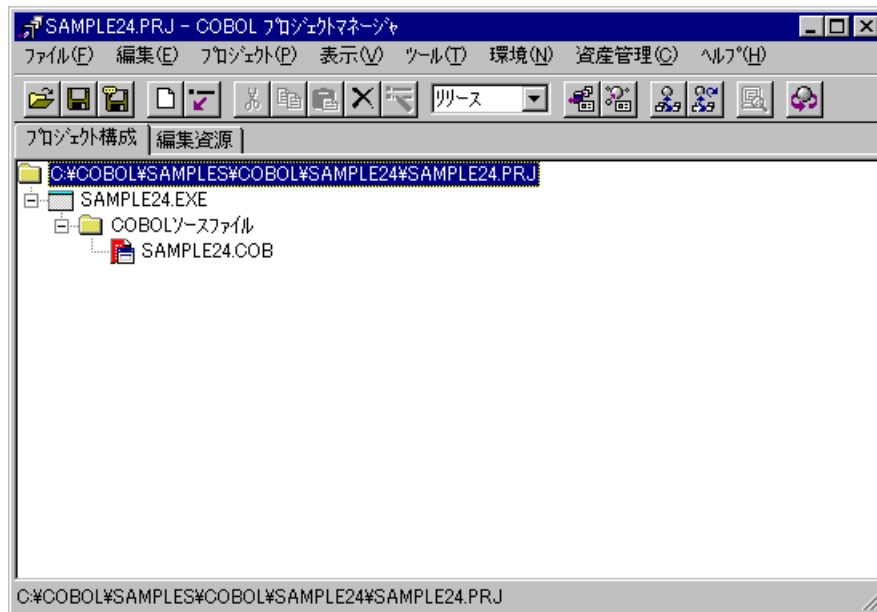
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル “SAMPLE24.PRJ” を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから “翻訳オプション” を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. [翻訳オプション] ダイアログの [COMサーバ名] ボタンをクリックします。
→ [COMサーバの設定] ダイアログが表示されます。



5. COM サーバ名を EXCEL として、型ライブラリ名には Excel97 が “C:\Program Files\Microsoft Office” にインストールされている場合、次のように設定します。

EXCEL = C:\Program Files\Microsoft Office\Office\Excel8.olb

Excel2000の型ライブラリ(ファイル名はExcel9.olb)を使用する場合や、Excel97を他のフォルダにインストールしてある場合は、[変更] ボタンをクリックして、型ライブラリの指定を変更してください。

6. プロジェクトマネージャの [プロジェクト] メニューから “ビルド” を選択します。
→ ビルド終了後、SAMPLE24.EXEが作成されていることを確認してください。



注意

Excel2002以降では、それ以前のものと一部仕様が異なります。Excel2002以降を使用する場合には次のようにしてください。

— プログラムの修正

提供されているSample24.cobの1100、1110行を削除あるいはコメントにして、コメントになっている1130、1140行を有効にしてください。

```
-----
001100      INVOKE セル "SET-VALUE"
001110                      USING PDATA(セルカラム位置)
001120* Excel2002以降を使用する場合は、こちらを使って下さい。
001130*      INVOKE セル "SET-VALUE"
001140*                      USING OMITTED PDATA(セルカラム位置)
-----
```

— 型ライブラリ

Excel2002以降の型ライブラリは、Excel.exeです。例えば、Excel2002が“C:\Program Files\Microsoft Office”にインストールされている場合、次のように設定します。

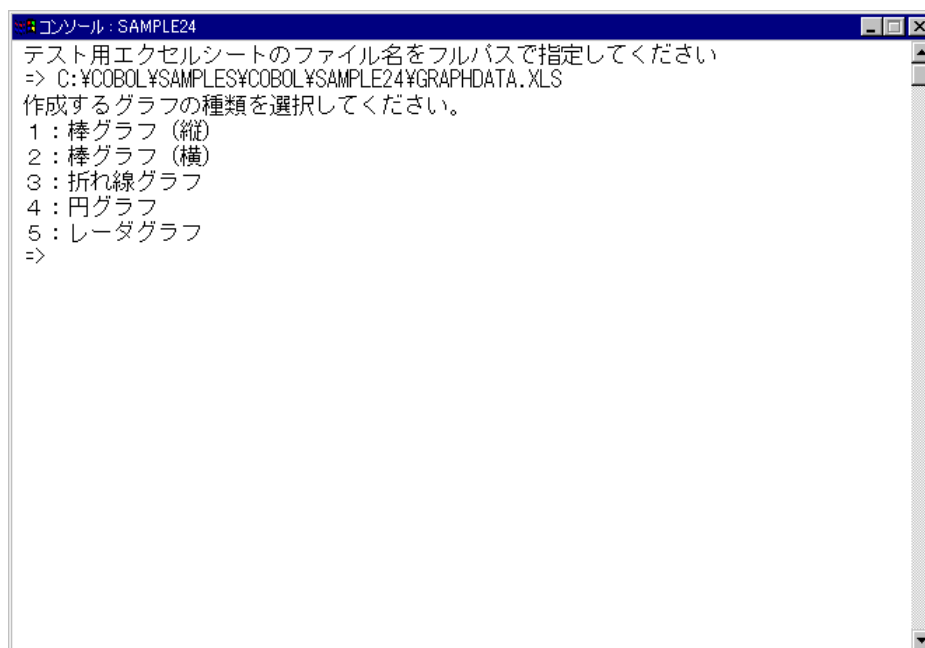
```
-----
EXCEL = C:\Program Files\Microsoft Office\Office\Excel.exe
-----
```

プログラムの実行

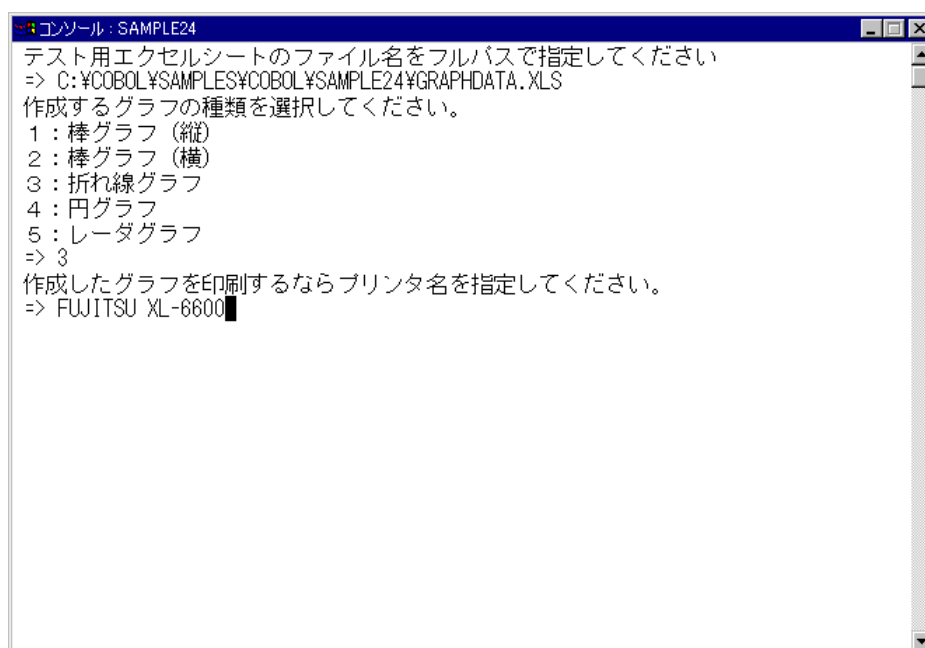
1. プロジェクトマネージャの〔プロジェクト〕メニューから“実行”を選択します。
→ Excelが起動します。また、COBOLのコンソールに次のメッセージが表示され、入力待ちの状態になります。



2. テスト用のExcelファイルのファイル名をフルパスで指定します。
→ 指定したテスト用のExcelファイルが開かれ、プログラムによるExcelの操作が行われます。その後で、COBOLのコンソールに次のメッセージが表示され、再度入力待ちの状態になります。



3. 作成したいグラフの種類を入力します。
→ グラフを描画し、描画が終了すると、COBOLのコンソールに次のメッセージが表示され、再度入力待ちになります。



4. グラフを印刷する場合、プリンタ名を指定します。印刷の必要がなければ、何も入力せずにENTERキーを押します。
→ Excelを終了させて、実行が終了します。プリンタ名を指定した場合は、Excelを終了する前にグラフが印刷されます。

例題25 COM連携-COBOLによるCOMサーバプログラムの作成

ここでは、本製品で提供するサンプルプログラム-例題25-について説明します。

例題25では、NetCOBOLのCOMサーバ機能を使って、COBOLプログラムをCOMサーバにする例を示します。

NetCOBOLのCOMサーバ機能では、COBOLのクラス定義をそのままCOMサーバに移行することができます。プロジェクトマネージャのCOMサーバ作成機能で必要な情報の設定を行って、ビルドしなおすだけで、選択したクラスがCOMサーバのインタフェースとして公開されます。

NetCOBOLのCOM機能の詳細については、“NetCOBOL 使用手引書”の“第25章 COM機能”を参照してください。

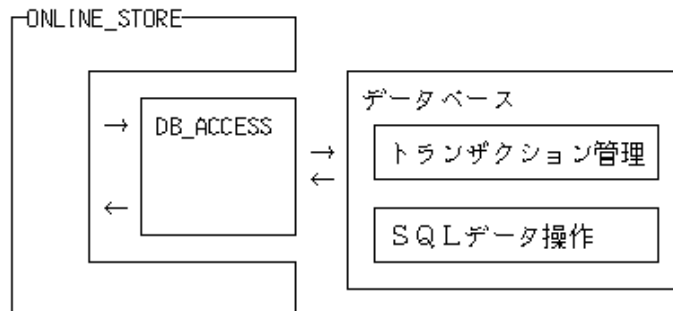
なお、このプログラムはODBCドライバを経由してデータベースにアクセスします。このため、このプログラムを動作させるためには、以下の製品が必要です。

- データベース
- データベースにODBCでアクセスするために必要な製品
- ODBCドライバ
- ODBCドライバマネージャ

ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL 使用手引書”の“第21章 リモートデータベースアクセス(ODBC)”を参照してください。

概要

例題プログラムは、2つのクラス定義で構成されています。



COBOLの2つのクラスのうち、ONLINE_STOREクラスがCOMサーバクラスです。ONLINE_STOREクラスは、オンラインストアのアプリケーションを構築するための次の機能を提供します。

- 認証処理
- 在庫確認
- オーダー登録
- オーダー清算

これらの機能を利用するクライアントプログラムとしては、COBOLプログラムはもちろんのこと、Visual C++(R)で作成したプログラム、Visual Basic(R)で作成したプログラムまたはASP(Active Server Pages)のVBscript(Visual Basic(R) Scripting Edition)などが使用できます。COBOLクライアントの例は例題26に、ASPクライアントの例は例題27に示します。

提供プログラム

- DB_ACCESS. COB(COBOL ソースファイル)
- ONLINE_STORE. COB(COBOL ソースファイル)

- STORESV1.PRJ(プロジェクトファイル)
- STORESV1.CBI(翻訳オプションファイル)
- STORESV1_DLL.CSI(COMサーバ情報ファイル)
- STORESV1.DEF(モジュール定義ファイル)
- SAMPLE25.TXT(プログラム説明書)

使用しているCOBOLの機能

- COMサーバ機能
- リモートデータベースアクセス
- *COM-ARRAYクラス

使用しているCOBOLの文

IF文、INVOKE文、INITIALIZE文、SET文、MOVE文、PERFORM文

埋込みSQL文(COMMIT文、CONNECT文、INSERT文、SELECT文、UPDATE文、ROLLBACK文、DISCONNECT文)

プログラムの翻訳・リンク・実行

プログラムを実行する前に

- ODBCドライバを経由してデータベースへアクセスできる環境を構築しておいてください。
- デフォルトで接続するサーバを設定し、そのサーバのデータベース上に次の4つの表を作成しておいてください。

— 顧客表

顧客表は、以下の形式で作成してください。

ユーザID	パスワード	←列の名前
可変長文字 32バイト	可変長文字 32バイト	←列の属性

↑
主キー

顧客表には次のデータを格納しておいてください。

ユーザID	パスワード
USER0001	USER0001
USER0002	USER0002
USER0003	USER0003
USER0004	USER0004
USER0005	USER0005
USER0006	USER0006
USER0007	USER0007
USER0008	USER0008
USER0009	USER0009
USER0010	USER0010

— 在庫表

在庫表は、以下の形式で作成してください。

製品番号	在庫数	←列の名前
固定長文字 10バイト	10進数整数 10桁	←列の属性

↑
主キー

在庫表には次のデータを格納しておいてください。

製品番号	在庫数
FMV2TXH111	900000
FMV2TXH161	100000
FMV2TXH151	500000
FMV2TXF111	45000
FMV2TXF161	300000
FMV2TXF151	60000
FMV2DXH111	90000
FMV2DXH161	55000
FMV2DXH151	990000
FMV2DXF111	10000
FMV2DXF161	777700
FMV2DXF151	200000
FMV2DXD111	690000
FMV2DXD161	870000
FMV2DXD151	619000
FMV2DXA111	2900000
FMV2DXA161	8760000
FMV2DXA151	100000
FMV3NA3LC0	10000
FMV3NA3LC6	300

— オーダー表

オーダー表は、以下の形式で作成してください。

オーダー番号	ユーザID	日付	←列の名前
固定長文字 12バイト	可変長文字 32バイト	固定長文字 14バイト	←列の属性

↑
主キー

オーダー表には、データを格納しておく必要はありません。

— オーダー明細表

オーダー明細表は、以下の形式で作成してください。

オーダー番号	製品番号	数量	←列の名前
固定長文字 12バイト	固定長文字 10バイト	10進数整数 10桁	←列の属性

オーダー明細表には、データを格納しておく必要はありません。

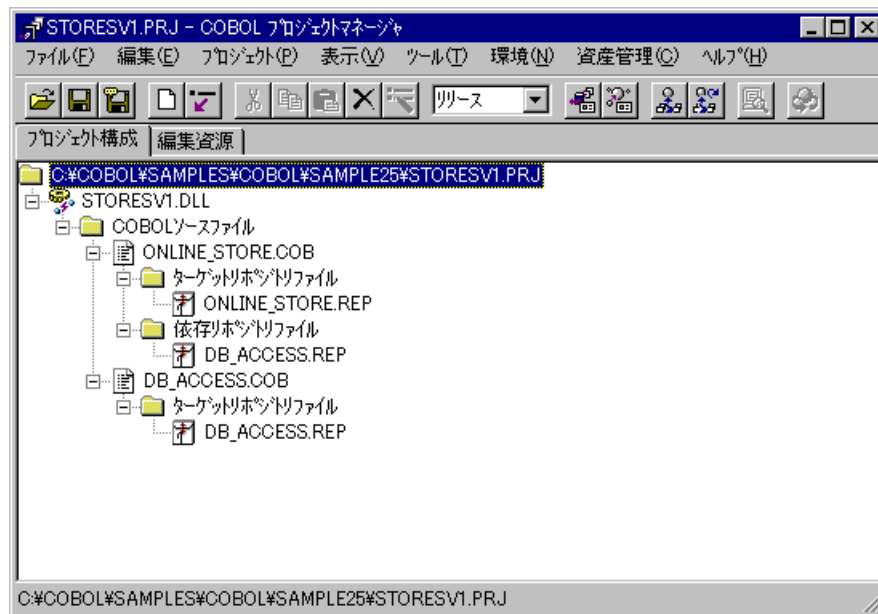
- ODBC情報ファイル設定ツール(SQLODBCS.EXE)を使用して、ODBC情報ファイル(ここではC:\YDBMSACS.INFとします)を作成してください。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用していきます。

なお、以降ではNetCOBOLのインストール先フォルダをC:\YCOBOLとして説明しています。フォルダ名がC:\YCOBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

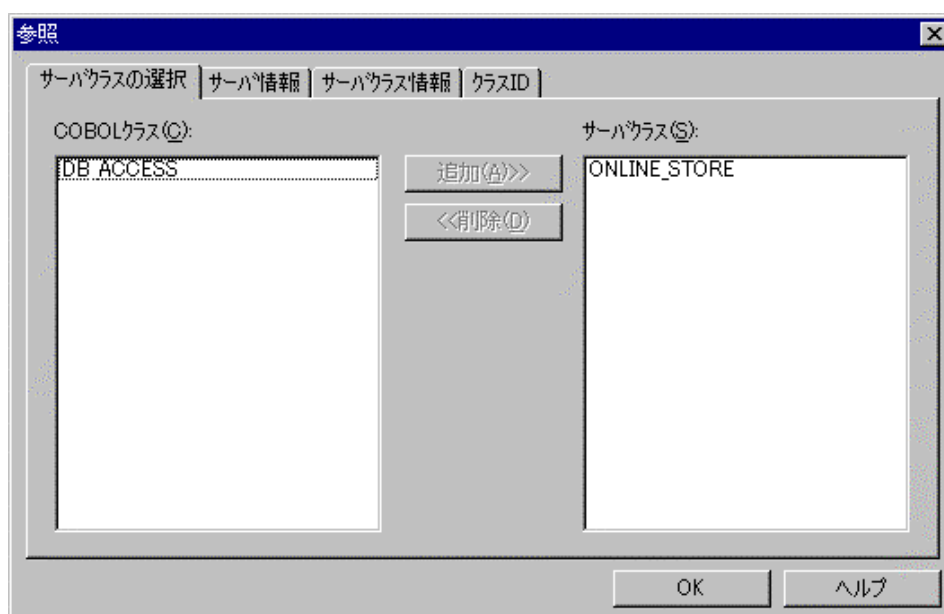
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“STORESV1.PRJ”を開きます。



3. 設定されているCOMサーバ情報を確認します。
ターゲットファイル(STORES.V1.DLL)を選択し、[プロジェクト] - [オプション] - [COMサーバ] メニューから“参照”を選択します。



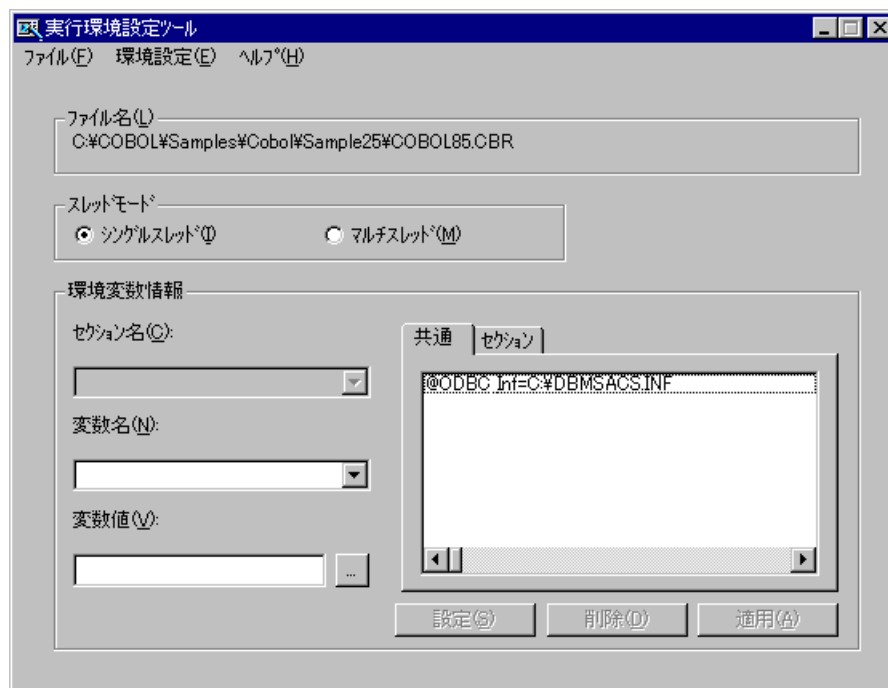
〔参照〕 ダイアログが開いて、設定されているCOMサーバ情報が参照できます。



4. プロジェクトマネージャの〔プロジェクト〕メニューから“ビルド”を選択します。
→ ビルド終了後、STORESV1.DLLが作成されていることを確認してください。

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの〔ツール〕メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。
2. 〔ファイル〕メニューの“開く”を選択し、ダイナミックリンクライブラリ (STORESV1.DLL) の存在するフォルダに、実行用の初期化ファイル (COBOL85.CBR) を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@ODBC_Inf (ODBC情報ファイルの指定)に、ODBC情報ファイル名を指定します。



4. 「適用」ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. 「ファイル」メニューの“終了”を選択し、実行環境設定ツールを終了します。

COMサーバの登録

作成したCOBOLアプリケーションをCOMサーバとして使用するためには、Windowsシステムへの登録が必要です。登録の方法は、COMサーバの使用形態により2つの方法があります。

- COMサーバとCOMクライアントを同一のマシンで使用する。
→ REGSVR32.EXEを使用して、システムのレジストリに登録します。詳細については“NetCOBOL 使用手引書”の“25.3.4 COMサーバの登録と削除”を参照してください。
- COMサーバをネットワーク接続された別のマシン上のCOMクライアントから使用する。
→ MTS (Microsoft (R) Transaction Server)を利用します。MTSエクスプローラを使用して、システムのレジストリおよびMTSに登録します。詳細については“NetCOBOL 使用手引書”の“25.4.2 MTS環境への登録方法”を参照してください。

例題26 COM連携-COBOLサーバプログラムの使用（COBOLクライアント）

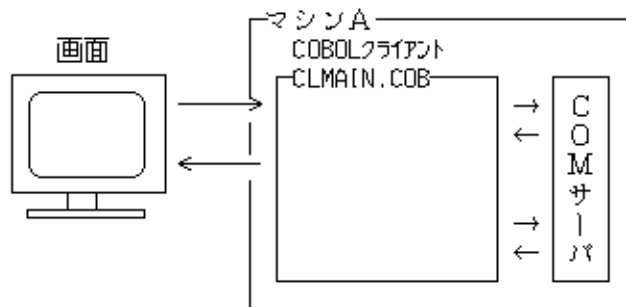
ここでは、本製品で提供するサンプルプログラム-例題26-について説明します。

例題26では、NetCOBOLのCOMクライアント機能を使って、例題25のCOBOLサーバプログラムを使用するクライアントプログラムの例を示します。

NetCOBOLのCOM機能の詳細については、“NetCOBOL 使用手引書”の“第25章 COM機能”を参照してください。

概要

例題25で作成したCOBOLサーバプログラムを使用して、オンラインストアのアプリケーションを作成します。クライアントプログラムは、スクリーン操作機能を使用して画面からデータの受け付け、サーバプログラムに処理を依頼します。サーバプログラムによる処理の結果は画面に表示されます。



提供プログラム

- CLMAIN.COB (COBOLソースプログラム)
- ORDERSHEET-INFO.CBL (COBOL登録集ファイル)
- PRODUCT-TABLE.CBL (COBOL登録集ファイル)
- SCREENS.CBL (COBOL登録集ファイル)
- SAMPLE26.PRJ (プロジェクトファイル)
- SAMPLE26.CBI (翻訳オプションファイル)
- SAMPLE26.TXT (プログラム説明書)

使用しているCOBOLの機能

- スクリーン機能
- COMクライアント機能
- *COM-ARRAYクラス

使用しているCOBOLの文

ACCEPT文(スクリーン機能)、DISPLAY文(スクリーン機能)、EVALUATE文、INVOKE文、IF文、PERFORM文、SET文

プログラムの翻訳・リンク・実行

プログラムを実行する前に

ネットワーク接続された別のマシン上のCOMサーバを使用する場合、このプログラムを実行するマシンにサーバの情報をインストールする必要があります。これは次の手順で行います。

1. COMサーバを登録したマシンでクライアント情報のインストールプログラムを作成します。

クライアント情報のインストールプログラムの作成方法の詳細は、“NetCOBOL 使用手引書”の“25.4.3 クライアントマシンへのインストール”を参照してください。

2. クライアント情報のインストールプログラムをこのプログラムを実行するマシンで実行します。

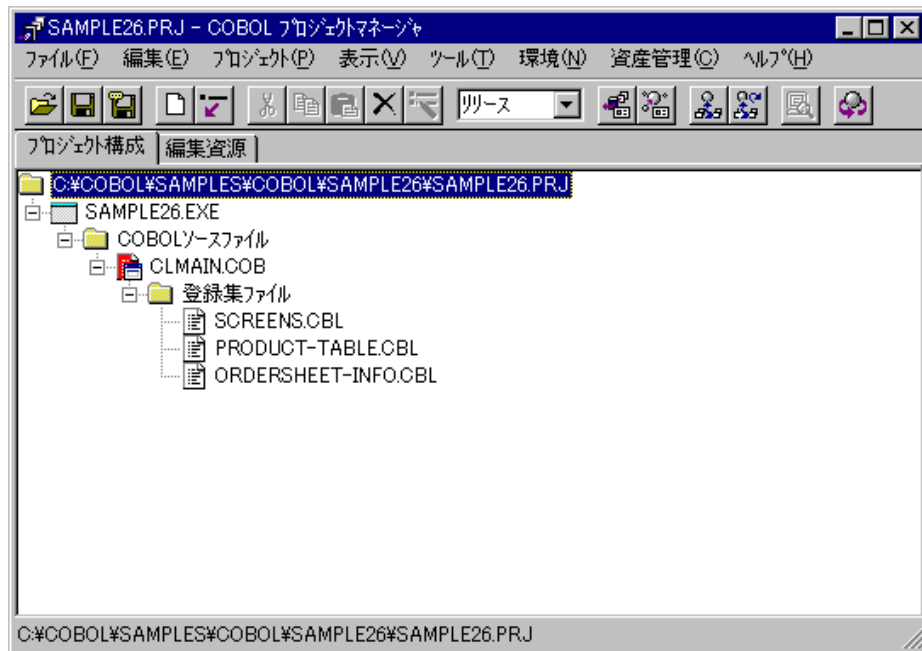
COMサーバと同じマシン上でこのプログラムを実行する場合は、この処理は必要ありません。

ビルド・リビルド

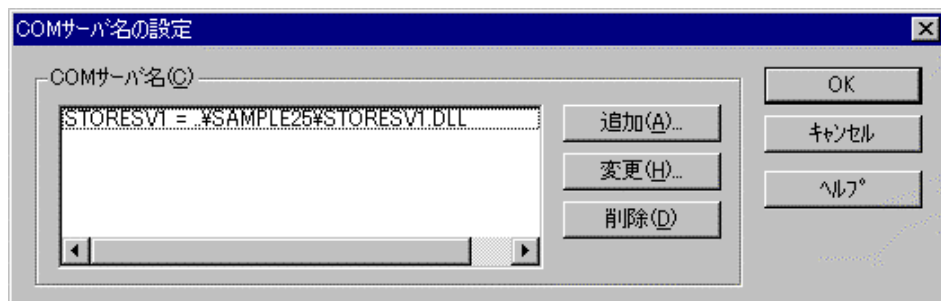
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE26.PRJ”を開きます。



3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. [翻訳オプション] ダイアログの[COMサーバ名] ボタンをクリックします。
→ [COMサーバの設定] ダイアログが表示されます。

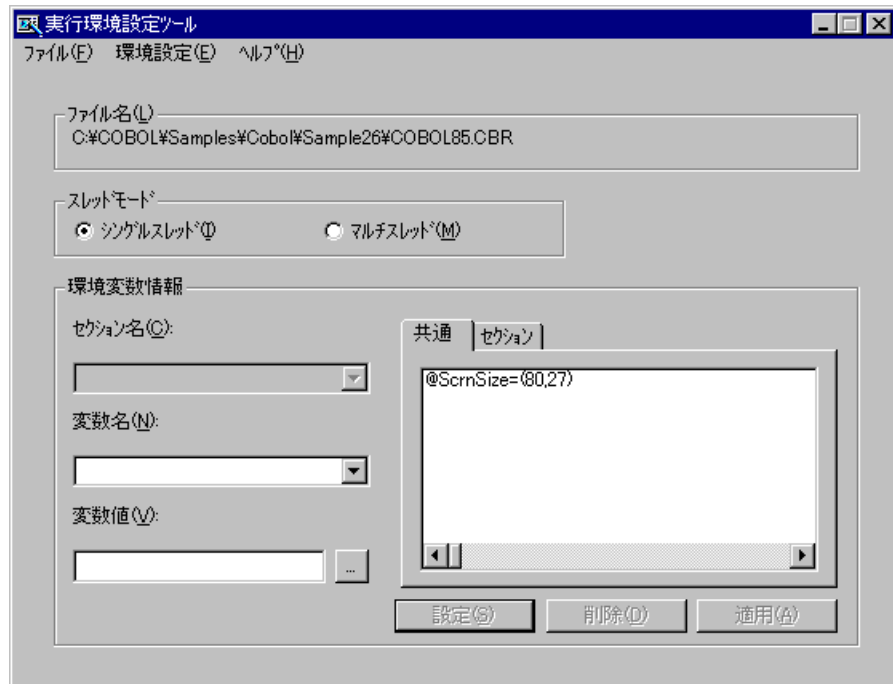


5. COMサーバ名STORESV1に、STORESV1.DLL (型ライブラリ)を指定します。確認後、[OK] ボタンをクリックします。
→ [翻訳オプション] ダイアログに戻ります。ここでも [OK] ボタンをクリックして、プロジェクトマネージャウィンドウに戻ります。
6. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。

→ ビルド終了後、SAMPLE26.EXEが作成されていることを確認してください。

実行環境情報の設定

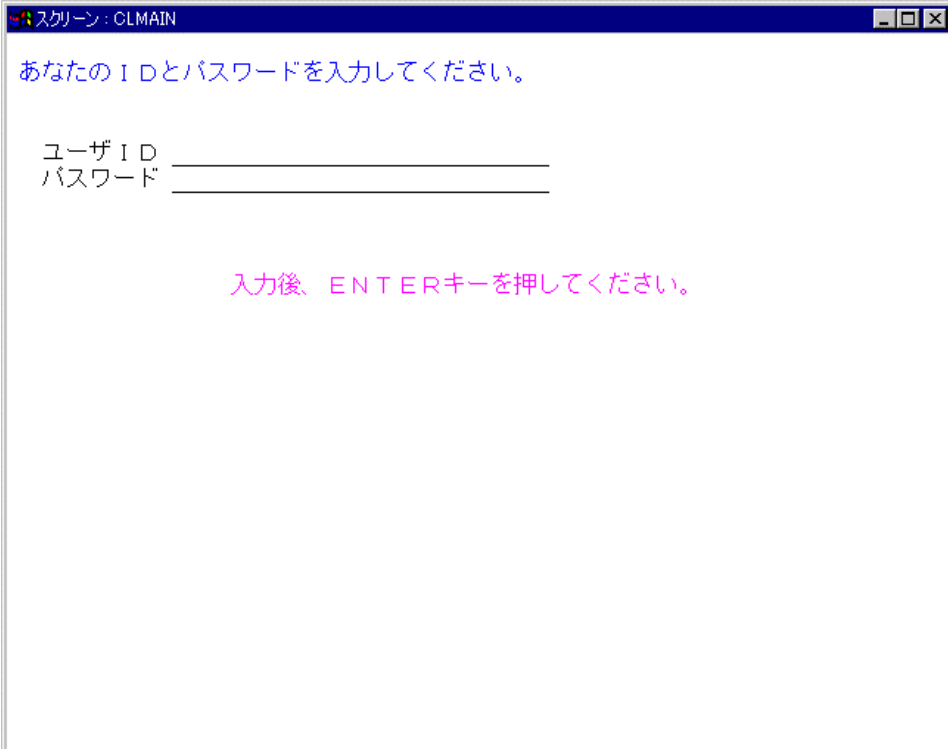
1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。
2. [ファイル]メニューの“開く”を選択し、実行可能プログラム(SAMPLE26.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@ScrnSize(スクリーン操作の論理画面の大きさ)に、“(80,27)”を指定します。



- COBOLクライアントプログラムをサーバプログラムと同じマシンで実行する場合、サーバプログラムの実行環境情報もここに設定する必要があります。その場合、環境変数情報@ODBC_Inf (ODBC情報ファイルの指定)に、ODBC情報ファイル名を指定してください。
4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
 5. [ファイル]メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

1. プロジェクトマネージャの[プロジェクト]メニューから“実行”を選択します。
→ 次の画面が表示されます。UserIDとPasswordを入力してENTERキーを押してください(入力フィールドの移動はカーソルキーまたはTABキーで行います)。UserIDはUSER0001～USER0010が使用できます。PasswordはUserIDと同じです。なお、パスワードは非表示になっているので注意してください。



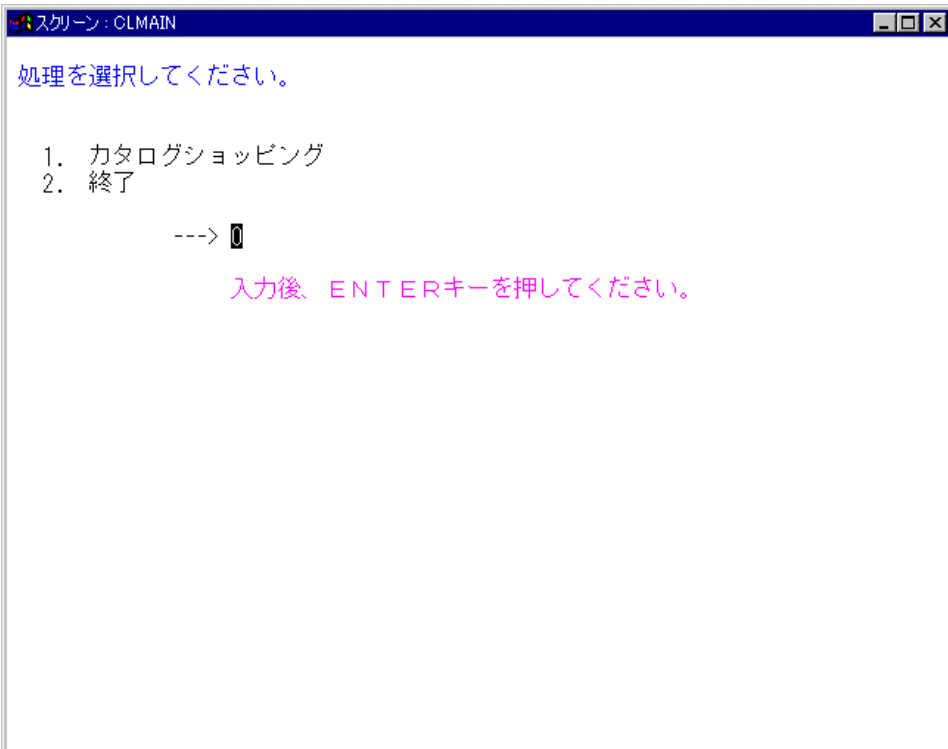
スクリーン: CLMAIN

あなたのIDとパスワードを入力してください。

ユーザID _____
パスワード _____

入力後、ENTERキーを押してください。

2. メニュー画面が表示されます。“1”を入力し、ENTERキーを押します。



スクリーン: CLMAIN

処理を選択してください。

1. カタログショッピング
2. 終了

---> 1

入力後、ENTERキーを押してください。

3. カタログ画面が表示されます。注文する個数を入力します。入力フィールドの移動はカーソルキーまたはTABキーで行います。入力終了後、ENTERキーを押します。

スクリーン: CLMAIN

数量を入力してください。

製品名	対象OS	単価	数量
FMV-6450TX2	WindowsNT	428000	
FMV-6450TX2	Windows98	408000	2
FMV-6450TX2	Windows95	408000	
FMV-6400TX2	WindowsNT	368000	
FMV-6400TX2	Windows98	348000	3
FMV-6400TX2	Windows95	348000	
FMV-6450DX2	WindowsNT	398000	
FMV-6450DX2	Windows98	378000	
FMV-6450DX2	Windows95	378000	1
FMV-6400DX2	WindowsNT	338000	
FMV-6400DX2	Windows98	318000	4
FMV-6400DX2	Windows95	318000	
FMV-6350DX2	WindowsNT	278000	
FMV-6350DX2	Windows98	258000	
FMV-6350DX2	Windows95	258000	
FMV-6366DX2c	WindowsNT	238000	
FMV-6366DX2c	Windows98	218000	
FMV-6366DX2c	Windows95	218000	
FMV-6366NA3/L	WindowsNT	648000	
FMV-6366NA3/L	Windows98	628000	

4. オーダー確認画面が表示されます。“Y”を入力してENTERキーを押します。

スクリーン: CLMAIN

オーダーを確認してください。

製品名	数量
FMV-6450TX2 Windows98	2
FMV-6400TX2 Windows98	3
FMV-6450DX2 Windows95	1
FMV-6400DX2 Windows98	4

よろしいですか? (Y/N)=> Y

5. オーダー控え画面が表示されます。ENTERキーを押すと、2. のメニュー画面に戻ります。

スクリーン: CLMAIN					
以下のオーダーを受け付けました。 □					
オーダー番号: ONUM11595039			2000/02/01 12:03		
FMV-6450TX2	Windows98	408000	2	816000	
FMV-6400TX2	Windows98	348000	3	1044000	
FMV-6450DX2	Windows95	378000	1	378000	
FMV-6400DX2	Windows98	318000	4	1272000	
数量合計				10	
支払い額				¥3,510,000	

6. 処理を終了する場合は、メニュー画面で“2”を入力し、ENTERキーを押します。

例題27 COM連携-COBOLサーバプログラムの使用(ASPクライアント)

ここでは、本製品で提供するサンプルプログラム-例題27-について説明します。

例題27では、NetCOBOLのCOMサーバ機能を使用して作成したCOMサーバを、ASP(Active Server Pages)のVisual Basic(R) Scripting Edition(以降ではVBScriptといいます)から呼び出して使用する例を示します。

なお、ASPとそこで使用するVBScriptの詳細については、市販の解説書を参考にしてください。
このプログラムを動作させるためには、以下の製品が必要です。

- 以下のいずれかの製品
 - Windows NT(R) Server
 - Windows(R) 2000 Server
 - Windows Server(TM) 2003
- Microsoft(R) Internet Information Server 4.0以上
(Windows NT(R) Serverの場合には、Microsoft(R) Windows NT(R) OptionPack 4.0が必要です。)

概要

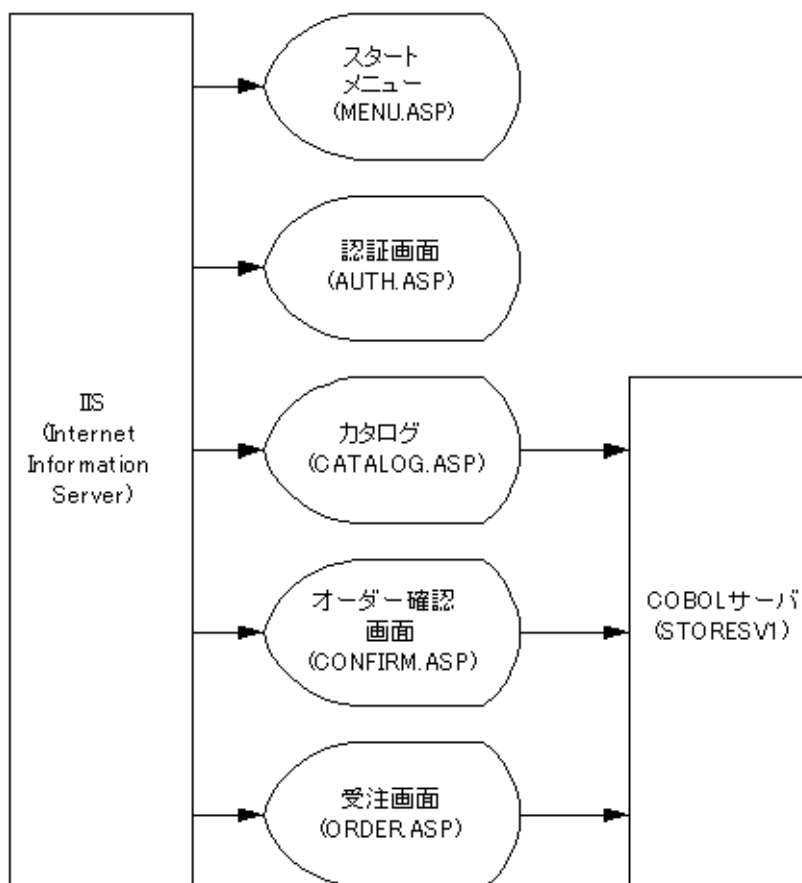
ASP(Active Server Pages)は、HTML文書にスクリプト言語を埋め込むことにより動的なWebアプリケーションを構築する方法の1つです。

ASPのVBScript中では、ASPの組み込みオブジェクトであるServerとそのメソッドCreateObjectを使用して、COMサーバのオブジェクトを生成することができます。生成したオブジェクトからCOMサーバの提供するメソッドの呼出しが可能となります。

この機能を使用して、例題25のCOBOLサーバプログラムを使ったオンラインストアのWebアプリケーションを作成します。

プログラムの構成

この例題プログラムは次の構成と呼出し関係を持っています。



提供プログラム

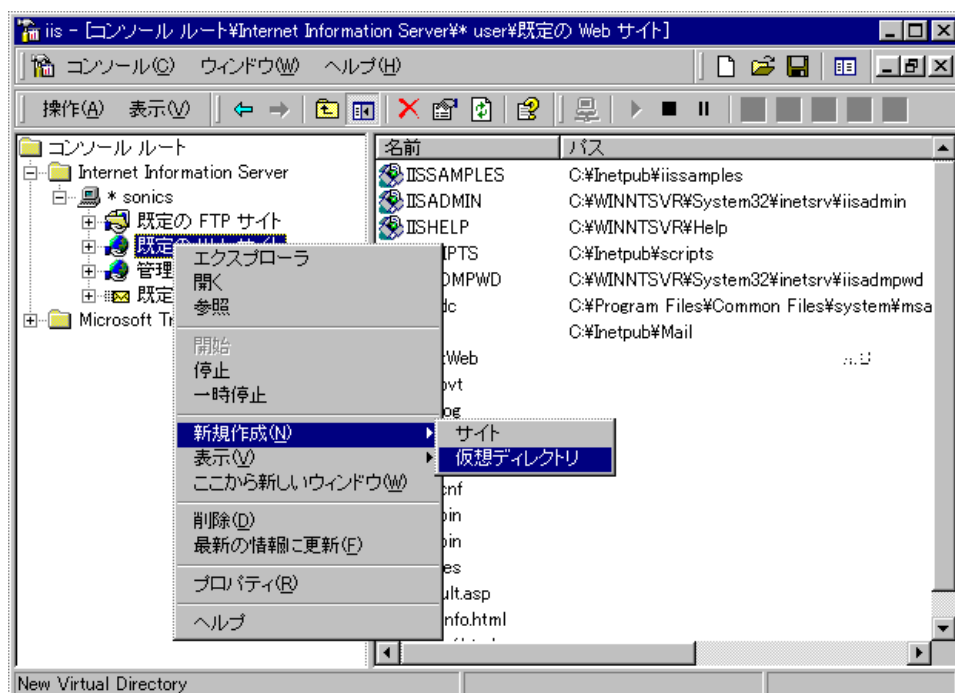
- MENU. ASP (ASPページファイル)
- AUTH. ASP (ASPページファイル)
- CATALOG. ASP (ASPページファイル)
- CONFIRM. ASP (ASPページファイル)
- ORDER. ASP (ASPページファイル)
- STYLE. CSS (スタイルシートファイル)
- CATALOGTITLE. GIF (画像ファイル)
- FJLOGO. GIF (画像ファイル)
- SAMPLE27. TXT (プログラム説明書)

プログラムを実行する前に

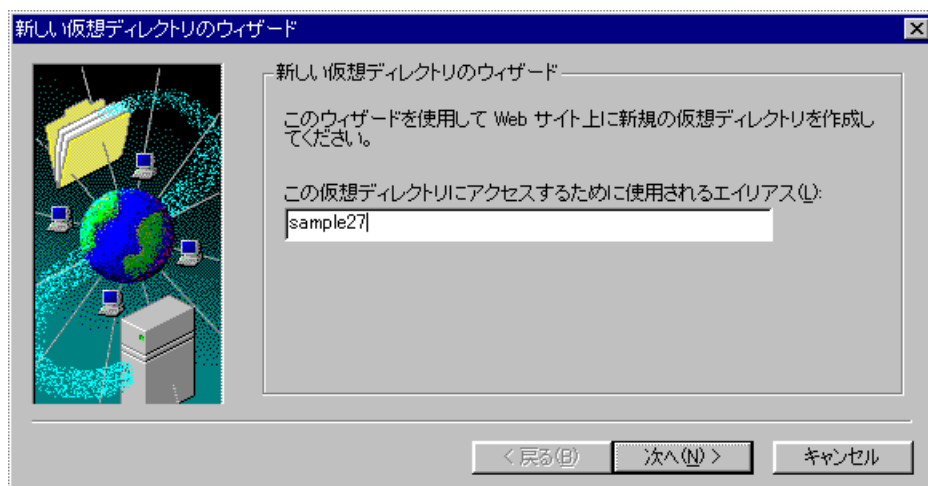
この例題では、例題25で作成したCOMサーバプログラムを使用します。例題25のプログラムをビルドして、COMサーバとしての登録や実行環境情報の設定をしておいてください。

次に例題27をインターネットサービスマネージャで登録します。登録する方法を示します。

1. インターネットサービスマネージャを起動して、“規定のWebサイト”を選択し、コンテンツメニューの「新規作成」から“仮想ディレクトリ”を選びます。



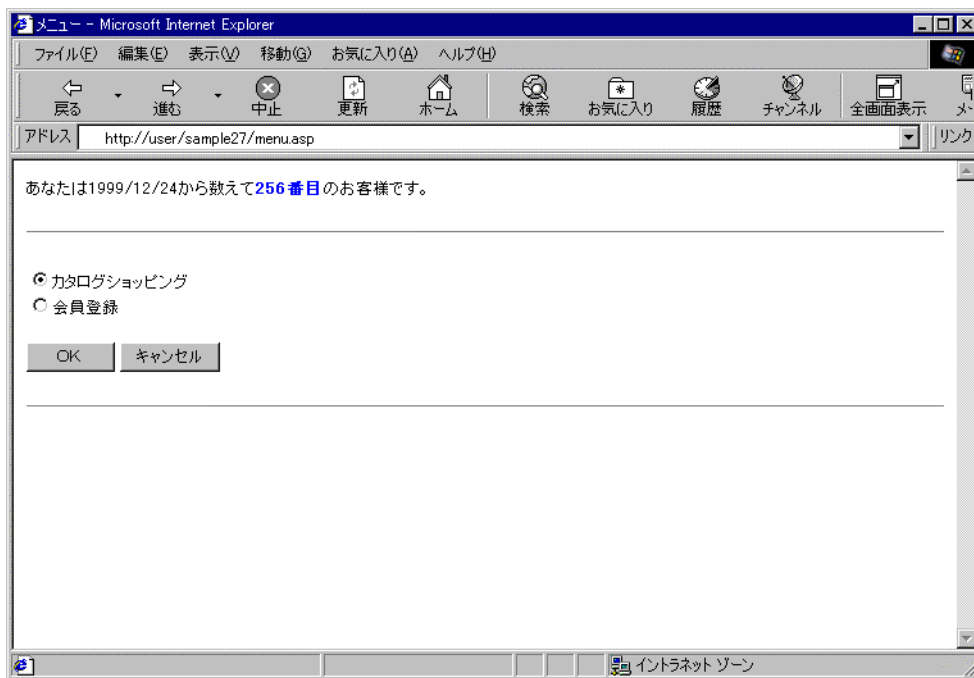
2. 仮想ディレクトリ名を入力し、次にASPページファイルがある例題27の物理パスを入力します。



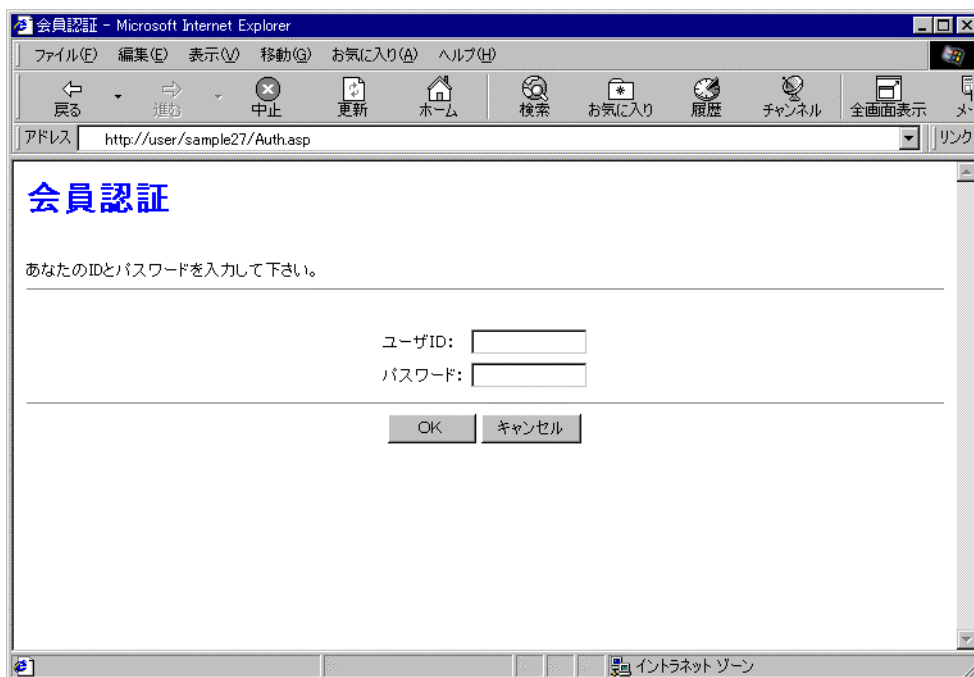
実行手順

ここでは、ドメイン名を“user”、仮想ディレクトリ名を“sample27”として登録します。
WWWブラウザは、Microsoft(R) Internet Explorerを使用しています。

1. URLに以下の情報を設定します。
メニュー画面が表示されるので、“カタログショッピング”を選択して、[OK] ボタンをクリックします。



2. 会員認証画面が表示されます。ユーザIDとパスワードを入力して「OK」ボタンをクリックします。ユーザIDは、USER0001～USER0010が使用できます。パスワードはユーザIDと同じです。
なお、パスワードは非表示になっているので注意してください。



3. カタログ画面が表示されます。注文する個数を入力して、「オーダー」ボタンをクリックします。

カタログ - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) 移動(Q) お気に入り(A) ヘルプ(H)

戻る 進む 中止 更新 ホーム 検索 お気に入り 履歴 チャンネル 全画面表示

アドレス http://user/sample27/Catalog.asp リンク

数量を入力してください。

製品名	スペック	モデル	単価	数量
FMV-6450TX2	Pentium-II 450MHz,64MB,4.3GB,Viper V550,100BASE-TX	WindowsNTモデル	428000	<input type="text" value="0"/>
		Windows98モデル	408000	<input type="text" value="0"/>
		Windows95モデル	408000	<input type="text" value="0"/>
FMV-6400TX2	Pentium-II 400MHz,64MB,4.3GB,Viper V550,100BASE-TX	WindowsNTモデル	368000	<input type="text" value="0"/>
		Windows98モデル	348000	<input type="text" value="3"/>
		Windows95モデル	348000	<input type="text" value="0"/>
FMV-6450DX2	Pentium-II 450MHz,32MB,4.3GB,RAGE PRO TURBO,サウンド,100BASE-TX	WindowsNTモデル	398000	<input type="text" value="2"/>
		Windows98モデル	378000	<input type="text" value="0"/>
		Windows95モデル	378000	<input type="text" value="0"/>
FMV-6400DX2	Pentium-II 400MHz,32MB,4.3GB,RAGE PRO TURBO,サウンド,100BASE-TX	WindowsNTモデル	338000	<input type="text" value="0"/>
		Windows98モデル	318000	<input type="text" value="1"/>
		Windows95モデル	318000	<input type="text" value="0"/>
FMV-6350DX2	Pentium-II 350MHz,32MB,4.3GB,RAGE PRO TURBO,サウンド,100BASE-TX	WindowsNTモデル	278000	<input type="text" value="5"/>
		Windows98モデル	258000	<input type="text" value="0"/>
		Windows95モデル	258000	<input type="text" value="0"/>
FMV-6366DX2c	Celeron 366MHz,32MB,4.3GB,RAGE PRO TURBO,サウンド,100BASE-TX	WindowsNTモデル	238000	<input type="text" value="0"/>
		Windows98モデル	218000	<input type="text" value="0"/>
		Windows95モデル	218000	<input type="text" value="0"/>

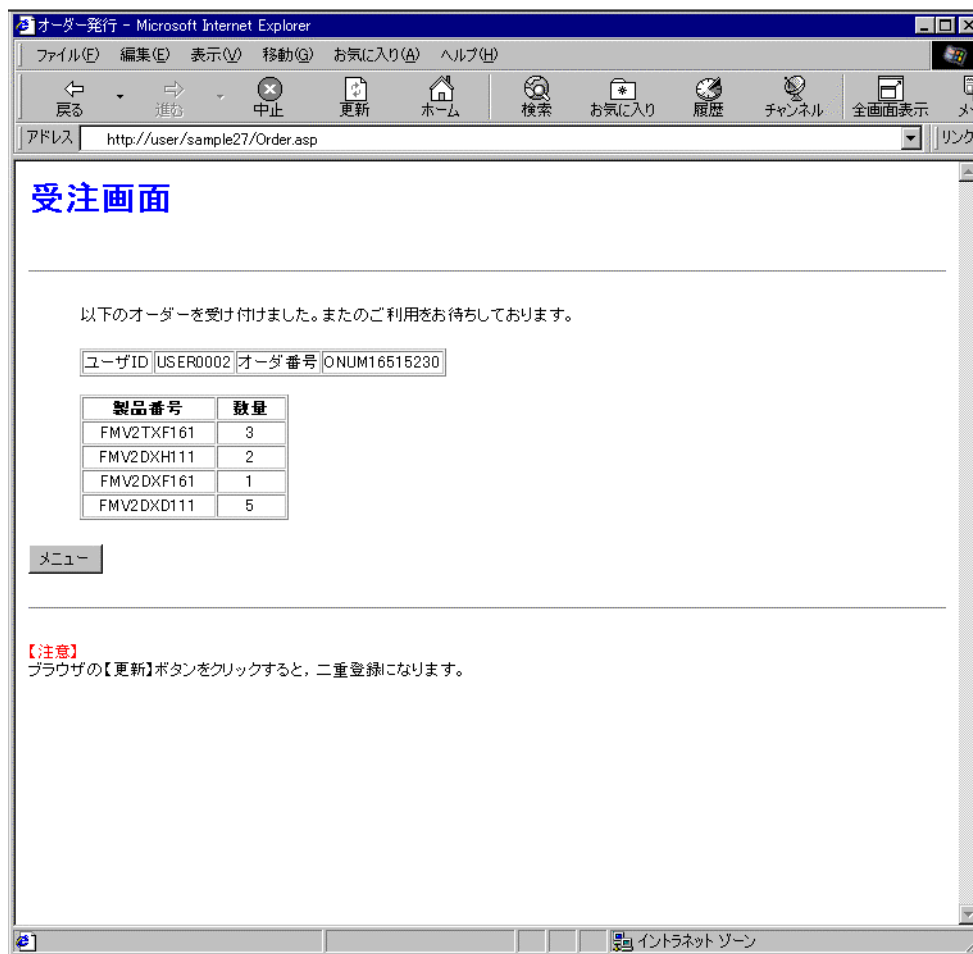
オーダー クリア メニュー

イントラネットゾーン

4. オーダー確認画面が表示されます。[オーダー発行] ボタンをクリックします。



5. 受注画面が表示されます。[メニュー] ボタンをクリックすると、1. のメニュー画面に戻ります。



例題28 COM連携-MTSによるトランザクション管理をするプログラム

ここでは、本製品で提供するサンプルプログラム-例題28-について説明します。

例題28では、COBOLによるCOMサーバプログラムのトランザクション管理をMTSで行う方法を示します。

COBOLアプリケーションで、MTS(Microsoft(R) Transaction Server)によるトランザクション管理を行う場合の詳細は“NetCOBOL 使用手引書”の“第25章 COM機能”を参照してください。

このプログラムを動作させるためには、以下の製品が必要です。

- 以下のいずれかの製品
 - Windows NT(R) Server
 - Windows(R) 2000 Server
 - Windows Server(TM) 2003
- Microsoft(R) Transaction Server 2.0以上
(Windows NT(R) Serverの場合には、Microsoft(R) Windows NT(R) OptionPack 4.0が必要です。)

なお、このプログラムはODBCドライバを経由してデータベースにアクセスします。

このため、このプログラムを動作させるためには、以下の製品が必要です。

- データベース
- データベースにODBCでアクセスするために必要な製品
- ODBCドライバ
- ODBCドライバマネージャ

ODBCドライバを使用するデータベースアクセスについては、“NetCOBOL 使用手引書”の“第21章 リモートデータベースアクセス(ODBC)”を参照してください。

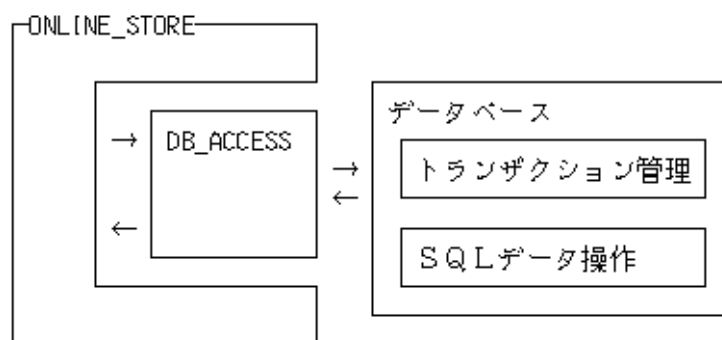
概要

このプログラムでは、例題25と同様にオンラインストアのアプリケーションを構築するための次の機能を提供します。

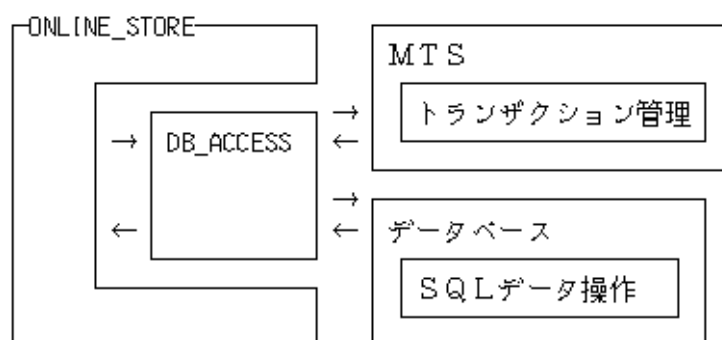
- 認証処理
- 在庫確認
- オーダー登録
- オーダー清算

ただし、このプログラムでは、トランザクションの管理をMTSの機能を使用して、COBOLプログラムから直接行っています。

例題25では、埋込みSQL文のCOMMIT文/ROLLBACK文を使用して、トランザクションの管理をデータベースに任せていました。これは埋込みSQL文に慣れた人にはわかりやすい方法ですが、トランザクション管理をデータベースに任せることにより、データベースの処理に負荷がかかります。



このプログラムでは、トランザクションの管理はMTSの機能を使用して、COBOLプログラム自身で行います。これにより、データベースの処理の負荷が軽減されるとともに、より詳細なトランザクションの管理が可能となります。



MTSの機能を使用して、COBOLアプリケーションからトランザクションを管理するには、次の2つの方法があります。

- COMサーバオブジェクトからオブジェクトが動作しているトランザクションを制御する。
- COMクライアントからCOMサーバが動作しているトランザクションを制御する。

ここここでは、前者の例を示します。

提供プログラム

- DB_ACCESS. COB (COBOLソースファイル)
- ONLINE_STORE. COB (COBOLソースファイル)
- STORESV2. PRJ (プロジェクトファイル)
- STORESV2. CBI (翻訳オプションファイル)
- STORESV2_DLL. CSI (COMサーバ情報ファイル)
- STORESV2. DEF (モジュール定義ファイル)
- SAMPLE28. TXT (プログラム説明書)

使用しているCOBOLの機能

- COMサーバ機能
- リモートデータベースアクセス
- *COM-ARRAYクラス

- オブジェクトコンテキストオブジェクト

使用しているCOBOLの文

IF文、INVOKE文、INITIALIZE文、SET文、MOVE文、PERFORM文

埋込みSQL文(CONNECT文、INSERT文、SELECT文、UPDATE文、ROLLBACK文、DISCONNECT文)

プログラムの翻訳・リンク・実行

プログラムを実行する前に

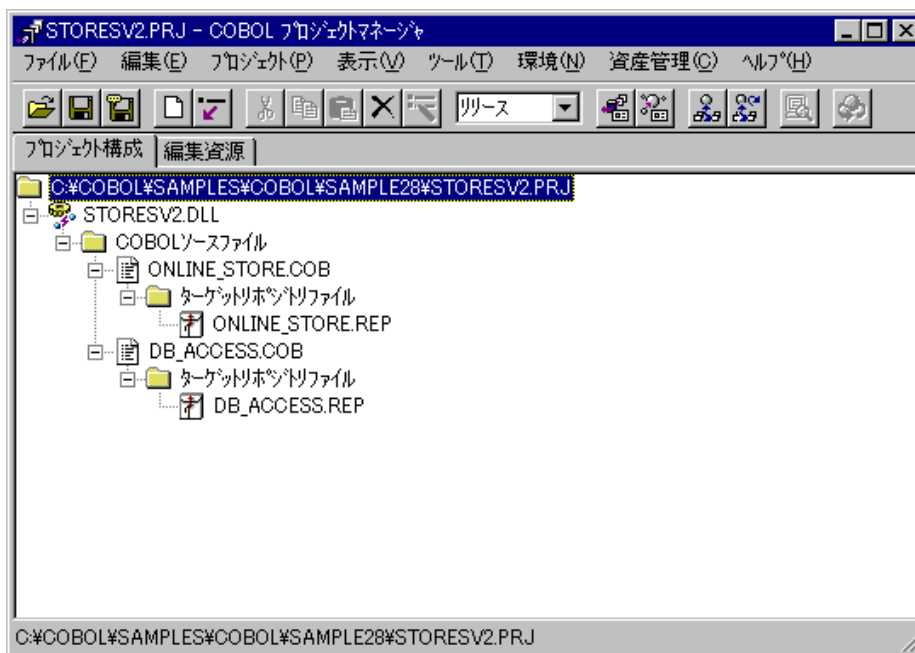
- ODBCドライバを経由してデータベースへアクセスできる環境を構築しておいてください。
- デフォルトで接続するサーバを設定し、そのサーバのデータベース上に次の4つの表を作成しておいてください。作成する表の形式および格納するデータについては“[例題25 COM連携-COBOLによるCOMサーバプログラムの作成](#)”を参照してください。
 - 顧客表
 - 在庫表
 - オーダー表
 - オーダー明細表
- ODBC情報ファイル設定ツール(SQLODBCS.EXE)を使用して、ODBC情報ファイル(ここではC:\FDBMSACS.INFとします)を作成してください。

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

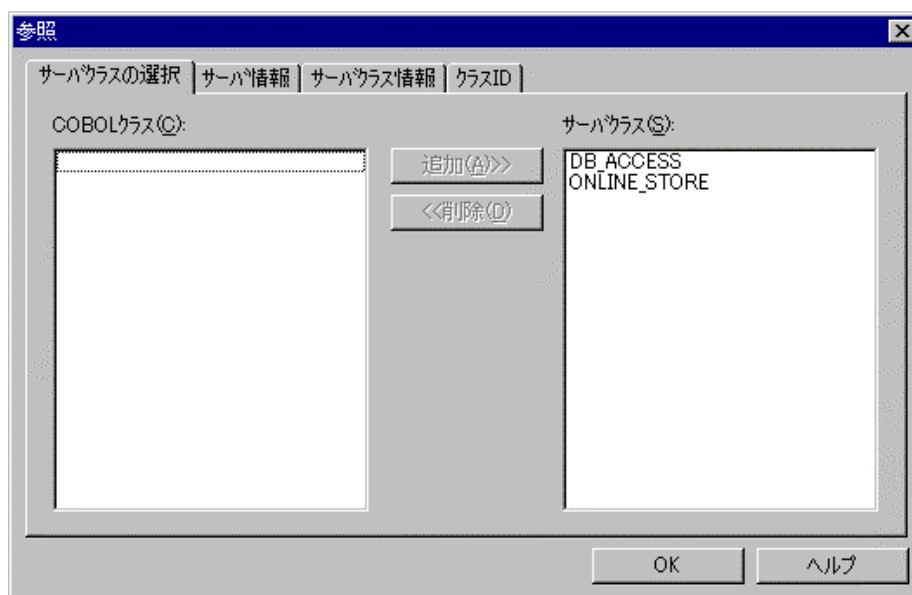
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“STORESV2.PRJ”を開きます。



3. 設定されているCOMサーバ情報を確認します。
ターゲットファイル(STORESV2.DLL)を選択し、[プロジェクト] - [オプション] - [COMサーバ] メニューから“参照”を選択します。



〔参照〕 ダイアログが表示され、設定されているCOMサーバ情報が参照できます。



4. プロジェクトマネージャの〔プロジェクト〕メニューから“ビルド”を選択します。
→ ビルド終了後、STORESV2.DLLが作成されていることを確認してください。

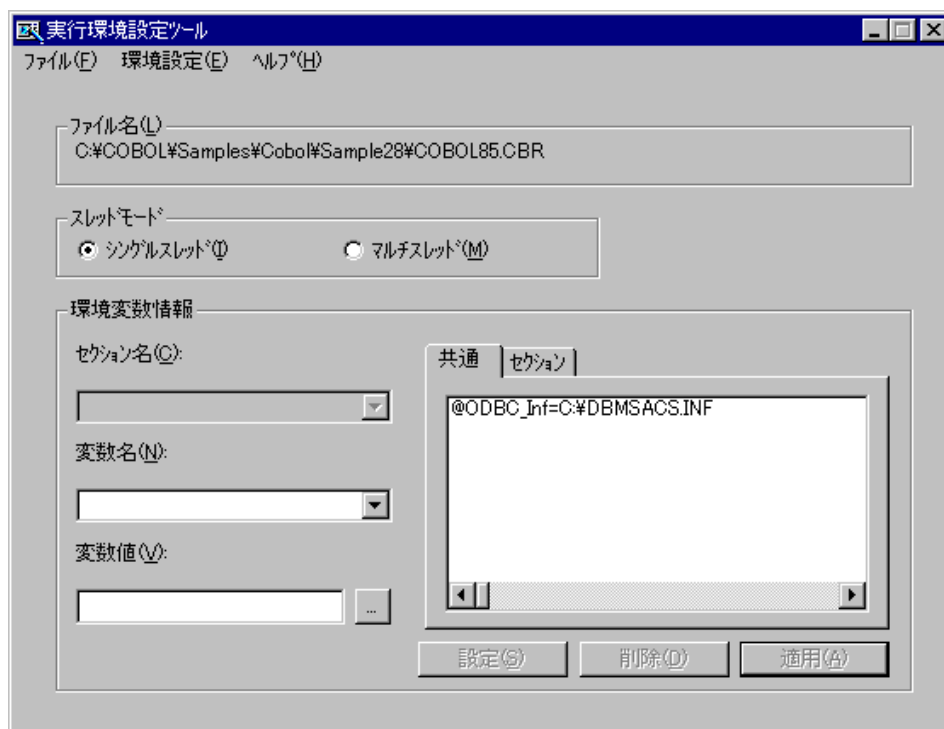
MTSへの登録

作成したCOBOLアプリケーションをCOMサーバとして使用するためには、WindowsシステムおよびMTSへの登録が必要です。またこの際、トランザクションの制御方法を指定します。

MTSエクスプローラを使用して、システムのレジストリおよびMTSに登録します。詳細については“NetCOBOL 使用手引書”の“25.4.2 MTS環境への登録方法”を参照してください。

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの〔ツール〕メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



2. [ファイル] メニューの“開く”を選択し、ダイナミックリンクライブラリ (SAMPLE28.DLL)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@ODBC_Inf(ODBC情報ファイルの指定)に、ODBC情報ファイル名を指定します。
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

クライアントプログラムの修正

例題26および例題27を次のように修正することによって、このプログラムのクライアントとして使用することができます。

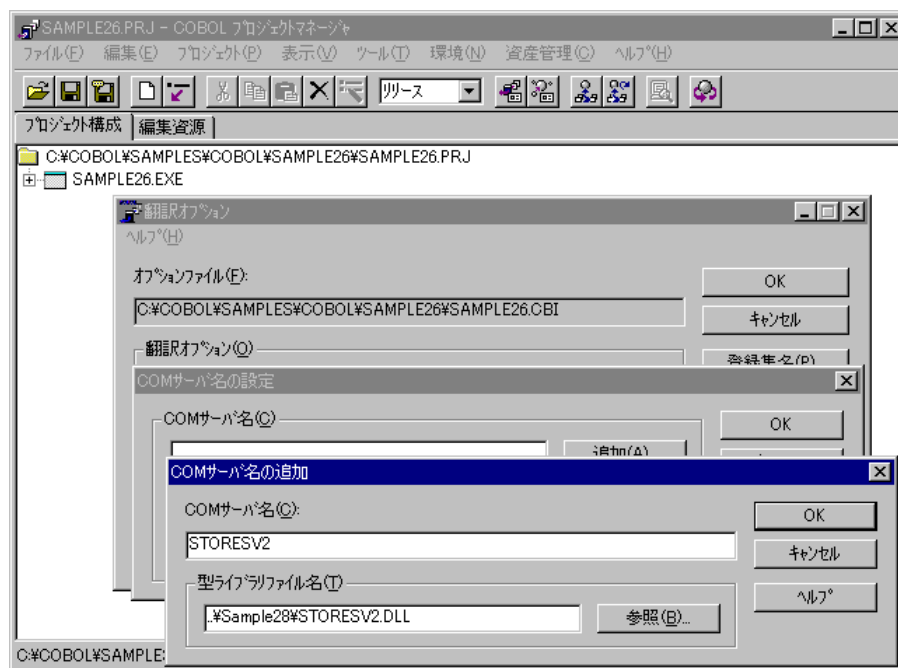
- 例題26 COBOLクライアントからの使用
 - CLMAIN.COBのリポトリ段落のクラス指定子を次のように修正します。

```

-----
000200      CLASS ONLINE_STORE AS “*COM:STORESV2:ONLINE_STORE”
                               ~~~~~
-----

```

- プロジェクトファイル“SAMPLE26.PRJ”を開き、COMサーバ名の設定“STORESV1”を削除します。代わりに、“STORESV2”を追加します。



- SAMPLE26.EXEをリビルドします。
- 例題28のプログラムについてのクライアント情報のインストールプログラムを作成し、これをクライアント側にインストールします。

● 例題27 ASPクライアントからの使用

COMオブジェクトの生成を行うCreateObjectの引数の指定を次のように修正します。

Catalog.asp

```
Set OLSService = Server.CreateObject("STORESV2.ONLINE_STORE")
~~~~~
```

Confirm.asp

```
Set OLSService = Server.CreateObject("STORESV2.ONLINE_STORE")
~~~~~
```

Order.asp

```
Set Obj = Server.CreateObject("STORESV2.ONLINE_STORE")
~~~~~
```


例題29 簡易アプリ間通信機能を使ったメッセージ通信

ここでは、本製品で提供するサンプルプログラム-例題29-について説明します。

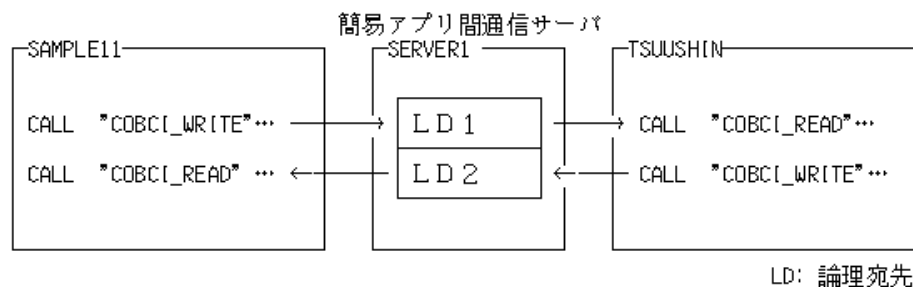
例題29では、簡易アプリ間通信機能を使って、アプリケーション間で論理宛先にメッセージを書き込んだり、論理宛先からメッセージを読み込んだりするプログラムの例を示します。メッセージ通信を行う場合の簡易アプリ間通信機能の使い方は、“NetCOBOL 使用手引書”の“20.4 簡易アプリ間通信機能の使い方”を参照してください。

概要

プログラム-SAMPLE29とプログラム-TSUUSHINの間でメッセージ通信を行います。

SAMPLE29は、サーバ“SERVER1”を接続し、論理宛先“MYLD1”にメッセージを書き込み、論理宛先“MYLD2”からメッセージを読み込みます。このとき、論理宛先“MYLD2”にメッセージの書き込みがなければ、60秒間メッセージを待ちます。論理宛先“MYLD2”からメッセージを読み込んだ後、論理宛先“MYLD1”から優先順位の高い順にメッセージを読み込みます。

TSUUSHINは、サーバ“SERVER1”を接続し、論理宛先“MYLD1”に書き込まれているメッセージを読み込み、論理宛先“MYLD1”および“MYLD2”にメッセージを書き込みます。



提供プログラム

- SAMPLE29. PRJ (プロジェクトファイル)
- SAMPLE29. COB (COBOL ソースプログラム)
- TSUUSHIN. COB (COBOL ソースプログラム)
- PRM_REC. CBL (登録集原文)
- SAMPLE29. INI (論理宛先定義ファイル)
- SAMPLE29. TXT (プログラム説明書)

使用しているCOBOLの機能

- 簡易アプリ間通信機能

使用しているCOBOLの文

CALL文、DISPLAY文、IF文、MOVE文

プログラムの翻訳・リンク・実行

プログラムを実行する前に

- 簡易アプリ間通信のサーバを起動しておいてください。
- 簡易アプリ間通信のサーバに論理宛先“LD1”および“LD2”を創成しておいてください。プログラム実行前のサーバの論理宛先情報は、以下のとおりです。

論理宛先情報

通信システム設定情報

最大格納メッセージ数: 999999999 最大待ち命令数: 999999999

論理宛先(L)	読込	書込	最大優先順位	最大格納メッセージ	現メッセージ	待ち命令
LD1	<input type="radio"/>	<input type="radio"/>	9	999999999	0	0
LD2	<input type="radio"/>	<input type="radio"/>	9	999999999	0	0

更新(R) キャンセル ヘルプ

- 論理宛先定義ファイル(SAMPLE29. INI)の相手マシン名の情報を論理宛先定義ファイル作成ユーティリティ(COBCIU32. EXE)を使って変更してください。
相手マシン名には、サーバが動作しているマシンのホスト名を指定し、[登録] ボタンをクリックします。初期状態では、lpfm0000となっていますが、動作環境によって書き換えてください。

クライアント定義

設定内容

サーバ名(S): 登録(D) キャンセル

相手マシン名(O): 削除(D) ヘルプ

論理宛先情報

自論理宛先(I): 相手論理宛先(R): 追加(A)

削除(E)

宛先情報一覧(L)

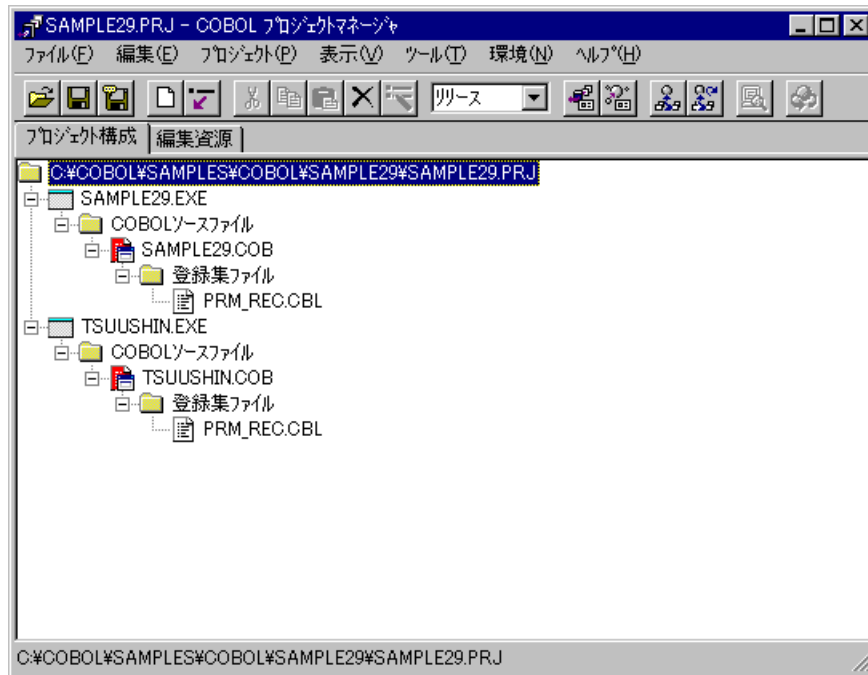
MYLD1=LD1
MYLD2=LD2

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。以降の説明で、フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

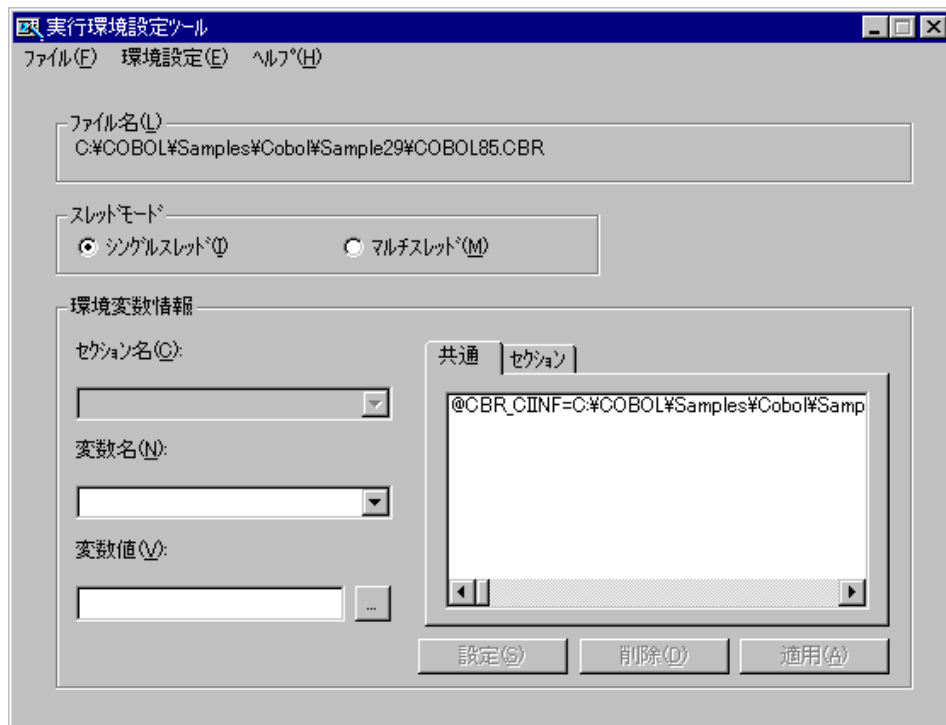
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE29. PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE29.EXEとTSUUSHIN.EXEが作成されていることを確認してください。

実行環境情報の設定

1. プロジェクトマネージャの「ツール」メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが起動されます。



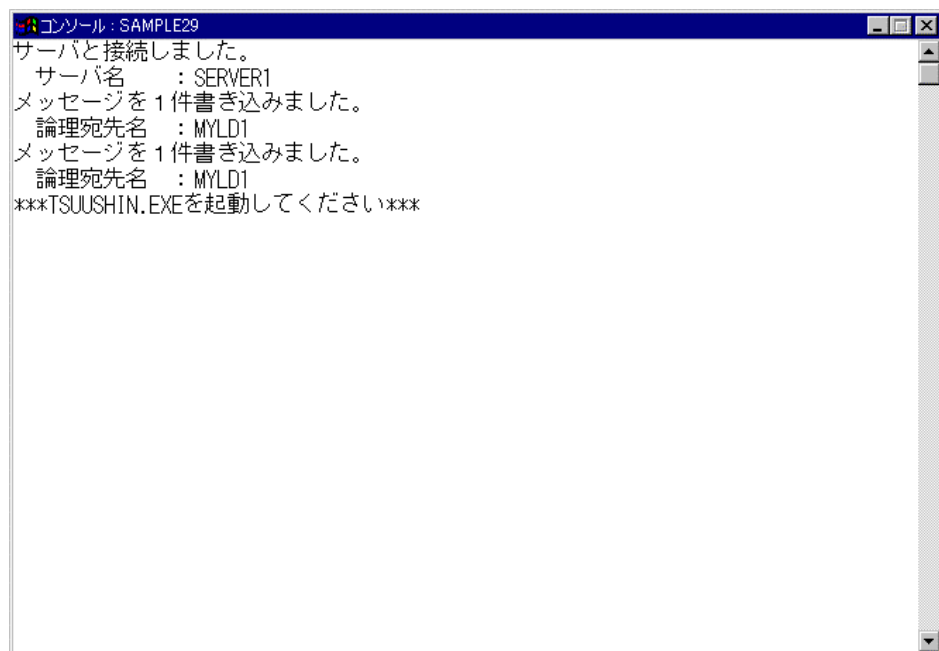
2. 「ファイル」メニューの“開く”を選択し、実行可能プログラム (SAMPLE29.EXE) が存在するフォルダに、実行用の初期化ファイル (COBOL85.CBR) を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@CBR_CIINF (論理宛先定義ファイルの指定) に、論理宛先定義ファイルのパス名を指定します。

この例題プログラムでは、使用する実行用の初期化ファイルが1つであり、指定する内容もSAMPLE29.EXEとTSUUSHIN.EXEプログラムで同じなので、この指定だけで問題ありません。

4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

1. プロジェクトマネージャでSAMPLE29.EXEを選択し、[プロジェクト]メニューから“実行”を選択します。
2. SAMPLE29を起動すると、データを2件サーバに書き込んだ後、“***TSUUSHIN.EXEを起動してください***”というメッセージがコンソールに表示されます。



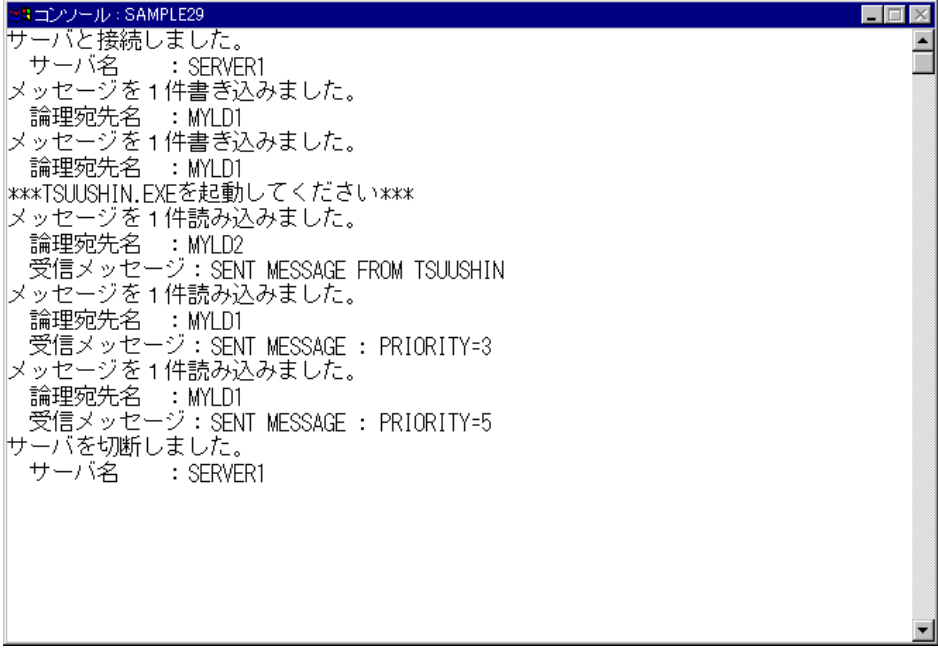
3. SAMPLE29が待ち状態になった後で、TSUUSHINを起動します。起動の方法は、1. と同じです。

実行結果

コンソールに表示された受信メッセージから以下のことを確認してください。

SAMPLE29のコンソールウィンドウ

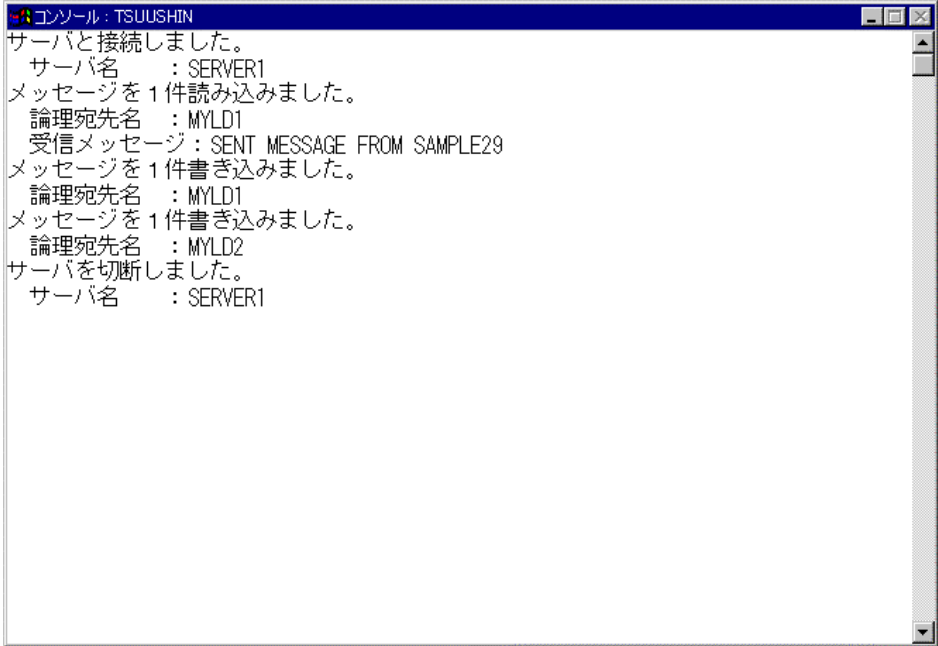
1. TSUUSHINからのメッセージを論理宛先“MYLD2”から読み込んだ。
2. 論理宛先“MYLD1”から優先順位の順番に従って、メッセージを読み込んだ。



```
CONSOLE: SAMPLE29
サーバと接続しました。
  サーバ名      : SERVER1
メッセージを1件書き込みました。
  論理宛先名    : MYLD1
メッセージを1件書き込みました。
  論理宛先名    : MYLD1
***TSUUSHIN.EXEを起動してください***
メッセージを1件読み込みました。
  論理宛先名    : MYLD2
受信メッセージ: SENT MESSAGE FROM TSUUSHIN
メッセージを1件読み込みました。
  論理宛先名    : MYLD1
受信メッセージ: SENT MESSAGE : PRIORITY=3
メッセージを1件読み込みました。
  論理宛先名    : MYLD1
受信メッセージ: SENT MESSAGE : PRIORITY=5
サーバを切断しました。
  サーバ名      : SERVER1
```

TSUUSHINのコンソールウィンドウ

1. SAMPLE29からのメッセージを論理宛先“MYLD1”から読み込んだ。
2. 論理宛先“MYLD1”と“MYLD2”にメッセージを書き込んだ。



```
CONSOLE: TSUUSHIN
サーバと接続しました。
  サーバ名      : SERVER1
メッセージを1件読み込みました。
  論理宛先名    : MYLD1
受信メッセージ: SENT MESSAGE FROM SAMPLE29
メッセージを1件書き込みました。
  論理宛先名    : MYLD1
メッセージを1件書き込みました。
  論理宛先名    : MYLD2
サーバを切断しました。
  サーバ名      : SERVER1
```

例題30 Unicodeを使用するプログラム

ここでは、本製品で提供するサンプルプログラム-例題30-について説明します。

例題30では、UCS-2のファイルレコードを入力し、それらを表示・印刷するプログラムの例を示します。

なお、このプログラムを動作させるためには、以下のオペレーティングシステムが必要です。

- Windows NT(R) 4.0 Service Pack 4以上
- Microsoft(R) Windows(R) 2000
- Microsoft(R) Windows(R) XP
- Microsoft(R) Windows Server(TM) 2003

概要

Unicode固有の漢字および英語の発音記号が格納されているファイル(UCS-2の行順ファイル)のレコードを読み出し、そのデータを出力します。画面には、UCS-2のデータを表示します。印刷ファイルには、UCS-2のデータの他に、レコード件数を示す数字をUTF-8で出力します。

提供プログラム

- SAMPLE30.COB(COBOLソースプログラム)
- SAMPLE30.PRJ(プロジェクトファイル)
- SAMPLE30.CBI(翻訳オプションファイル)
- COBOL85.CBR(実行用の初期化ファイル)
- INDATA(入力ファイル)
- SAMPLE30.TXT(プログラム説明書)

使用しているCOBOLの機能

- プログラム間連絡機能
- 組込み関数機能
- 小入出力機能(コンソールウィンドウ)
- 印刷ファイル
- 行順ファイル(参照)
- 内部プログラム
- プロジェクト管理機能

使用しているCOBOLの文

CALL文、ACCEPT文、DISPLAY文、PERFORM文、IF文、EVALUATE文、GO TO文、MOVE文、COMPUTE文、OPEN文、CLOSE文、READ文、WRITE文、EXIT文

プログラムの翻訳・リンク・実行

プログラムを実行する前に

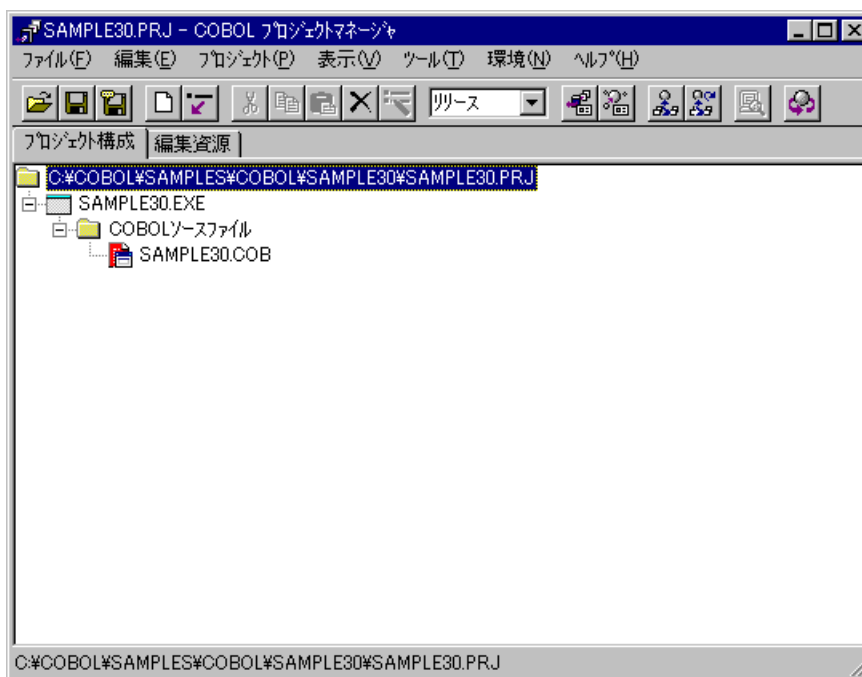
印刷ファイルの内容が標準のプリンタに出力されます。“標準のプリンタ”の設定を確認してください。

ビルド・リビルド

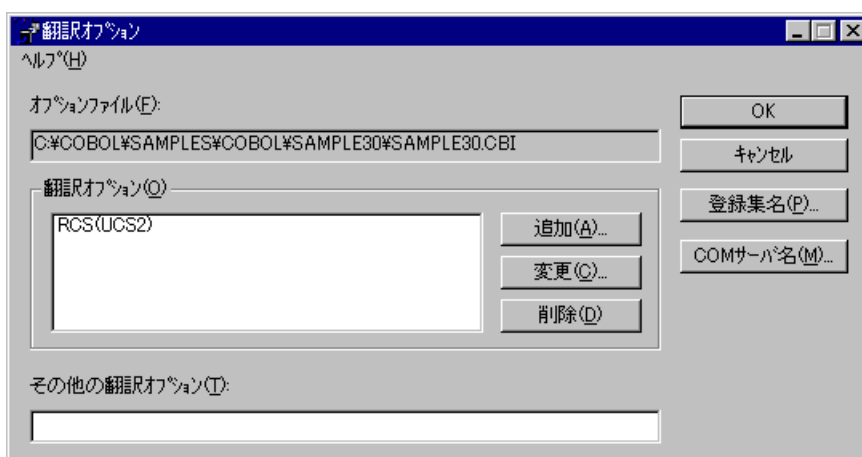
翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、以降ではNetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。フォルダ名がC:\¥COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE30.PRJ”を開きます。



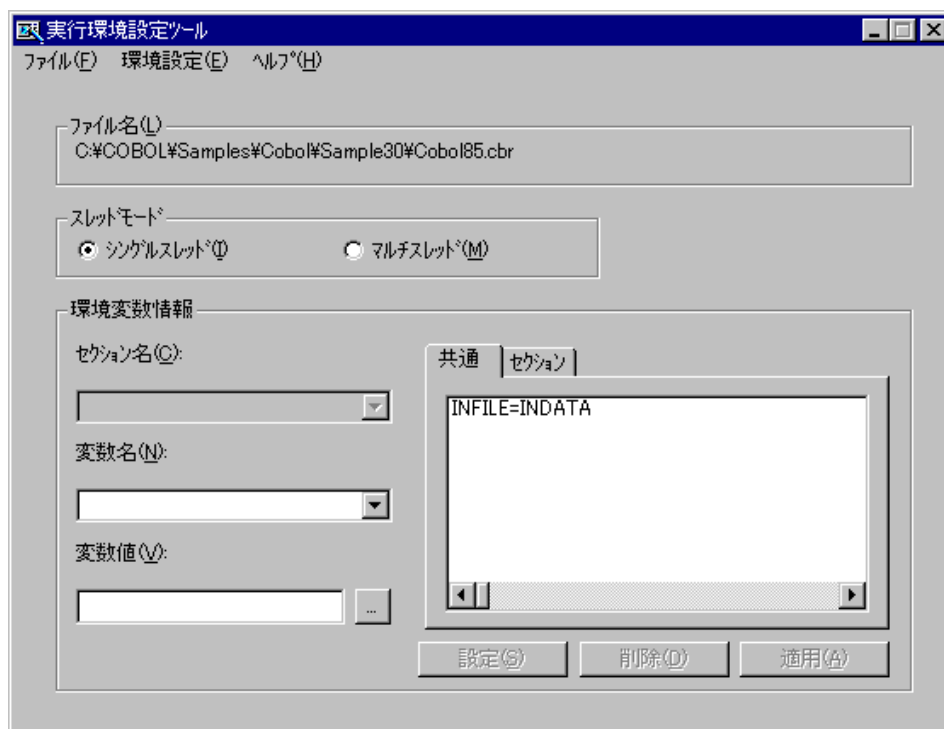
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。



4. 翻訳オプションRCS(UCS2)を指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE30.EXEが作成されていることを確認してください。

実行環境情報の設定

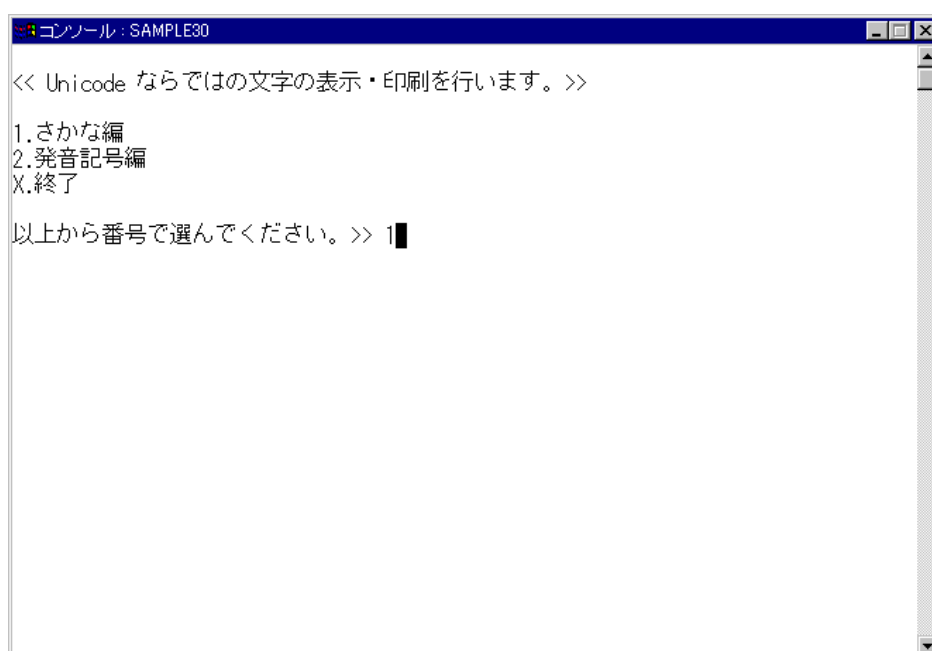
1. プロジェクトマネージャの [ツール] メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。



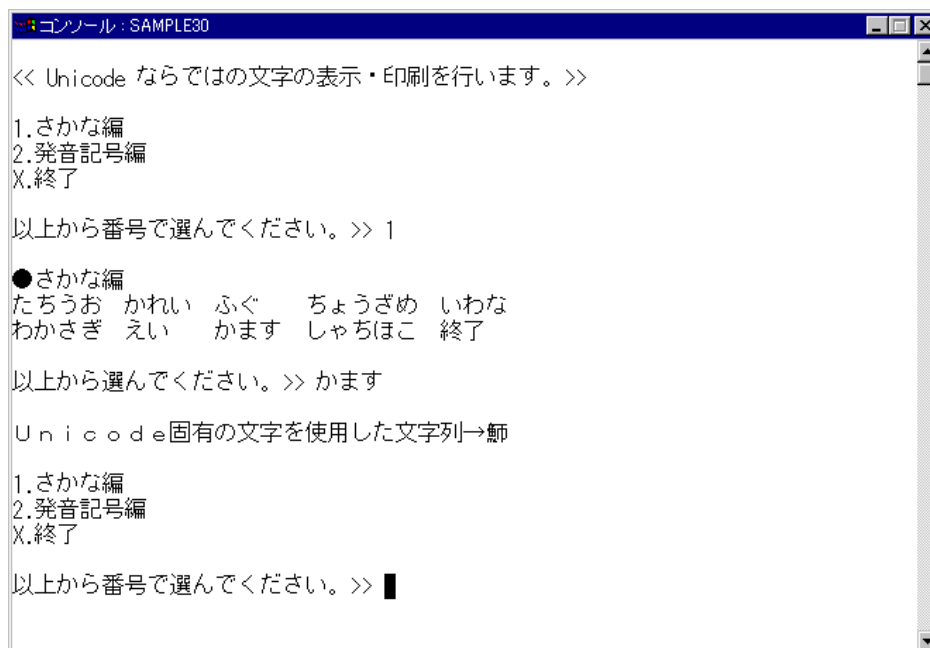
2. [ファイル] メニューの“開く”を選択し、実行可能プログラム (SAMPLE30. EXE) が存在するフォルダに、実行用の初期化ファイル (COBOL85. CBR) を作成します。
3. 共通タブを選択し、以下を設定します。
 - ファイル識別名 INFILE にデータファイル (行順ファイル) のファイル名 (INDATA) を指定します。
4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

プログラムの実行

1. プロジェクトマネージャの [プロジェクト] メニューから“実行”を選択します。
 - COBOL コンソール画面が開き、以下のメッセージを表示して入力待ちになります。



2. 表示する文字列の種類を選択してください。
- “1”を入力し、ENTERキーを押すと、日本語文字列がいくつかひらがなで提示されます。それらのうちからひとつを選んで入力すると、対応する漢字が画面に表示されます。



3. 終了する場合、“X”を入力します。
- 実行が終了すると、印刷ファイルの内容が標準のプリンタに出力されます。

例題31 メッセージボックスの出力

ここでは、本製品で提供するサンプルプログラム-例題31-について説明します。

例題31では、プログラム間連絡機能を使って、Windowsシステム関数を呼び出し、メッセージボックスを出力するプログラムの例を示します。プログラム間連絡機能の詳細については、“NetCOBOL 使用手引書”の“第10章 サブプログラムを呼び出す～プログラム間連絡機能～”を参照してください。

なお、この例題ではCOBOLプログラムの復帰値をバッチファイルで参照します。

概要

STDCALL呼出し規約を使用し、CALL文でWindowsシステム関数の“MessageBoxA”（末尾に“A”が付いていることに注意してください）を呼び出します。[はい] / [いいえ] / [キャンセル] ボタンを持つメッセージボックスを出力し、メッセージボックスからの復帰値をCALL文のRETURNING指定で受け取ります。さらにその復帰値に対応する値をCOBOLプログラムからの復帰値として、バッチファイルで参照します。

COBOLプログラムの復帰値はバッチファイルではERRORLEVELという名前で参照することができます。以下は“SAMPLE31.BAT”の一部です。

```
:START
start /w MsgBox.exe
@rem 復帰値が9999以上ならCOBOLプログラムを再度呼び出す
if errorlevel 9999 goto START
@rem 復帰値が9以上なら『いいえ』が押された。
if errorlevel 9 goto NG
echo 『はい』が押されました。
```

提供プログラム

- MSGBOX. COB (COBOLソースプログラム)
- SAMPLE31. PRJ (プロジェクトファイル)
- SAMPLE31. CBI (翻訳オプションファイル)
- SAMPLE31. BAT (起動用バッチファイル)
- SAMPLE31. TXT (プログラム説明書)

使用しているCOBOLの機能

- COBOLプログラムからCプログラムを呼び出す方法
- STDCALL呼出し規約
- BY VALUEでのパラメタの受渡し
- CALL文のRETURNING指定
- 特殊レジスタPROGRAM-STATUS
- プロジェクト管理機能



注意

- メッセージ本文とメッセージタイトルの文字列の末尾には、“X” 00” または LOW-VALUEを格納しなければなりません。文字列を部分参照して、末尾に“X” 00” またはLOW-VALUEを格納する必要があります。
- メッセージボックスを表示する関数名は、“MessageBoxA” です。小文字を有効に

するため、翻訳オプションNOALPHALまたはALPHAL(WORD)を指定する必要があります。

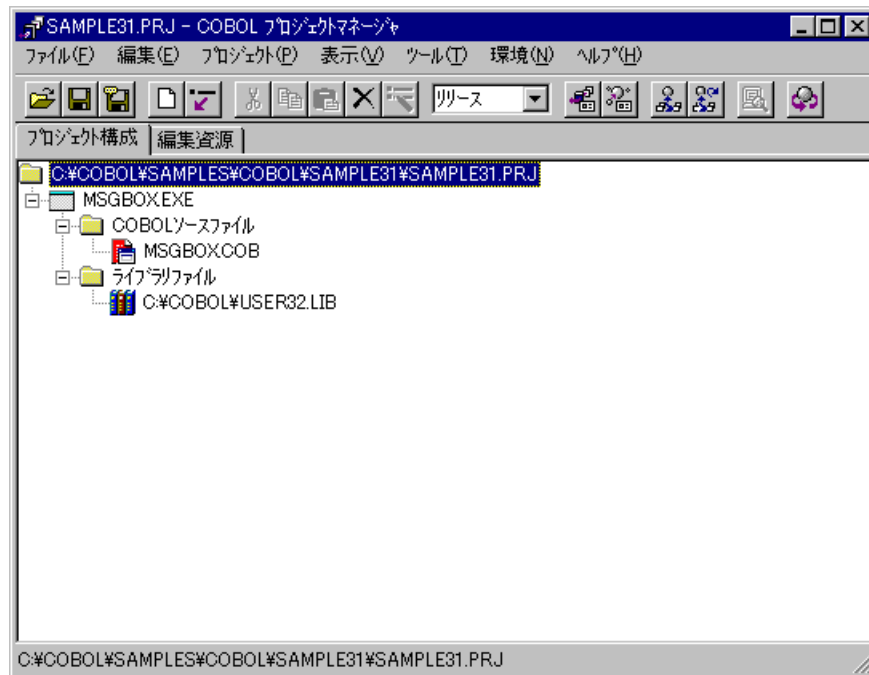
プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用していきます。

なお、以降ではNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

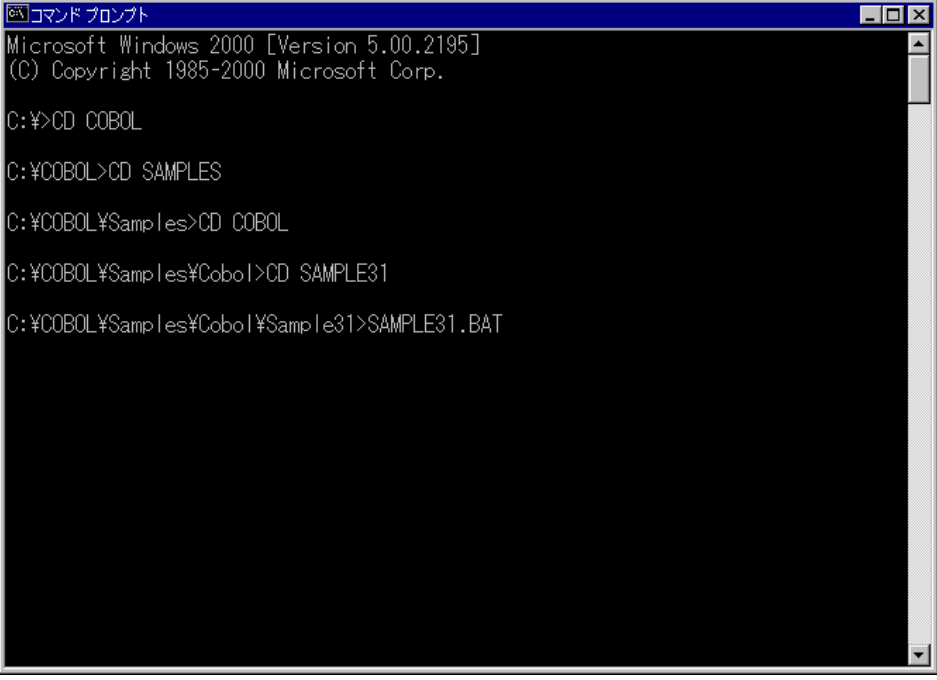
1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE31.PRJ”を開きます。



3. プロジェクトマネージャの「プロジェクト」メニューから“ビルド”を選択します。
→ ビルド終了後、MSGBOX.EXEが作成されていることを確認してください。

プログラムの実行

1. コマンドプロンプトを開き、実行可能プログラムと同じフォルダにあるバッチファイル“SAMPLE31.BAT”を実行します。



```

コマンド プロンプト
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>CD COBOL

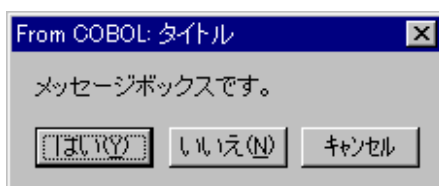
C:\COBOL>CD SAMPLES

C:\COBOL\SAMPLES>CD COBOL

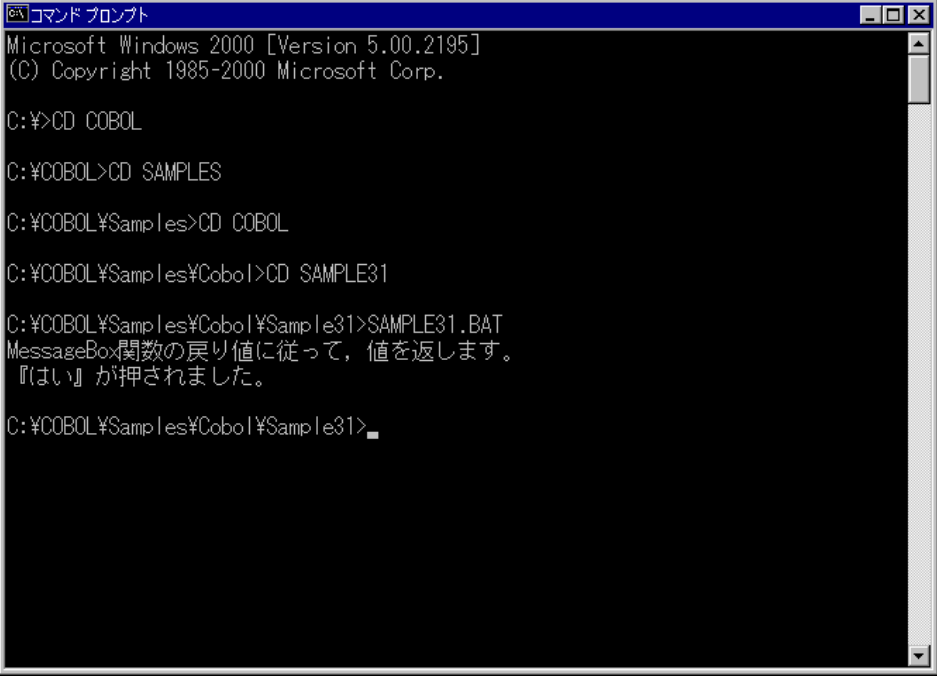
C:\COBOL\SAMPLES\COBOL>CD SAMPLE31

C:\COBOL\SAMPLES\COBOL\SAMPLE31>SAMPLE31.BAT
  
```

2. 以下のメッセージボックスが表示されます。[はい]、[いいえ] または [キャンセル] ボタンをクリックします。



3. [はい] または [いいえ] のボタンをクリックすると、どちらのボタンが押されたかがコマンドプロンプトに表示されます。[はい] ボタンをクリックすると、次のように表示されます。



```

コマンド プロンプト
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>CD COBOL

C:\COBOL>CD SAMPLES

C:\COBOL\SAMPLES>CD COBOL

C:\COBOL\SAMPLES\COBOL>CD SAMPLE31

C:\COBOL\SAMPLES\COBOL\SAMPLE31>SAMPLE31.BAT
MessageBox関数の戻り値に従って、値を返します。
『はい』が押されました。

C:\COBOL\SAMPLES\COBOL\SAMPLE31>
  
```

4. [キャンセル] ボタンをクリックした場合、実行可能プログラムが再度実行されます。

例題32 他のプログラムの起動

ここでは、本製品で提供するサンプルプログラム-例題32-について説明します。

例題32では、プログラム間連絡機能を使って、Windowsシステム関数を呼び出し、他のプログラムあるいはバッチファイルを起動して、その終了コードを受け取るプログラムの例を示します。プログラム間連絡機能の詳細については、“NetCOBOL 使用手引書”の“第10章 サブプログラムを呼び出す～プログラム間連絡機能～”を参照してください。

概要

起動するプログラムあるいはバッチファイルのパス名と、必要ならコマンド文字列を入力します。これを引数に指定してWindowsシステム関数を呼び出し、指定したプログラムあるいはバッチファイルを起動します。また、起動に成功した場合、その実行が終了するまで待つ、終了コードを受け取ります。

提供プログラム

- SAMPLE32. PRJ(プロジェクトファイル)
- SAMPLE32. COB(COBOLソースプログラム)
- SAMPLE32. TXT(プログラム説明書)

使用しているCOBOLの機能

- COBOLプログラムからCプログラムを呼び出す方法
- STDCALL呼出し規約
- BY VALUEでのパラメタの受渡し
- CALL文のRETURNING指定
- STORED-CHAR-LENGTH関数
- プロジェクト管理機能

プログラムの翻訳・リンク・実行

プログラムを実行する前に

起動するプログラムとして、以下の例題プログラムの実行可能ファイルを使用します。

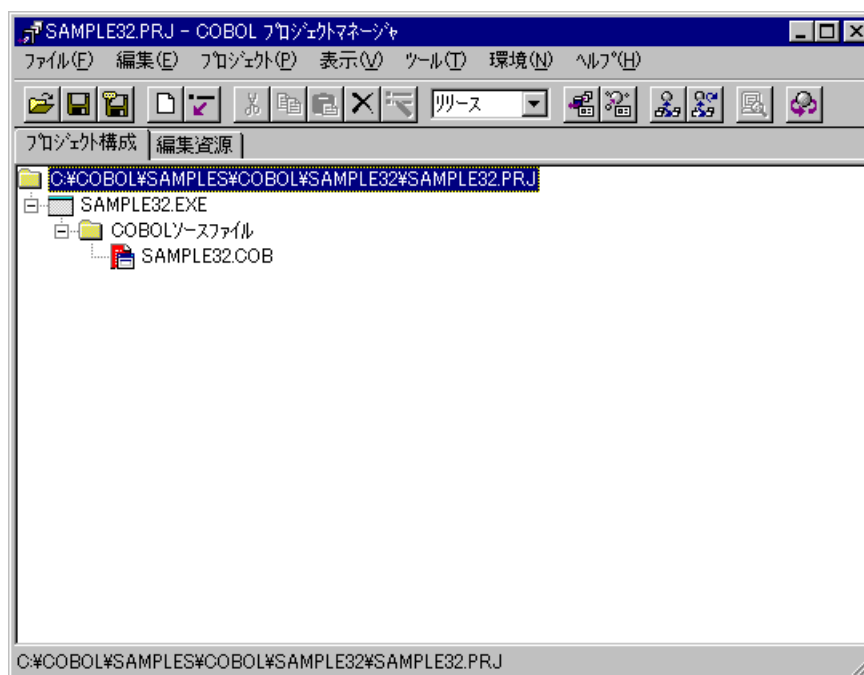
- 例題6のSAMPLE6. EXE
- 例題31のMSGBOX. EXE

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルはNetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。以降の説明で、フォルダ名がC:\COBOLとなっているところは、これをNetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAMPLE32. PRJ”を開きます。



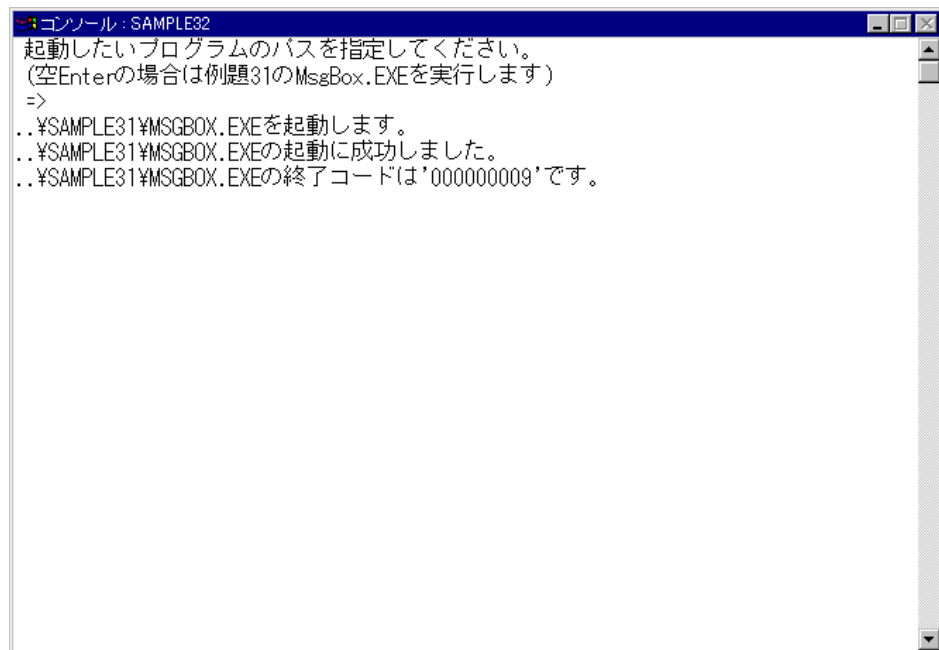
3. プロジェクトマネージャの〔プロジェクト〕メニューから“ビルド”を選択します。
→ ビルド終了後、SAMPLE32.EXEが作成されていることを確認してください。

プログラムの実行

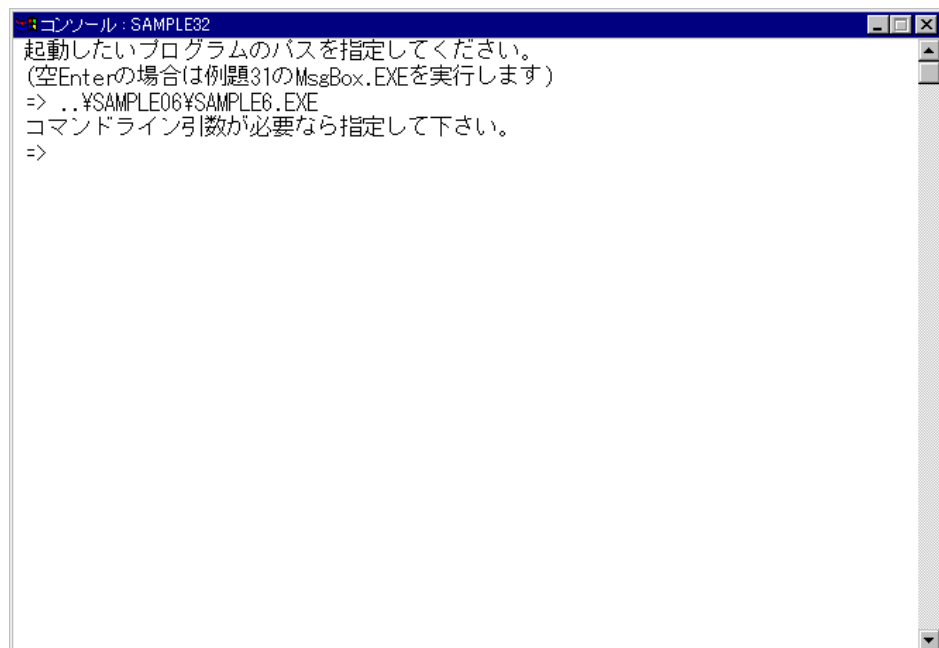
1. プロジェクトマネージャの〔プロジェクト〕メニューから“実行”を選択します。
→ 次の表示が現れて入力待ちになります。



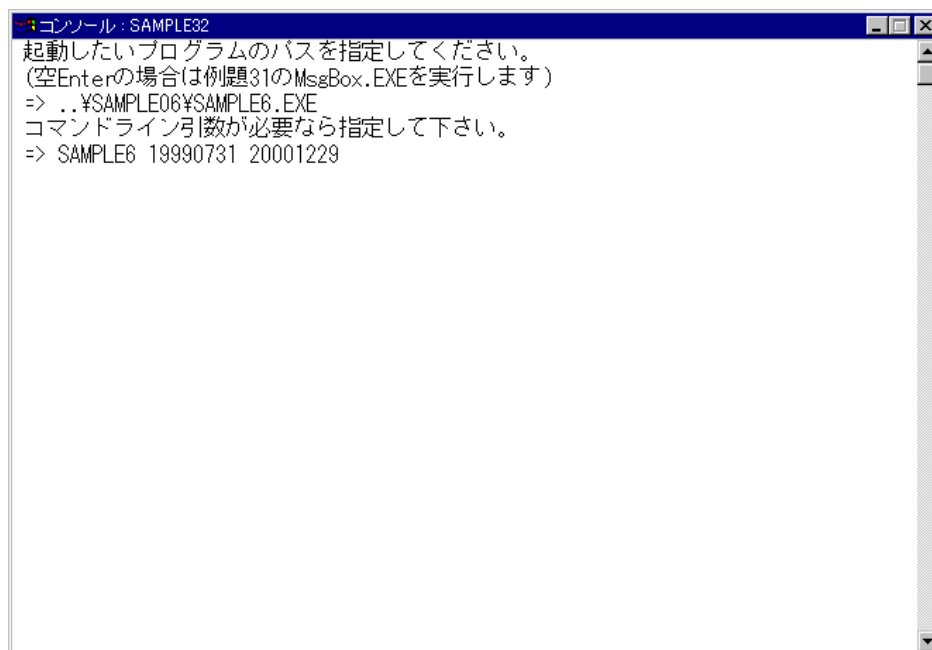
2. 起動するプログラムあるいはバッチファイルのパス名を入力します。ここでは環境変数 PATHの指定は無効であるため、絶対パスまたは、SAMPLE32.EXEを実行したフォルダからの相対パスを指定する必要があります。
3. 何も入力しないでENTERキーを押すと、例題31のMSGBOX.EXEを実行します。
4. MSGBOX.EXEを起動する旨のメッセージが表示され、例題31と同様のメッセージボックスが表示されます。メッセージボックスのボタンのどれかをクリックすると、例題31のMSGBOX.EXEの終了コードが示されて、プログラムが終了します。[いいえ] ボタンをクリックした場合、次のように表示されます。



5. 起動するプログラムやバッチファイルのパス名を明示的に指定した場合、コマンド行引数の入力を促すメッセージが表示されて、入力待ちになります。コマンド行引数が必要ならここで入力します。不要な場合は、何も入力しないでENTERキーを押してください。ここでは、例題6のSAMPLE6.EXEを実行します。



6. 例題6のSAMPLE6.EXEは2つのコマンド行引数が必要なため、ここで指定します。コマンド行引数をプログラム名に続けて指定することに注意してください。

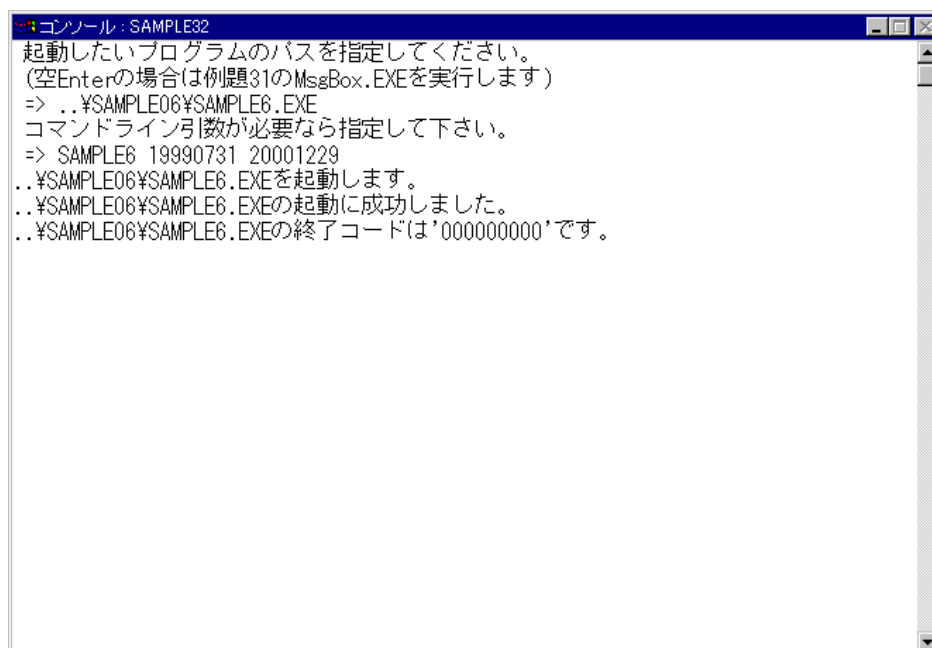


```

コンソール : SAMPLE32
起動したいプログラムのパスを指定してください。
(空Enterの場合は例題31のMsgBox.EXEを実行します)
=> ..¥SAMPLE06¥SAMPLE6.EXE
コマンドライン引数が必要なら指定して下さい。
=> SAMPLE6 19990731 20001229

```

7. SAMPLE6.EXEを起動する旨のメッセージが表示され、SAMPLE6.EXEが実行されます(システムのコンソールが開かれて実行結果が出力されます)。実行が終了すると、SAMPLE6.EXEの終了コードが表示されます。



```

コンソール : SAMPLE32
起動したいプログラムのパスを指定してください。
(空Enterの場合は例題31のMsgBox.EXEを実行します)
=> ..¥SAMPLE06¥SAMPLE6.EXE
コマンドライン引数が必要なら指定して下さい。
=> SAMPLE6 19990731 20001229
..¥SAMPLE06¥SAMPLE6.EXEを起動します。
..¥SAMPLE06¥SAMPLE6.EXEの起動に成功しました。
..¥SAMPLE06¥SAMPLE6.EXEの終了コードは'00000000'です。

```

COBOL Webサブルーチンの例題プログラム

NetCOBOLでは、以下のプログラムをCOBOL Webサブルーチンを使用したサンプルとして提供しています。

[例題 CGIサブルーチンを使ったプログラム](#)

[例題 ISAPIサブルーチンを使ったプログラム](#)

[例題 SAFサブルーチンを使ったプログラム](#)

[例題 セッション管理機能を使ったプログラム](#)

例題 CGIサブルーチンを使ったプログラム

ここでは、本製品で提供するCOBOL CGIサブルーチンのサンプルプログラムについて説明します。
COBOL CGIサブルーチンを使って、Webサーバとブラウザ間の情報を交換するプログラム例を示します。

概要

Webブラウザより名前、趣味、性別を受け取り、受け取った情報を元にWebサーバ上で結果出力用ページを作成してWebブラウザに表示します。

提供プログラム

- CGISMP01.COB (COBOL ソースプログラム)
- CGISMP01.HTM (呼出し用ページ)
- CGISMP01_1.HTM (結果出力用ページ)
- COBOL85.CBR (実行用の初期化ファイル)
- CGISMP01.TXT (プログラム説明書)

使用しているCGIサブルーチン

- COBW3_INIT
- COBW3_GET_VALUE_XX
- COBW3_SET_CNV_XX
- COBW3_PUT_HTML
- COBW3_PUT_TEXT
- COBW3_FREE

プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。
なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。以降の説明で、フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“CGISMP01.PRJ”を開きます。
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. 翻訳オプションLIBに、COBOL CGIサブルーチンの登録集ファイル(COBW3.cb1)が格納されているフォルダを指定します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトにインポートライブラリF3BICWSR.LIBが指定されていることを確認します。
6. プロジェクトマネージャの[プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、CGISMP01.EXEが作成されていることを確認してください。

[参照] “COBOL Webサブルーチン使用手引書”の“2.4.1 翻訳およびリンク”

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。
2. [ファイル]メニューの“開く”を選択し、実行可能プログラム(CGISMP01.EXE)が存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@MessOutFileに、出力メッセージの格納ファイル名を指定します。
 - 環境変数情報@WinCloseMsgに、OFFを指定します。
 - 環境変数情報@CBR_CGI_LOGFILEに、ログファイル名を指定します。

— 環境変数情報@CBR_CGI_SEVERITYに、重要度を指定します。

4. [適用] ボタンをクリックします。

→ 設定した内容が実行用の初期化ファイルに保存されます。

5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

[参照] “COBOL Webサブルーチン 使用手引書”の“2.4.2 CGIサブルーチンの環境変数設定”

プログラムを実行する前に

呼出し用ページ (CGISMP01. HTM) のFORMタグのACTION属性に指定されているWebアプリケーションのパスを、実際に実行するWebサーバの仮想パスに変更してください。

プログラムの実行

Webサーバで指定されたフォルダにCGISMP01. EXE、CGISMP01. HTM、CGISMP01_1. HTM、COBOL85. CBRの各ファイルをコピーします。

WebブラウザでWebサーバのURLを入力し、CGISMP01. HTMを指定すると簡易アンケート画面が表示されます。

各項目を入力または選択し、[実行] ボタンをクリックすると入力した内容がWebサーバに送られ、その結果がWebブラウザに表示されます。

Webサーバへの送信前に入力した内容を消去したい場合は [書き直し] ボタンをクリックしてください。

[参照] “COBOL Webサブルーチン 使用手引書”の“2.4.3 CGIアプリケーションの実行”

例題 ISAPIサブルーチンを使ったプログラム

ここでは、本製品で提供するCOBOL ISAPIサブルーチンのサンプルプログラムについて説明します。

概要

COBOL ISAPIサブルーチンを使って、Cookieを使用したアプリケーション間のデータの引継ぎやサーバやブラウザ情報などを取得する方法を示します。

提供プログラム

- ISAMAIN. cob (COBOLソースプログラム)
- ISAINIT. cob (COBOLソースプログラム)
- ISATERM. cob (COBOLソースプログラム)
- ISASTART. htm (呼出し用ページ)
- ISARPLY1. htm (結果出力用ページ)
- ISARPLY2. htm (結果出力用ページ)
- ISAERROR. htm (結果出力用ページ(エラー処理用))
- ISASMP1. def (モジュール定義ファイル)
- COBOL85. cbr (実行用の初期化ファイル)
- ISASMP1. txt (プログラム説明書)

使用しているISAPIサブルーチン

- COBW3_INIT
- COBW3_SET_CNV_NX
- COBW3_PUT_HTML
- COBW3_RECEIVE_HEADER
- COBW3_GET_REQUEST_INFO
- COBW3_SET_COOKIE_XX
- COBW3_GET_COOKIE_XX
- COBW3_FREE

プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\COBOLとして説明しています。以降の説明で、フォルダ名がC:\COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“ISASMP1.prj”を開きます。
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. 翻訳オプションLIBに、COBOL ISAPIサブルーチンの登録集ファイル(COBW3.cb1、ISAPIINF.cb1、ISAPICTX.cb1、ISAPIFLG.cb1)が格納されているフォルダを指定します。
翻訳オプションALPHALの設定で“英大文字と等価に扱う”を選択し、さらに“WORD - COBOLの語”を選択します。また、翻訳オプションTHREADの設定で、“MULTI - マルチスレッドとする”を選択します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトにインポートライブラリF3BISAPI.libとモジュール定義ファイルISASMP1.defが指定されていることを確認します。
6. プロジェクトマネージャの[プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、ISASMP1.DLLが作成されていることを確認してください。

[参照] “COBOL Webサブルーチン使用手引書” の “3. 4. 1 翻訳およびリンク”

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。
2. [ファイル]メニューの“開く”を選択し、ISASMP11.DLLが存在するフォルダに、実行用の初期化ファイル(COBOL85.cbr)を作成します。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@MessOutFileに、出力メッセージの格納ファイル名を指定します。
 - 環境変数情報@WinCloseMsgに、OFFを指定します。
 - 環境変数情報@CBR_ISAPI_LOGFILEに、ログファイル名を指定します。
 - 環境変数情報@CBR_ISAPI_SEVERITYに、重要度を指定します。
4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル]メニューの“終了”を選択し、実行環境設定ツールを終了します。

[参照] “COBOL Webサブルーチン使用手引書” の “3. 4. 3 ISAPIサブルーチンの環境変数設定”

また、例題が入ったフォルダを物理パスとした適切な仮想ディレクトリをIISに登録してください。IISへの仮想ディレクトリの登録方法は、“COBOL Webサブルーチン使用手引書” の “3. 4. 2 IISの設定” を参照してください。

プログラムを実行する前に

呼出し用ページ(ISASTART.htm)のFORMタグのACTION属性に指定されているWebアプリケーションのパスを、実際に実行するWebサーバの仮想パスに変更してください。

プログラムの実行

IISで指定されたフォルダに例題の各ファイルをコピーします。

IISが起動されていることを確認し、Webブラウザで呼出し用ページ(ISASTART.htm)をIISに登録した仮想ディレクトリで構成されるURLを指定して表示します。あとは画面の指示に従い「実行」ボタンを押してください。

[参照] “COBOL Webサブルーチン使用手引書” の “3. 4. 4 ISAPIアプリケーションの実行”



注意

この例題はCookieをサポートしていないWebブラウザおよびWebブラウザがCookieを受け入れない設定になっている場合、正しく動作しません。

例題の解説

例題の解説をするために、この例題で使用している呼出し用ページ(ISASTART.htm)、結果出力用ページ(ISARPLY1.htm、ISARPLY2.htm)およびCOBOLプログラム(ISAMAIN.cob)を以下に提示します。ただし、COBOLプログラムは、入口名がHttpExtensionProcのプログラム(ISAMAIN.cob)だけ提示します。入口名がGetExtensionVersionおよびTerminateExtensionのプログラム(ISAINIT.cobおよびISATERM.cob)は“COBOL Webサブルーチン使用手引書”の“3. 2. 1 GetExtensionVersion”および“3. 2. 3 TerminateExtension”のひな形と同じため、省略します。

呼出し用ページ(ISASTART.htm)

```
<HTML>
<HEAD>
  <TITLE> 初期画面</TITLE>
</HEAD>

<BODY>
  ISAPI サブルーチンを使用したサンプルです。<BR>
```

動作の確認を行いたい場合は、実行ボタンを押してください。

終了したい場合は、ブラウザを閉じてください。


```
<FORM METHOD="POST" ACTION="isasmpl1.dll">  
  <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">  
</FORM>  
</BODY>  
</HTML>
```

結果出力用ページ (ISARPLY1.htm)

```
<HTML>  
<HEAD>  
  <TITLE> 初回画面</TITLE>  
</HEAD>  
  
<BODY>  
<FONT COLOR=BLUE> はじめてご利用いただきありがとうございます。</FONT><BR>  
継続してご利用される場合は、実行ボタンを押してください。<BR>  
終了したい場合は、ブラウザを閉じてください。<BR>  
  
<FORM METHOD="POST" ACTION="isasmpl1.dll">  
  <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">  
</FORM>  
</BODY>  
</HTML>
```

結果出力用ページ (ISARPLY2.htm)

```
<HTML>  
<HEAD>  
  <TITLE> 初回以降の画面</TITLE>  
</HEAD>  
  
<BODY>  
現在の状況は次のとおりです。<BR>  
<TABLE BORDER=2>  
<TR>  
  <TH>アクセス元ホスト名</TH>  
  <TH>ブラウザ</TH>  
  <TH>アクセス回数</TH>  
</TR>  
<TR>  
  <TD>//COBOL// ホスト名//COBOL//</TD>  
  <TD>//COBOL// ブラウザ名//COBOL//</TD>  
  <TD>//COBOL// アクセス回数//COBOL//</TD>  
</TR>  
</TABLE>  
<BR>
```

継続してご利用される場合は、実行ボタンを押してください。〈BR〉
 終了したい場合は、ブラウザを閉じてください。〈BR〉
 (注: ブラウザを閉じた場合、カウンタはリセットされます。また、
 異なるブラウザや別のマシン上のブラウザでアクセスした場合、
 カウンタ値は異なります)

```
<FORM METHOD="POST" ACTION="isasmpl1.dll">
  <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">
</FORM>
</BODY>
</HTML>
```

COBOL プログラム (ISAMAIN.cob)

```
000010*-----*
000020* ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 1999-2003 *
000030* *
000040* ファイル名 : ISAMAIN.COB *
000050* 概要 : ISAPI サブルーチンの例題 *
000060*-----*
000070 IDENTIFICATION DIVISION.
000080 PROGRAM-ID. "HttpExtensionProc".
000090 ENVIRONMENT DIVISION.
000100 DATA DIVISION.
000110 WORKING-STORAGE SECTION.
000120 COPY COBW3.
000130*
000140 01 HTMLFILENAME PIC X(64).
000150 01 PATHNAME PIC X(256).
000160 01 PATHSIZE PIC 9(05).
000170 01 COPYSTARTPOS PIC 9(05).
000180 01 LEFTLENGTH PIC 9(05).
000190 01 アクセス回数 PIC 9(05).
000200 01 WORK-CNTR1 PIC 9(4) COMP-5.
000210 01 WORK-CNTR2 PIC 9(4) COMP-5.
000220 01 CONVERT-VALUE PIC X(1024).
000230 01 CONVERT-LENGTH PIC S9(4) COMP-5.
000240 01 SANITIZED-DATA PIC X(1024).
000250 01 SANITIZED-LENGTH PIC S9(4) COMP-5.
000260*
000270 LINKAGE SECTION.
000280 COPY ISAPICTX.
000290*
000300 PROCEDURE DIVISION WITH STD CALL LINKAGE USING ISAPI-CTX-CNT.
000310*
000320 ISAPISAMPLE1-START.
000330*
000340* I S A P I サブルーチンパラメタ域の初期化
000350 MOVE LOW-VALUE TO COBW3.
000360 MOVE FUNCTION ADDR(ISAPI-CTX-CNT) TO COBW3-CONTEXT.
000370*
```

```
000380*   I S A P I サブルーチン作業環境の設定及び
000390*   W E B パラメタの獲得
000400       CALL "COBW3_INIT" USING COBW3.
000410*
000420       MOVE SPACE TO PATHNAME.
000430*
000440       MOVE "Your Access Counter" TO      COBW3-COOKIE-NAME.
000450       CALL "COBW3_GET_COOKIE_XX" USING COBW3.
000460       IF COBW3-STATUS NOT = ZERO AND COBW3-STATUS NOT = 8820 THEN
000470           MOVE "ISAERROR.htm" TO HTMLFILENAME
000480           PERFORM 画面出力処理
000490       ELSE IF COBW3-SEARCH-FLAG-NON THEN
000500           MOVE 1 TO COBW3-COOKIE-VALUE
000510           PERFORM アクセスカウンタ登録処理
000520           MOVE "ISARPLY1.htm" TO HTMLFILENAME
000530           PERFORM 画面出力処理
000540       ELSE
000550           PERFORM 継続画面出力処理
000560       END-IF.
000570*
000580 終了位置.
000590*
000600*   I S A P I サブルーチン作業領域の解放
000610       CALL "COBW3_FREE" USING COBW3.
000620*
000630 ISAPISAMPLE1-END.
000640       MOVE 1 TO PROGRAM-STATUS.
000650       EXIT PROGRAM.
000660*
000670 継続画面出力処理 SECTION.
000680*   各種変換データの登録
000690*   アクセス回数の取得と登録
000700       COMPUTE アクセス回数 = FUNCTION NUMVAL (COBW3-COOKIE-VALUE).
000710       ADD 1 TO アクセス回数.
000720       MOVE アクセス回数 TO COBW3-COOKIE-VALUE.
000730       MOVE ZERO TO COBW3-COOKIE-VALUE-LENGTH.
000740*
000750*   アクセスカウンタの登録
000760       PERFORM アクセスカウンタ登録処理.
000770       MOVE NC"アクセス回数" TO COBW3-CNV-NAME-N.
000780       MOVE アクセス回数 TO COBW3-CNV-VALUE.
000790       PERFORM 変換データ登録.
000800*
000810*   リモートホスト名の取得と登録
000820       SET COBW3-REMOTE-HOST TO TRUE.
000830       CALL "COBW3_GET_REQUEST_INFO" USING COBW3.
000840       IF COBW3-STATUS NOT = ZERO THEN
000850           MOVE "ISAERROR.htm" TO HTMLFILENAME
000860           PERFORM 画面出力処理
000870           GO TO 終了位置
000880       END-IF.
000890       MOVE NC"ホスト名" TO COBW3-CNV-NAME-N.
```



```
000900*
000910* クロスサイトスクリプティング脆弱性対策
000920     MOVE COBW3-REQUEST-INFO TO CONVERT-VALUE.
000930     MOVE COBW3-REQUEST-INFO-LENGTH TO CONVERT-LENGTH.
000940     PERFORM サニタイズ処理.
000950     MOVE SANITIZED-DATA      TO COBW3-CNV-VALUE.
000960     PERFORM 変換データ登録.
000970*
000980* ブラウザ名の取得と登録
000990     MOVE "USER-AGENT" TO COBW3-HEADER-NAME.
001000     CALL "COBW3_RECEIVE_HEADER" USING COBW3.
001010     IF COBW3-STATUS NOT = ZERO THEN
001020         MOVE "ISAERROR.htm" TO HTMLFILENAME
001030         PERFORM 画面出力処理
001040         GO TO 終了位置
001050     END-IF.
001060     MOVE NC"ブラウザ名" TO COBW3-CNV-NAME-N.
001070*
001080* クロスサイトスクリプティング脆弱性対策
001090     MOVE COBW3-HEADER-VALUE TO CONVERT-VALUE.
001100     MOVE COBW3-HEADER-VALUE-LENGTH TO CONVERT-LENGTH.
001110     PERFORM サニタイズ処理.
001120     MOVE SANITIZED-DATA      TO COBW3-CNV-VALUE.
001130     PERFORM 変換データ登録.
001140*
001150* HTMLファイルの出力
001160     MOVE "ISARPLY2.htm" TO HTMLFILENAME.
001170     PERFORM 画面出力処理.
001180*
001190 継続画面出力処理終了.
001200     EXIT.
001210*
001220*
001230 アクセスカウンタ登録処理 SECTION.
001240* 有効期限を設定すると、ブラウザ終了後もアクセスカウンタの
001250* 内容が残せます。ただし、ブラウザが異なると意味がありません。
001260     CALL "COBW3_SET_COOKIE_XX" USING COBW3.
001270     IF COBW3-STATUS NOT = ZERO THEN
001280         MOVE "ISAERROR.htm" TO HTMLFILENAME
001290         PERFORM 画面出力処理
001300         GO TO 終了位置
001310     END-IF.
001320 アクセスカウンタ登録処理終了.
001330     EXIT.
001340*
001350 変換データ登録 SECTION.
001360     CALL "COBW3_SET_CNV_NX" USING COBW3.
001370     IF COBW3-STATUS NOT = ZERO THEN
001380         MOVE "ISAERROR.htm" TO HTMLFILENAME
001390         PERFORM 画面出力処理
001400         GO TO 終了位置
001410     END-IF.
```

```
001420 変換データ登録終了.
001430      EXIT.
001440*
001450 画面出力処理 SECTION.
001460*   アプリケーションが配置されている物理パスの取得と
001470*   HTML文書名の編集
001480      IF PATHNAME = SPACE THEN
001490          PERFORM 物理パス名取得
001500      END-IF.
001510      MOVE SPACE TO COBW3-HTML-FILENAME.
001520      MOVE PATHNAME(1:PATHSIZE) TO COBW3-HTML-FILENAME.
001530      COMPUTE COPYSTARTPOS = PATHSIZE + 1.
001540      MOVE "¥" TO COBW3-HTML-FILENAME(COPYSTARTPOS:1).
001550      COMPUTE COPYSTARTPOS = COPYSTARTPOS + 1.
001560      COMPUTE LEFTLENGTH = 256 - COPYSTARTPOS.
001570      MOVE HTMLFILENAME TO COBW3-HTML-FILENAME(COPYSTARTPOS:256).
001580*
001590*   HTML文書の出力
001600      CALL "COBW3_PUT_HTML" USING COBW3.
001610*
001620 画面出力処理終了.
001630      EXIT.
001640*
001650 物理パス名取得 SECTION.
001660      MOVE SPACE TO PATHNAME.
001670      SET COBW3-PHYSICALPATH TO TRUE.
001680      CALL "COBW3_GET_REQUEST_INFO" USING COBW3.
001690      IF COBW3-STATUS = ZERO THEN
001700          MOVE COBW3-REQUEST-INFO TO PATHNAME
001710          MOVE COBW3-REQUEST-INFO-LENGTH TO PATHSIZE
001720      END-IF.
001730*
001740 物理パス名取得終了.
001750      EXIT.
001760
001770 サニタイズ処理 SECTION.
001780      MOVE 0                      TO      WORK-CNTR1 .
001790      MOVE 1                      TO      WORK-CNTR2 .
001800      MOVE SPACE                 TO      SANITIZED-DATA .
001810      PERFORM WITH TEST BEFORE
001820          VARYING WORK-CNTR1 FROM 1 BY 1
001830          UNTIL WORK-CNTR1 > CONVERT-LENGTH
001840      EVALUATE CONVERT-VALUE(WORK-CNTR1:1)
001850          WHEN "<"
001860              MOVE "&lt;"          TO      SANITIZED-DATA(WORK-CNTR2:4)
001870              ADD 4                TO      WORK-CNTR2
001880          WHEN ">"
001890              MOVE "&gt;"          TO      SANITIZED-DATA(WORK-CNTR2:4)
001900              ADD 4                TO      WORK-CNTR2
001910          WHEN "&"
001920              MOVE "&amp;"        TO      SANITIZED-DATA(WORK-CNTR2:5)
001930              ADD 5                TO      WORK-CNTR2
```

```

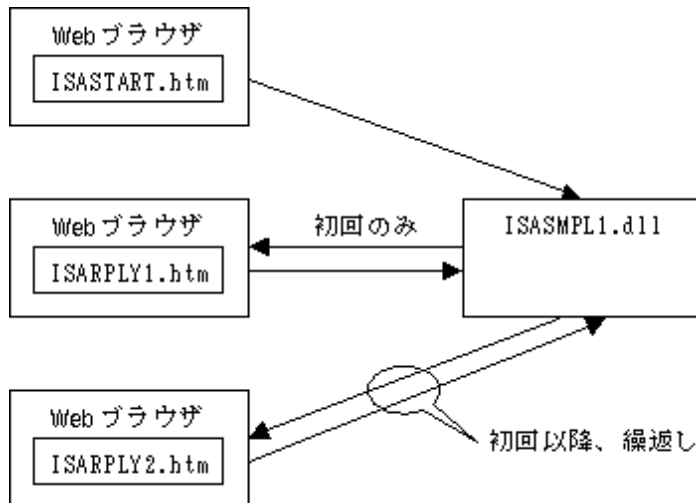
001940          WHEN OTHER
001950          MOVE CONVERT-VALUE (WORK-CNTR1:1)
001960                                TO      SANITIZED-DATA (WORK-CNTR2:1)
001970          ADD 1                                TO      WORK-CNTR2
001980          END-EVALUATE
001990          END-PERFORM.
002000          MOVE WORK-CNTR2                                TO      SANITIZED-LENGTH.
002010          サニタイズ処理終了.
002020          EXIT.

```

この例題で行っている処理は、次のとおりです。

- Cookieデータの取得
Webブラウザから送信されるCookieデータの取得を行います。
- Cookieデータの登録
- 結果出力用ページの出力
Cookieデータの内容に応じて、出力する結果出力用ページを切り分けます。また、必要に応じて変換データを登録して結果出力用ページの編集を行います。

画面と処理の遷移は次のとおりです。



例題における個々の機能の使い方と使用目的について、簡単に説明します。

Cookieデータの取得

この例題で使用するCookie名は” Your Access Counter” です。したがって、このCookieデータを取得するためには、COBW3-COOKIE-NAMEに上記Cookie名を編集し、” COBW3_GET_COOKIE_XX” を呼び出します(440行目～450行目)。

この例題では、Cookieデータの有無を初回 (ISASTART.htmからの初めての起動) の判定およびアクセス回数の保持に使用しています。初回アクセスの場合は、Cookieデータがないため、COBW3_GET_COOKIE_XXを呼び出してもCookieデータは見つからない点を利用して、初回処理を行っています(490行目～530行目)。一方、初回以降はCookieデータが必ず送信されるので、この点を利用します(540行目～550行目)。なお、初回処理でカウンタの値を1に設定し、初回以降で、1ずつ増加させます。

Cookieデータの登録

登録したいCookie名をCOBW3-COOKIE-NAMEに、内容をCOBW3-COOKIE-VALUEに設定し、COBW3_SET_COOKIE_XXを呼び出します。なお、この例題では、Cookieデータは” Your Access Counter” だけであるので最初に設定した値をそのまま使用します(440行目)。ここで登録されたCookieデータは、結果出力用ページの出力時にWebブラウザに送信されます。

リクエスト情報の取得

取得方法は非常に簡単で、取得したい情報の条件名を指定し、COBW3_GET_REQUEST_INFOを呼び出すだけです。情報の取得に成功すると、該当する情報がCOBW3-REQUEST-INFOに設定されます。

この例題では、2つのリクエスト情報を取得しています。ひとつは、仮想ディレクトリに対応する物理パスの情報で、アプリケーション(ISASMP11.d11)と同じパスに格納されている結果出力用ページのパス名を決定するために使用しています(1660行目～1720行目)。IIS配下のWebアプリケーションではカレントディレクトリが不定のため、この情報を元にパス名を決定するか絶対パス指定などを行う必要があります。もうひとつは、画面に表示する項目として、Webブラウザが実行されているホスト名を取得しています(820行目～880行目)。

ヘッダ情報の取得

この場合も取得方法は非常に簡単で、取得したいHTTPヘッダ名をCOBW3-HEADER-NAMEに設定し、COBW3_RECEIVE_HEADERを呼び出すだけです。HTTPヘッダ情報の取得に成功すると、その情報がCOBW3-HEADER-VALUEに設定されます。

この例題では、画面に表示する項目としてWebブラウザの情報の取得に使用しています。Webブラウザの情報は” User-Agent” ヘッダから取得できます(990行目～1050行目)。

例題 SAFサブルーチンを使ったプログラム

ここでは、本製品で提供するCOBOL SAFサブルーチンのサンプルプログラムについて説明します。

概要

COBOL SAFサブルーチンを使って、Cookieを使用したアプリケーション間のデータの引継ぎやサーバやブラウザ情報などを取得する方法を示します。

提供プログラム

- SAFMAIN. cob (COBOL ソースプログラム)
- SAFSTART. htm (呼出し用ページ)
- SAFRPLY1. htm (結果出力用ページ)
- SAFRPLY2. htm (結果出力用ページ)
- SAFERROR. htm (結果出力用ページ(エラー処理用))
- COBOL85. cbr (実行用の初期化ファイル)
- SAFSMPL1. txt (プログラム説明書)

使用しているSAFサブルーチン

- COBW3_INIT
- COBW3_SET_CNV_NX
- COBW3_PUT_HTML
- COBW3_RECEIVE_HEADER
- COBW3_GET_REQUEST_INFO
- COBW3_SET_COOKIE_XX
- COBW3_GET_COOKIE_XX
- COBW3_FREE

プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。以降の説明で、フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“SAFSMPL1.PRJ”を開きます。
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. 翻訳オプションLIBに、COBOL SAFサブルーチンの登録集ファイル(COBW3.cb1、COBW3SAF.cb1)が格納されているフォルダを指定します。翻訳オプションALPHALの設定で“英大文字と等価に扱う”を選択し、さらに“WORD - COBOLの語”を選択します。また、翻訳オプションTHREADの設定で、“MULTI - マルチスレッドとする”を選択します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトにオブジェクトC:\¥COBOL\F3BICBDM.objおよびインポートライブラリF3BINSRT.libが指定されていることを確認します。
6. プロジェクトマネージャの[プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、SAFSMPL1.DLLが作成されていることを確認してください。

[参照] “COBOL Webサブルーチン使用手引書”の“4.4.1 翻訳およびリンク”

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの[ツール]メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。

2. [ファイル] メニューの“開く”を選択し、SAFSMPL1.DLLが存在するフォルダに、実行用の初期化ファイル(COBOL85.CBR)を作成します。
 3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@MessOutFileに、出力メッセージの格納ファイル名を指定します。
 - 環境変数情報@WinCloseMsgに、OFFを指定します。
 - 環境変数情報@CBR_SAF_LOGFILEに、ログファイル名を指定します。
 - 環境変数情報@CBR_SAF_SEVERITYに、重要度を指定します。
 4. [適用] ボタンをクリックします。
 - 設定した内容が実行用の初期化ファイルに保存されます。
 5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。
- [参照] “COBOL Webサブルーチン使用手引書”の“4.4.3 SAFサブルーチンの環境変数設定”

また、例題が入ったフォルダを物理パスとした適当な仮想ディレクトリをNESに登録してください。また、Webアプリケーションを実行する上で必要な設定をNESに対して行ってください。NESの設定については、“COBOL Webサブルーチン使用手引書”の“4.4.2 NESの設定”を参照してください。

プログラムを実行する前に

呼出し用ページのFORMタグのACTION属性に記述された拡張子をNESに登録するか、または、NESに登録した拡張子に合わせて呼出し用ページを修正してください。

プログラムの実行

NESに設定したフォルダに例題の各ファイルをコピーします。

NESが起動されていることを確認し、Webブラウザから呼出し用ページ(SAFSTART.htm)を表示します。あとは画面の指示に従い[実行]ボタンを押してください。

【補足】 この例題の動作コード系をUnicodeにする場合は、翻訳時に翻訳オプション”RCS(UCS2)”を追加し、すべてのHTML文書をUnicodeに変更してください。



この例題はCookieをサポートしていないWebブラウザまたはWebブラウザがCookieを受け入れない設定になっていると正しく動作しません。

例題の解説

例題の解説をするために、この例題で使用している呼出し用ページ(SAFSTART.htm)、結果出力用ページ(SAFRPLY1.htm、SAFRPLY2.htm)およびCOBOLプログラム(SAFMAIN.cob)を以下に提示します。

呼出し用ページ(SAFSTART.htm)

```
<HTML>
<HEAD>
  <TITLE> 初期画面</TITLE>
</HEAD>

<BODY>
SAF サブルーチンを使用したサンプルです。<BR>
動作の確認を行いたい場合は、実行ボタンを押して下さい。<BR>
終了したい場合は、ブラウザを閉じて下さい。<BR>

<FORM METHOD="POST" ACTION="safsmpl1.cobap1">
  <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">
</FORM>
</BODY>
```

</HTML>

結果出力用ページ (SAFRPLY1. htm)

```
<HTML>
<HEAD>
  <TITLE> 初回画面</TITLE>
</HEAD>

<BODY>
<FONT COLOR=BLUE> はじめてご利用いただきありがとうございます。</FONT><BR>
継続してご利用される場合は、実行ボタンを押してください。<BR>
終了したい場合は、ブラウザを閉じてください。<BR>

<FORM METHOD="POST" ACTION="safsmp11.cobap1">
  <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">
</FORM>
</BODY>
</HTML>
```

結果出力用ページ (SAFRPLY2. htm)

```
<HTML>
<HEAD>
  <TITLE> 初回以降の画面</TITLE>
</HEAD>

<BODY>
現在の状況は次のとおりです。<BR>
<TABLE BORDER=2>
<TR>
  <TH>アクセス元ホスト名</TH>
  <TH>ブラウザ</TH>
  <TH>アクセス回数</TH>
</TR>
<TR>
  <TD>//COBOL//ホスト名//COBOL//</TD>
  <TD>//COBOL//ブラウザ名//COBOL//</TD>
  <TD>//COBOL//アクセス回数//COBOL//</TD>
</TR>
</TABLE>
<BR>

継続してご利用される場合は、実行ボタンを押してください。<BR>
終了したい場合は、ブラウザを閉じてください。<BR>
（注：ブラウザを閉じた場合、カウンタはリセットされます。また、
異なるブラウザや別のマシン上のブラウザでアクセスした場合、
カウンタ値は異なります）

<FORM METHOD="POST" ACTION="safsmp11.cobap1">
```

```

    <INPUT TYPE="SUBMIT" NAME="GO" VALUE="実行">
</FORM>
</BODY>
</HTML>

```

COBOL プログラム (SAFMAIN.cob)

```

000010*-----*
000020* ALL RIGHTS RESERVED, COPYRIGHT (C) FUJITSU LIMITED 2000-2003 *
000030* *
000040* ファイル名 : SAFMAIN.COB *
000050* 概要 : SAFサブルーチンの例題 *
000060*-----*
000070 IDENTIFICATION DIVISION.
000080 PROGRAM-ID. SAF-MAIN.
000090 ENVIRONMENT DIVISION.
000100 DATA DIVISION.
000110 WORKING-STORAGE SECTION.
000120 COPY COBW3.
000130*
000140 01 HTMLFILENAME PIC X(64).
000150 01 PATHNAME PIC X(256).
000160 01 PATHSIZE PIC 9(05).
000170 01 COPYSTARTPOS PIC 9(05).
000180 01 LEFTLENGTH PIC 9(05).
000190 01 アクセス回数 PIC 9(05).
000200 01 WORK-CNTR1 PIC 9(4) COMP-5.
000210 01 WORK-CNTR2 PIC 9(4) COMP-5.
000220 01 CONVERT-VALUE PIC X(1024).
000230 01 CONVERT-LENGTH PIC S9(4) COMP-5.
000240 01 SANITIZED-DATA PIC X(1024).
000250 01 SANITIZED-LENGTH PIC S9(4) COMP-5.
000260*
000270 LINKAGE SECTION.
000280 01 SAFCTX POINTER.
000290*
000300 PROCEDURE DIVISION USING SAFCTX.
000310*
000320 SAFSAMPLE1-START.
000330*
000340* S A F サブルーチンパラメタ域の初期化
000350 MOVE LOW-VALUE TO COBW3.
000360 SET COBW3-CONTEXT TO SAFCTX.
000370*
000380* S A F サブルーチン作業環境の設定及び
000390* W E B パラメタの獲得
000400 CALL "COBW3_INIT" USING COBW3.
000410*
000420 MOVE SPACE TO PATHNAME.
000430*
000440 MOVE "YOUR ACCESS COUNTER" TO COBW3-COOKIE-NAME.

```



```
000450      CALL "COBW3_GET_COOKIE_XX" USING COBW3.
000460      IF COBW3-STATUS NOT = ZERO AND COBW3-STATUS NOT = 8820 THEN
000470          MOVE "SAFERROR.htm" TO HTMLFILENAME
000480          PERFORM 画面出力処理
000490      ELSE IF COBW3-SEARCH-FLAG-NON THEN
000500          MOVE 1 TO COBW3-COOKIE-VALUE
000510          PERFORM アクセスカウンタ登録処理
000520          MOVE "SAFRPLY1.htm" TO HTMLFILENAME
000530          PERFORM 画面出力処理
000540      ELSE
000550          PERFORM 継続画面出力処理
000560      END-IF.
000570*
000580 終了位置.
000590*
000600*   S A Fサブルーチン作業領域の解放
000610      CALL "COBW3_FREE" USING COBW3.
000620*
000630 SAFSAMPLE1-END.
000640*
000650      EXIT PROGRAM.
000660*
000670 継続画面出力処理 SECTION.
000680*   各種変換データの登録
000690*   アクセス回数の取得と登録
000700      COMPUTE アクセス回数 = FUNCTION NUMVAL (COBW3-COOKIE-VALUE).
000710      ADD 1 TO アクセス回数.
000720      MOVE アクセス回数 TO COBW3-COOKIE-VALUE.
000730      MOVE ZERO TO COBW3-COOKIE-VALUE-LENGTH.
000740*
000750*   アクセスカウンタの登録
000760      PERFORM アクセスカウンタ登録処理.
000770      MOVE NC"アクセス回数" TO COBW3-CNV-NAME-N.
000780      MOVE アクセス回数 TO COBW3-CNV-VALUE.
000790      PERFORM 変換データ登録.
000800*
000810*   リモートホスト名の取得と登録
000820      SET COBW3-REMOTE-HOST TO TRUE.
000830      CALL "COBW3_GET_REQUEST_INFO" USING COBW3.
000840      IF COBW3-STATUS NOT = ZERO THEN
000850          MOVE "SAFERROR.htm" TO HTMLFILENAME
000860          PERFORM 画面出力処理
000870          GO TO 終了位置
000880      END-IF.
000890      MOVE NC"ホスト名" TO COBW3-CNV-NAME-N.
000900*
000910*   クロスサイトスクリプティング脆弱性対策
000920      MOVE COBW3-REQUEST-INFO TO CONVERT-VALUE.
000930      MOVE COBW3-REQUEST-INFO-LENGTH TO CONVERT-LENGTH.
000940      PERFORM サニタイズ処理.
000950      MOVE SANITIZED-DATA      TO COBW3-CNV-VALUE.
000960      PERFORM 変換データ登録.
```

```
000970*
000980* ブラウザ名の取得と登録
000990     MOVE "USER-AGENT" TO COBW3-HEADER-NAME.
001000     CALL "COBW3_RECEIVE_HEADER" USING COBW3.
001010     IF COBW3-STATUS NOT = ZERO THEN
001020         MOVE "SAFERROR.htm" TO HTMLFILENAME
001030         PERFORM 画面出力処理
001040         GO TO 終了位置
001050     END-IF.
001060     MOVE NC"ブラウザ名" TO COBW3-CNV-NAME-N.
001070*
001080* クロスサイトスクリプティング脆弱性対策
001090     MOVE COBW3-HEADER-VALUE TO CONVERT-VALUE.
001100     MOVE COBW3-HEADER-VALUE-LENGTH TO CONVERT-LENGTH.
001110     PERFORM サニタイズ処理.
001120     MOVE SANITIZED-DATA      TO COBW3-CNV-VALUE.
001130     PERFORM 変換データ登録.
001140*
001150* HTMLファイルの出力
001160     MOVE "SAFRPLY2.htm" TO HTMLFILENAME.
001170     PERFORM 画面出力処理.
001180*
001190 継続画面出力処理終了.
001200     EXIT.
001210*
001220*
001230 アクセスカウンタ登録処理 SECTION.
001240* 有効期限を設定すると、ブラウザ終了後もアクセスカウンタの
001250* 内容が残せます。ただし、ブラウザが異なると意味がありません。
001260     CALL "COBW3_SET_COOKIE_XX" USING COBW3.
001270     IF COBW3-STATUS NOT = ZERO THEN
001280         MOVE "SAFERROR.htm" TO HTMLFILENAME
001290         PERFORM 画面出力処理
001300         GO TO 終了位置
001310     END-IF.
001320 アクセスカウンタ登録処理終了.
001330     EXIT.
001340*
001350 変換データ登録 SECTION.
001360     CALL "COBW3_SET_CNV_NX" USING COBW3.
001370     IF COBW3-STATUS NOT = ZERO THEN
001380         MOVE "SAFERROR.htm" TO HTMLFILENAME
001390         PERFORM 画面出力処理
001400         GO TO 終了位置
001410     END-IF.
001420 変換データ登録終了.
001430     EXIT.
001440*
001450 画面出力処理 SECTION.
001460* アプリケーションが配置されている物理パスの取得と
001470* HTML文書名の編集
001480     IF PATHNAME = SPACE THEN
```

```

001490      PERFORM 物理パス名取得
001500      END-IF.
001510      MOVE SPACE TO COBW3-HTML-FILENAME.
001520      MOVE PATHNAME(1:PATHSIZE) TO COBW3-HTML-FILENAME.
001530      COMPUTE COPYSTARTPOS = PATHSIZE + 1.
001540      MOVE "¥" TO COBW3-HTML-FILENAME(COPYSTARTPOS:1).
001550      COMPUTE COPYSTARTPOS = COPYSTARTPOS + 1.
001560      COMPUTE LEFTLENGTH = 256 - COPYSTARTPOS.
001570      MOVE HTMLFILENAME TO COBW3-HTML-FILENAME(COPYSTARTPOS:256).
001580*
001590*   HTML文書の出力
001600      CALL "COBW3_PUT_HTML" USING COBW3.
001610*
001620   画面出力処理終了.
001630      EXIT.
001640*
001650   物理パス名取得 SECTION.
001660      MOVE SPACE TO PATHNAME.
001670      SET COBW3-PHYSICALPATH TO TRUE.
001680      CALL "COBW3_GET_REQUEST_INFO" USING COBW3.
001690      IF COBW3-STATUS = ZERO THEN
001700          MOVE COBW3-REQUEST-INFO TO PATHNAME
001710          MOVE COBW3-REQUEST-INFO-LENGTH TO PATHSIZE
001720      END-IF.
001730*
001740   物理パス名取得終了.
001750      EXIT.
001760
001770   サニタイズ処理 SECTION.
001780      MOVE 0                      TO      WORK-CNTR1 .
001790      MOVE 1                      TO      WORK-CNTR2 .
001800      MOVE SPACE                 TO      SANITIZED-DATA .
001810      PERFORM WITH TEST BEFORE
001820          VARYING WORK-CNTR1 FROM 1 BY 1
001830          UNTIL WORK-CNTR1 > CONVERT-LENGTH
001840      EVALUATE CONVERT-VALUE(WORK-CNTR1:1)
001850          WHEN "<"
001860              MOVE "&lt;"          TO      SANITIZED-DATA(WORK-CNTR2:4)
001870              ADD 4                TO      WORK-CNTR2
001880          WHEN ">"
001890              MOVE "&gt;"          TO      SANITIZED-DATA(WORK-CNTR2:4)
001900              ADD 4                TO      WORK-CNTR2
001910          WHEN "&"
001920              MOVE "&amp;"       TO      SANITIZED-DATA(WORK-CNTR2:5)
001930              ADD 5                TO      WORK-CNTR2
001940          WHEN OTHER
001950              MOVE CONVERT-VALUE(WORK-CNTR1:1)
001960                  TO      SANITIZED-DATA(WORK-CNTR2:1)
001970              ADD 1                TO      WORK-CNTR2
001980      END-EVALUATE
001990      END-PERFORM.
002000      MOVE WORK-CNTR2              TO      SANITIZED-LENGTH.

```

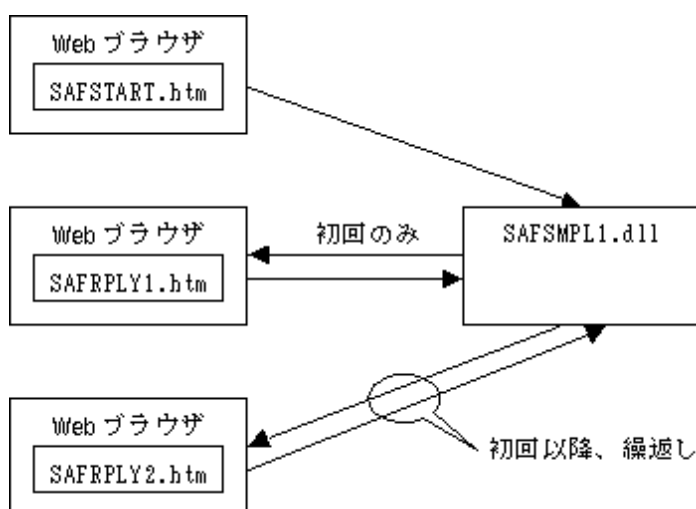
002010 サニタイズ処理終了.

002020 EXIT.

この例題で行っている処理は、次のとおりです。

- Cookieデータの取得
Webブラウザから送信されるCookieデータの取得を行います。
- Cookieデータの登録
- 結果出力用ページの出力
Cookieデータの内容に応じて、出力する結果出力用ページを切り分けます。また、必要に応じて変換データを登録して結果出力用ページの編集を行います。

画面と処理の遷移は次のとおりです。



例題における個々の機能の使い方と使用目的について、簡単に説明します。

Cookieデータの取得

この例題で使用するCookie名は” Your Access Counter” です。したがって、このCookieデータを取得するためには、COBW3-COOKIE-NAMEに上記Cookie名を編集し、” COBW3_GET_COOKIE_XX” を呼び出します(440行目～450行目)。

この例題では、Cookieデータの有無を初回(SAFSTART.htmからの初めて起動)の判定およびアクセス回数の保持に使用しています。初回アクセスの場合は、Cookieデータがないため、COBW3_GET_COOKIE_XXを呼び出してもCookieデータは見つからない点を利用して、初回処理を行っています(490行目～540行目)。一方、初回以降はCookieデータが必ず送信されるので、この点を利用します(550行目～560行目)。なお、初回処理でカウンタの値を1に設定し、初回以降で、1ずつ増加させます。

Cookieデータの登録

登録したいCookie名をCOBW3-COOKIE-NAMEに、内容をCOBW3-COOKIE-VALUEに設定し、COBW3_SET_COOKIE_XXを呼び出します。なお、この例題では、Cookieデータは” Your Access Counter” だけであるので最初に設定した値をそのまま使用します(440行目)。ここで登録されたCookieデータは、HTMLファイル出力時にWebブラウザに送信されます。

リクエスト情報の取得

取得方法は非常に簡単で、取得したい情報の条件名を指定し、COBW3_GET_REQUEST_INFOを呼び出すだけです。情報の取得に成功すると、該当する情報がCOBW3-REQUEST-INFOに設定されます。

この例題では、2つのリクエスト情報を取得しています。ひとつは、仮想パスに対応する物理パスの情報で、アプリケーション(SAFSMPL1.dll)と同じパスに格納されている結果出力用ページのパス名を決定するために使用しています(1660行目～1720行目)。NES配下のアプリケーションではカレントディレクトリが不定のため、この情報を元にパス名を決定するか絶対パスを指定する必要があります。もうひとつは、画面に表示する項目として、Webブラウザが実行されているホ

スト名を取得しています(820行目～880行目)。

ヘッダ情報の取得

この場合も取得方法は非常に簡単で、取得したいHTTPヘッダ名をCOBW3-HEADER-NAMEに設定し、COBW3_RECEIVE_HEADERを呼び出すだけです。HTTPヘッダ情報の取得に成功すると、その情報がCOBW3-HEADER-VALUEに設定されます。

この例題では、画面に表示する項目としてWebブラウザの情報の取得に使用しています。Webブラウザの情報は” User-Agent” ヘッダから取得できます(990行目～1050行目)。

例題 セッション管理機能を使ったプログラム

ここでは、本製品で提供するCOBOL Webサブルーチンのセッション管理機能を使ったプログラムについて説明します。

例題では、認証処理を行い、セッション管理機能を使用してデータを引き継ぐアプリケーションの例を示します。

COBOL Web サブルーチンを利用したアプリケーションでセッション管理機能を実現するには、COBOL ISAPIサブルーチンまたはCOBOL SAFサブルーチンを使用します。提供されているサンプルプログラムではISAPIアプリケーションを想定したものとなっています。SAF アプリケーションとして使用する場合には、プログラムとHTMLファイルの一部を変更する必要があります。各変更方法についてはプログラムやHTMLファイルにコメントとして記述されているので参照して下さい。

セッション管理機能についての説明や各APIの使い方の詳細は、“COBOL Webサブルーチン使用手引書”を参照してください。

概要

Web ブラウザから預入／払戻金額を入力し、その残高をファイルに書き出すオンライン業務を行います。

サンプルプログラムは、次の3つの部分からなります。

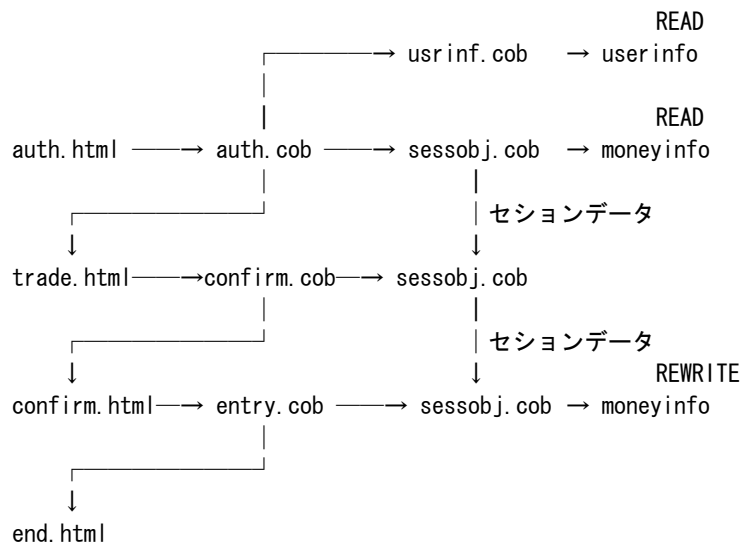
- 認証処理
認証処理を行い、セッションを開始します。
ここではWeb ブラウザから入力されたユーザIDとファイルから読み込んだ取引前の残高をセッションデータとして登録します。
- 確認表作成処理
確認表を作成します。
ここではWeb ブラウザから入力されたデータと取引後の残高をセッションデータとして登録します。
- 更新処理
更新処理を行います。
セッションデータとして引き継いだユーザIDと取引後の残高をもとにファイルの更新処理を行い、セッションを終了します。

提供プログラム

- AUTH. COB (COBOL ソースプログラム)
- CONFIRM. COB (COBOL ソースプログラム)
- ENTRY. COB (COBOL ソースプログラム)
- ISAINIT. COB (COBOL ソースプログラム)
- ISATERM. COB (COBOL ソースプログラム)
- SESSOBJ. COB (COBOL ソースプログラム)
- USRINF. COB (COBOL ソースプログラム)
- GETDATA. CBL (登録集ファイル)
- SESSDATA. CBL (登録集ファイル)
- USER-INFO. CBL (登録集ファイル)
- AUTH. HTML (HTML ファイル)
- AUTHFAIL. HTML (HTML ファイル)
- CONFIRM. HTML (HTML ファイル)
- END. HTML (HTML ファイル)
- ILLIGALACCESS. HTML (HTML ファイル)
- INPUTERROR. HTML (HTML ファイル)
- SYSTEMERROR. HTML (HTML ファイル)
- TRADE. HTML (HTML ファイル)

- UNDERCONSTRUCTION. HTML (HTML ファイル)
- USEDERROR. HTML (HTML ファイル)
- COBOL85. CBR (実行用の初期化ファイル)
- AUTH. DEF (モジュール定義ファイル)
- CONFIRM. DEF (モジュール定義ファイル)
- ENTRY. DEF (モジュール定義ファイル)
- WSESSION. PRJ (プロジェクトファイル)
- WSESSION. CBI (オプションファイル)
- MONEYINFO (データファイル)
- USERINFO (データファイル)
- WSESSION. TXT (プログラム説明書)

プログラムの呼出し関係



使用しているSAFサブルーチン

- COBW3_INIT
- COBW3_GET_VALUE_XX
- COBW3_GET_VALUE_NX
- COBW3_START_SESSION
- COBW3_END_SESSION
- COBW3_SET_SESSION_DATA
- COBW3_GET_SESSION_INFO
- COBW3_GET_SESSION_DATA
- COBW3_SET_CNV_XX
- COBW3_SET_CNV_XN
- COBW3_GET_REQUEST_INFO
- COBW3_PUT_HTML
- COBW3_FREE

プログラムの翻訳・リンク・実行

ビルド・リビルド

翻訳およびリンクは、プロジェクトマネージャのビルド機能を使用して行います。

なお、プロジェクトファイルは、NetCOBOLのインストール先フォルダをC:\¥COBOLとして説明しています。以降の説明で、フォルダ名がC:\¥COBOLとなっているところは、NetCOBOLをインストールしたフォルダに変更してください。また、現状ISAPIアプリケーションを想定したものとなっているため、ライブラリファイルにF3BISAPI.LIBを指定しています。SAFアプリケーションとして

使用する場合、この部分をF3BINSRT.LIBに変更して下さい。

1. プロジェクトマネージャを起動します。
2. プロジェクトファイル“WSESSION.PRJ”を開きます。
3. プロジェクトファイルを選択し、[プロジェクト] - [オプション] メニューから“翻訳オプション”を選択します。
→ [翻訳オプション] ダイアログが表示されます。
4. 翻訳オプションLIBに、COBOL ISAPI サブルーチンの登録集ファイル (COBW3.cb1、ISAPIINF.cb1、ISAPICTX.cb1、ISAPIFLG.cb1) が格納されているフォルダを指定します。
翻訳オプションALPHALの設定で“英大文字と等価に扱う”を選択し、さらに“WORD - COBOLの語”を選択します。また、翻訳オプションTHREADの設定で、“MULTI - マルチスレッドとする”を選択します。確認後、[OK] ボタンをクリックします。
→ プロジェクトマネージャウィンドウに戻ります。
5. プロジェクトにインポートライブラリF3BISAPI.LIBが指定されていることを確認します。
6. プロジェクトマネージャの [プロジェクト] メニューから“ビルド”を選択します。
→ ビルド終了後、プロジェクトに登録した各DLL (ダイナミックリンクライブラリ) が作成されていることを確認してください。

[参照] “COBOL Webサブルーチン使用手引書”の“3.4.1 翻訳およびリンク”

サーバプログラムの実行環境の設定

1. プロジェクトマネージャの [ツール] メニューから“実行環境設定ツール”を選択します。
→ 実行環境設定ツールが表示されます。
2. [ファイル] メニューの“開く”を選択し、例題で提供された実行用の初期化ファイル (COBOL85.CBR)を開きます。
3. 共通タブを選択し、以下を設定します。
 - 環境変数情報@MessOutFileに、出力メッセージの格納ファイル名を指定します。
 - 環境変数情報@WinCloseMsgに、OFFを指定します。
 - 環境変数情報@CBR_ISAPI_LOGFILEに、ログファイル名を指定します。
 - 環境変数情報@CBR_ISAPI_SEVERITYに、重要度を指定します。
4. [適用] ボタンをクリックします。
→ 設定した内容が実行用の初期化ファイルに保存されます。
5. [ファイル] メニューの“終了”を選択し、実行環境設定ツールを終了します。

[参照] “COBOL Webサブルーチン使用手引書”の“3.4.3 ISAPIサブルーチンの環境変数設定”

また、例題が入ったフォルダを物理パスとした適切な仮想ディレクトリをIISに登録してください。IISの設定については、“COBOL Webサブルーチン使用手引書”の“3.4.2 IISの設定”を参照してください。

プログラムを実行する前に

呼出し用ページ (ISASTART.htm) のFORMタグのACTION属性に指定されているWebアプリケーションのパスを、実際に実行するWebサーバの仮想パスに変更してください。

プログラムの実行

IISに設定したフォルダに例題の各ファイルをコピーします。

例題では、ドメイン名を“user”、仮想ディレクトリ名を“wsession”としてサーバに登録しています。

1. URL に以下の情報を設定して実行キーを押します。

アドレス	http://user/wsession/auth.html
------	--------------------------------

2. 会員認証画面が表示されるのでユーザIDとパスワードを入力して、[OK] ボタンをクリックします。ここで、入力できるユーザIDはUSER0001からUSER0030までです。パスワードは

ユーザIDと同じです。

会員認証

貴方のIDとパスワードを入力して下さい。会員登録がまだの方は、[会員登録](#)を行って下さい。

ユーザID:	<input type="text" value="USER0001"/>
パスワード:	<input type="password" value="*****"/>

3. [OK] ボタンをクリックすると取引画面が表示されます。ここで、金額を入力し預入または払戻を選択し、[OK] ボタンをクリックします。

取引

現在の残高は次のとおりです。金額を入力して、「預入」又は「払戻」を選択して下さい。

残高:	¥0
金額:	<input type="text" value="5000"/>

1. ☒ 預入

2. ☐ 払戻

4. [OK] ボタンをクリックすると、確認画面が表示されます。内容を確認し、ファイルを更新するなら確認、更新しないなら取消を選択し、[OK] ボタンをクリックします。

確認

内容は次のとおりです。確認後、「確認」又は「取消」を選択して下さい。

預入金額:	¥5,000
取引後残高:	¥5,000

1. ☒ 確認

2. ☐ 取消

5. 終了画面が表示されます。

終了

ご利用ありがとうございました。

ご利用明細

ユーザID	USER0001
取引区分	預入
取引額	¥5,000
残高	¥5,000

[会員認証に戻る](#)

索引

記号

*>..... 19
*COM-ARRAYクラス..... 128, 142
/EXPORT..... 49, 53
@CBR_CIINF..... 149
@CBR_SCR_KEYDEFFILE..... 16
@MGPRM..... 22

A

ACCEPT文..... 2, 15, 23, 26, 29, 128, 152

C

CALL文..... 18, 23, 96, 106, 147, 152
CLOSE文..... 6, 10, 15, 26, 29,
39, 43, 52, 96, 106, 152
COBOL ISAPIサブルーチン..... 106
COBOLサーバプログラムの使用 (ASPクライアント)
..... 134
COBOLサーバプログラムの使用 (COBOLクライアン
ト)..... 128
COBOLによるCOMサーバプログラムの作成..... 122
COBOLプログラム間の呼出し..... 18
CollectionSize-Getメソッド..... 61
Collectクラス..... 61
COMMIT文..... 43, 123
COMPUTE文..... 23, 52, 152
COMクライアント機能..... 115, 118
COMクライアント機能..... 128
COMサーバ機能..... 142
COM連携..... 71, 115, 118, 128, 134, 141
COM連携..... 122
CONNECT文..... 39, 43, 123, 143
COPY文..... 6, 10, 23

D

DELETE文..... 43
Dictクラス..... 62
DISCONNECT文..... 39, 43, 123, 143
DISPLAY文..... 2, 10, 15, 18, 23,
26, 115, 118, 128, 147, 152
DIVIDE文..... 23

E

Element-Getメソッド..... 62, 64
Element-Insertメソッド..... 64
ElementNo-Getメソッド..... 65
Element-PutAtメソッド..... 62, 64
Element-PutLastメソッド..... 65
EVALUATE文..... 128, 152
Excel連携..... 71
Excelを操作するプログラム..... 115, 118
EXIT文..... 2, 6, 10, 15, 18, 23, 26,
29, 52, 96, 106, 152

F

FETCH文..... 39, 43
FirstElement-Getメソッド..... 61
FirstKey-Getメソッド..... 63
FORMAT句付き印刷ファイル..... 34
FORMLIB..... 11

G

GO TO文..... 96, 106, 152
GO TO文..... 6, 10, 15, 18, 23, 26, 29

I

IF文..... 2, 10, 15, 23, 26, 29, 52, 96,
106, 115, 118, 123, 128, 143, 147, 152
INITIALIZE文..... 123, 143
INSERT文..... 123, 143
INVOKE文..... 57, 66, 79, 87, 90, 115,
118, 123, 128, 143

J

JMPCINT2..... 48, 52
JMPCINT3..... 48, 52

L

LastElement-Getメソッド..... 65
LastKey-Getメソッド..... 63
LIB..... 11, 27, 37
Listクラス..... 63

M

MAIN..... 7
MeFt..... 10
MOVE文..... 6, 10, 15, 18, 23, 52, 96,
106, 123, 143, 147, 152
MTSによるトランザクション管理をするプログラ
ム..... 141

N

NextElement-Getメソッド..... 61

O

ODBC情報ファイル設定ツール..... 40, 44
OPEN文..... 6, 10, 15, 18, 26, 29, 39,
43, 52, 96, 106, 152

P

PERFORM文..... 2, 10, 23, 52, 96, 106, 115,
118, 123, 128, 143, 152
PreviousElement-Getメソッド..... 61

R

READ文..... 6, 10, 18, 26, 52, 96, 106, 152
Remove-Allメソッド..... 63, 65
Remove-Atメソッド..... 63, 65
REWRITE文..... 52
REWRITE文..... 106
ROLLBACK文..... 39, 43, 123, 143

S

SELECT文..... 43, 123, 143

SET文.....57, 66, 79, 87, 90, 96,
106, 115, 118, 123, 128, 143
SRF.....21
START文.....106
STOCK表.....39, 43
STRING文.....26

U

Unicode.....152
UPDATE文.....43, 123, 143

V

Visual Basic(R)からの呼出し.....48
Visual Basic(R)を使った簡易ATM端末処理機能
.....51

W

WRITE文....6, 10, 15, 18, 26, 29, 52, 106, 152

い

印刷ファイル.....18, 29, 31

う

ウィンドウ情報ファイル.....12
埋込みSQL文.....39, 43, 123, 143
埋込み例外宣言.....39, 43

え

永続オブジェクト.....86, 89

お

オブジェクトコンテキストオブジェクト.....143
オブジェクト指向プログラミング機能
.....57, 66, 79, 87, 90
オブジェクト指向プログラム.....57, 68, 78
オブジェクト定義.....58, 66, 79, 87, 90
オブジェクトの永続化.....86, 89
オブジェクトの生成.....57, 66, 71, 79
オブジェクトプロパティ.....66, 79, 87, 90

か

カーソル宣言.....39
カーソル宣言.....43
外部データ.....106
外部ファイル.....106
外部ファイルイベントログ.....106
カプセル化.....57, 66, 71, 78
画面入出力.....10
簡易アプリ間通信機能.....147
環境変数の操作.....26

き

行順ファイル.....6, 18

く

クラス階層.....60
クラス定義.....58, 66, 79, 87, 90
クラスライブラリ.....60

け

継承.....66, 71, 78

こ

コマンド行引数の受取り方.....23

コマンド行引数の取出し.....23
コレクションクラス.....60
コンソールウィンドウ.....2, 29

さ

索引ファイル.....6, 15, 18, 87, 106

し

実行時パラメタの受渡し.....18
実行履歴.....9
自由形式.....18
小入出力機能.....2, 10, 18, 29

す

スクリーン機能.....128
スクリーン操作機能.....15, 79

た

対話型デバッグ.....2
多重継承.....71
多態.....78
他のプログラムの起動.....159

ち

注記行.....19
帳票印刷.....10

て

データベース機能.....39
データベース機能を使ったプログラム.....43
データロックサブルーチン.....106

と

登録集の取込み.....10, 18

な

内部プログラム.....23

ひ

表指定ホスト変数.....43
表示ファイル機能.....10
標準入出力を使ったデータ処理.....2

ふ

ファクトリ定義.....87, 90
複数行指定ホスト変数.....43
プリンタ情報ファイル.....12
プログラム間連絡機能.....18, 156
プロジェクト管理機能.....10, 15, 18,
23, 26, 29, 32, 34, 39, 43, 48, 52, 57, 66,
71, 79, 87, 90, 118, 152, 156, 159

ま

マルチスレッドプログラミング.....94, 103

め

メソッド定義.....58, 66, 79, 87, 90
メソッドの行内呼出し.....66, 79, 87, 90
メソッド呼出し.....57, 66, 71, 79
メッセージボックス.....10, 18
メッセージボックスの出力.....156

り

リポジトリ段落.....57, 66, 79, 87, 90
リモートデータベースアクセス.....142

リモートデータベースアクセス.....39, 43
リモートデータベースアクセス(ODBC)機能.....90
 れ
例外処理.....71

ろ

論理宛先定義ファイル.....148
論理宛先定義ファイル作成ユーティリティ....148

