Micro FocusR Net ExpressR

入門書 - その他のトピック

第 1 版
2003 年 5 月

このマニュアルでは、Net Express の機能に関するその他のチュートリアルを説明します。

第 1 章 : はじめに

この『*入門書 - その他のトピック*』では、Net Express に含まれる他のツールのチュートリアルを紹介します。

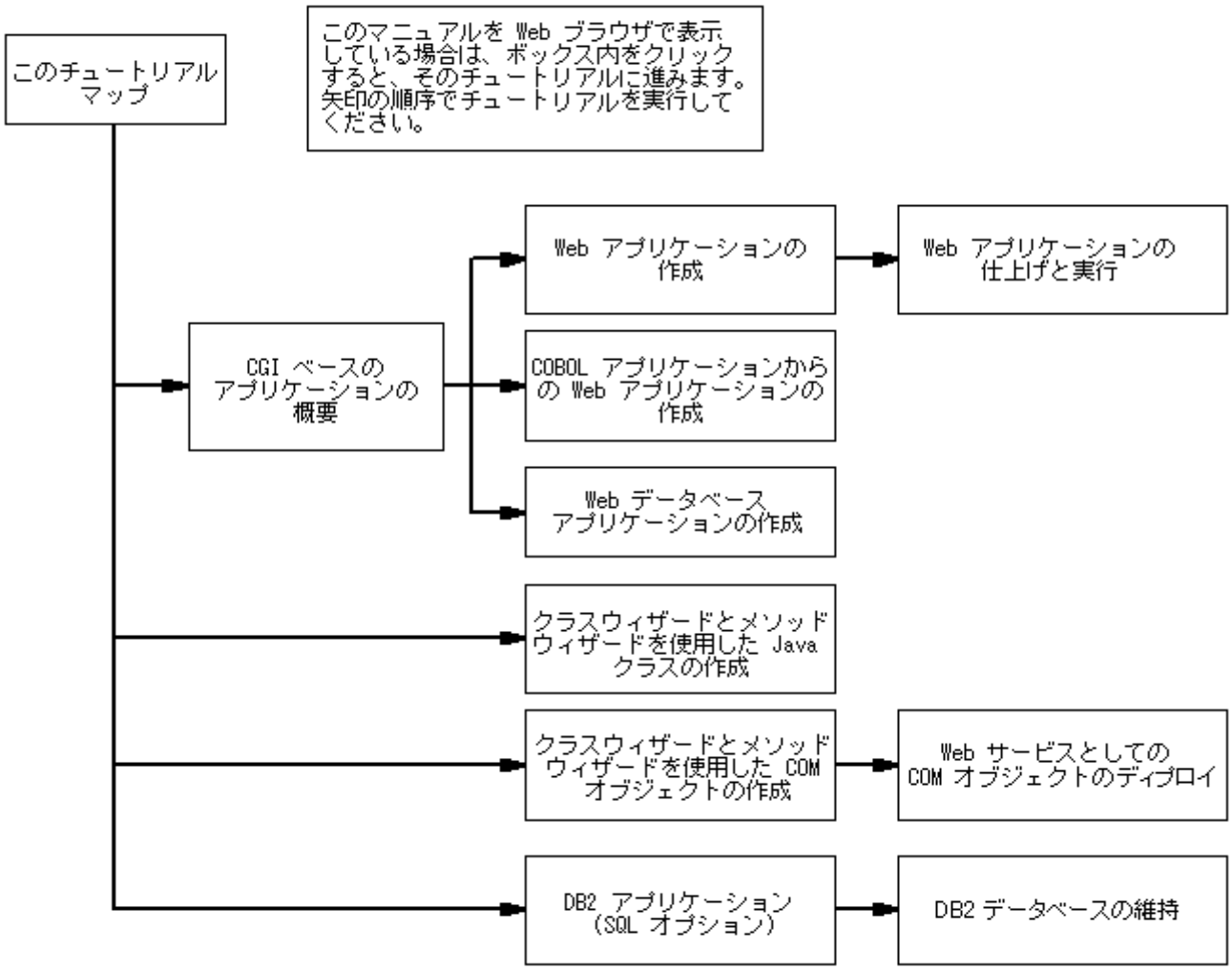
1.1 入門書のチュートリアル

『*入門書*』のチュートリアルを実行してからこのマニュアルのチュートリアルに進んでください。必ず、『*チュートリアルの概要*』の章を読み、『*Net Express の使用方法*』を行ってください。

1.2 チュートリアルマップ

次の図に、このマニュアルの各チュートリアルを示します。各チュートリアルは、1 つの章にまとめられています。 次図のボックスをクリックすると各章に進むことができます。次の図は、チュートリアルの相関関係を表しています。複数のチュートリアルが矢印で結合されている場合は、矢印の順序でチュートリアルを実行してください。

『入門書 - その他のトピック』のチュートリアルは、最も一般的な作業を実行しながら、入門書で説明する各機能を簡単に紹介するために作成されています。 各機能の詳細は、Windows の [**スタート**] メニューの [ヘルプ] から表示できます。



第 2 章 : CGI ベースのアプリケーションの概要

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

2.1 CGI の概要

『***入門書***』マニュアルのチュートリアルでは、Web アプリケーションなどの分散アプリケーションに使用できる構成要素ベースの技術について説明しています。『***入門書 - その他のトピック***』マニュアルのこの章では、Web アプリケーションの旧技術について説明します。

サーバ側プログラムは、通常、Web ブラウザからの要求に応じて Web サーバからロードされるプログラムです。このプログラムは、Web サーバマシンで実行されます。通常は、ブラウザでロードされた Web フォームに指定されたプログラムが要求され、そのフォームにユーザが入力したデータが処理されます。

プログラムを Web ブラウザで起動するためのプロトコルは、いくつかあります。以前から確立されているプロトコルの 1 つに Common Gateway Interface (CGI) プロトコルがあります。このプロトコルでは、プログラムを個別のプロセスとして実行します。コマンド行から呼び出せる限り、プログラムをどの言語 (例 シェルスクリプト、C 言語、COBOL 言語) で作成してもかまいません。CGI に準拠すると、どの Web サーバとも互換できます。

CGI の他に Internet Server Application Programming Interface (ISAPI) や Netscape Server Application Programming Interface (NSAPI) も使用できます。Net Express などの開発環境で CGI プログラムを ISAPI (または NSAPI) に変更する場合は、別のオプションを設定してリビルドするのみでかまいません。そのため、選択するプロトコルは重要ではありません。開発中には CGI プログラムを使用し、運用環境では ISAPI または NSAPI に切り替えることもできます。「CGI」と「CGI プログラム」という用語は、通常、「サーバ側プログラム」という用語と同義で使用されます。このマニュアルで説明する一連のチュートリアルの残りの章でもこの用語を使用します。

CGI プログラムを作成するには、通常、非常に複雑な作業が必要ですが、Net Express では、この作業が自動化されています。Net Express では、CGI が COBOL 言語で作成されます。

2.2 Net Express の CGI

Net Express では、次の 3 通りの方法で CGI Web アプリケーションを作成できます。

- 最初から作成 - ユーザのブラウザに表示する HTML フォーム、および、データを受け付けて処理した後に結果を表示するサーバ側プログラムを作成する必要があります。

- 既存の COBOL プログラムから作成 - HTML フォーム、および、データを受け付けて既存の処理プログラムに渡した後に結果を表示するサーバ側プログラムを作成する必要があります。
- 既存のデータベースから作成 - HTML フォーム、および、データを受け付けてデータベース内の情報を検索した後に結果を表示するサーバ側プログラムを作成する必要があります。

上記のどの場合もインターネットアプリケーションウィザードを使用します。最初の方法では、まず HTML ページウィザードと Form Designer を使用して、フォームを独自に設計します。その後のインターネットアプリケーションウィザードの使用方法は、上記のどの場合でもほとんど同じです。ウィザードに既存の要素 (フォーム、プログラム、またはデータベース) を指定すると、アプリケーションの残りの部分が自動的に生成されます。

この方法を使用した場合は、Net Express に組み込まれた標準テンプレートに基づいてフォームとフォーム処理コードが作成されます。フォームとフォーム処理コードを個別の条件に合わせてカスタマイズする場合は、後でこのフォームとフォーム処理コードを編集できます。

この一連のセッションでは、インターネットアプリケーションウィザードの 3 通りの使用方法について説明します。これらのセッションでアプリケーションをテストする場合には、どの Web ブラウザも使用できます。

2.3 Solo

アプリケーションのテストには、どの Web サーバソフトウェアも使用できます。ここでは、Net Express に組み込まれた Web サーバソフトウェア「Solo」を使用します。Solo は、Net Express を使用して開発された CGI ベースのアプリケーションをテストするために特別に設計されたソフトウェアです。この Web サーバは、実際に運用するためのソフトウェアではありません。Solo は、構成または設定する必要がありません (自動的に設定された Solo の構成を確認するには、付録『*Web サーバの構成*』を参照してください)。

2.4 サーバ側プログラム

Net Express で作成したサーバ側プログラムでは、Micro Focus の 2 つの拡張 COBOL 構文 (拡張 ACCEPT 文と EXEC HTML 文) を使用してフォームの入出力を処理します。EXEC HTML 文は、埋め込み HTML とも呼ばれ、文内に記述された HTML を出力します。フォームには [送信] ボタンを指定し、エンドユーザがこのボタンをクリックしたときに Web サーバ上のプログラムが実行されるように設定する必要があります。ACCEPT 文を指定すると、フォームからデータを入力できます。

Micro Focus COBOL には、拡張 DISPLAY 文も用意されていますが、EXEC HTML 文の方がより強力です。

2.4.1 構造

サーバ側プログラムは、通常、次の要素などで構成されています。

1. 入力フォームからデータを入力するための ACCEPT 文
2. 入力フォームに関連付けられたデータ項目から作業領域にデータを転記するためのコード
3. 挿入したビジネスロジック
4. フォームに関連付けられたデータ項目に作業領域から結果を転記するためのコード
5. 出力フォームを作成するための EXEC HTML 文

次の点に注意してください。

- メインループはありません。エンドユーザが入力フォームを送信するたびに、プログラムが再実行されます。
- ACCEPT 文では、フォームを読み取りできません。また、ACCEPT 文では、フォームが読み取られるまで待機することもありません。エンドユーザが [送信] ボタンをクリックすると、データが入力されたフォームが Web ブラウザからプログラムに送信されます。送信されたデータがある場合は、ACCEPT 文によってデータが受け付けられます。それ以外の場合は、ACCEPT 文による処理はありません。
- EXEC HTML 文では、フォームが出力されません。フォームが作成されるのみです。プログラムが終了すると、Web サーバからブラウザにフォームが送信されます。

2.4.2 対称と非対称

アプリケーションでは、入力と出力に同じフォームを使用することも、別のフォームを使用することもできます。

Net Express では、入出力に同一のフォームを使用するアプリケーションを対称アプリケーションと呼びます。このアプリケーションは、通常、エンドユーザが実行形式ファイルを実行すると起動します。通常は、実行形式ファイルにリンクした Web ページを作成します。エンドユーザがこの Web ページをクリックすると、プログラムが 1 回実行され、ブラウザにフォームが表示されます。エンドユーザがフォームに入力して [送信] ボタンをクリックすると、プログラムが再実行されてデータが処理され、結果が表示されます。フォームは、エンドユーザが再度使用できる状態に戻ります。

入力と出力に別のフォームを使用するアプリケーションを非対称アプリケーションと呼びます。このアプリケーションは、通常、エンドユーザが入力フォームをロードすると、起動します。通常は、フォームにリンクした Web ページを作成します。エンドユーザがこの Web ページをクリックすると、フォームがロードされます。エンドユーザがフォームに入力して [送信] ボタンをクリックすると、プログラムが実行されてデータが処理され、結果が反映された出力フォームが表示されます。次のクエリーを入力するには、エンドユーザが入力フォームを再度ロードします。(入力フォームのすべての入力フィールドが出力フォームに含まれている場合は、入力フォームをリロードしないで出力フォームを入力に使用することもできます。プログラムは、ブラウザからデータを受信するのみなので、入力フォームと出力フォームの相違を認識しません。)

1 つのアプリケーションで複数の入力フォームを使用する場合は、フォームごとにサーバ側プログラムを作成する必要があります。

2.5 次の作業

次の章『*Web アプリケーションの作成*』に進んでください。

第 3 章 : Web アプリケーションの作成

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

3.1 概要

この章と次の章では、Web アプリケーションを作成します。

この章では、HTML ページウィザードと Form Designer を使用してユーザインターフェイスを作成し、インターネットアプリケーションウィザードを使用してデータ処理プログラムを作成します。 IDE、Web ブラウザ、および Net Express に付属の Web サーバソフトウェア「Solo」を使用して、アプリケーションを実行し、インターフェイスの表示状態を確認します。

次の章『*Web アプリケーションの仕上げと実行*』では、このプログラムにビジネスロジックを追加して、完全なアプリケーションを実行します。

HTML ページウィザードを使用して、Web ブラウザで表示するフォームを作成します。 これらがアプリケーションのユーザインターフェイスになります。ウィザードから Form Designer に移り、ユーザインターフェイスのレイアウトの設計や編集を実行できます。

また、インターネットアプリケーションウィザードを使用して、フォームを処理するサーバ側 COBOL プログラムを作成します。

HTML フォームでは、コントロールの順序を指定できますが、正確なレイアウトはそのフォームを表示するブラウザによって決定されます。 ただし、Form Designer には、正確なレイアウトを指定するための機能があります。

サーバ側プログラムは、Web アプリケーションの中心的機能を果たします。 サーバ側プログラムは、入力フォームからのデータを受け付け、データを出力フォームに出力します。 入力と出力の両方に 1 つのフォームを使用する特殊な場合もあります。 このセッションでは、このようなアプリケーションを作成します。

3.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウなどのウィンドウが開いている場合は、閉じます。

3.3.1 プロジェクトとフォームの作成

『***入門書***』マニュアルの『*Net Express の使用方法*』の章で説明した方法でプロジェクトを作成することもできますが、すぐにフォームを作成する場合は、Net Express でプロジェクトを作成できます。 フォームの作成方法は、次のとおりです。

1. [**ファイル > 新規作成**] をクリックし、「新規作成」ダイアログボックスで「**HTML ページ**」を選択し、[**OK**] をクリックします。
2. プロジェクトを作成するかどうかを確認するメッセージに対して [**OK**] をクリックします。
3. 「**HTML プロジェクト**」が選択されていることを確認します。プロジェクト名として「**Goform**」を入力し、プロジェクトを格納するフォルダとして「**Net Express\Base\Demo\Goform**」を入力して、[**作成**] をクリックします。
4. ディレクトリを作成するかどうかを確認するメッセージに対して [**はい**] をクリックします。(このセッションを実行したことがある場合は、既存のプロジェクトを上書きするかどうかを確認するメッセージが表示されます。[**はい**] をクリックします。)

   このプロジェクトの「プロジェクト」ウィンドウが表示されます。HTML ページウィザードの最初のページも表示されます。

5. 「**Positional Form.htm**」(設定によっては、拡張子 **.htm** が表示されないことがあります) をクリックし、[**次へ**] をクリックします。

   通常は、HTML フォームの設計時に、配置するオブジェクトの順序のみを指定します。正確な位置は、フォームを表示するブラウザによって決定されます。ただし、Net Express では、フォームの一部を *位置フォーム* として指定できます。位置フォーム内のオブジェクトは、配置した場所どおりにブラウザに表示されます。ここで「**Positional Form.htm**」をクリックすると、フォームの大部分に 1 つの位置フォームが配置されます。

   Web 用語では、ユーザがデータを入力するフィールドを含む Web ページをフォームと呼びます。フォームと、フォーム内の位置フォームは、区別してください。

6. デフォルトの HTML ファイル名を **mypage** に変更し、「**クロスプラットフォーム**」を選択したままで、[**次へ**] をクリックします。

   この操作によって、HTML フォームの名前が **mypage.htm** に設定されます。HTML ファイルの一般的な拡張子は、**.htm** です。

   「**クロスプラットフォーム**」を指定すると、Net Express によって位置フォーム内のオブジェクトが確実に希望の位置に配置されます。クロスプラットフォームのフォームでは、HTML テーブルを使用して配置を確定します。動的 HTML では、スタイル属性が使用されます。HTML テーブルは、多くのブラウザでサポートされます。動的 HTML では、より正確な位置を指定できますが、インターネットエクスプローラ 4.0 以降でのみサポートされます。作成したフォームはエンドユーザ側で別のブラウザを使用して表示される場合があるので、注意してください。

7. **mypage.htm** または **mypage.mff** がすでに存在することを表すメッセージが表示された場合は、[**はい**] をクリックして上書きします。

   このセッションを以前に実行している場合は、これらのメッセージが表示されることがあります。それ以外の場合は、次のダイアログボックスが表示されます。

次のダイアログボックスには、選択内容の概要が表示されます。

8. [**終了**] をクリックします。

   HTML ウィザードによってフォームが生成され、終了します。生成されたファイルの名前がプロジェクトに追加され、プロジェクトの依存関係が検索されます。

   IDE に「フォーム設計」ウィンドウが開きます。このウィンドウには、背景となる点線のグリッドと、3 つの関連するウィンドウが表示されます。3 つのタブをもつツールバーも追加で表示されます。このツールバーをオブジェクトツールバーと呼びます。ツールバー上には、フォームに追加できるオブジェクトが表示されます。メニューバーには、[**ページ**] メニューと [**配置**] メニューが追加で表示されます。

   Net Express に付属の Web サーバソフトウェア「Solo」をツールバーから起動することもできます (Windows のタスクバートレイに  が表示されます)。ここでは、このアイコンを無視してもかまいません。Form Designer では、必ずバックグラウンドで Solo が実行されます。

3.3.2 この段階での IDE の状態

IDE は、図 3-1 のように表示されます。

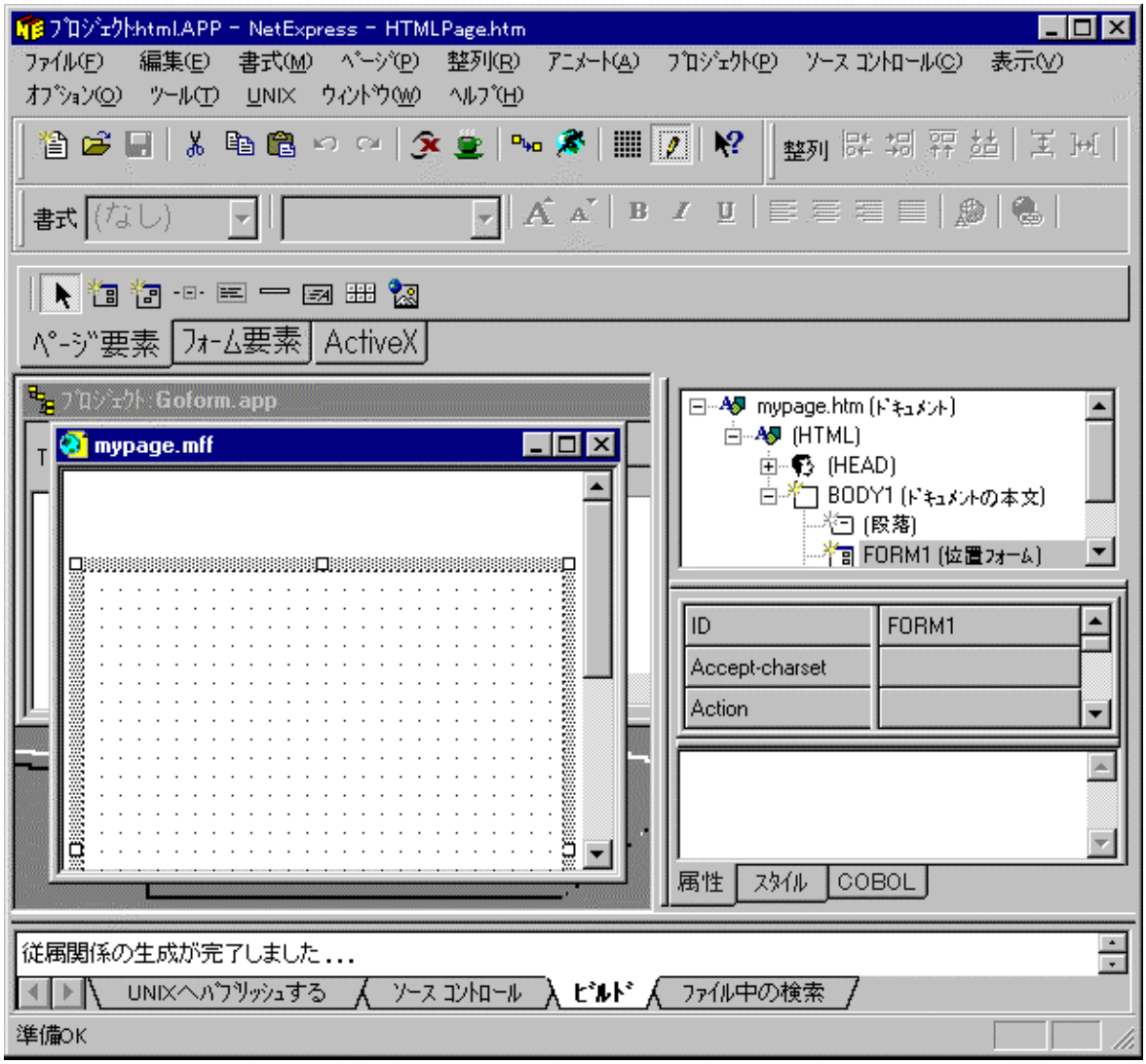


図 3-1 : Form Designer のウィンドウが開いた IDE

新しい 4 つのウィンドウは、次のとおりです。

- 「フォーム設計」ウィンドウ

  背景に点線のグリッドが描画されたウィンドウです。ここでフォームを設計します。作成されたフォーム **mypage.htm** がロードされ、設計の準備が整います。フォームの大部分の領域が、ハイライトされた四角形で囲まれていることに注意してください。この範囲が位置フォームです。

- 「コントロールツリー」ウィンドウ

右上のウィンドウには、フォームのコントロールがすべて表示されます。コントロール名を変更するには、そのコントロールを右クリックし、表示されるコンテキストメニューで [**名前を変更**] を選択します。

- 「属性」ウィンドウ

  「コントロールツリー」ウィンドウの下部のウィンドウには、「フォーム設計」ウィンドウで現在選択されているオブジェクトの属性が表示されます。属性名は左下に、対応する値は右下に、表示されます。 値フィールド内をクリックすると、属性を変更できます。

- 「ヘルプ」ウィンドウ

  「属性」ウィンドウのヘルプが表示される右下のウィンドウです。「属性」ウィンドウ内のフィールドをクリックすると、ヘルプウィンドウ内にそのフィールドのヘルプが表示されます。

「フォーム設計」ウィンドウをいったん端にドラッグして、プロジェクトの変化を確認することもできます。IDE 内でウィンドウを移動したり、サイズを変更したりして、すべてのウィンドウを明確に表示することもできます。

3.3.3 フォームへのコントロールの追加

フォームの設計方法は、次のとおりです。

1. オブジェクトツールバーの [**フォーム要素**] タブをクリックし、他のタブの前に表示します。
2. 位置フォーム (「フォーム設計」ウィンドウ内の点線の四角形) 内をクリックし、作成したオブジェクトが位置フォーム内に配置されるように、確実に選択します。
3. オブジェクトツールバーの [テキスト入力] ボタン  をクリックして、「フォーム設計」ウィンドウ内をクリックします。(ツールバー上のボタンにマウスポインタをしばらく合わせておくと、ボタン名のツール情報が表示されます。) このフィールドに表示される最初のテキスト「**テキスト入力**」を削除します。

   クリックした場所にエントリフィールドが作成されます。エンドユーザは、エントリフィールドにデータを入力したり、表示したりします。 エントリフィールドを配置後に移動するには、マウスポインタをエントリフィールドの境界に合わせてマウスボタンを押したままドラッグします。 エントリフィールドをウィンドウの右側に配置します。

   「属性」ウィンドウの「名前」属性に注意してください (「名前」属性が表示されていない場合は、リスト内をクリックし、矢印キーを使用して、リスト内をスクロールしてください)。「名前」属性は、**input1** です。input1 は、後で作成する COBOL プログラム内のこのフィールドに対するデータ名です。

4. 点線の四角形内の別の場所をクリックして再度選択し、その後で作成したオブジェクトが位置フォーム内に配置されるように選択します。

5. 同様に、[テキスト] ボタン  をクリックし、エントリフィールドの左側をクリックして、入力フィールドのラベルを作成します。 エントリフィールドに重ならないように注意してください。重なる場合は、エントリフィールドの境界をドラッグして、そのフィールドを移動するか、サイズを変更します。
6. ラベルフィールドのデフォルトのキャプション「**新規テキスト**」を「**名前**」に変更します。(新しい行を開始するので、**Enter** キーは押さないでください。)
7. マウスポインタをキャプションの左上に合わせ、エントリフィールドの右下にドラッグして、キャプションとエントリフィールドの両方を選択します。 必要な場合は、両方が表示されるように「フォーム」設計ウィンドウのサイズを大きくしてください。

   四角形が一次的に表示され、両方のフィールドがハイライトされます。

8. [**配列 > 配置**] をクリックし、[**トップ**] をクリックします。

   2 つのフィールドが整列します。

9. 点線の四角形を再度選択し、2 つ目のエントリフィールドとラベルを上記のように追加して、ラベルのキャプションを「Greeting」に設定します。

   この 2 つ目のエントリフィールドに対する「属性」ウィンドウ内の「名前」属性は、**input2** です。

10. 2 つのキャプションを互いに整列させることも、2 つのエントリフィールドを互いに整列させることもできます。どちらの場合も、2 つのフィールドを選択し (必ず両方のフィールドを四角形で囲んでください。確実に囲まないと、フィールドが選択されません)、[**整列 > 配置**] をクリックし、[**左**] をクリックします。
11. 点線の四角形を再度選択し、ツールバーの [送信] ボタン  をクリックし、フォーム内をクリックしてボタンを配置します。 このボタンは、ウィンドウの下部に配置してください。

    図 3-2 のようなフォームが表示されます。

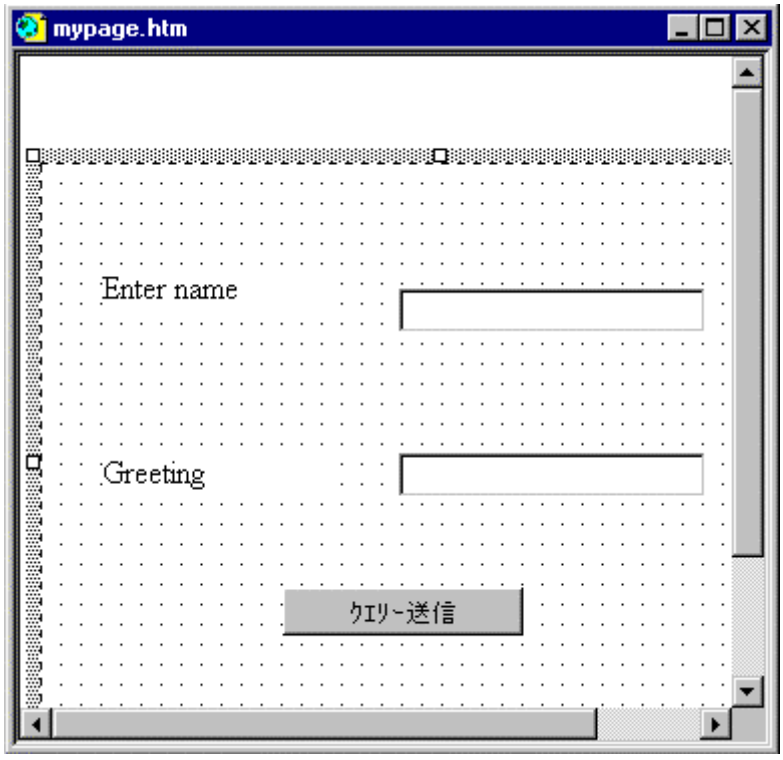


図 3-2 :「フォーム設計」ウィンドウで作成されたフォーム

12. [**ファイル > 上書き保存**] をクリックして、フォームを保存します。
13.「フォーム設計」ウィンドウの ×ボタンをクリックします。

「フォーム設計」ウィンドウ、および、関連するウィンドウがすべて閉じます。ウィンドウをすべて閉じる必要はありません。続いて操作を実行する場合は、開いていてもかまいません。ただし、ウィンドウを閉じておくと、画面を見やすくできます。

3.3.4 サーバ側プログラムの作成

インターネットアプリケーションウィザードを使用して、サーバ側プログラムを作成します。

1. [**ファイル > 新規作成**] をクリックし、「新規作成」ダイアログボックスで「**インターネットアプリケーション**」を選択して、[**OK**] をクリックします。

   インターネットアプリケーションウィザードの最初のページが表示されます。このページでは、『*CGI ベースのアプリケーションの概要*』の章で説明するとおり、3 通りのアプリケーション作成方法から 1 つを選択します。

2.「**Net Express が作成した HTML からサーバ側プログラムを生成**」を選択します。

   ダイアログボックスの下部では、「**HTML フォーム**」が自動的に選択されています。

3. [**次へ**] をクリックします。

表示されているページでは、サーバ側プログラムの名前を指定し、使用する入力フォームと 1 つ以上の出力フォームを指定できます。

4. 「ファイル名」フィールドに **myprog.cbl** を入力します。

   このファイル名は、サーバ側プログラム名になります。

5. 「入力ファイル/フォーム」フィールドと「出力ファイル」フィールドで **mypage** が選択 (ハイライト) されていることを確認します。

   以前にプロジェクトフォルダを使用していない限り、存在するフォームは **mypage** のみです。「入力ファイル/フォーム」フィールドは、自動的に **mypage** に設定されます。

   サーバ側プロジェクトでは、1 つの入力フォームと 1 つ以上の出力フォームを使用します。このようなプログラム (入力と出力に 1 つのフォームのみを使用するプログラム) を対称サーバ側プログラムと呼びます。

6. [**次へ**] をクリックします。

   このダイアログボックスには、選択内容の概要が表示されます。

7. [**終了**] をクリックします。

   インターネットアプリケーションウィザードによってプログラムが生成され、終了します。生成されたファイルの名前がプロジェクトに追加され、プロジェクトの依存関係が検索されます。インターネットアプリケーションウィザードが終了します。

3.3.5 作成したファイル

ここまでは、ユーザインターフェイスを作成しました。ウィザードと Form Designer によって作成され、プロジェクトに追加されたファイルは、次のとおりです (これらのファイルをすべて表示するには、プロジェクトウィンドウのツリービューで「+」記号をクリックして、ツリービューを展開する必要がある場合があります)。

- **mypage.htm** - HTML フォームです。
- **myprog.cbl** - サーバ側プログラムです。

次のファイルは、右側のペインのみに表示されます。

- **mypage.mff** - サーバ側プログラムの生成に必要な追加情報が含まれています。

プロジェクトのビルド時に作成された次のオブジェクトファイルもプロジェクトに追加されています。

- **myprog.obj** - コンパイル済みのサーバ側プログラムです。

- **myprog.exe** - 実行形式ファイルです。

また、複数のコピーファイル (拡張子 **.cp***) も作成されています。これらのファイルは、**.cbl** ファイル内の注釈に記述されています。

上記のファイル名は、小文字で表記されていますが、プロジェクト内では、大文字の場合があります。ファイル名については、大文字小文字が区別されません。

フォームを更新する必要がある場合は、プロジェクトウィンドウの **mypage.htm** をダブルクリックし、Form Designer を開きます。更新したフォームを保存すると、コピーファイルが再生成されるので、コピーファイルを直接編集しないでください。ウィザードで生成した **.cbl** プログラムは、編集できます。

3.3.6 フォームのテスト

Web ブラウザに表示されたフォームの確認方法は、次のとおりです。

1. 「プロジェクト」ウィンドウの **mypage.htm** をダブルクリックします。

   フォームが編集用に再度開きます。ここでは、フォームに変更を加えません。[**ページ**] メニューと [**配置**] メニューをメニューバーに再度表示するために、フォームを編集用に開いています。

2. [**ページ > プレビュー**] をクリックします。

   この操作によって、Web ブラウザが開いていない場合は、Web ブラウザが起動され、フォームがロードされます。Solo も起動していない場合は、起動します。

   詳細については、次のセッション『*Web アプリケーションの仕上げと実行*』を参照してください。ここでは、フォームがブラウザでどのように表示されるかのみを確認します。

   2 つのエントリフィールドに「:f-Input1」と「:f-Input2」が格納されています。これらの値は無視してかまいません。サーバ側プログラムによってフォームが表示されたときに同時に表示されるデータのプレースホルダです。

3. ブラウザを終了し、「フォーム設計」ウィンドウの ボタンをクリックします。

   この段階では、まだビジネスロジックが追加されていませんが、サーバ側プログラムの実行を確認できます。

4. ツールバーの をクリックします。

   Net Express で **.exe** ファイルがビルドされます。

5. [**アニメート > 実行**] をクリックします。

6. 「アニメーションの起動」ダイアログボックスの [**OK**] をクリックします。

   サーバ側プログラムが実行され、ブラウザにフォームが表示されます。 この段階では、プルースホルダが表示されなくなっていることに注意してください。 IDE は、自動的に最小化されます。

7. 最初のフィールドにユーザ名を入力し、[**クエリーの送信**] をクリックします。

   プログラムが実行され、フォームが再表示されます。ただし、2 つ目のフィールドにはビジネスロジックを追加していないため、何も表示されません。 この操作は、次のセッション『*Web アプリケーションの仕上げと実行*』で実行します。

8. ブラウザを終了し、最小化された IDE を復元して、[**アニメート > アニメート停止**] をクリックします。

3.4 次に進む前に

作成した Web アプリケーションを正しく動作させるには、次の章『*Web アプリケーションの仕上げと実行*』に進んでください。

作業を中断する場合は、Net Express やプロジェクトを閉じてかまいません。その場合は、次の章に進んだときに、Net Express やプロジェクトを再度開いてください。

Windows のタスクバートレイの [Solo] アイコン  を右クリックし、ポップアップメニューの [**終了**] をクリックして、Solo を閉じることもできます。

第 4 章 : Web アプリケーションの仕上げと実行

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

4.1 概要

前のセッションでは、HTML ウィザードと Form Designer を使用してフォームを設計し、インターネットアプリケーションウィザードを使用して COBOL プログラムを作成しました。このセッションでは、作成した COBOL プログラムを編集して、ビジネスロジックを追加します。その後で、Web ブラウザと Web サーバソフトウェア「Solo」を使用し、完成したアプリケーションを実行してデバッグします。アプリケーションと Web サーバソフトウェアは、どちらもローカルマシンで実行します。

インターネットアプリケーションウィザードで **.cbl** ファイル内に COBOL 原始プログラムを作成します。この原始プログラムには、フォームを入出力するためのコードが含まれています。その後で、この **.cbl** ファイルを編集し、フォームから入力されたデータを処理して出力データを作成するためのビジネスロジックを追加します。このフォーム処理プログラムとビジネスロジックを完全に切り離すために、CALL 文のみを追加して他のプログラムを呼び出すこともできます。

4.2 準備

Net Express を閉じている場合は、再度開きます。**Net Express\Base\Demo\Goform** ディレクトリ内の **Goform** プロジェクトを開きます。デフォルトでは、「開く」ダイアログボックスに **.app**、**.cbl** および **.cpy** ファイルが表示されるので、**Goform** ファイルも表示されます。

使用中のコンピュータで Solo 以外の Web サーバソフトウェアを実行している場合は、それらのソフトウェアを終了します。これらのソフトウェアを終了しない場合は、Solo の実行時に障害が発生することがあります。

Web ブラウザを先に開いておく必要はありません。

4.3.1 COBOL プログラムの編集

ビジネスロジックを COBOL プログラムに追加する場合は、どのテキストエディタでも使用できますが、ここでは、IDE を使用します。

1. 「プロジェクト」ウィンドウの **myprog.cbl** をダブルクリックします。

テキストウィンドウが開き、プログラムのソースが表示されます。必要な場合は、ソースを見やすいようにウィンドウのサイズを変更してください。[**表示 > COPY ファイルをすべて非表示**] の次にあるボタンのようなアイコンが押されていることを確認します (押されていない場合は、クリックします)。

手続き部と注釈を確認するには、数分かかります。

2. [**表示 > COPY ファイルをすべて非表示**] をクリックします。

   [COPY ファイルをすべて非表示] が解除され、コピーファイルが展開されます。これらのコピーファイルは、**.cbl** ファイルとともにインターネットアプリケーションウィザードで生成されたファイルです。

   ([COPY ファイルをすべて非表示] の解除は、[ファイル] メニューの [**コピーファイル**] を使用して展開されていた COPY 文、つまり、プログラムを以前にコンパイルしたときにソースに含まれていた COPY 文のみに影響します。前のセッションの『*フォームのテスト*』でプロジェクトをビルドしたときに、プログラムがコンパイルされます。

   しばらくの間、展開したコピーファイルを確認します。

   Convert-Input 節では、Input-Conversion 節が実行され、フォームに関連付けられたデータ項目からプログラム独自の作業領域にデータが転記されます。同様に Mypage-Cvt 節では、フォームに関連付けられたデータ項目にデータが転記されます。ソース内の注釈は、各コピーファイルの内容を表しています。

   このプログラムでは、入力と出力に対して同じフォームを使用することに注意してください。2 つのエントリフィールドは、F-INPUT1 と F-INPUT2 です。入力時には 2 つのエントリフィールドに入力されたデータが INPUT1 と INPUT2 に転記され、出力時には INPUT1 と INPUT2 のデータが 2 つのエントリフィールドに転記されます。

3. Process-Business-Logic節では、カーソルを EXIT 文の上にある空白行に移動し、**Enter** キーを押して空白行を作成します。カーソルを 12 カラム目 (EXIT の「E」の上) に移動し、次の行を挿入します。

   ```
   string "Hello " Input1 delimited by size into Input2
   ```

   実際のアプリケーションでは、通常、ここで副プログラムを呼び出し、フォームからのデータを処理して、表示データを計算または検索します。

4. テキストウィンドウを閉じます。ファイルを保存するかどうかを確認するメッセージに対して [**はい**] をクリックします。

4.3.2 アプリケーションのビルド

アプリケーションのビルド方法は、次のとおりです。

1. [**プロジェクト > すべてをリビルド**] をクリックします。

   Net Express によってプログラムが保存およびコンパイルされ、実行形式ファイルがビルドされます。「出力」ウィンドウに「リビルドの完了」というメッセージが表示された後で、次に進みます。

4.3.3 アプリケーションの実行

アプリケーションの実行方法は、次のとおりです。

1. [**アニメート > 実行**] をクリックします。

   このアプリケーションではフォームを 1 つしか使用しないので、「アニメーションの起動」ダイアログボックスには実行開始場所として実行形式ファイルのみが表示されます。

2. 「アニメーションの起動」ダイアログボックスの [**OK**] をクリックします。

   この操作によって Solo と Web ブラウザが起動します。エンドユーザが実行形式ファイルへのリンクをクリックした場合と同様にプログラムを実行します。 プログラムによってフォームが表示されます。 IDE は、自動的に最小化されます。

   Solo の実行に問題がある場合は、最小化された IDE を復元し、[**ヘルプ > ヘルプトピック**] をクリックして、ヘルプの索引で「Solo トラブルシューティング」を検索してください。

3. Web ブラウザの「**名前**」フィールドを選択し、ユーザ名を入力して [**クエリーの送信**] をクリックします。

   プログラムが実行されます。「**Greeting**」フィールドの「Hello」の後に入力したユーザ名が追加され、Web ブラウザにフォームが再表示されます。

4. 「**名前**」フィールドをクリックして、ユーザ名を別の名前に変更して、[**クエリーの送信**] をクリックします。

   プログラムが再度実行され、「Hello」の後に新しい名前が表示されます。

5. 最小化された IDE を復元し、[**アニメート > アニメート停止**] をクリックします。

4.3.4 アプリケーションのデバッグ

アプリケーションのデバッグ方法は、次のとおりです。

1. [**アニメート > アニメーションの起動**] をクリックします。
2. 「アニメーションの起動」ダイアログボックスの [**OK**] をクリックします。

Web ブラウザが起動し、IDE に **myprog.cbl** のソースが表示され、デバッグの準備が整います。 通常は、デバッグを実際に開始する前に、プログラムを一度に実行してフォームを表示します。

3. [**アニメート > 実行**] をクリックします。

   IDE が最小化され、アニメートしないでプログラムが実行されて、Web ブラウザにフォームが表示されます。 フォームは初期状態で表示されます (前の操作で表示されたフォームが置換されますが、出力データは格納されたままになります)。

   [**実行**] は、IDE 内でアプリケーションを実行するための機能です。 デバッグ中に使用して、アニメートしないでブレークポイントまで実行することもできます。 サーバ側プログラムのデバッグ時には、この機能によって最初にブレークポイントとしてプログラムの再入が処理されます。

4. Web ブラウザの「**名前**」フィールドを選択し、ユーザ名を入力して [**クエリーの送信**] をクリックします。

   最小化された IDE が復元し、**myprog.cbl** のソースが再度デバッグできる状態に戻ります。

5. ツールバーの  を使用して **myprog.cbl** の STOP RUN 文の実行直前までをステップ実行します。

   このプログラムは、あまり長くないので、ステップ実行にかかる時間は 1 分未満です。 埋め込み HTML (EXEC HTML 文) の実行には、数秒かかります。

6. STOP RUN 文をステップ実行します。

   IDE が最小化され、完成した Web フォーム (「**Greeting**」は更新済み) が表示されます。

7. 最小化された IDE を復元し、[**アニメート > アニメート停止**] をクリックします。

4.4 次に進む前に

プロジェクトを閉じます。

タスクバートレイの [Solo] アイコン  を右クリックし、ポップアップメニューの [**終了**] をクリックして、Solo を閉じます。

一連のチュートリアルは、ここまでです。 別のチュートリアルに進むには、第 1 章『*はじめに*』の『*チュートリアルマップ*』を参照してください。

第 5 章 : COBOL アプリケーションからの Web アプリケーションの作成

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

5.1 概要

この章では、既存の COBOL アプリケーションを Web アプリケーションに変換するには、インターネットアプリケーションウィザードを使用します。

インターネットアプリケーションウィザードを使用して、フォーム、および、既存の COBOL アプリケーションの連絡節からフォームを処理する COBOL プログラムを作成します。フォーム処理プログラムと元のプログラムをともにサーバ側プログラムとして使用します。

既存の COBOL プログラムは、旧来のプログラムでも、このアプリケーションを新しく作成する第 1 段階として作成したプログラムでもかまいません。

既存のプログラムには副プログラムが必要です。この副プログラムの連絡節には、ユーザインターフェイスのデータを定義します。通常、使用する既存のプログラムは、ファイル処理プログラムまたはデータベース処理プログラムです。既存のアプリケーションでは、このプログラムを呼び出してユーザのクエリーを渡した後に結果を返す別のプログラムで画面を処理します。一方、新しい Web ベースのアプリケーションでは、作成したフォームとプログラムが元の画面処理プログラムと同じ役割を果たします。

インターネットアプリケーションウィザードは、Windows GUI アプリケーションから Web アプリケーションへの変換に適していません。この作業については、『***移行ガイド***』オンラインマニュアルの『*はじめに*』の章を参照してください。

このセッションでは、COBOL 索引ファイルに保存されたアカウントデータにアクセスする既存の COBOL 副プログラム **acctopen.cb**を起動します。変換したアプリケーションを実行すると、エンドユーザ側では、1 つのフィールドをもつ HTML フォームが表示されます。エンドユーザがアカウント番号を入力して、[**クエリーの送信**] をクリックすると、このアカウントを担当する営業担当者の名前と、そのアカウントの最新の注文に関する情報がフォームに返され、表示されます。この情報は、アプリケーションによって **Net Express\Base\Demo\Formx** のデータファイル (**acctsasc.vsm**と **acctsasc.idx**) 内で検索されます。

5.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウなどのウィンドウが開いている場合は、閉じます。

Web ブラウザや Solo を先に実行しておく必要はありません。

5.3.1 プロジェクト、フォーム、およびプログラムの作成

『*Net Express の使用方法*』の章で説明した方法でプロジェクトを作成することもできますが、すぐにアプリケーションを作成する場合は、Net Express でプロジェクトを作成できます。 アプリケーションの作成方法は、次のとおりです。

1. [**ファイル > 新規作成**] をクリックし、「新規作成」ダイアログボックスで「**インターネットアプリケーション**」を選択して、[**OK**] をクリックします。

   インターネットアプリケーションウィザードの最初のページが表示されます。このページでは、『*CGI ベースのアプリケーションの概要*』の章で説明するとおり、3 通りのアプリケーション作成方法から 1 つを選択します。

2. 「**アプリケーション全体 (HTML クライアントとサーバプログラム)**」を選択します。

   ダイアログボックスの下部では、デフォルトで「**COBOL ソースファイル**」が選択されています。

3. [**次へ**] をクリックします。
4. プロジェクトを作成するかどうかを確認するメッセージに対して [**はい**] をクリックします。
5. 「**空プロジェクト**」(デフォルト設定) が選択されていることを確認します。 プロジェクト名として「**Account**」を入力し、プロジェクトを格納するフォルダとして「**Net Express\Base\Demo\Formx**」を入力して、[**作成**] をクリックします。

   このフォルダは、Net Express のインストール時に作成されます。このチュートリアル用に用意された COBOL プログラム **acctopen.cbl**は、このフォルダ内に格納されています。 このセッションを実行したことがある場合は、既存のプロジェクトを上書きするかどうかを確認するメッセージが表示されます。 [**はい**] をクリックします。

   表示されたページでは、フォームとフォーム処理プログラムを作成する既存のプログラムを指定できます。

6. [**参照**] をクリックし、**acctopen.cbl**をダブルクリックして、「**原始プログラム**」フィールドを設定します。そして、[**次へ**] をクリックします。
7. [**個別入出力フォーム**] をクリックし、デフォルトの「**クロスプラットフォーム (テーブル形式)**」を使用して、[**次へ**] をクリックします。

   「**クロスプラットフォーム**」を指定すると、Net Express によって位置フォーム内のオブジェクトが確実に希望の位置に配置されます。 クロスプラットフォームのフォームでは、HTML テーブルを使用して配置を確定します。動的 HTML では、スタイル属性が使用されます。 HTML テーブルは、多くのブラウザでサポートされます。動的 HTML では、より正確な位置を指定できますが、インターネットエクスプローラ 4.0 以降でのみサポートされます。 作成したフォームはエンドユーザ側で別のブラウザを使用して表示される場合があるので、注意してください。

表示されたページには、作成されるファイルが表示されます。作成されるファイルのうちの 2 ファイルは、データ入力用の HTML フォームと結果表示用の HTML フォームです。もう 1 つのファイルは、フォーム処理と Acctopen の呼び出しを実行する COBOL プログラムです。

8. [**次へ**] をクリックします。

5.3.2 フィールドの属性の追加

開いているページには、元のプログラムのデータ項目が表示されます。また、右側には、生成されるフォームの対応するフィールドが表示されます。最初は、最上位の項目のみが表示されます。

1. 左側のペインで **LS-OLD-MF-REC** の「+」をクリックしてから、**OMF-ORDER** の「+」をクリックします。

   ツリービューが展開され、**ls-old-mf-rec** に従属するすべての項目が 図 5-1 のように表示されます。

図 5-1 : データ項目と対応するフィールド

左側のペインには、**acctopen.cbl**の連絡節で宣言されたデータ項目が表示されています。右側のペインには、インターネットアプリケーションウィザードで生成されるフォーム上の対応するフィールドが表示されます。「**ラベル名**」カラムには、フィールドに

表示されるラベルが表示されます。これらのラベルは、宣言からデータ名に設定されています。ここでは、これらのラベルをよりわかりやすいラベルに変更し、その他の属性を設定します。

2. 右側のペインで「**ラベル名**」フィールドの **OMF-ACCTNO** をクリックし、「**アカウント**」と入力します。**Enter** キーを押します。

   「**コントロールタイプ**」の下部にあるエントリについては、デフォルトの「**編集**」を使用するので、変更する必要はありません。このエントリは、このデータ項目がフォームのテキストエントリフィールドで表されることを示します。

   「**入出力**」の下部にあるエントリについては、デフォルトの「**両方**」を使用するので、変更する必要はありません。このエントリは、このデータ項目がアプリケーションのフォームからのデータ入力の受け付けとそのフォームへの出力データの送信の両方に使用されることを表します。このアプリケーションは、非対称アプリケーション (入力と出力に別のフォームを使用) のため、対応するフィールドが入力フォームと出力フォームの両方に表示されます。

3. 右側のペインで「**ラベル名**」フィールドの **OMF-SALESREP** をクリックし、「**担当者**」と入力します。**Enter** キーを押します。

   「**コントロールタイプ**」の下部にあるエントリについては、デフォルトの「**編集**」を使用するので、変更する必要はありません。

4. 「**入出力**」フィールドをクリックして **OMF-SALESREP** (ここでは、**担当者**) を選択し、ドロップダウンリストから [**出力**] を選択します。

   このエントリは、このデータ項目がアプリケーションのフォームに出力データを送信することを表します。このアプリケーションは、非対称アプリケーションのため、対応するフィールドは出力フォームのみに表示されます。

5. 左側のペインで **OMF-ORDER** を右クリックし、ポップアップメニューの [**出力**] をクリックします。

   この方法で集団フィールドの属性を設定すると、その集団フィールドの従属項目が影響を受けます。右側のペインの [**入出力**] カラムが更新され、この属性が項目に対して表示されます。

6. [**次へ**] をクリックします。

   開いたページには、選択内容の概要が表示されます。

7. [**終了**] をクリックします。

インターネットアプリケーションウィザードによってフォームとプログラムが生成され、終了します。生成されたファイルの名前がプロジェクトに追加され、プロジェクトの依存関係が検索されます。インターネットアプリケーションウィザードが終了します。

5.3.3 作成したファイル

ここまでは、ユーザインターフェイスを作成しました。インターネットアプリケーションウィザードによって作成され、プロジェクトに追加されたファイルは、次のとおりです (これらのファイルをすべて表示するには、「プロジェクト」ウィンドウのツリービューで「+」記号をクリックして、ツリービューを展開する必要がある場合があります)。

- **acctopen_input.htm**- 入力フォームです。
- **acctopen_output.htm**- 出力フォームです。
- **acctopen_server.cbl**- **acctopen.cbl**とともにコンパイルし、リンクさせると、サーバ側プログラムになります。

次のファイルは、右側のペインのみに表示されます。

- **acctopen_input.mff**- サーバ側プログラムの生成に必要な追加情報が含まれています。
- **acctopen_output.mff**- サーバ側プログラムの生成に必要な追加情報が含まれています。

プロジェクトには、元のプログラム **acctopen.cbl**も追加されています。

プロジェクトのビルド時に作成された次のオブジェクトファイルもプロジェクトに追加されています。

- **acctopen_server.obj**- コンパイル済みのフォーム処理プログラムです。
- **acctopen_server.exe**- サーバ側プログラムの実行形式ファイルです。
- **acctopen.obj**- コンパイル済みの元のプログラムです。

また、複数のコピーファイル (拡張子 **.cp***) も作成されています。これらのファイルは、**.cbl**ファイル内の注釈に記述されています。

上記のファイル名は、小文字で表記されていますが、プロジェクト内では、大文字の場合があります。ファイル名については、大文字小文字が区別されません。

フォームを更新する必要がある場合は、「プロジェクト」ウィンドウの **acctopen_input.htm**または **acctopen_output.htm**をダブルクリックして、Form Designer を開きます。更新したフォームを保存すると、コピーファイルが再生成されるので、コピーファイルを直接編集しないでください。ウィザードで生成した **.cbl**プログラムは、編集できます。

5.3.4 アプリケーションのビルド

アプリケーションのビルド方法は、次のとおりです。

1. [**プロジェクト > すべてをリビルド**] をクリックします。

   Net Express でプログラムがコンパイルされ、実行可能ファイルがビルドされます。「出力」ウィンドウに「リビルドの完了」というメッセージが表示された後で、次に進みます。

5.3.5 アプリケーションの実行

アプリケーションの実行方法は、次のとおりです。

1. [**アニメート > 実行**] をクリックします。

   このアプリケーションでは、入出力に個別のフォームを使用するので、「アニメーションの起動」ダイアログボックスには実行開始場所として入力フォームが表示されます。

2. 「アニメーションの起動」ダイアログボックスの [**OK**] をクリックします。

   この操作によって Solo と Web ブラウザが起動します。エンドユーザが入力フォームへのリンクをクリックした場合と同様に入力フォーム「**acctopen_input.htm**」がブラウザにロードされます。

   Solo の実行に問題がある場合は、Net Express の [**ヘルプ**] メニューの [**ヘルプトピック**] をクリックして、ヘルプの索引で「Solo トラブルシューティング」を検索してください。

3. Web ブラウザの「**アカウント番号**」フィールドをクリックし、「A1」を入力して [**クエリーの送信**] をクリックします。(このアプリケーションでは、大文字と小文字が区別されるので、「a1」ではなく「A1」と入力してください。)

   IDE が自動的に最小化され、プログラムが実行されます。アカウント「A1」の営業担当者とこのアカウントの最新の注文に関する情報が出力フォームに格納され、このフォームに Web ブラウザが表示されます。

4. 「**アカウント番号**」フィールドに「A1 ～ A9」の番号入力を試行してください。

   このファイルには、A1 ～ A9 のアカウントのレコードが含まれています。既存のプログラム「**acctopen.cbl**」には、「B1」などの他の番号を入力した場合に「**担当者**」フィールドに「見つかりません」が返されるようなコードが含まれています。

5. 最小化された IDE を復元し、[**アニメート > アニメート停止**] をクリックします。

5.4 次に進む前に

プロジェクトを閉じます。

タスクバートレイの [Solo] アイコン  を右クリックし、ポップアップメニューの [**終了**] をクリックして、Solo を閉じます。

一連のチュートリアルは、ここまでです。別のチュートリアルに進むには、第 1 章『*はじめに*』の『*チュートリアル マップ*』を参照してください。

第 6 章 : Web データベースアプリケーションの作成

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

6.1 概要

この章では、既存のデータベースに対して Web インターフェイスを作成するには、インターネットアプリケーションウィザードを使用します。

フォームとサーバ側プログラムを作成して Web 経由で既存のデータベースにアクセスするには、インターネットアプリケーションウィザードを使用します。

オープンデータベースコネクティビティ (ODBC) がサポートされるデータベースシステムで作成されたデータベースを使用する必要があります。このようなデータベースの例として、Oracle、Sybase、Informix、DB2、Microsoft SQL (読みは「シークエル」) Server、Microsoft Access などが挙げられます。

インターネットアプリケーションウィザードを使用すると、フォーム設計やプログラミングを手作業で実行する必要がありません。ウィザードによってデータベース構造が検索され、フォームとサーバ側プログラムが作成されます。ユーザは、可能な操作を指定するためのオプションを設定するのみです。

生成されたアプリケーションは、データベースを問い合わせて (このオプションを指定した場合) 更新できます。元々のデータベースツールを使用し、データベースを問い合わせて更新することもできます。

作業を始める前に、独自のデータソースを設定する必要があります。詳細については、ヘルプを参照してください。ヘルプでは、2 つのデータソース (**UserSample1** と **UserSample2**) を作成することを推奨しています。Net Express には、卸売り会社の顧客情報を格納した Microsoft Access データベースなどの複数のサンプルデータベースが含まれています。顧客は、小売店です。このセッションでは、Web アプリケーションを作成して、このデータベースを問い合わせて更新します。ここでは、その他に XDB データベースも使用します。以後では、Microsoft Access データベースではなくXDB データベースを使用する場合に必要な入力値について適宜説明します。

データベースには、データソース名 (DSN) **UserSample2** (Microsoft Acces サンプルデータベース **\Net Express\Base\Demo\SMPLDATA\access\sample.mdb**) 経由でアクセスします。

6.1.1 用語

このセッションのデータベースに関する説明では、データベース用語「*行*」と「*カラム*」を使用します。データが Web ブラウザのフォームに表示された場合は、対応する用語として「*レコード*」と「*フィールド*」を使用します。

6.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウやテキストウィンドウが開いている場合は、閉じます。

Web ブラウザや Solo を先に実行しておく必要はありません。

6.3.1 プロジェクト、フォーム、およびプログラムの作成

『*Net Express の使用方法*』の章で説明した方法でプロジェクトを作成することもできますが、すぐにアプリケーションを作成する場合は、Net Express でプロジェクトを作成できます。アプリケーションの作成方法は、次のとおりです。

1. [**ファイル > 新規作成**] をクリックし、「新規作成」ダイアログボックスで「**インターネットアプリケーション**」を選択して、[**OK**] をクリックします。

   インターネットアプリケーションウィザードの最初のページが表示されます。このページでは、『*CGI ベースのアプリケーションの概要*』の章で説明するとおり、3 通りのアプリケーション作成方法から 1 つを選択します。

2. 「**アプリケーション全体 (HTML クライアントとサーバプログラム)**」を選択し、ダイアログボックスの下部で「**SQL データベース**」を選択します。
3. [**次へ**] をクリックします。
4. プロジェクトを作成するかどうかを確認するメッセージに対して [**はい**] をクリックします。
5. 「**空プロジェクト**」(デフォルト設定) が選択されていることを確認します。プロジェクト名として「**Daw**」を入力し、プロジェクトを格納するフォルダとして「**Net Express\Base\Demo\Daw**」を入力して、[**作成**] をクリックします。
6. ディレクトリを作成するかどうかを確認するメッセージに対して [**はい**] をクリックします。(このセッションを実行したことがある場合は、既存のプロジェクトを上書きするかどうかを確認するメッセージが表示されます。[**はい**] をクリックします。)
7. 生成するすべてのファイルの基本名として「**小売**」を、すべてのフォームに表示するタイトルとして「**小売店**」を入力して、[**次へ**] をクリックします。

   表示されたページに、使用中のコンピュータのデータソースが一覧表示されます。右側には、多くのタブが表示されます。前面に [**クエリー**] タブが表示されていることを確認します。(デフォルトでは、前面に表示されます。)

8. **UserSample2** をダブルクリックします。

   リストが展開され、このデータベース内のテーブルが表示されます。

9. [**顧客**] をダブルクリックします。

   リストが展開され、「顧客」テーブルのカラムが表示されます。

   SQL (読みは「シークエル」) の SELECT 文が生成され、右側に表示されます。 SQL は、データベースを問い合わせるための言語です。 Net Express には、OpenESQL という機能があります。OpenESQL は、SQL を COBOL ソースに含めるためのコンパイラプリプロセッサです。 インターネットアプリケーションウィザードを使用する場合は、SQL に精通している必要はありません。ウィザードによって必要な SQL が作成されます。

   この段階では、文は完成していません。 文は、次の段階で完成し、インターネットアプリケーションウィザードで使用する COBOL 原始プログラムに含まれる SQL の基礎として使用されます。

10. ツリービューの「**顧客**」を右クリックし、ポップアップメニューの [**すべてのカラムを選択**] をクリックします。

    すべてのカラム名にチェックマークが表示され、「クエリー」フィールド内の文が完成します。 この文によって「顧客」テーブルのすべてのカラムが表示されます。

    このクエリーは、COBOL プログラムに埋め込むクエリーですが、この段階で次のように対話的に実行することもできます。

11. [**クエリーの実行**] (「インターネットアプリケーションウィザード」ダイアログボックスの左上付近) をクリックします。

    [**実行結果**] タブが前面に開き、「顧客」テーブルのすべての行のすべてのカラムが表示されます。

    原始プログラムに埋め込むために生成中の SQL 文では、この操作を実行します。ただし、エンドユーザが実行時に指定できる選択条件 (表示する行 -表示する顧客の詳細情報- の選択) も処理されます。 これらの文によって、テーブルからのデータがプログラムの作業場所節に読み込まれます。他の COBOL 文によってこの作業場所節から Web フォームにデータが転記されます。

    このダイアログボックスは、Net Express の他の機能 (OpenESQL アシスタント) でも使用されます。 この機能を使用して SQL クエリーを作成し、これらのクエリーを対話的に実行するか、または、作成した COBOL 原始プログラム内に指定します。

    ここで作成したクエリーより複雑なクエリーの作成方法については、このマニュアルを読み終えた後で、『***データベースアクセス***』オンラインマニュアルの OpenESQL アシスタントに関するチュートリアルを参照してください。

12. [**次へ**] をクリックします。

表示されたページで、アプリケーションで使用する Web フォームを指定します。

「**クロスプラットフォーム**」を指定すると、Net Express によって位置フォーム内のオブジェクトが確実に希望の位置に配置されます。 クロスプラットフォームのフォームでは、HTML テーブルを使用して配置を確定します。動的 HTML では、スタイル属性が使用されます。HTML テーブルは、多くのブラウザでサポートされます。動的 HTML では、より正確な位置を指定できますが、インターネットエクスプローラ 4.0 以降でのみサポートされます。作成したフォームはエンドユーザ側で別のブラウザを使用して表示される場合があるので、注意してください。ここでは、デフォルト設定を使用します。

フォームには、2 つの標準フォームがあります。単一レコードビュー (別称「レコードビュー」) フォームには、テーブルの 1 行からのデータ (ここでは、顧客の詳細情報) が表示されます。テーブルビューフォームには、複数の行のデータが表形式で表示されます。ここでは、デフォルト設定を使用し、両方のフォームを使用します。

13. [**次へ**] をクリックします。

    表示されたページで、ユーザに対して、データベースへの問い合わせのみを許可するか、または、更新も許可するかを指定します。ここでは、デフォルト設定を使用し、更新も許可します。

14. [**次へ**] をクリックします。

    開いたページには、選択内容の概要が表示されます。

15. [**終了**] をクリックします。

    インターネットアプリケーションウィザードによってフォームとプログラムが生成され、終了します。生成されたファイルの名前がプロジェクトに追加され、プロジェクトの依存関係が検索されます。インターネットアプリケーションウィザードが終了します。

6.3.2 作成したファイル

ここまでは、ユーザインターフェイスを作成しました。インターネットアプリケーションウィザードによって作成され、プロジェクトに追加されたファイルは、次のとおりです (これらのファイルをすべて表示するには、「プロジェクト」ウィンドウのツリービューで「+」記号をクリックして、ツリービューを展開する必要がある場合があります)。

- **retailform.htm** - HTML 形式のレコードビューフォームです。
- **retailform.cbl** - このフォームのサーバ側プログラムです。
- テーブルビューフォームに関する同様のファイルセット (すべてのファイルの基本名は「**retaillist**」)

次のファイルは、右側のペインのみに表示されます。

- **retailform.mff** - サーバ側プログラムの生成に必要な追加情報が含まれています。
- **retaillist.mff** - サーバ側プログラムの生成に必要な追加情報が含まれています。

プロジェクトのビルド時に作成された次のオブジェクトファイルもプロジェクトに追加されています。

- **retailform.obj** - コンパイル済みのサーバ側プログラムです。
- **retailform.exe** - 実行形式ファイルです。
- テーブルビューフォームに関する同様のファイルセット (すべてのファイルの基本名は「**retaillist**」)

また、複数のコピーファイル (拡張子 **.cp***) も作成されています。これらのファイルは、**.cbl** ファイル内の注釈に記述されています。

上記のファイル名は、小文字で表記されていますが、プロジェクト内では、大文字の場合があります。ファイル名については、大文字小文字が区別されません。

フォームを更新する必要がある場合は、「プロジェクト」ウィンドウの **retailform.htm** または **retaillist.htm** をダブルクリックして、Form Designer を開きます。更新したフォームを保存すると、コピーファイルが再生成されるので、コピーファイルを直接編集しないでください。ウィザードで生成した **.cbl** プログラムは、編集できます。

6.3.3 アプリケーションのビルド

アプリケーションのビルド方法は、次のとおりです。

1. [**プロジェクト > すべてをリビルド**] をクリックします。

   Net Express によってプログラムが保存およびコンパイルされ、実行形式ファイルがビルドされます。「出力」ウィンドウに「リビルドの完了」というメッセージが表示された後で、次に進みます。

6.3.4 アプリケーションの実行

アプリケーションの実行方法は、次のとおりです。

1. [**アニメート > 実行**] をクリックします。
2. 「アニメーションの起動」ダイアログボックスの [**OK**] をクリックします。

   Solo と Web ブラウザが起動し、プログラム Retailform が実行されます。このプログラムの単一レコードビューフォーム **retailform.htm** が表示され、IDE が自動的に最小化されます。

   まず、このフォームには、テーブルの最初の行 (CustID が ALWAO の行) から顧客の詳細情報が表示されます。

Web ブラウザのフォントサイズに応じて、レコードビューフォーム内の顧客 ID の 5 文字の一部が表示されない場合があります。 たとえば、「ALWAO」が「ALWA」、「MERRG」が「MERR」のように表示されることがあります。「CustID」フィールドをクリックし、左右の矢印キーを使用して、横方向にスクロールすると、顧客 ID 全体が表示されます。

3. フォームの右上の [**テーブルビュー**] をクリックします。

   テーブルビューフォーム **retaillist.htm** がブラウザにロードされます。 このテーブルには、最初の 10 名の顧客 (ALWAO ～ CONSH) が表示されます。

4. フォームの右上の [**レコードビュー**] をクリックします。

   単一レコードビューフォーム **retailform.htm** が再度ロードされます。

6.3.5 データベース内の移動

両方のフォームには、テーブル内を前後に移動するための標準のコントロールが用意されています。 レコードビューフォーム **retailform.htm** には、テーブルの更新と問い合わせを実行するための機能もあります。 これらの機能の一部は、フォームの作成にオプションで設定できます。 すべての機能を設定するには、すべてのデフォルト値を使用します。

ここでは、これらの標準コントロールの一部を使用します。 単一レコードビューフォームが表示されていることを確認してください。「整列条件」フィールドのデフォルト値は、「CustID」です。 このデフォルト値を使用すると、テーブルの行が「CustID」順に表示されます。

必要に応じて、フォームを正しく表示するために、Web ブラウザを全画面表示してください。

1. フォームの上部の [**>**] をクリックします。

   このフォームに、テーブル内の次の顧客 (CustID が ANDRC の行) が表示されます。

2. [**テーブルビュー**] をクリックします。

   テーブルビューフォームがロードされ、ANDRC 以降の顧客 10 名分が表示されます。

3. [**>>**] をクリックします。

   フォームに、テーブルの末尾 (最後の 10 名の顧客 TOPOF ～ WITAE) が表示されます。

4. [**>**] をクリックします。

   フォームが再表示されますが、変更されていません。 テーブルビューでテーブルの末尾を超えて表示しようとしても、その先の情報は表示されません。

5. [**<<**] をクリックします。

   フォームに、テーブルの先頭 (先頭の 10 名の顧客 ALWAO ～ CONSH) が表示されます。

6. [**>**] をクリックします。

   フォームに次の 10 名の顧客 EASTC ～ HANOP が表示されます。

7. [**レコードビュー**] をクリックします。

   レコードビューがロードされ、顧客 EASTC が表示されます。

8. [**>>**] をクリックします。

   フォームにテーブル内の最後の顧客 WITAE が表示されます。

9. [**>**] をクリックします。

   フォームにテーブル内の最初の顧客 ALWAO が表示されます。レコードビューでテーブルの末尾を超えて表示しようとすると、テーブルの先頭の情報に戻ります。

10. 「CustID」フィールドの値を現在の ALWAO から「BE」に変更して、[**クエリー**] をクリックします。

    「CustID」がアルファベット順で「CustID」フィールドに入力した値以降の文字で始まる最初の顧客がフォームに表示されます。ここでは、顧客 BERgt が表示されます。

11. [テーブルビュー] をクリックします。

    テーブルビューがロードされ、BERgt 以降の顧客 10 名分が表示されます。

12. 顧客 EASTC の最初のカラムで **EASTC** をクリックします。

    単一レコードビューフォームに顧客 EASTC が表示されます。

6.3.6 データベース内の更新

アプリケーションの作成時にデフォルト設定を使用して、ユーザによるデータベースの更新を許可しました。ここでは、単一レコードビューフォームを使用してレコードの挿入、削除、または更新を実行します。

1. 「アドレス」フィールドの「35 King George」に「Street」を追加して、フォームの下部にある [**レコードの更新**] をクリックします。

レコードが再表示されます。

2. [**テーブルビュー**] をクリックします。

   テーブルビューが再度ロードされ、表示されたデータベースに変更が反映されていることがわかります。

6.3.7 データベースのフィルタ処理

レコードビューフォームの最上部にあるフィールドを使用し、表示したくないレコードをフィルタ処理によって非表示にしたり、レコードの順序を決定するためのフィールドを指定したりできます。

たとえば、カナダの顧客のみを表示するには、都市順に表示します。

1. [**レコードビュー**] をクリックします。

   レコードビューフォームが表示されます。フィルタの設定前に [**画面をクリア**] をクリックして、顧客データのフィールドをクリアできます。ただし、指定したフィールドのみが表示されるので、この操作は必要ありません。

2. ドロップダウンリストを使用して、「整列条件」フィールドを「**都市**」、「フィルタの種類」フィールドを「=」(デフォルト) に、「フィルタ条件」フィールドを「**国**」に設定します。
3. 「国」フィールドの値を「**カナダ**」に変更して、[**<<**] をクリックします。

   フォームが表示され、テーブルに保存されたカナダ在住の最初の 10 名 (都市順) の顧客が表示されます。この場合は、East Vancouver の MERRG が表示されます。

   「フィルタ条件」フィールドは、自動的に「(既存)」に変更されます。そのため、このフィールドの値を変更するまで、次以降の各手順で、選択済みの行が使用されます。次以降の手順では、テーブル内の位置を変更するのみで、新しいフィルタを設定しません。

4. 「都市」フィールドの値を **T** に変更して、[**クエリー**] をクリックします。

   フォームに、都市順に並べたときに都市名の先頭の文字が「T」以降の最初のカナダの顧客が表示されます。この場合は、Tsawassen の BOTTM が表示されます。

5. [**テーブルビュー**] をクリックします。

   テーブルビューフォームがロードされます。都市順で Tsawassen 以降のカナダの顧客が表示されます。表示される顧客は、3 名です。

   新しいフィルタを設定します。フィルタを設定するには、「フィルタ条件」フィールドを「(既存)」から「フィールド名」に再度戻します。

6. [**レコードビュー**] をクリックします。
7. 「フィルタ条件」フィールドを「**国**」に変更し、「国」フィールドに **UK** と入力して、[**<<**] をクリックします。

   フォームに都市順で最初の英国の顧客 Hedge End の ISLAT が表示されます。

8. [**テーブルビュー**] をクリックします。

   テーブルビューフォームがロードされます。都市順で最初の英国の顧客 10 名分が表示されます。

   次に、元のようにテーブル全体を表示できるようにして、顧客 ID 順にレコードを並べます。

9. [**レコードビュー**] をクリックします。
10. 「整列条件」フィールドを「**CustID**」に変更し、「フィルタ条件」フィールドを「**(なし)**」に変更して、[**<<**] をクリックします。

    フォームが再表示され、テーブル内の最初の 10 名 (顧客 ID 順) の顧客が表示されます。

11. [**テーブルビュー**] をクリックします。

    テーブルビューフォームがロードされ、最初の 10 名の顧客が表示されます。テーブル全体が顧客 ID 順で再表示されます。

12. 最小化された IDE を復元し、Net Express の [**アニメート**] メニューで [**アニメート停止**] をクリックします。

6.4 次に進む前に

プロジェクトを閉じます。

タスクバートレイの [Solo] アイコン  を右クリックし、ポップアップメニューの [**終了**] をクリックして、Solo を閉じます。

一連のチュートリアルは、ここまでです。別のチュートリアルに進むには、第 1 章『*はじめに*』の『*チュートリアルマップ*』を参照してください。

第 7 章 : ウィザードを使用した Java クラスの作成

チュートリアルは、第 1 章『はじめに』の『チュートリアルマップ』に表示されている矢印の順に読み進んでください。

7.1 概要

この章では、クラスウィザードとメソッドウィザードを使用して Java クラスを作成し、ビジネスルーチンをラップします。

クラスウィザードで Java クラスの作成を選択すると、Java クラスにラップされた COBOL クラスが作成されます。この COBOL クラスにアプリケーションをラップします。JSP からビジネスルーチンを呼び出すには、対応する Java メソッドを呼び出します。この Java メソッドで COBOL メソッドを呼び出し、COBOL メソッドからアプリケーションを呼び出します。JSP 側では、Java で作成されたビジネスルーチンと同様に処理されます。このウィザードでは、Java と COBOL のクラスとメソッドが作成されます。

Net Express には、Java コンパイラが含まれていませんが、Java コンパイラを統合できます。Net Express で Java コンパイラを呼び出す方法については、『分散コンピューティング』マニュアルの『Java と COBOL の連携』の章を参照してください。Net Express のビルドまたはリビルド機能を使用すると、Java クラスが COBOL プログラムのビルド処理の一部としてシームレスにコンパイルされます。

Net Express では、COBOL プログラム内に Java ドメインを指定できます。Java ドメインを指定すると、COBOL プログラム内で Java クラスを宣言できます。そのため、Java と Net Express COBOL を相互運用できます (相互呼び出しが可能)。従来の手続き型 COBOL とオブジェクト指向 COBOL の両方で可能です。COBOL から Java を呼び出す方法、または、Java から COBOL を呼び出す方法については、『分散コンピューティング』マニュアルを参照してください。

7.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウなどのウィンドウが開いている場合は、閉じます。

7.3 Java クラスの作成

Java クラスの作成方法は、次のとおりです。

1. OO COBOL ツールバーを表示します ([表示 > ドッキングできるウィンドウ] をクリックし、「OO COBOL」をチェックします)。

2. [ファイル > 新規作成 > クラス] の順にクリックするか、OO COBOL ツールバーの  をクリックして、クラスウィザードを起動します。
3. ウィザードの最初のページで、作成するクラスの種類として「Java」を選択します (図 7-1 を参照)。

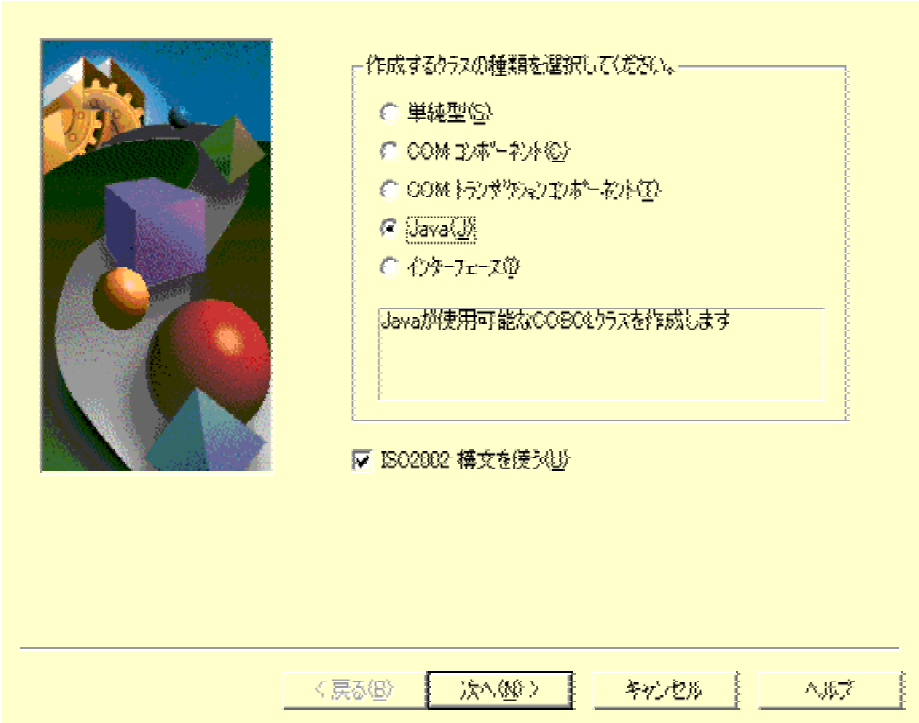


図 7-1：Java が選択されたクラスウィザード

4. ウィザードの他のページでは、Java クラス、COBOL クラスの名前、および、ビルドした COBOL クラスを格納する実行形式ファイルの名前を指定します。

次に OO COBOL ツールバーの  をクリックして、メソッドウィザードを起動し、メソッドを作成します。 ウィザードでメソッドを作成する場合は、COBOL クラス内にメソッドが作成されます。また、Java クラス内に、対応するメソッドが作成され、そのクラスを呼び出すことができます。 後でコードを挿入するために、空の COBOL メソッドが作成されます。 各ビジネスルーチンに対して 1 つのメソッドを作成し、ビジネスルーチンを呼び出すための行を挿入します。

7.4 次に進む前に

一連のチュートリアルは、ここまでです。 別のチュートリアルに進むには、第 1 章『はじめに』の『チュートリアルマップ』を参照してください。

第 8 章 : ウィザードを使用した COM オブジェクトの作成

チュートリアルは、第 1 章『はじめに』の『チュートリアルマップ』に表示されている矢印の順に読み進んでください。

8.1 概要

この章では、クラスウィザードとメソッドウィザードを使用して COM オブジェクトを作成します。

アプリケーションを COM オブジェクトにする場合は、クライアントアプリケーションを起動してその COM オブジェクトを呼び出します。COM ではオブジェクトを処理するので、COM オブジェクトとしてディプロイする COBOL アプリケーションを最も簡単に準備するには、相関するオブジェクト指向クラスのセットとして作成します。従来の手続き型 COBOL で作成された既存のアプリケーションを使用する場合は、オブジェクト指向のクラス内に各 COBOL プログラムをラップできます。その場合は、クライアントアプリケーションでクラスのメソッドを呼び出し、そのメソッドから手続き型 COBOL ルーチンを呼び出します。Net Express には、このラッパークラスを作成して COM オブジェクトとして登録するための機能があります。この機能は、クラスウィザードとメソッドウィザードです。

8.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウなどのウィンドウが開いている場合は、閉じます。

8.3 COM オブジェクトの作成

COM オブジェクトの作成方法は、次のとおりです。

1. OO COBOL ツールバーを表示します ([表示 > ドッキングできるウィンドウ] をクリックし、「OO COBOL」をチェックします)。
2. [ファイル > 新規作成 > クラス] の順にクリックするか、OO COBOL ツールバーの  をクリックして、クラスウィザードを起動します。
3. クラスを COM オブジェクトとしてのみディプロイする場合は、ウィザードの最初のページで「COM コンポーネント」を選択します (図 8-1 を参照)。クラスをトランザクションコンポーネントとしてディプロイする場合は、「COM トランザクションコンポーネント」を選択します。

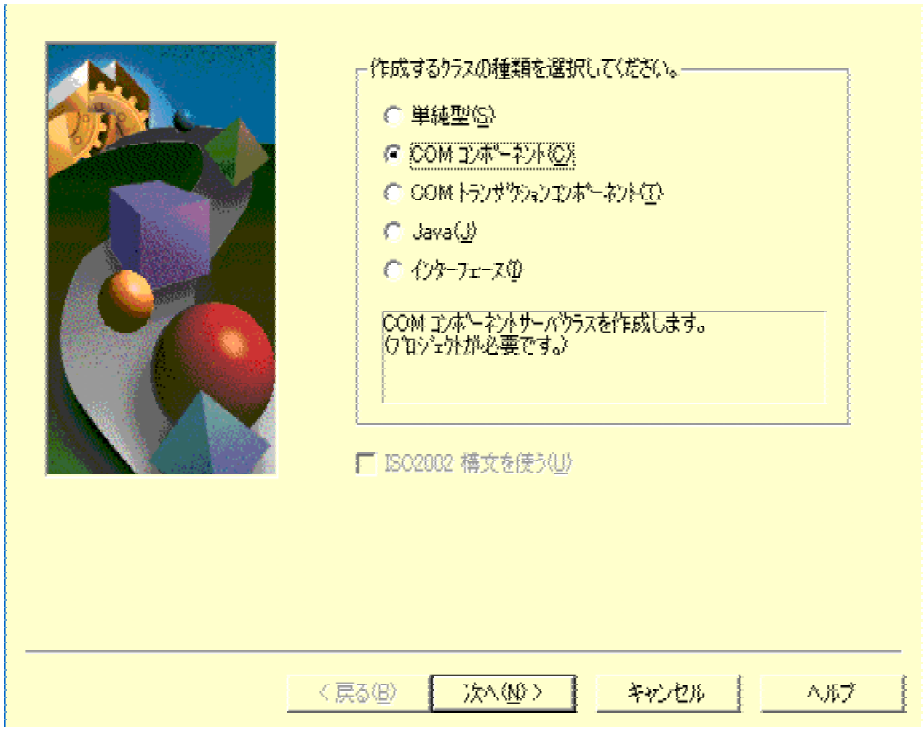


図 8-1：COM コンポーネントが選択されたクラスウィザード

4. ウィザードの他のページでは、ビルドした COBOL クラスを格納する実行形式ファイルの名前などの詳細情報を指定します。
5. クラスを作成すると、.reg ファイルも作成されます。このファイルを実行して、クラスを COM オブジェクトとして登録します。クラスをリモートオブジェクト (DCOM を使用したネットワーク経由での呼び出しが可能) として指定すると、.reg ファイルが 2 つ作成されます。

次に OO COBOL ツールバーの  をクリックして、メソッドウィザードを起動し、メソッドを作成します。このウィザードでは、コードを挿入するための空のメソッドを作成します。各ビジネスルーチンに対して 1 つのメソッドを作成し、ビジネスルーチンを呼び出すための行を挿入します。

8.4 次に進む前に

必要に応じて、.NET を使用してクライアントアプリケーションを作成して COM オブジェクトを使用することができます。クライアントアプリケーションの作成方法については、『入門書』の『COM 用の .NET クライアント』に進んでください。

COM オブジェクトを Web サービスとして公開する方法については、第 9 章『Web サービスとしての COM オブジェクトのディプロイ』に進んでください。

作業を中断する場合は、Net Express やプロジェクトを閉じてかまいません。その場合は、次の章に進んだときに、Net Express やプロジェクトを再度開いてください。

第 9 章 : Web サービスとしての COM オブジェクトのディプロイ

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

9.1 概要

この章では、クラスウィザードとメソッドウィザードを使用して Web サービスとして作成した COM オブジェクトをディプロイします。

Net Express で Web サービスを作成するには、Interface Mapping Toolkit を使用することをお奨めします。その他の方法との比較については、『***分散コンピューティング***』マニュアルの『*はじめに*』の章を参照してください。 Web サービスを Enterprise Server にディプロイしない場合は、Microsoft SOAP Toolkit を使用して Web サービスを作成することもできます。 クラスウィザードとメソッドウィザードを使用して COM オブジェクトを作成し、Microsoft SOAP Toolkit を使用してその COM オブジェクトを Web サービスにすることもできます。

Microsoft SOAP Toolkit は、www.microsoft.comからダウンロードできます。 このツールキットには、既存の Web サービスにアクセスするための COM コンポーネントのセットが含まれています。 また、既存の COM コンポーネントから WSDL ファイルを生成し、その COM コンポーネントを Web サービスとして実装する WSDL ジェネレータも含まれています。 Net Express では、COBOL から COM コンポーネントへアクセスしたり、COBOL で COM コンポーネントを新しく作成したりできます。 そのため、COBOL から Microsoft SOAP Toolkit を使用できます。 Micro Focus COBOL では、COM と Java を相互運用するために、オブジェクト指向プログラミング用に拡張された COBOL 構文を使用します。 ただし、ユーザがオブジェクト指向プログラミングに精通している必要はありません。 Web サービスの作成時に Net Express のクラスウィザードとメソッドウィザードを使用すると、必要なすべての OO 構文が作成されます。 COBOL から Web サービスを呼び出す場合は、新しい動詞「**INVOKE**」を使用します。 Net Express でサポートされる COM と COBOL の相互運用については、『***分散コンピューティング***』マニュアルの『COM、COBOL、および .NET』を参照してください。

このチュートリアルでは、新聞などの広告媒体に対する簡単な広告 Web サービス (広告の掲載、変更、および検索が可能) を例として使用します。ここでは、『*ウィザードを使用した COM オブジェクトの作成*』の章の説明に従い、クラスウィザードとメソッドウィザードを使用して COM コンポーネントを作成し、その COM コンポーネントを登録して、**.dll** ファイルをビルドしてあることを仮定します。 Visual Basic に実装された Web サービスクライアントも提供されます。 COM コンポーネント AdService には、次のメソッドが含まれています。

PlaceAd	広告を掲載し、その広告を識別するための一意な ID を返します。
ChangeAd	特定の広告の詳細を変更します。

DeleteAd	広告を削除します。
RetrieveAdById	広告の詳細を取得します。
RetrieveFirstAdId	最初の広告の ID を取得します。
RetrieveNextAdId	次の広告の ID を取得します。
RetrieveFirstAdByEmail	特定の電子メール ID で掲載された最初の広告の ID を取得します。
RetrieveNextAdByEmail	特定の電子メール ID で掲載された次の広告の ID を取得します。

ここでは、これらのすべてのメソッドを Web サービスの機能として実装します。その場合は、Microsoft SOAP Toolkit のユーティリティである WSDL ジェネレータを実行する必要があります。

9.2 Web サービス用の WSDL ファイルの作成

WSDL ファイルの作成方法は、次のとおりです。

1. WSDL ジェネレータを起動します。最初のウィンドウは、図 9-1 のように表示されます。

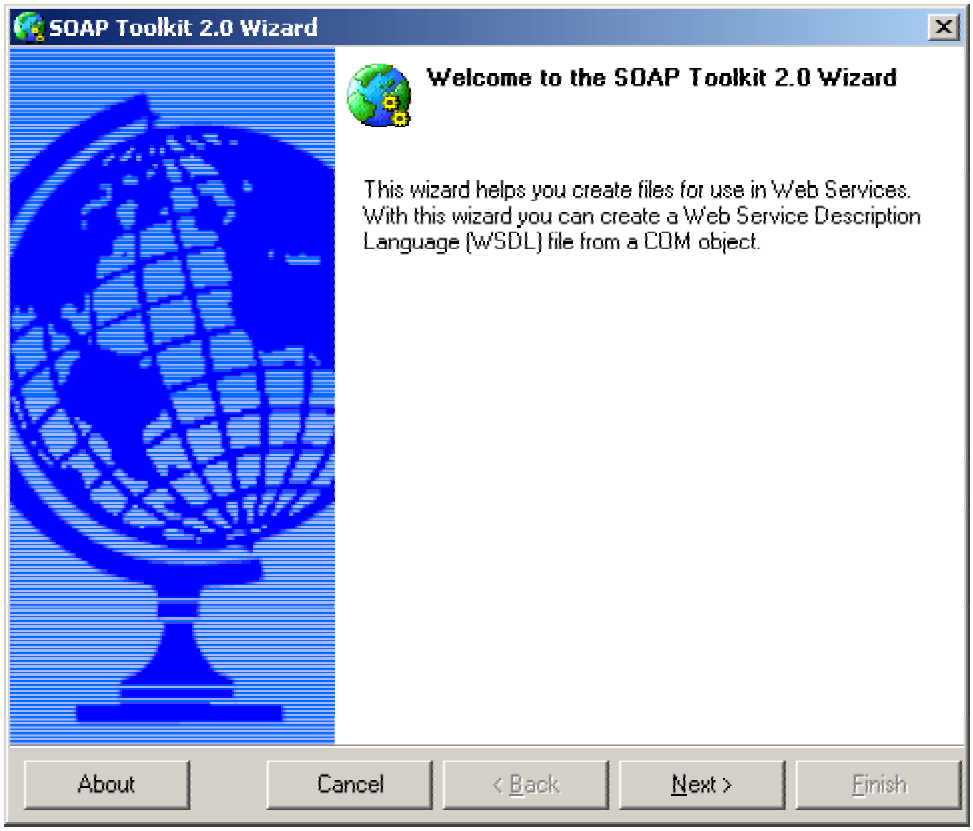


図 9-1：WSDL ジェネレータの最初の画面

2. [**次へ**] をクリックします。図 9-2 のウィンドウが表示されます。このウィンドウでは、Web サービスとして実装する COM コンポーネントを指定します。

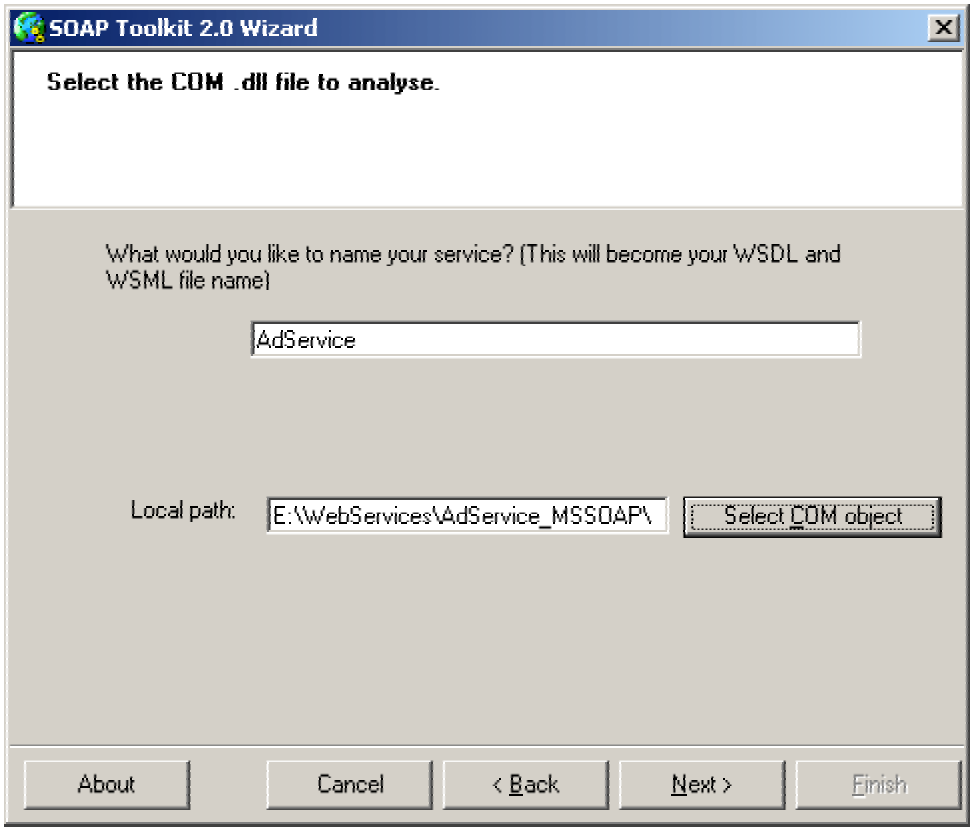


図 9-2：WSDL ジェネレータの 2 番目の画面

3. Net Express を使用してビルドする **.dll** ファイルが格納されたディレクトリのパスを入力します。この画面では、Web サービスに付ける名前も指定します。この名前が、WSDL ファイル内でサービス名として使用されます。ここでは、**AdService** という名前を指定します。
4. [**次へ**] をクリックします。図 9-3 のページが表示されます。このページでは、Web サービスの機能として実装するメソッドを選択できます。すべてのメソッドを実装しない方がよい場合もありますが、ここでは、すべてを選択します。

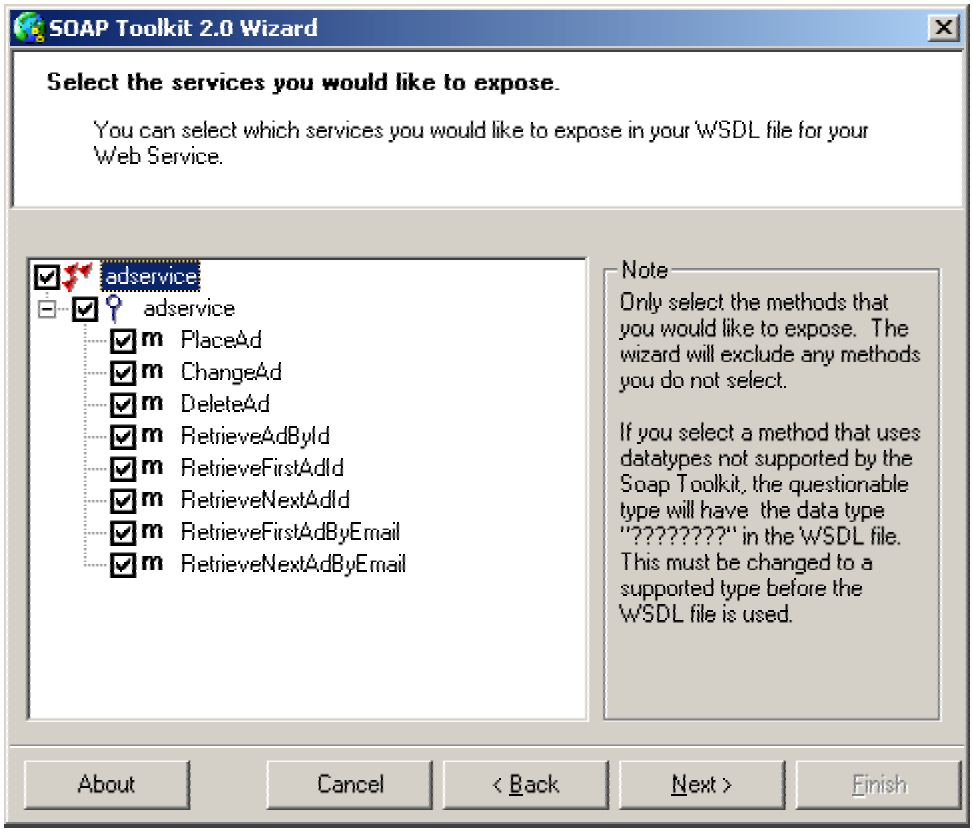


図 9-3：WSDL ジェネレータの 3 番目の画面

5. [**次へ**] をクリックします。図 9-4 のページが表示されます。この画面で重要なフィールドは、「**リスナー URI**」フィールドです。このフィールドに、サーバ名、および、WSDL ファイルと関連ファイルを格納するサーバ上の位置を指定します。クライアントは、この位置に接続して Web サービスにアクセスします。この例では、クライアントとサーバを同じマシンで実行するので、サーバ名として Localhost を指定します。通常は、個別のサーバ名または IP アドレスを指定します。使用される位置は、WSDL ファイルの位置として設定される、サーバの仮想ディレクトリの名前です。

   他のフィールドには、デフォルト値を使用します。

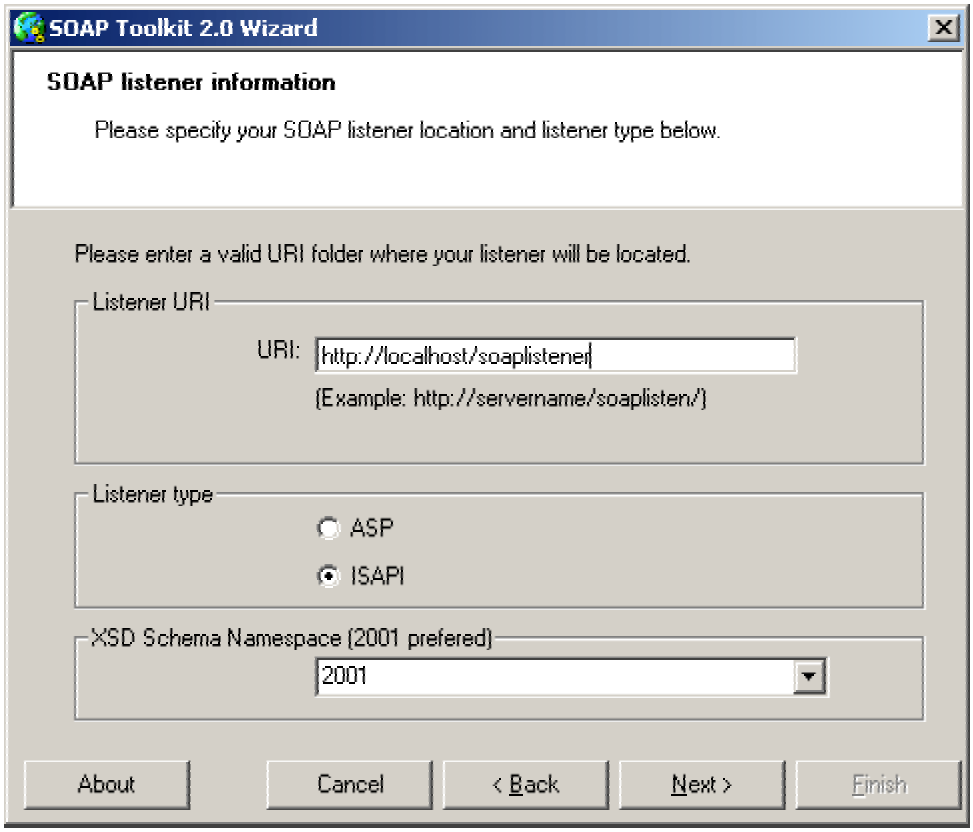


図 9-4：WSDL ジェネレータの 4 番目の画面

6. **[次へ] をクリックします。WSDL ジェネレータの最後の画面が表示されます。この画面では、作成したファイルの格納場所を指定します。このページは、図 9-5 のように表示されます。**

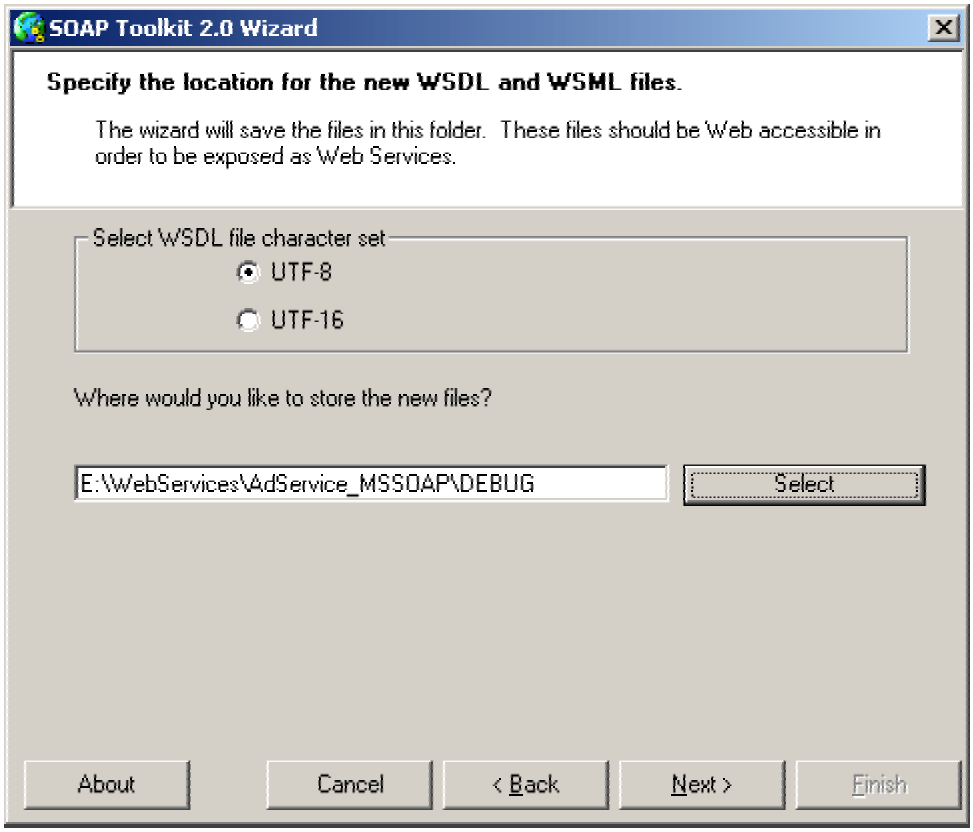


図 9-5：WSDL ジェネレータの 5 番目の画面

WSDL ファイルからは、2 つのファイル (**AdService.wsdl** と **AdService.wsml**) が生成されます。WSML ファイルは、Microsoft SOAP Toolkit に固有のファイルで、実行時に Web サービスの機能を提供する COM コンポーネントに関する情報が格納されています。その COM コンポーネントは、サーバ上で WSDL ファイルと同じ位置に格納する必要があります。

SOAP Toolkit と WSDL ジェネレータの使用方法については、Microsoft SOAP Toolkit のユーザガイドを参照してください。

9.3 次に進む前に

Interface Mapping Toolkit の「WSDL からクライアントを生成」機能を使用してクライアントを生成することもできます。この生成方法については、『*入門書*』マニュアルの『Web サービス用に生成した COBOL クライアント』の章に進んでください。

クライアントを生成しない場合には、一連のチュートリアルはここまでです。別のチュートリアルに進むには、第 1 章『*はじめに*』の『*チュートリアルマップ*』を参照してください。

第 10 : DB2 アプリケーション (SQL オプション)

注意：この章に記載された機能は、日本語版の Net Express 製品では提供されていません。

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

このセッションを実行するには、SQL オプションをインストールする必要があります。 ここでは、ユーザがメインフレーム上の DB2 に精通していることを前提に説明します。

10.1 概要

この章では、PC での DB2 アプリケーションの維持と実行について説明します。また、XDB Link 技術を使用してメインフレームにアクセスするための PC の設定方法やメインフレームで DB2 アプリケーションを直接実行するための PC の設定方法についても説明します。

この章で使用するデモアプリケーションは、メインフレームからダウンロードできるような簡単な DB2 アプリケーションで、 PC で実行します。

10.2 準備

Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウなどのウィンドウが開いている場合は、閉じます。

10.3.1 プロジェクトの作成

プロジェクトの作成方法は、次のとおりです。

1. [**ファイル > 新規作成**] をクリックし、「新規作成」ダイアログボックスで「**プロジェクト**」を選択して、[**OK**] をクリックします。
2. 「**空プロジェクト**」が選択されていることを確認します。プロジェクト名として **sqldemo** を入力し、プロジェクトを格納するフォルダとして **Net Express\mfsql\demo** を入力して、[**作成**] をクリックします。
3. [**プロジェクト > プロパティ**] をクリックします。
4. [**SQL 指令**] をクリックします。ESQL プリプロセッサのドロップダウンメニューで「**XDB**」を選択し、[**OK**] をクリックします。
5. 「プロジェクトのプロパティ」ダイアログボックスで [**OK**] をクリックします。

10.3.2 プロジェクトへのファイルの追加

プロジェクトへのファイルの追加方法は、次のとおりです。

1. [**プロジェクト > プロジェクトへファイルの追加**] をクリックします。
2. 「**ファイルを追加**」ダイアログボックスで **Net Express\mfsql\demo** フォルダを開きます。「**ファイルの種類**」フィールドを必ず「**COBOL プログラム**」に設定します。
3. **test1.cbl**を選択して [**追加**] をクリックします。 The file and its dependencies are added to the project.

10.3.3 プロジェクトのビルド

プロジェクトのビルド方法は、次のとおりです。

1. [**プロジェクト > すべてをリビルド**] をクリックします。

   『*Net Express の使用方法*』の章の説明のとおり、ソースファイル (ここでは COBOL ファイル) に対して正しいコンパイラが自動的に呼び出されます。 ECM 環境では、各 COBOL ファイルに対して COBOL コンパイラから SQL プリプロセッサが呼び出されます。

   ビルドの終了時には、「リビルドの完了」というメッセージが表示されます。

10.3.4 XDB サーバの起動

XDB サーバを起動する方法は、次のとおりです。

1. [**ツール > SQL for DB2**] をクリックします。
2. [**サーバの起動**] をクリックします。

XDB サーバは、[コントロール パネル] で [**Micro Focus XDB Server for NX**] をクリックしても起動できます。

10.3.5 アプリケーションの実行

アプリケーションの実行方法は、次のとおりです。

1. 「プロジェクト」ウィンドウの右側のペインで **test1.cbl**をクリックして選択します。
2. [**アニメート > 実行**] をクリックします。
3. [**OK**] をクリックします。

   アプリケーションが実行されます。「アプリケーション出力」ウィンドウが開き、アプリケーションからの画面出力が表示されます。 このアプリケーションは、データベースへの従業員レコードの挿入、苗字の取得と表示、および、従業員レコードの削除を実行する簡単なアプリケーションです。 進捗状況に応じてメッセージが表示されます。

次でも「アプリケーション出力」ウィンドウを使用するので、このウィンドウを開いたままにしていてかまいません。

10.3.6 アプリケーションのデバッグ

アプリケーションのデバッグ方法は、次のとおりです。

1. **test1.cbl**が選択されていることを確認し、[**アニメート > アニメーションの起動**] をクリックします。

   アプリケーションの実行時と同じ状態が発生しますが、この場合は、「ソースビュー」ウィンドウが開き、**test1.cbl**のソースが表示されます。 この段階では、「アプリケーション出力」ウィンドウにアプリケーションの最初の画面が表示されていません。これは、COBOL アプリケーションの最初の実行文で実行が一時停止されているためです。

   表示されるソースは、元のソースで前処理後のソースではありません。

2.  を数回クリックします。

   この操作によって、他のアプリケーションと同様に、DB2 アプリケーションをステップ実行できます。 他のアプリケーションに対して使用できるすべてのデバッグ機能を DB2 アプリケーションに適用できます。ここでは、このアプリケーションのデバッグをこの段階で終了し、実行を完了します。

3.  をクリックします。

   デバッグしないでアプリケーションが終了します。 前と同様に「アプリケーション出力」ウィンドウにアプリケーションからメッセージが出力されます。

10.3.7 XDB サーバの終了

完了した後に、XDB サーバをシャットダウンする方法は、次のとおりです。

1. [**ツール > SQL for DB2**] をクリックします。
2. [**サーバの停止**] をクリックします。

10.3.8 XDB Link によるメインフレームへのアクセスの設定

XDB サーバを使用しないでメインフレームに直接接続する方法は、次のとおりです。

1. 次のように、ゲートウェイプロファイルユーティリティを使用して、メインフレームでの DB2 の位置に関する情報のログを取ります。
   1. [**オプション > SQL for DB2 > XDB LINK**] をクリックします。

2. ユーザ **INSTALL** を使用してローカルの XDB サーバにログインします。パスワードは指定しません。
3. [**登録**] をクリックします。
4. メインフレームの DBA と**ヘルプ**情報を参照して、各フィールドに入力します。

2. オプションユーティリティによって、使用する位置とプロトコルを次のように定義します。
   1. [**オプション > SQL for DB2 > クライアント**] をクリックします。
   2. [**セキュリティ**] タブをクリックして、「**クライアントのセキュリティ**」をオンにします。
   3. [**接続**] タブをクリックし、通信プロトコルとして **DRDA** を選択します。
   4. 「**接続位置**」と「**システム位置**」の値が、**GPROF** ユーティリティで指定した値と同じであることを確認します。

10.4 次に進む前に

「ソースビュー」ウィンドウを閉じて、「アプリケーション出力」ウィンドウを非表示にします。

SQL ウィザードを使用して DB2 データベースを維持する方法については、第 11 章『*DB2 データベースの維持*』に進んでください。

作業を中断する場合は、Net Express やプロジェクトを閉じてかまいません。その場合は、次の章に進んだときに、Net Express やプロジェクトを再度開いてください。

第 11 章 : DB2 データベースの維持

注意:この章に記載された機能は、日本語版の Net Express 製品では提供されていません。

チュートリアルは、第 1 章『*はじめに*』の『*チュートリアルマップ*』に表示されている矢印の順に読み進んでください。

このセッションを実行するには、SQL オプションをインストールする必要があります。 ここでは、ユーザが SQL とメインフレーム上の DB2 に精通していることを前提に説明します。

11.1 概要

この章では、 SQL ウィザードを使用して、既存の DB2 データベースに対するテーブルやクエリーの定義と維持を実行します。

具体的には、ウィザードを使用して、組み込みの Tutorial データベースに Pensioner テーブルを追加し、このテーブルでクエリーを作成して実行します。

SQL ウィザードとこのセッションで使用するサンプルデータベースは、プロジェクトに関連付けられていませんが、プロジェクトをあらかじめロードして IDE メニュー機能を使用可能にする必要があります。ロードするプロジェクトでは、「**SQL**」チェックボックスが選択されている必要があります。ここでは、『*DB2 アプリケーション (SQL オプション)*』の章で作成したプロジェクトを使用します。

11.2 準備

このデモでは、『*DB2 アプリケーション (SQL オプション)*』の章で作成してビルドしたプロジェクトを使用します。

1. Net Express を閉じている場合は、再度開きます。「プロジェクト」ウィンドウとその他に開いているウィンドウを閉じます。
2. 『*チュートリアルの概要*』の章で説明されているどれかの方法を使用して、**sqldemo** プロジェクトを開きます。
3. 『*DB2 アプリケーション (SQL オプション)*』の章の『*XDB サーバの起動*』で説明するとおり、SQL サーバを起動します。

11.3.1 SQL ウィザードの起動

SQL ウィザードの起動方法は、次のとおりです。

1. [**ツール > SQL for DB2**] をクリックし、[**SQL ウィザード**] をクリックします。

SQL ウィザードが起動し、図 11-1 のような「カタログ参照」ダイアログボックスが表示されます。

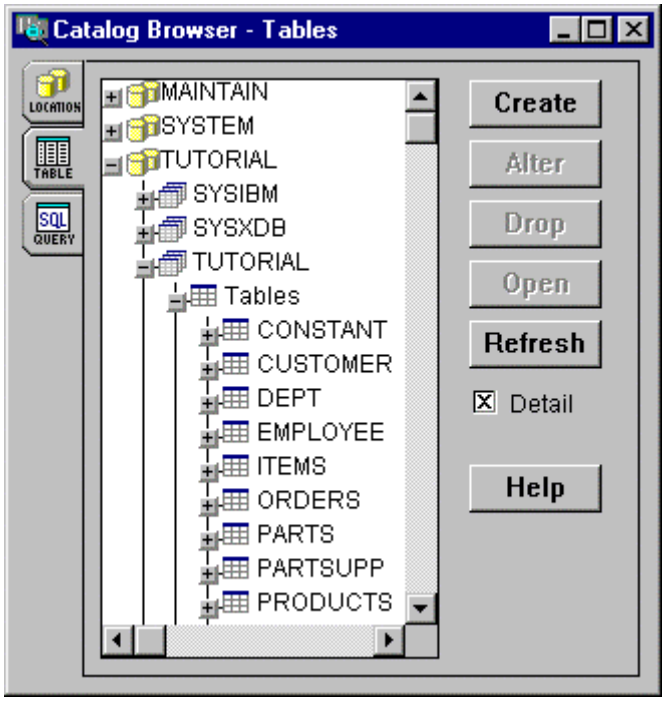


図 11-1：「カタログ参照」ダイアログボックス

11.3.2 テーブルの新規作成

テーブルの作成方法は、次のとおりです。

1. [**テーブル**] タブ  をクリックして、このタブを確実に選択します。
2. [**作成**] をクリックします。
3. 「テーブル名」フィールドに **PENSIONERS** を入力します。
4. このダイアログボックス内のフィールドへ移動するには、Tab キーを押すか、フィールドをクリックします。ダイアログボックスの下部では、テーブルに表示するフィールドを定義します。ここでは、3 つのフィールドを定義します。

   テーブル内の空の行 (「**フィールド名**」というカラムヘッダーから始まる) に、次の値を入力します。

フィールド名	データ型	長さ	NULL 以外	備考
PensionerId	Char	5	Unique	<空白のまま>

行の一番右側のカラムまでスライダを移動すると、「フィールド名」以外のカラムが左側に移動し、「フィールド名」と一番右の「備考」のみが表示されます。

5. **Tab** キーを押して次の行を表示し、次の値を入力します。フィールド名を入力すると、スライダが左側に移動し、すべてのフィールドが表示されます。

フィールド名	データ型	長さ	NULL 以外	備考
PensionerAddress	Varchar	100	True	<空白のまま>

6. 3 番目の行を同じように入力します。

フィールド名	データ型	長さ	NULL 以外	備考
PensionerAge	Varchar	3	False	<空白のまま>

11.3.3 主キーの定義

テーブルの主キーの定義方法は、次のとおりです。

1. 「**指標とキー**」フィールドの右側にある [**作成**] をクリックします。

   「指標の作成」ダイアログボックスが開きます。

2. 「指標名」フィールドに **PensionerId** を入力します。
3. 「非指標カラム」リストボックスの **PensionerId** をクリックし、  をクリックします。

   この操作によって、このフィールドが「指標カラム」リストに移動します。

4. [**型**] の下部にある [**主キー**] をクリックします。

   図 11-2 に示すようなダイアログボックスが表示されます。

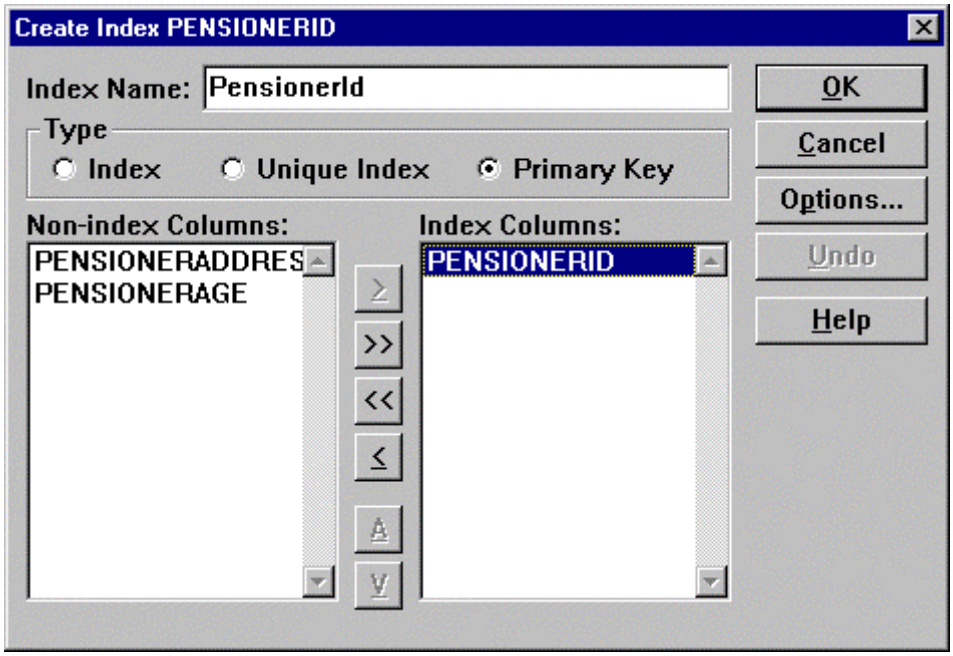


図 11-2 : 主キーの「指標の作成」ダイアログボックス

5. [**OK**] をクリックします。

「**指標とキー**」のリストに主キー「**+PensionerId**」が追加されます。図 11-3 に示すようなダイアログボックスが表示されます。

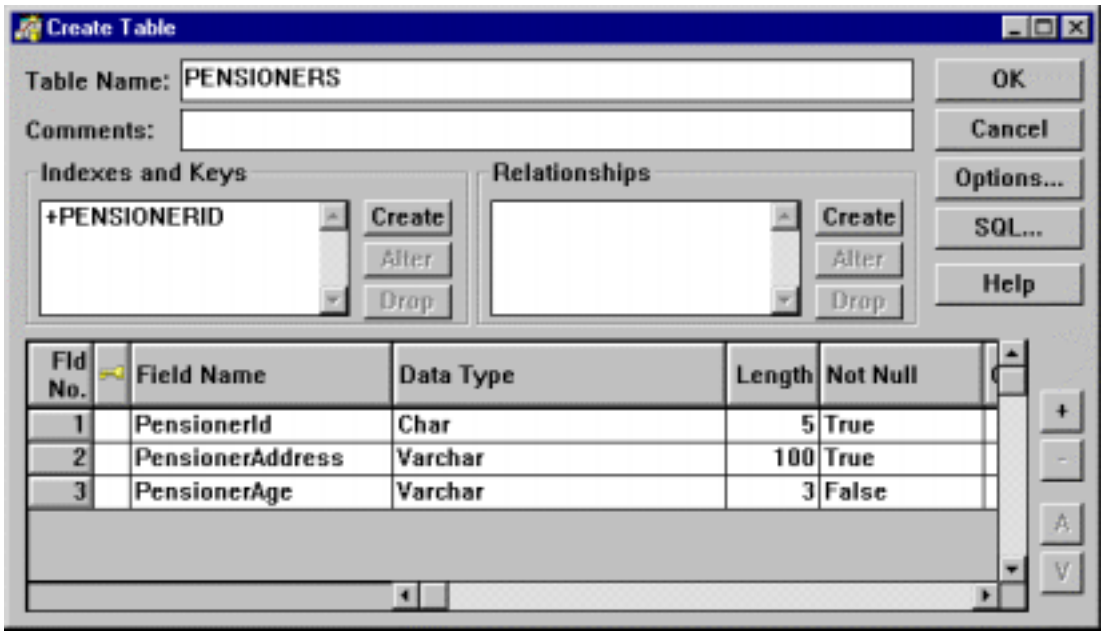


図 11-3 :「テーブルの作成」ダイアログボックス

11.3.4 二次指標の定義

テーブルの二次指標の定義方法は、次のとおりです。

1. 「**指標とキー**」フィールドの右側にある [**作成**] をクリックします。

   「指標の作成」ダイアログボックスが再度開きます。

2. 「指標名」フィールドに **Altindex** を入力します。
3.  をクリックし、「指標カラム」にすべてのフィールドを追加します。

   図 11-4 に示すようなダイアログボックスが表示されます。

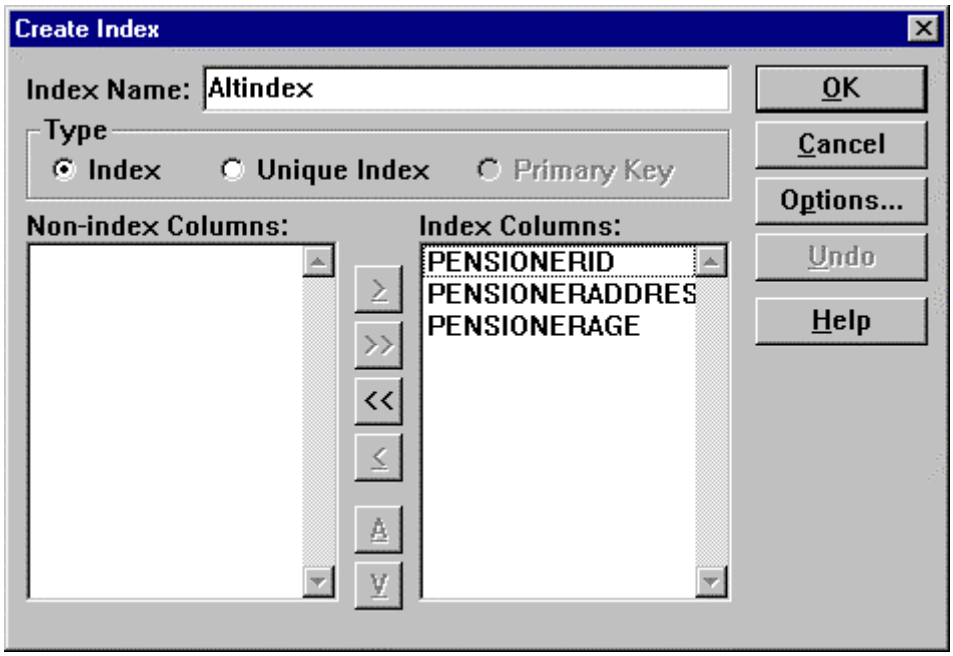


図 11-4：二次指標の「指標の作成」ダイアログボックス

4. [**OK**] をクリックします。

   [**指標とキー**] リストに指標「**+AltIndex**」が追加されます。

11.3.5 SQL の表示

このテーブルを新規作成するための SQL の表示方法は、次のとおりです。

1. [**SQL...**] をクリックします。
2. このビューウィンドウを閉じます。

11.3.6 テーブルの作成の終了

テーブルの作成の終了方法は、次のとおりです。

1. [**OK**] をクリックします。

   You are returned to the Catalog Browser.

2. [**更新**] をクリックします。

   Tutorial データベースのテーブルリストに新しいテーブル PENSIONERS が追加された状態で表示されます。 必要に応じてスライダをプルダウンしてください。

3. テーブルの詳細を展開するには、PENSIONERS という名前のテーブルにある「+」記号をクリックします。「**カラム**」と「**指標**」も同様に展開します。

11.3.7 テーブルへのデータの追加

このテーブルへのデータの保存方法は、次のとおりです。

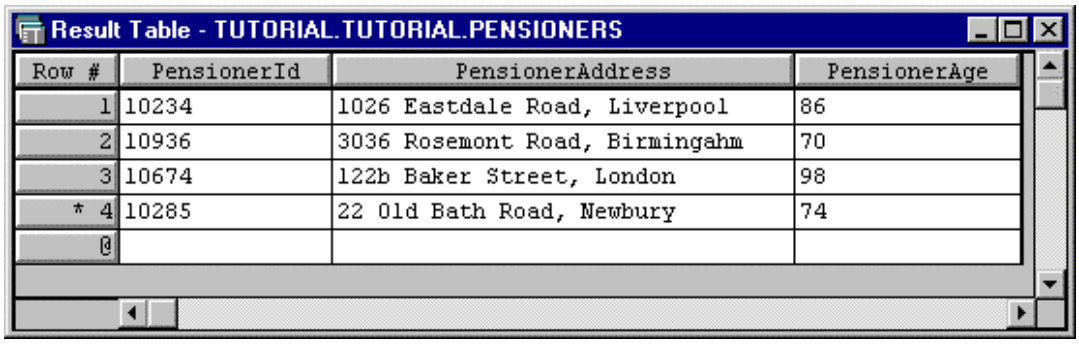
1. 「カタログ参照」の **PENSIONERS** をクリックし、[**開く**] をクリックします。

   「結果テーブル」ウィンドウが表示され、2 つの新しいメニュー [**レコード**] と [**カラム**] が [SQL ウィザード] メニューに追加されます。結果テーブルには、通常、テーブル内のレコードが表示されますが、この段階ではレコードが表示されていません。

2. [**レコード > 編集を許可**] をクリックします。

   この操作によって、テーブルにレコードを入力できるようになります。

3. 図 11-5 のようにレコードを入力します。



Row #	PensionerId	PensionerAddress	PensionerAge
1	10234	1026 Eastdale Road, Liverpool	86
2	10936	3036 Rosemont Road, Birmingahm	70
3	10674	122b Baker Street, London	98
* 4	10285	22 Old Bath Road, Newbury	74
0			

図 11-5 : Pensioners 結果テーブル

4. 「結果テーブル」ウィンドウを閉じます。

11.3.8 クエリーの作成

このテーブルへのクエリーの作成方法は、次のとおりです。

1. [**クエリー**] タブ  をクリックします。
2. [**新規作成**] をクリックします。

   「テーブルの追加」ダイアログボックスが表示されます。

3. 「テーブル名」フィールドの **PENSIONERS** をクリックして、[**追加**] をクリックします。

   「**PENSIONERS(P1)**」ウィンドウが表示されます。

4. [**完了**] をクリックします。
5. 「**PENSIONERS(P1)**」ウィンドウで **PensionerId** と **PensionerAddress** をクリックします。

   これらのエントリがリストに追加されます。

6. **PensionerId** のすぐ下にある「**条件**」フィールドに **10234** を入力し、**Tab** キーを押します。

   エントリの変化に注意してください。

7. **PensionerId** のすぐ下にある「**または**」フィールドに **10674** を入力し、**Tab** キーを押します。

   図 11-6 に示すようなウィンドウが作成されます。

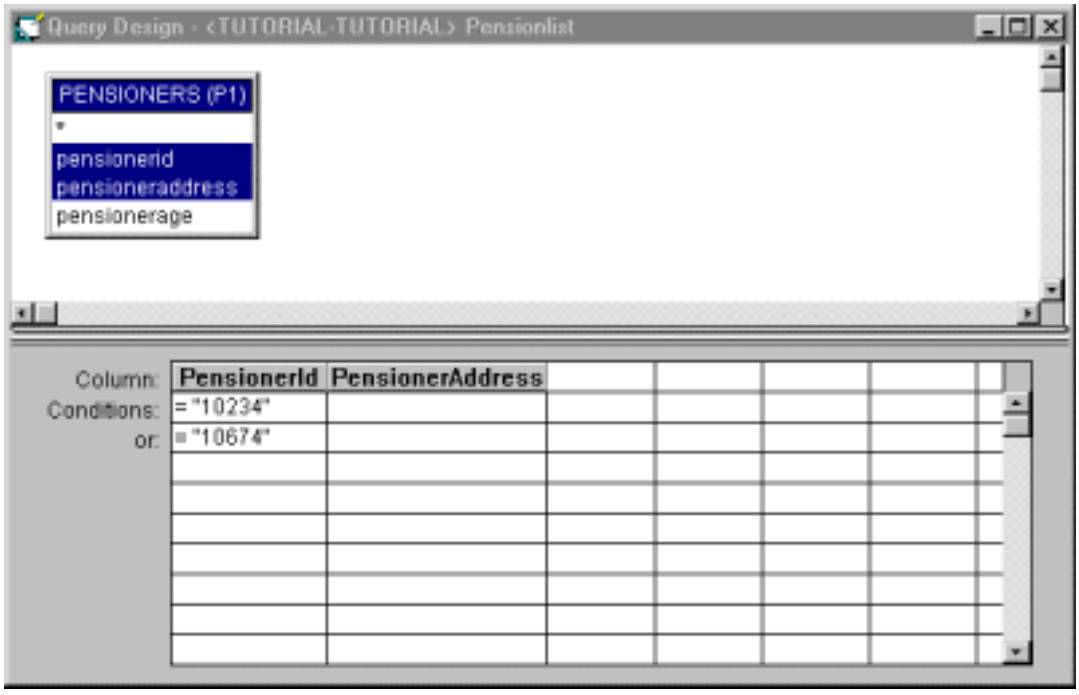


図 11-6：「クエリーデザイン」ウィンドウ

8. SQL ウィザードのメニューで [**ファイル > 名前を付けて保存**] をクリックし、クエリー名として **Pensionlist** を入力し、[**OK**] をクリックします。
9. 「**クエリーデザイン**」ウィンドウを閉じます。
10. [**更新**] をクリックします。

    Tutorial データベースのクエリーリストに新しいクエリー Pensionlist が追加された状態で表示されます。

11.3.9 クエリーの実行

クエリーの実行方法は、次のとおりです。

1. リストの **Pensionlist** をクリックし、[**バッチ**] をクリックします。

   クエリーで指定した条件に一致したレコードが結果テーブルに表示されます。

2. 「結果テーブル」ウィンドウを閉じます。

11.4 次に進む前に

SQL ウィザードを閉じ、SQL サーバを停止します。

一連のチュートリアルは、ここまでです。別のチュートリアルに進むには、第 1 章『*はじめに*』の『*チュートリアル マップ*』を参照してください。

付録 A : Web サーバの構成

Web アプリケーションをテストするには、Web ソフトウェアが必要です。Net Express に付属の Solo をお奨めします。Solo は、次のように自動的に構成されるので、手動で構成する必要がありません。Solo を使用する場合は、この付録を基礎情報として活用してください。他の Web サーバソフトウェアを使用する場合は、次のように構成してください。

Web サーバには、2 つの Web 共有を設定し、アプリケーションが格納されているプロジェクトディレクトリを指定します。一方の Web 共有にはプロジェクトディレクトリを、他方の Web 共有にはプロジェクトビルドディレクトリを指定します。Solo の場合は、起動時にこれらの Web 共有が自動的に設定されます。Web 共有に指定されるディレクトリは、Net Express で最後にロードされたプロジェクトです。他の Web サーバを使用する場合は、次のように構成してください。

プロジェクトのビルドディレクトリは、プロジェクトを作成すると自動的に作成されます。前の説明のとおり、Form Designer などのツールでファイルを作成すると、これらのファイルが適切なディレクトリに自動的に格納されます。

2 つの Web 共有は、次のとおりです。

- **/COBOL/**

  プロジェクトのソースディレクトリ。読み取りアクセス権が設定されています。このディレクトリにアプリケーションの HTML ページを保存します。たとえば、『*Web アプリケーションの作成*』の章で作成した Goform プロジェクトの場合は、**\Program Files\Micro Focus\Net Express\Base\Demo\Goform** です。

- **/cgi-bin/**

  プロジェクトのビルドディレクトリ。読み取りアクセス権と実行アクセス権が設定されています。このディレクトリにアプリケーションのサーバ側実行形式ファイルを保存します。Goform プロジェクトの場合は、**c:\Program Files\Micro Focus\Net Express\Base\Demo\Goform\DEBUG** です。
