



Net Express
インターネット アプリケーション

Micro Focus NetExpress™

インターネット アプリケーション

Micro Focus®

第 3 版

1998 年 10 月

Copyright © 1999 Micro Focus Limited. All rights reserved.

本文書、ならびに使用されている固有の商標と商品名は国際法で保護されています。

Micro Focus は、このマニュアルの内容が公正かつ正確であるよう万全を期しておりますが、このマニュアルの内容は予告なしに随時変更されることがあります。

このマニュアルに述べられているソフトウェアはライセンスに基づいて提供され、その使用および複製は、ライセンス契約に基づいてのみ許可されます。特に、Micro Focus 社製品のいかなる用途への適合性も明示的に本契約から除外されており、Micro Focus はいかなる必然的損害に対しても一切責任を負いません。

Micro Focus® は、Micro Focus Limited の登録商標です。Correlate™、Dialog System™、Form Designer™、Form Express™、Micro Focus COBOL™、NetExpress™、および Solo™ は、Micro Focus Limited の商標です。

Java™ および JavaBeans™ は、Sun Microsystems, Inc. の商標です。

Microsoft®、Windows®、および Windows for Workgroups® は、Microsoft Corporation の登録商標です。

Netscape Communications™、Netscape™、Netscape Navigator™ および Netscape Communications のロゴは、Netscape Communications Corporation の商標です。

Visual Basic™、Transaction Server™ および Windows NT™ は、Microsoft Corporation の商標です。

IBM® は、International Business Machines Corporation の登録商標です。

UNIX® は、X/Open Company Limited の登録商標です。

Copyright© 1987-1999 Micro Focus

All Rights Reserved.

序文

このマニュアルでは、NetExpress のツールである Form Designer、Form Express および Solo を使用してインターネット アプリケーションを作成し、デバッグする方法について説明します。

対象読者

このマニュアルは、インターネットや組織内のイントラネットで実行するためのアプリケーションを作成する COBOL プログラマを対象としています。このマニュアルでは、インターネットやイントラネットの専門的知識は特に必要ありません。ただし、Web ブラウザを使用して WWW や組織内のイントラネットをナビゲートした経験があると、この概念をより簡単に理解することができます。

このマニュアルの使用方法

このマニュアル全体の基本的な概念を理解するには、まず第 1 章をお読みください。NetExpress のインターネットプログラミング ツールに関する簡単な紹介については、第 2 章をお読みください。第 3 章では、ページに含めることができるさまざまなコントロール (HTML、Java アプレット、ActiveX) や Form Designer で使用するさまざまな出力形式の違いについて説明します。

アプリケーションの作成方法については、第 4 章、第 5 章、第 6 章をお読みください。第 4 章では Form Designer で新規アプリケーションを作成する方法を、第 5 章ではデータ アクセス ウィザードを使用する方法を説明します。第 6 章では、Form Express でレガシー アプリケーションを基にしたアプリケーションを作成する方法を示します。Form Express やデータ アクセス ウィザードで作成したアプリケーションは Form Designer で修正できるので、これらのツールを使用する場合にも第 5 章をお読みください。

サーバー側のプログラミングに関しては、第 7 章をお読みください。Form Designer、データ アクセス ウィザード、および Form Express を使用すると、サーバー側のプログラミングを通じてさらに機能を追加できるスケルトン アプリケーションを生成できます。

標準的な CGI プログラムをリコンパイルして ISAPI アプリケーションまたは NSAPI アプリケーションに変換する方法については、第 8 章をお読みください。ISAPI や NSAPI を使用しない場合には、この章をとばしてください。

Form Designer を他の HTML エディタと併用する場合には、第 9 章をお読みください。

フォームにイベント処理を追加する場合には、第 10 章をお読みください。これはオプションです。現在インターネットで実行されているほとんどのアプリケーションでは、フォームのイベント処理を使用しません。これらの技法を使用すると、さらに洗練されたアプリケーションを作成できます。また、第 11 章で説明する関数を使用して、クライアント側の確認機能をフォームに追加することもできます。

最終的に Web サーバーでアプリケーションを実装する場合、第 12 章をお読みください。

表記規則

このユーザー ガイドでは、次の書体や規則を使用します。

- 入力するテキストは、次のように表示します。

```
cat script_name | more
```

斜体テキストはコマンドの一部として入力する変数を表します。

- コマンド行やコード例でオプション入力するテキストは、角かっこ ([[]]) で囲みます。次の式では、オプションの単語 NOT を入力しない場合、*column_name* は *pattern_value* に設定されます。一方、NOT を入力した場合、*column_name* は *pattern_value* 以外に設定されます。

```
column_name [NOT] LIKE pattern_value
```

- 特定のモデルやオペレーティング システムだけに適用する項や段落については、段落のすぐ前に太字斜体の項目名を記載します。たとえば、次のようになります。

UNIX

この段落は UNIX システムにのみ適用されます。

目次

序文.....	ii
対象読者.....	ii
このマニュアルの使用方法.....	ii
表記規則.....	iii
第1章 インターネット プログラミングの概要.....	1-1
1.1 概要.....	1-1
1.2 インターネット アプリケーションの内容.....	1-2
1.2.1 複雑なアプリケーション.....	1-3
1.3 実行フロー.....	1-5
第2章 インターネット アプリケーションの開発.....	2-1
2.1 概要.....	2-1
2.2 支援ツール.....	2-1
2.2.1 HTML ページの作成.....	2-1
2.2.2 サーバー側プログラムの作成.....	2-2
2.2.3 インターネット アプリケーションのデバッグとテスト.....	2-2
第3章 フォームと HTML.....	3-1
3.1 概要.....	3-1
3.2 コントロールとデータ.....	3-1
3.3 HTML、Java および ActiveX.....	3-2
3.4 フォームの出力型.....	3-3
第4章 新規アプリケーションの作成.....	4-1
4.1 概要.....	4-1

4.2 Form Designer.....	4-1
4.2.1 Form Designer のウィンドウ	4-2
4.2.2 配置編集とフロー編集	4-4
4.2.3 プロパティとリンク データ項目	4-4
4.3 フォームとサーバー側プログラムの作成	4-5
4.3.1 フォームのペイント例	4-7
4.4 スケルトン アプリケーションの実行	4-11
4.4.1 サーバー側プログラムのビルド	4-11
4.4.2 アプリケーションの実行	4-11
4.4.3 アプリケーションのデバッグ	4-12
4.4.4 例	4-13
4.5 非対称サーバー側プログラムの作成	4-13
4.5.1 例	4-14
4.6 サーバー側のスケルトン プログラムへの機能追加	4-16
4.6.1 例	4-17
第5章 データ アクセス アプリケーション	5-1
5.1 概要	5-1
5.2 データ アクセス アプリケーションの作成	5-2
5.2.1 アプリケーションの生成	5-2
5.2.2 基本アプリケーションの実行	5-5
5.2.3 第 2 のフォーム セットの生成	5-6
5.2.4 アプリケーションの修正	5-7
5.2.4.1 フォームの変更	5-7
5.2.4.2 サーバー側プログラムの修正	5-10
5.2.4.3 新規アプリケーションの実行	5-12

第6章 レガシー コードの再使用.....	6-1
6.1 概要.....	6-1
6.2 新しいインターネット アプリケーションの作成.....	6-2
6.3 データの選択.....	6-3
6.3.1 エントリ フィールドへのフィールド割り当て.....	6-6
6.3.2 オプション ボタンへのフィールド割り当て.....	6-6
6.3.3 チェック ボックスへのフィールド割り当て.....	6-6
6.3.4 COBOL テーブルの割り当て.....	6-7
6.3.5 選択コントロールへのフィールド割り当て.....	6-7
6.4 アプリケーションの生成.....	6-7
6.5 アプリケーション例の作成.....	6-8
6.5.1 アプリケーション例のテスト.....	6-11
6.6 アプリケーションの拡張.....	6-13
6.6.1 COBOL プログラムの編集.....	6-13
6.6.2 フォームの編集.....	6-14
6.6.3 アプリケーション例の拡張.....	6-14
第7章 サーバー側のプログラミング.....	7-1
7.1 概要.....	7-1
7.2 リソースの競合.....	7-1
7.3 サーバー側プログラムへの入力.....	7-2
7.3.1 構文.....	7-3
7.3.2 例.....	7-3
7.4 サーバー側プログラムからの出力.....	7-3
7.4.1 EHTML の使用方法.....	7-4
7.4.1.1 代入マーカー.....	7-5

7.4.1.2 EHTML プリプロセッサ指令	7-7
7.4.2 DISPLAY 文の使用法	7-7
7.4.2.1 例	7-8
7.5 アプリケーションの状態の維持	7-9
7.5.1 非表示フィールド	7-10
7.5.2 cookie	7-11
7.5.2.1 cookie の例	7-12
7.5.3 サーバー側の状態機構	7-13
7.5.3.1 クライアント状態レコードの保存と検索	7-14
7.5.3.2 クライアント状態レコードの削除	7-22
第8章 CGI、ISAPI、および NSAPI プログラム	8-1
8.1 概要	8-1
8.2 ISAPI および NSAPI サーバー側プログラムの作成	8-2
8.2.1 ISAPI 用コンパイラ指令の設定	8-2
8.2.2 NSAPI 用コンパイラ指令の設定	8-3
8.2.3 ISAPI および NSAPI プログラムのリンク	8-4
第9章 Form Designer による出力の編集	9-1
9.1 概要	9-1
9.1.1 Form Designer への HTML ページの転記	9-1
9.1.2 Form Designer からの HTML ページの転記	9-2
第10章 クライアント側のプログラミング	10-1
10.1 概要	10-1
10.2 JavaScript の概要	10-2
10.3 スクリプト アシスタント	10-2
10.3.1 [イベント] タブ、[メソッド] タブ、[プロパティ] タブ、および [スタイル] タブ	10-3

10.3.1.1	オブジェクト ビュー	10-7
10.3.1.2	イベント ビュー	10-7
10.3.1.3	メソッド ビュー	10-7
10.3.1.4	プロパティ ビュー	10-8
10.3.1.5	スタイル ビュー	10-8
10.3.1.6	イベント ハンドラ ビュー	10-9
10.3.1.6.1	HTML コントロールのイベント ハンドラ	10-9
10.3.1.6.2	ActiveX コントロールのイベント ハンドラ	10-10
10.3.2	[スクリプト] タブ	10-10
10.4	クロスプラットフォーム互換性	10-12
10.5	例	10-12
10.5.1	例 1 プロパティを設定するイベント ハンドラの追加	10-12
10.5.2	例 2 プロパティを取得するためのイベント ハンドラの追加	10-14
10.5.3	例 3 配色を設定するためのイベント ハンドラの追加	10-16
第11章	フォームの確認	11-1
11.1	組み込みの確認関数	11-1
11.2	確認例	11-2
11.2.1	例 1 オプションの小数を確認する方法	11-2
11.2.2	例 2 必須の Visa カード番号を確認する方法	11-3
11.2.3	例 3 必須フィールドへのデータ入力確認	11-3
11.3	ユーザー定義確認関数の追加	11-4
第12章	アプリケーションの実装	12-1
12.1	概要	12-1
12.1.1	他の Web サーバーを使用したデバッグ	12-1
12.1.2	作業例	12-1

12.2 サポートされているサーバー	12-2
12.3 実装とデバッグに関するガイド	12-2
12.3.1 実装手順ガイド (Windows サーバー).....	12-3
12.3.2 実装手順ガイド (UNIX サーバー).....	12-4
12.3.3 デバッグ手順ガイド	12-5
12.3.4 ランタイム システムの選択	12-6
12.3.5 Web サーバーの設定.....	12-7
12.3.6 実装用コピーの作成	12-9
12.3.7 アプリケーションの URL 変更	12-10
12.3.8 実装済みアプリケーションへの sstate の追加.....	12-11
12.3.9 アプリケーションのリビルド	12-12
12.3.10 NSAPI アプリケーションを実行する前に	12-17
12.3.10.1 NSAPI サーバー構成ファイルの修正	12-18
12.3.11 ISAPI に準拠した Object COBOL アプリケーションの実装.....	12-20
12.4 アプリケーションの実装	12-22
12.4.1 ISAPI および NSAPI アプリケーションにおける従属関係の検索.....	12-23
12.4.2 ODBC データ ソースの使用	12-25
12.5 UNIX へのアプリケーションのパブリッシュ.....	12-25
12.6 アプリケーションのデバッグ	12-26
12.6.1 CGI プログラムのアニメート	12-26
12.6.2 ISAPI および NSAPI プログラムのアニメート	12-28
付録A: WWW の概要.....	A-1
A.1 Web と商取引	A-2
A.1.1 イン트라ネット	A-2
A.1.2 エクストラネット	A-2

A.2 Web とクライアント サーバー	A-2
A.3 メインフレームの復活	A-3
A.4 Web と NetExpress	A-3
A.5 Web が機能する原理	A-4
A.6 Web ブラウザ	A-5
A.7 URL	A-5
A.7.1 Web サイト	A-5
A.7.2 ドキュメント URL	A-6
A.8 URL の分析	A-6
A.8.1 プロトコル	A-6
A.8.2 ホスト コンピュータ	A-6
A.8.3 パス	A-7
A.8.4 オプションのインターネット ポート	A-7
A.9 HTML	A-8
A.9.1 HTML タグ	A-8
A.9.2 HTML の概要	A-8
A.9.2.1 ドキュメント構造	A-8
A.9.3 タイトルと見出し	A-9
A.9.3.1 段落	A-9
A.9.3.2 リスト	A-9
A.10 Web のマルチメディア	A-9
A.10.1 グラフィックス	A-10
A.10.2 オーディオ	A-10
A.10.3 ビデオ	A-10
A.10.4 アプレット	A-10

A.10.5 3-D バーチャル リアリティ	A-10
A.11 Java.....	A-10
A.11.1 COBOL および Java.....	A-11
A.12 HTML フォーム: レガシー プログラムへのゲートウェイ.....	A-11
A.13 Common Gateway Interface (CGI).....	A-11
A.14 NetExpress および CGI.....	A-12
A.15 詳細情報.....	A-13
付録B: 実装例とデバッグ例	B-1
B.1 概要.....	B-1
B.1.1 Microsoft Internet Server での ISAPI の実装.....	B-1
B.1.1.1 Internet Server での Web 共有リソースの設定.....	B-2
B.1.1.2 実装可能なバージョンのアプリケーション作成.....	B-2
B.1.1.3 アプリケーションの実装.....	B-4
B.2 別の Web サーバーでのデバッグ.....	B-4
B.2.1 Microsoft Internet Server での CGI プログラムのアニメート.....	B-5
B.2.2 Microsoft Internet Server での ISAPI プログラムのアニメート.....	B-7
B.2.3 NSAPI プログラムのアニメート.....	B-11

第1章 インターネット プログラミングの概要

この章では、インターネット アプリケーションの内容とそのさまざまな構成要素について説明します。

1.1 概要

インターネット アプリケーションは、標準的なインターネット プロトコルを使用してサーバーにクライアントを接続するクライアント サーバー アプリケーションです。ここでいうインターネット アプリケーションは、WWW で公に使用できるインターネット用アプリケーションやイントラネット用アプリケーションと、まったく同じ方法で作成することができます。イントラネット アプリケーションは組織のイントラネットで実行されるもので、組織のスタッフだけが使用することができます。このマニュアルで使用する「インターネット アプリケーション」という用語は、純粋なインターネット用アプリケーションまたはイントラネット用アプリケーションを指します。WWW の概要については、付録の WWW の概要 を参照してください。

インターネット アプリケーションは、"Thin client" と "Thick Server" で構成されます。エンドユーザーが対話に使用するクライアント側で必要なのは、ユーザー インターフェイスだけです。クライアントは、Web ブラウザで実行されます。Web ブラウザは、インターネットにアクセスするための標準ツールです。すべての処理は、サーバー側で行われます。このサーバー側に組織のデータが置かれます。

インターネット アプリケーションでは、クライアントとサーバー間の通信に標準的なインターネット プロトコルを使用するため、アプリケーションをクロスプラットフォームにすることができます。サーバー側プログラムを、Micro Focus COBOL で作成すると、Windows NT や UNIX サーバーで実行することができます (ただし、UNIX でアプリケーションを実行するには、Micro Focus COBOL for UNIX を購入する必要があります)。

インターネット アプリケーションのサーバー側プログラムは、マシンの Web サーバー ソフトウェアを介してクライアントと通信します。COBOL プログラムとこれを実行する Web サーバーの間のインターフェイスをプログラムする必要はありません。次にあげる 3 つの業界標準の Web サーバー インターフェイスをそのまま (コードを変更せずに) 使用することができます。

- Common Gateway Interface (CGI)
- Internet Server Application Program Interface (ISAPI)
- Netscape Server Application Program Interface (NSAPI)

上記のインターフェイスについては、「CGI、ISAPI および NSAPI プログラム」の章で詳しく説明します。デフォルトの NetExpress でアプリケーションを作成すると、CGI 用 (CGI は、すべての Web サーバーでサポートされています) にビルドされるので、CGI を使用して開発とデバッグを行うことをお勧めします。NetExpress のコンパイラ設定とビルド設定を変更し、プログラムをリビルドすると、COBOL CGI プログラムを ISAPI プログラムまたは NSAPI プログラムに変換することができます。

クライアント側のユーザー インターフェイスは、以下の構成要素を使用して作成することができます。

- 標準的な HTML フォーム。フォーム対応の Web ブラウザで使用可能なため、Windows、UNIX、Macintosh または OS/2 を使用するエンドユーザーで実装できます。
- ActiveX コントロール。ActiveX 対応の Web ブラウザで使用することができます (ActiveX は、現在、32 ビットの Windows プラットフォームでしか使用できません)。
- Java アプレット。Java 対応の Web ブラウザで使用することができます (インターネット エクスプローラ と Netscape Navigator を含みます)。

詳細については、「フォームと HTML」の章を参照してください。

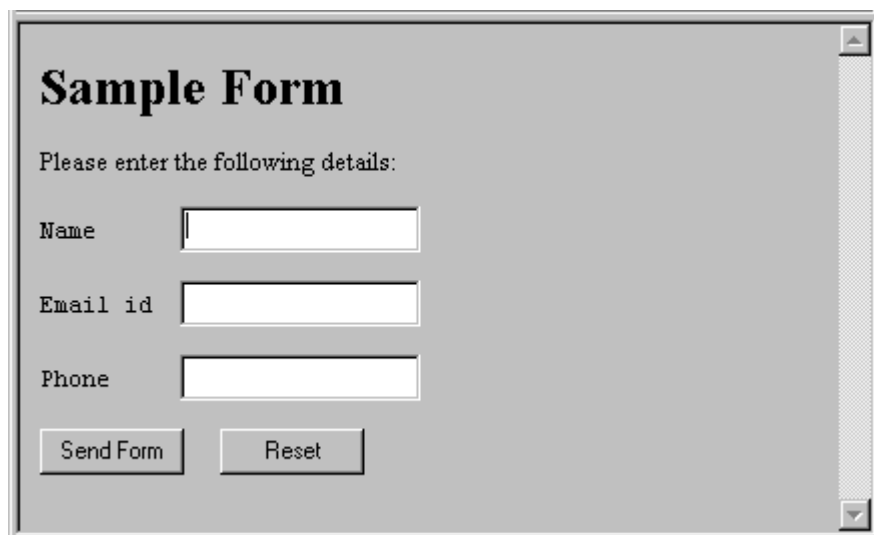
Form Designer を使用すると、クライアント側の機能を追加するためのスクリプトを簡単に作成することができます。共通確認関数を実行したり (「フォームの確認」の章を参照)、スクリプト アシスタントを使用してスクリプトを追加することができます (「クライアント側プログラミング」の章を参照)。

1.2 インターネット アプリケーションの内容

インターネット アプリケーションは、クライアント サーバーアプリケーションです。これは、次の 2 つに分割することができます。

- フォーム
- サーバー側プログラム

フォームは、エンドユーザー側で表示される部分です。フォームは、Web ブラウザで表示され、エンドユーザーがデータを入力するためのコントロールを装備しています。下図は、フォーム例を示します。



The image shows a web browser window displaying a form titled "Sample Form". The form has a title bar and a scroll bar on the right. The content of the form is as follows:

Sample Form

Please enter the following details:

Name

Email id

Phone

図 1-1 インターネット アプリケーションのフォーム例

上記のフォームでは、エンドユーザーが [Send Form] ボタンをクリックすると、フォームに入力された情報がパッケージ化され、サーバー側プログラムに送信されます。サーバー側プログラムは、フォーム、Web ページのリンクなどにより起動された場合にだけ実行されます。サーバー側プログラムは、フォームの情報を処理し、エンドユーザーにページを返します。

結果は、プログラムの内容に応じて、返されたフォームにテキストなどの形式で表示されます。

1.2.1 複雑なアプリケーション

上記のアプリケーション例は、非常に簡単なものです。実際のアプリケーションは、通常、より複雑です。また、互いにリンクされた複数のフォームやサーバー側プログラムで構成されている場合もあります。サーバー側プログラムは、次の 2 種類に分類されます。

- 対称

対称サーバー側プログラムでは、入力と出力に対して同じフォームを使用します。たとえば、データベースの問い合わせプログラムや更新プログラムでは、レコードや SQL クエリーのためのフィールド セットを使用します。データベースに問い合わせるデータの入力とプログラムから返される結果の出力は、これらの同じフィールドを使用して行います。

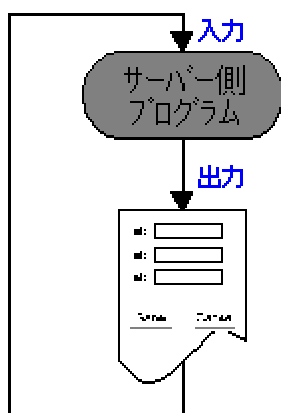


図 1-2 対称サーバー側プログラム

- 非対称

非対称サーバー側プログラムでは、入力と出力に異なるフォームを使用します。たとえば、最初のフォームに顧客の詳細情報を入力すると、新規注文用の別のフォームが表示される注文入力プログラムなどがあげられます。

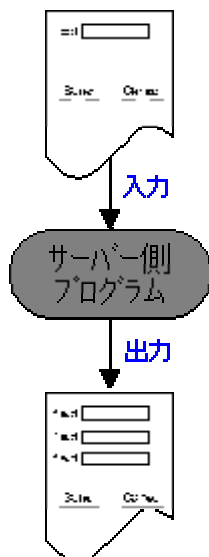


図 1-3 非対称サーバー側プログラム

非対称サーバー側プログラムを作成する場合、別個のフォームとサーバー側プログラムを連鎖させて、複雑なアプリケーションをビルドすることができます。下図は、最初のプログラムの出力を受けて 2 番目のサーバー側プログラムが起動されるアプリケーションを示します。2 番目のプログラムは、処理パスに応じて異なるフォームを出力します。

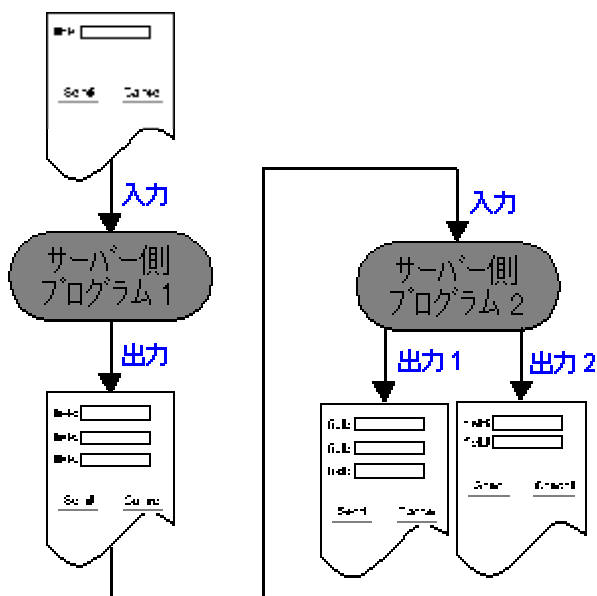


図 1-4 複雑なアプリケーション

注記:

Web ブラウザで表示できるのは、HTML ページです。HTML ページには、複数のフォームを含めることができます。ただし、サーバー側プログラムは、複数のフォームから入力を受け取ることができません。サーバー側プログラ

ムは、ブラウザにページを返します。このページにも複数のフォームを含めることができます。

サーバー側プログラムに対する入力の基本単位は、常にフォームです。また、出力の基本単位は常にページです。ほとんどの場合、ページに含まれるフォームは 1 つです。アプリケーションの作成中はページよりフォームでの作業が多くなるため、上図ではページだけを記載してあります。

1.3 実行フロー

インターネット アプリケーションは、従来の CICS アプリケーションに似ています。フォームを BMS 入力画面に、サーバー側プログラムを CICS トランザクションに置き換えて考えると、コントロールのフローは非常に似ています。CICS トランザクションと同様、サーバー側プログラムは、データを処理し、結果を返す間しか実行されません。複雑な CICS アプリケーションは、複数の入力画面と複数のトランザクションで構成されます。エンドユーザーからはアプリケーションが常に実行されているように見えますが、実際のアプリケーションは、短時間だけ実行されて終了するトランザクションが連続した状態で作成されています。

対称サーバー側プログラム（この用語については、前項を参照してください）に基づいたアプリケーションは、次のように機能します。

1. エンドユーザーが、サーバー側プログラムを起動するリンクを Web ページでクリックします。
2. サーバー側プログラムが実行され、アプリケーションのフォームを含む HTML ページを返します。
3. エンドユーザーは、フォームに記入後、ボタンをクリックしてこのフォームを送信します。

この操作により、サーバー側プログラムが再度実行されます。

4. サーバー側プログラムは、データを取得し、フォームでエンドユーザーのブラウザへ返します。

上記の手順以外にも、さまざまな流れがあります。たとえば、NetExpress で提供する CGI アプリケーション例 (netexpress¥base¥demo¥cgi¥cgiprg1.appに格納されています) を起動すると、簡単な HTML ページを返す CGI プログラムを実行する入力フォームが表示されます。前項で簡単に説明した非対称サーバー側プログラムは、より複雑な例です。重要なことは、サーバー側プログラムは結果を返すまでの間しか実行されないということです。

第2章 インターネット アプリケーションの開発

この章では、インターネット アプリケーションを作成するための開発手順の概要と、NetExpress ツールを利用して迅速なインターネット アプリケーションを開発する方法について説明します。

2.1 概要

NetExpress では、すぐにインターネット アプリケーションを作成することができます。NetExpress には、次の作業に利用できるツールが含まれています。

- HTML ページとサーバー側プログラムの作成
- インターネット アプリケーションのサブルーチンの使用とレガシー コードの再使用
- アプリケーションのデバッグとテスト

NetExpress でインターネット アプリケーションを開発する場合、3 種類のアプローチ方法があります。

- Form Designer とインターネット アプリケーション ウィザードを使用して、最初から新規アプリケーションを開発する（「新規アプリケーションの作成」の章で説明しています）。
- 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、既存のサブルーチンを再使用する（「レガシーコードの再使用」の章で説明しています）。
- 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、SQL データベースにアクセスするアプリケーションを開発する（「データ アクセス アプリケーション」の章で説明しています）。

2.2 支援ツール

この項では、インターネット アプリケーションの作成を支援する NetExpress の 2 つのツールについて簡単に紹介します。

2.2.1 HTML ページの作成

Form Designer を使用すると、インターネット アプリケーションのユーザー インターフェイスを作成することができます。Form Designer は、WYSIWYG の HTML エディタと HTML フォーム ペインタです。Form Designer では、標準的な HTML フォーム コントロール、ActiveX コントロール、Java アプレットを使用することができます。

Form Designer では、ページの各 HTML フォーム コントロールに対して COBOL データ型を指定することができます。アプリケーションのサーバー プログラムを作成するときに、ページの各コントロールについてデータ項目が宣言されます。Form Designer の基礎知識については、『入門書』の「具体的な手順」を参照してください。

Form Designer でインターネット アプリケーションを作成する方法は、次のとおりです。

1. アプリケーションのファイルをすべて格納する NetExpress プロジェクトを作成します。
2. Form Designer を起動し、作成する HTML ページの種類を入力して (既存の HTML ページをテンプレートとして使用することもできます)、ページにフォームをペイントします。
3. 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、サーバー側のスケルトン プログラムを作成します。
4. NetExpress の IDE を使用して、サーバー側プログラムを編集し、機能を追加します。
5. プログラムをビルドします。
6. Solo Web サーバーでアプリケーションのテストとデバッグを行います (アプリケーションのテストとデバッグにはどの Web サーバーでも使用できますが、Web サーバーの構成や使用に精通していない場合は、Solo を使用すると便利です)。
7. Form Designer でフォームを編集し、調整します。このとき、フォーム コントロールに対する変更を反映するために、コード ジェネレータによりサーバー側プログラムのデータ宣言が変更されます。ただし、手動で追加したアプリケーション コードはまったく変更されません。
8. サーバー側プログラムを編集し、調整します。

Form Designer の使用方法の詳細については、「新規アプリケーションの作成」の章を参照してください。

2.2.2 サーバー側プログラムの作成

インターネット アプリケーション ウィザードでは、インターネット アプリケーションのサーバー側プログラムを COBOL で作成します。これらのプログラムの作成方法は、3 種類あります。

- Form Designer を使用してペイントしたフォームでサーバープログラムを作成する。
- 既存のコードからサーバー プログラムとフォームを作成する。
- データベースからサーバー プログラムとフォームを作成する。

インターネット アプリケーション ウィザードの基礎知識については、『入門書』の「具体的な手順」を参照してください。

2.2.3 インターネット アプリケーションのデバッグとテスト

Solo Web サーバーを使用すると、開発に使用したマシンでインターネット アプリケーションをデバッグできます。Solo は、NetExpress で機能するように設計されており、作業中のプロジェクトと同じフォルダを使用するように自動的に構成されます。Solo を使用する場合、セットアップや Web サーバーの実行に関する知識は必要ありません。

Solo の基礎知識については、『入門書』の「具体的な手順」を参照してください。また、Solo については、NetExpress のオンラインヘルプでも説明されています。[ヘルプ] メニューの [ヘルプ トピック] をクリックし、ヘルプの [目次] で [プログラミング]、[インターネット アプリケーションのプログラミング]、[インターネット アプリケーションのデバッグ] の順に選択します。

Solo は、プロダクション Web サーバーとして使用することを目的としていません。実際に使用するアプリケーションを実装する準備ができたなら、市販の Web サーバー ソフトウェアを使用してください。また、アプリケーションのユニット テストも、使用する Web サーバー ソフトウェアで行う必要があります。

「アプリケーションの実装」の章では、Web サーバーにアプリケーションを移植する方法について説明します。

第3章 フォームと HTML

この章では、HTML フォームの基本的な概念について説明します。また、Form Designer のさまざまな出力オプションについても説明します。

3.1 概要

フォームは、アプリケーションのユーザー インターフェイスです。最も簡単なフォームとしては、いくつかのコントロール（入力フィールド、オプション ボタンやチェック ボックスなど）と、フォームを送信するためのプッシュ ボタンを含むものがあげられます。フォームは、送信されると、Web サーバーに対してサーバー側プログラムの起動要求を送信し、フォームに入力されたデータをまとめて、サーバーに送信します。

このように簡単なフォームを使用する場合、GUI やイベント駆動型プログラミングに関する知識はまったく必要ありません。フォームは、Form Designer でペイントすることができるため、COBOL 以外の知識は必要ありません。

エンドユーザーがデータを入力している間にコントロールが互いに対話するような、より複雑なフォームをデザインすることもできます。たとえば、特定のオプション ボタンを選択しない限り、特定のフィールドを使用禁止にすることができます。このようなフォームを使用する場合、HTML Web ページのスクリプト言語である JavaScript を学習する必要があります。この場合も、イベント ハンドラを簡単にセット アップしてフォームのイベントに接続できる Form Designer の スクリプト アシスタントが役立ちます。イベント ハンドラのセット アップについては、「クライアント側のプログラミング」の章で説明します。

3.2 コントロールとデータ

フォームの各コントロールには名前と値があります。エンドユーザーがフォームを送信すると、フォームの情報は、サーバー側プログラムに一对の名前と値として送信されます。COBOL 構文の拡張機能を使用すると、フォームのコントロールの名前を、サーバー側プログラムの COBOL データ項目に直接、関連付けることができます。フォームのデータがサーバー側プログラムにポストされると、データ項目はフォームのコントロールの値に設定されます。この方法については、「サーバー側プログラミング」の章で説明します。

COBOL と Form Designer では、現在、次の種類のコントロールを簡単に使用できるようなサポートを提供していません。

- 入力フィールド
- 選択ボックス
- オプション ボタン
- チェック ボックス

- プッシュボタン
- 入力イメージ (HTML 4.0 のみ)

これらは、HTML フォームで使用できる基本的なコントロールです。フォームのペイントについては、「新規アプリケーションの作成」の項で説明します。

3.3 HTML、Java および ActiveX

HTML ページには、標準的な HTML フォームコントロール以外に、ActiveX コントロールと Java アプレットを追加することもできます。ActiveX コントロールと Java アプレットは、ページの対話性を向上させるためのもう 1 つの方法です。下表は、ActiveX コントロールと Java アプレットの主な違いをまとめたものです。これらの技術はインターネット上で使用することもできますが、企業イントラネットに適しています。ただし、その場合、企業イントラネットの全ユーザーに正しいブラウザがインストールされていることが必要です。

	ActiveX コントロール	Java アプレット
プラットフォーム	Windows プラットフォームのインターネット エクスプローラだけでサポートされます。	複数のオペレーティング システムのインターネット エクスプローラと Netscape Navigator でサポートされます。 ブラウザによっては、Java の全機能をサポートしていないものがあります。
セキュリティ	ユーザーは、ブラウザで ActiveX コントロールの無効化を選択できます。また、ユーザーは、ActiveX コントロールを個別に無効化することができます。ActiveX コントロールについては、オリジネータを確認するためのデジタル署名をすることができます。	ユーザーは、ブラウザで Java アプレットの無効化を選択できます。Java には、アプレットが、実行されている PC の情報を修正したり、読み込んだりしないようにする厳密なセキュリティ モデルもあります。
機能性	完全な Windows API にアクセスすることができます。そのため、ActiveX コントロールは、PC で実行されている他のプログラムやアプリケーションのどの関数でも実行することができます。	Java を使用すると、非常に洗練された GUI アプリケーションをビルドすることができます。ただし、セキュリティ モデルにより Java アプレットの実行内容は制限されます。

Form Designer を使用すると、ActiveX コントロールと Java アプレットを HTML コントロールと同じようにページに追加することができます。また、Form Designer の プロパティ エディタ を使用すると、ActiveX コントロールと Java アプレットにプロパティを設定することもできます。

3.4 フォームの出力型

Form Designer とインターネット アプリケーション ウィザードでは、ページのフォーム配置情報を記録するための HTML フォーマットを選択することができます。

- クロスプラットフォーム出力

フォームのすべての要素を正確に配置するために、フォームを HTML テーブルで出力します。

- ダイナミック HTML

フォームの各要素の正確なサイズと位置をマークするために、スタイル属性を使用します。

クロスプラットフォーム出力は、さまざまなブラウザでサポートされています。ダイナミック HTML 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 でしかサポートされていません。

ページ ウィザード (Form Designer の場合) とインターネット アプリケーション ウィザードを使用すると、最初にフォームを作成するときに使用するフォーマットを選択することができます。また、Form Designer で HTML ページを開き、「ページのプロパティ」を変更して、再度保存すると、フォーマットの切り替えができます。「ページのプロパティ」ダイアログ ボックスは、[ページ] メニューの [プロパティ] をクリックすると表示されます。

第4章 新規アプリケーションの作成

この章では、Form Designer でインターフェイスをペイントし、インターネット アプリケーション ウィザードでサーバー側のスケルトン プログラムを生成して、新規のインターネット アプリケーションを作成する方法について説明します。

4.1 概要

NetExpress を使用すると、非常に簡単にインターネットやイントラネットのアプリケーションを作成することができます。NetExpress では、必要なフォームをペイントしたり、サーバー側プログラムを自動作成することができます。インターネット アプリケーション ウィザードにより生成されるスケルトン コードは、フォームからのデータを受け付け、COBOL データ項目に格納します。また、これらのデータ項目の情報をフォームに出力します。この場合、入力と出力の間でデータを処理するコードを追加する必要があります。

インターネット アプリケーション ウィザードでは、2 種類のスタイルのプログラムを作成することができます。

- 対称 - 入力として受け付けたフォームを出力として返します。
- 非対称 - 入力フォームとは別のフォームが出力されます。

この章で説明するアプリケーションの作成手順は、次のとおりです。

1. フォームをペイントします。
2. スケルトン アプリケーションを実行します。
3. 非対称サーバー側プログラムを作成します (アプリケーションによっては、非対称サーバー側プログラムを使用しないものがあります)。
4. サーバー側プログラムに機能を追加します。

各過程で、簡単な実行例を使用して説明します。この章では、「ここで [OK] ボタンをクリックしてください。」というように詳細までは説明しません。全体的な過程について説明します。Form Designer と NetExpress の詳しい手順ごとの指示については、オンライン ヘルプを参照してください。この章を読み始める前に Form Designer や NetExpress の概要が必要な場合は、『入門書』の「具体的な手順」を参照してください。

4.2 Form Designer

この項では、Form Designer について紹介します。NetExpress では、新しい HTML ページを作成するときや既存の HTML ページを編集するときに Form Designer が起動されます。この項の 3 つのサブ項目では、次の内容を説明します。

- Form Designer のウィンドウ
- 配置編集とフロー モード編集
- プロパティとリンク データ項目

4.2.1 Form Designer のウィンドウ

次の画面表示は、NetExpress の Form Designer で HTMLページを開いた状態を示します。

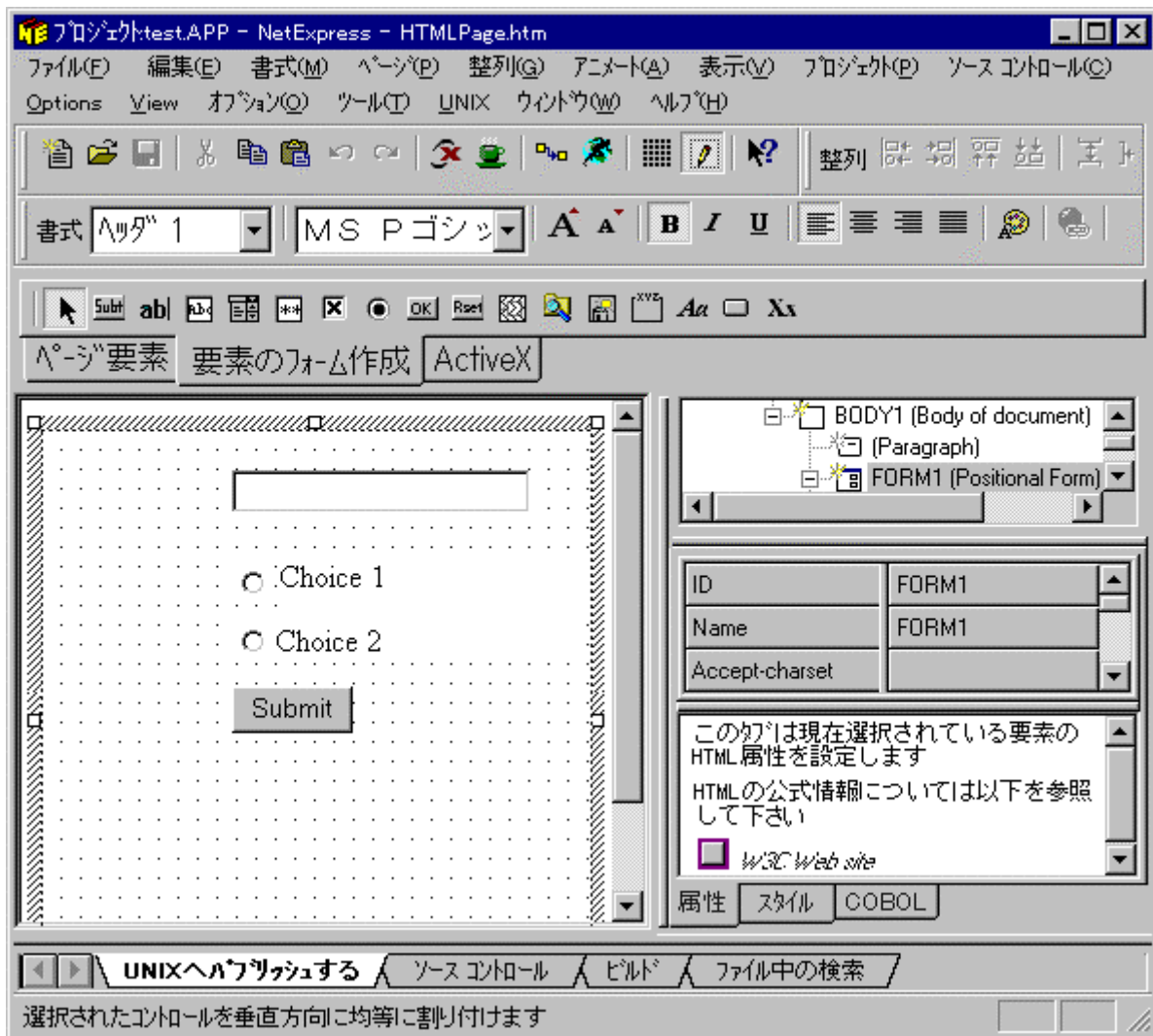
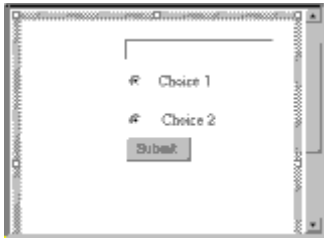


図 4-1 Form Designer

Form Designer の各部名称は、次のとおりです。

- ページ デザイン領域



現在作業しているページやフォームを編集するための領域です。

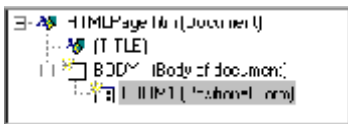
- プロパティ エディタ



ページ ツリー ビューで現在選択されている要素のプロパティをすべて表示します。通常、ページ デザイン領域またはページ ツリー ビューのどちらかで要素を選択すると、両方のビューで選択したことになります。ただし、<BODY> のように、ページ デザイン領域でレンダリングされていない要素には、該当しません。

プロパティは、別々のペインにグループ化されるので、下部のタブをクリックして選択します。「属性」ペインには、HTML 属性が表示されます。「スタイル」ペインには、要素の CSS スタイル設定が表示されます。フォームまたはフォーム要素を選択した場合、「COBOL」ペインを使用して、リンクされたデータ項目のプロパティを設定できます（「プロパティとリンク データ項目」を参照してください）。Java アプレットか ActiveX コントロールを選択した場合、「プロパティ」ペインでは、表示されたプロパティをすべて設定できます。

- ページ ツリー ビュー



ページの全要素の階層を表示します。ページ デザイン領域の要素は、ページ ツリー ビューに表示された同じ要素をクリックして選択することもできます。ページ ツリー ビューで右クリックしてから、ポップアップ メニューの [要素をフィルタ処理] をクリックすると、ページ ツリー ビューで異なる型の要素をフィルタ処理することができます。

- 構成要素パレット



このパレットを使用すると、ページやフォームに要素を追加できます。要素は、さまざまなタブにグループ化されます。[ページ要素] タブには、ページのどの位置にでも配置できる HTML 要素が含まれます。[

フォーム要素] タブには、HTML フォームにだけ配置することができる HTML 要素が含まれます。

- 書式ツールバー



選択したテキストの書式を変更することができます。段落から見出しへの変更、フォントと色の変更、太字、斜体、下線の使用、テキストの整列の変更などを行うことができます。ハイパーリンクを追加することもできます。

4.2.2 配置編集とフロー編集

Form Designer は、HTML エディタとしても、また HTML フォーム ペインタとしても使用することができます。これら 2 種類の要求に応えるために、Form Designer には 2 つの編集モードがあります。

- 配置

コントロールやテキスト ラベルを位置フォームのどの場所にもドロップすることができます。フォームの要素は、マウスを使用してドラッグします。

- フロー

テキスト エディタを使用しているときと同様に、テキストの入力、挿入、上書き、削除ができます。

フローは、デフォルトの編集モードです。Form Designer では、ページで位置フォームが選択されている場合にだけ、配置編集が可能です。HTML ページ ウィザードを使用して新しいページを作成するときに位置フォーム テンプレートを選択した場合、Form Designer は、単一の位置指定フォームを含む空のページを作成します。

また、[フォーム要素] 構成要素パレットの [位置フォーム] ボタンをクリックすると、HTML ページのどの場所にも位置フォームを挿入することができます。Form Designer は、テキストの現在のカーソル位置にフォームを挿入します。位置フォームの上または下をクリックすると、フォームの前または後にテキストやイメージを追加することができます。

フロー モードで編集する場合、[フォーム要素] パレットのフォーム要素をクリックして、ページにフォーム要素を追加することもできます。この場合、HTML フォームをページの現在のカーソル位置に挿入するかどうかをたずねるダイアログ ボックスが Form Designer により表示されます。フォームの配置編集はできませんが、段落、空白文字、テーブルなどを使用して、フォームに要素を配置することができます。

4.2.3 プロパティとリンク データ項目

Form Designer では、COBOL サーバー側プログラムのデータ項目にフォームのコントロールをリンクさせることができます。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、Form Designer で作成したフォームからサーバー側プログラムを生成する場合、このプログラムは、フォームで命名されたコントロールに対するデータ項目を使って作成されます。

インターネット アプリケーション ウィザードは、「Name」属性を参照し（コントロールに「Name」属性がない場合は「ID」属性）、サーバー側プログラムでそのコントロールを表すために同じ名前の COBOL データ項目を生成します。「Name」属性と「ID」属性は、どちらもプロパティ エディタの「属性」ペインで設定します。データ項目のピクチャは、COBOLPicture プロパティ（プロパティ エディタの「COBOL」ペイン）から採用されます。COBOLPicture プロパティが空白の場合、リンクされたデータ項目は作成されません。

送信したフォームがサーバー側プログラムで入力として使用されると、リンクされたデータ項目は、コントロールの値に設定されます。フォームがサーバー側プログラムにより出力されると、フォームのコントロールは、サーバー側プログラムのリンクされたデータ項目の値に設定されます。

下図は、入力フィールド、2 つのオプション ボタン、2 つの実行ボタンを持つ入力フォームと、3 つの入力フィールドをもつ出力フォームを示します。これらのフォームの間には、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たリンク データ項目をもつサーバー側プログラムがあります。

アプリケーションのエンドユーザーが [送信] ボタンをクリックすると、コントロールの値がサーバー側プログラムに渡されます。複数のオプション ボタンには共通の名前が付けられ、サーバー側プログラムの 1 つのデータ項目で表されます。複数の送信ボタンも 1 つの名前とデータ項目を共有します。サーバー側プログラムが戻す出力フォームには、リンクされたコントロールの値によりエン트리 フィールドが設定されます。

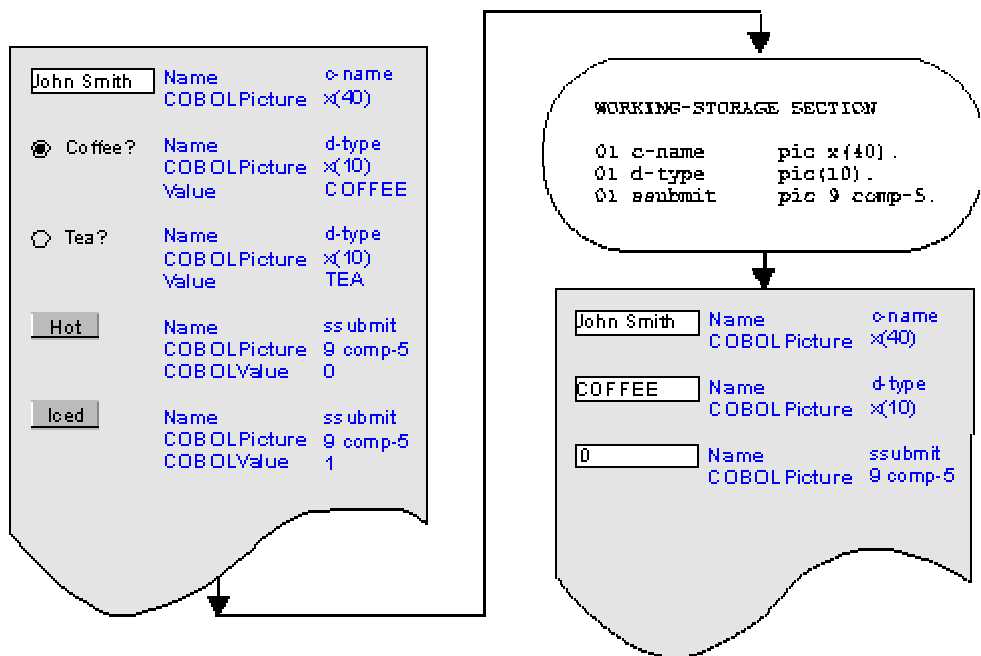


図 4-2 リンク データ項目へのコントロールのマップ方法

4.3 フォームとサーバー側プログラムの作成

この項では、フォームとサーバー側プログラムを作成するための基本的な手順を操作例を交えて説明します。Form

Designer の主な目的は、フォームを作成し、COBOL サーバー側プログラムで使用することですが、HTML を読み込んで編集することもできます。ページ ウィザードを使用して新しいページを作成する場合、HTML ページをテンプレートとして使用することができます。また、NetExpress のプロジェクト ディレクトリに HTML ページをコピーしてプロジェクトに追加し、Form Designer で直接編集することもできます。

上記の機能により、アプリケーションのページを柔軟にデザインし、作成することができます。プロのレイアウト デザイナが作成したページをもとに Form Designer でフォームを追加することもできます。ページでサーバー側プログラムと対話する部分はフォームだけなので、この章では、簡潔に説明するために、位置フォームを 1 つだけもつ空白の HTML ページを取り上げます。

フォームに対するサーバー側プログラムを生成するには、次の選択を行う必要があります。

- 入力フォームのあるページと出力するページが異なる非対称アプリケーションと、1 つのフォームとページを使用する対称アプリケーションのどちらを作成するか。対称アプリケーションと非対称アプリケーションについては、「インターネット プログラミングの概要」の章を参照してください。
- 入力と出力に使用するフォーム。

非対称アプリケーションでは、2 ページが同じサーバー側プログラムを共有します。サーバー側プログラムを生成するときに、HTML ページ、プログラムへの入力に使用する入力フォーム、別の出力ページを指定します。非対称アプリケーションの追加情報については、「非対称サーバー側プログラムの作成」の項を参照してください。対称アプリケーションのサーバー側プログラムを作成するときは、入力と出力に同じフォームとページを指定してください。

次の説明は、上級者を対象としています。詳しい手順ごとの説明が必要な場合は、オンライン ヘルプを参照してください。Form Designer や NetExpress を使用した経験がない場合は、『入門書』の「具体的な手順」を参照してください。

1. アプリケーション ファイルを格納する NetExpress プロジェクトを設定します。
2. [ファイル] メニューの [新規作成] をクリックして、「新規作成」ダイアログ ボックスから HTML ページを選択し、ページ ウィザードを起動します。
3. ページ ウィザードのプロンプトに記入します。

「HTML ページの新規作成」ダイアログ ボックスで [全般] タブと [フレーム] タブに表示されたテンプレート ファイルをもとにページを作成することができます。また、[既存のページ] タブをクリックして、既存の HTML ファイルをもとにすることもできます。

4. ページやページに含まれるフォームを編集します。
5. ページを保存します。
6. アプリケーションの入力フォームと出力フォームをもつページを作成したら、[ファイル] メニューの [新規作成] をクリックし、インターネット アプリケーションを選択して、インターネット アプリケーション ウ

ィザードを起動します。

7. インターネット アプリケーション ウィザードで [サーバー プログラム] を選択し、[次へ] をクリックします。
8. 「サーバー プログラムの生成」ページで、サーバー プログラムに対して入力ページを 1 つと入力フォームを選択します (ページには複数のフォームを含めることができます)。1 つまたは複数の出力ページを選択します。対称アプリケーションの場合、入力と出力に同じページを選択します。

4.3.1 フォームのペイント例

この章の残りの部分では、次のフォーム例を使用して、NetExpress でインターネット アプリケーションをビルドする原理を説明します。このフォームは、ホット ドリンクの注文に関する詳細情報を入力するためのものです。将来、企業のイントラネットを通してオフィスのコーヒーマシンが接続されることを想定して作成されています。完成したフォームは、次のようになります。

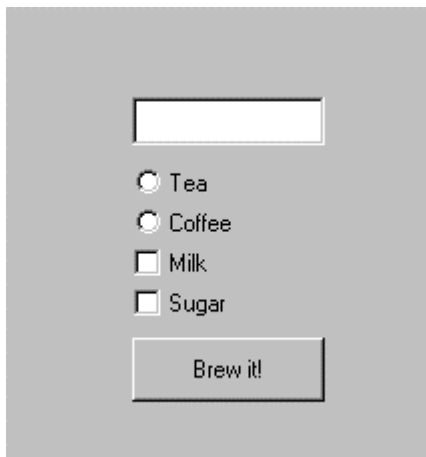


図 4-3 飲料のフォーム例

次に示す過程では、フォームのペイント方法を手順ごとに詳しく説明していません。Form Designer を使用した経験がない場合は、『入門書』の「具体的な手順」を参照してください。Form Designer を使用したほとんどの作業に関する詳細な手順は、Form Designer のヘルプに記載されています。[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を表示し、[目次] タブで [プログラミング]、[インターネット アプリケーション プログラミング]、[Form Designer] の順にクリックします。Form Designer の多くのダイアログ ボックスや入力フィールドでは、コンテキスト ヘルプも使用できます。ダイアログ ボックスの疑問符アイコンをクリックしてから、ヘルプを表示する項目をクリックします。または、フィールドにカーソルを合わせてから、[F1] キーを押します。

このアプリケーションの完全なバージョンは、¥netexpress¥base¥demo¥bevord_h に格納されています。

飲料のフォーム例をペイントする方法は、次のとおりです。

1. NetExpress を起動して、bevord.app という空プロジェクトを作成します。

2. [ファイル] メニューで [新規作成] をクリックし、「新規作成」ダイアログ ボックスから [HTML ページ] を選択してページ ウィザードを起動します。

次のように、ページ ウィザードに詳細を記入します。

- テンプレートとしての位置フォーム
- クロスプラットフォーム HTML 出力
- bevord_h の HTML ファイル名

[完了] をクリックすると、ウィザードは bevord_h.htm を生成し、Form Designer で開きます。

3. フォームにテキスト入力を追加します ([フォーム要素] パレットから)。プロパティ エディタを使用して、次のように設定します。

- Name を customername に設定します (「属性」ペインで)
- COBOLPicture を X(60) に設定します (「COBOL」ペインで)

サーバー側プログラム bev.cbl を後で作成すると、インターネット アプリケーション ウィザードはデータ項目 customername を生成します。フォームが送信されると、データ項目 customername は、このテキスト入力の内容に設定されます。

4. [フォーム要素] パレットの [テキスト] ボタンを使用し、スパン要素を [テキスト入力] の左側に追加し、スパンの内側をクリックします。"Name" と入力します。

スパンを使用して、位置フォームにラベルを追加します。

5. フォームにオプション ボタンを追加します。キャプションを「紅茶」に変更します。

デフォルトのキャプション テキストは、フォームにコントロールをドロップするときすでに選択されていますが、上書きすることができます。続いてキャプションを変更するには、キャプション テキストの中をクリックして、テキスト カーソルを表示する必要があります。修正するテキストを選択し、新しく上書きします。

オプション ボタン自体をクリックし (またはページ ツリー ビューでオプション ボタンを選択し)、次のプロパティを設定します。

- Name を beveragetype に設定します。
- Value を TEASELECTED に設定します。
- COBOLPicture を X(20) に設定します。

フォームの送信時に、このオプション ボタンが選択されている場合、データ項目 beveragetype には

"TEASELECTED" が設定されます。

6. フォームに 2 番目のオプション ボタンを追加します。キャプションを「コーヒー」に変更します。次のプロパティを設定します。

- Name を `beveragetype` に設定します。
- Value を `COFFEESELECTED` に設定します。
- `COBOLPicture` を `X(20)` に設定します。

フォームの送信時にこのオプション ボタンが選択されている場合、データ項目 `beveragetype` には "COFFEESELECTED" が設定されます。

7. フォームにチェック ボックスを追加します。キャプションを「ミルク」に変更します。次のプロパティを設定します。

- Name を `milkrequest` に設定します。
- Value を `1` に設定します。
- `COBOLPicture` を `9` に設定します。

フォームの送信時にこのチェック ボックスが選択されている場合、データ項目 `milkrequest` には `1` が設定されます。

8. フォームにチェック ボックスを追加します。キャプションを「砂糖」に変更します。次のプロパティを設定します。

- Name を `sugarrequest` に設定します。
- Value を `1` に設定します。
- `COBOLPicture` を `9` に設定します。

フォームの送信時にこのチェック ボックスが選択されている場合、データ項目 `sugarrequest` には `1` が設定されます。

9. [送信ボタン] を追加します。ボタンのキャプションに「注ぐ」を指定します。次のプロパティを設定します。

- Name を `Submit` に設定します。
- Value を `1` に設定します。
- `COBOLPicture` を `9` に設定します。

これにより、ボタンをクリックしたときに 1 に設定される COBOL データ項目送信が bev_h.cbl で宣言されます。

10. ツールバーの [保存] ボタンをクリックして、フォームを保存します。

ページの詳細は、すべて bevord_h.htm に保存されます。サーバー側プログラムのリンク データ項目を作成するために必要な情報は、bevord_h.mff に格納されます。

これらのファイルは、プロジェクトに追加されます。.mff ファイルは、ソース プール (「プロジェクト」ウィンドウの右側) だけに追加されます。.htm ファイルは、「プロジェクト」ウィンドウの両方のペインに追加されます。NetExpress では、.mff ファイルと .htm ファイルのどちらのファイルを開いても、Form Designer を起動することができます。

ここまでで、フォームを使用する簡単な CGI (サーバー側プログラム) を生成することができます。

11. [ファイル] メニューの [新規作成] をクリックし、[インターネット アプリケーション] を選択して、インターネット アプリケーション ウィザードを起動します。
12. 次のように、ウィザードに詳細を記入します。

- サーバー プログラム
- 入力ファイル bevord_h.htm
- 入力フォーム FORM1 (これは、作成したフォームのデフォルト名です)
- 出力フォーム bevord_h.htm
- ファイル名 bev_h.cbl

ウィザードの [完了] をクリックすると、サーバー側プログラムに対して次のファイルが生成されます。

- bev.cbl
- bev.cpf
- bev.cpv
- bev.cpy

.cbl ファイルはプログラムで、その他はフォーム データのコピーファイルです。詳細については、この章の後半で説明します。

関連付けられた CGI アプリケーションを実行すると、エンドユーザー側で表示されるフォームをプレビューすることができます。フォームを Web ブラウザに直接読み込んだ場合 (たとえば Windows のエクスプローラでフォーム ファイルをダブルクリックした場合)、フィールドには次のように表示されます。

:f-customername

これは、CGI プログラムからフォームに送信されたデータを代入するパラメータです。フォームを CGI プログラムで表示した場合、入力フィールドには、空白文字が、プログラムから返される情報が表示されます。プログラムの実行については、次の項で説明します。

4.4 スケルトン アプリケーションの実行

この段階では、実用可能なインターネット アプリケーションのスケルトンが作成されているはずですが。

- フォーム
- サーバー側プログラム

サーバー側プログラムをビルドし、実行して、問題がないかテストすることができます。

4.4.1 サーバー側プログラムのビルド

この章では、サーバー側プログラムを実行とアニメートが最も簡単な CGI プログラムとしてビルドする方法について説明します。プログラムを実際に使用する場合は、プログラムを ISAPI や NSAPI に変更することができます。これについては、「CGI、ISAPI および NSAPI プログラム」の章にまとめてあります。

サーバー側プログラムのビルド方法は、次のとおりです。

- NetExpress の [プロジェクト] メニューで [リビルド] をクリックします。

これで、アプリケーションを実行することができます。

4.4.2 アプリケーションの実行

アプリケーションを実行するためには、ローカル マシンで Web サーバーを起動する必要があります。Web サーバーとして Solo を使用しない場合、「アプリケーションの実装」の章の説明にしたがって、Web サーバーを構成し、NetExpress アプリケーションをデバッグします。また、他の Web サーバーを使用するための簡単な説明については、NetExpress のオンライン ヘルプを参照してください。[ヘルプ] メニューの [ヘルプ トピック] をクリックしてから、[目次] タブの [プログラミング]、[インターネット アプリケーションのプログラミング]、[インターネット アプリケーションのデバッグ]、[方法]、[インターネット アプリケーションのアニメート] の順に選択してください。

以前に、マシンでインターネット アプリケーションを実行した経験がない場合には、『入門書』の Solo Web サーバーの「具体的な手順」を参照してください。

Solo を使用してアプリケーションを実行する方法は、次のとおりです。

- NetExpress の [アニメート] メニューで [実行] をクリックします。

4.4.3 アプリケーションのデバッグ

NetExpress の Animator を使用してプログラム全体をステップ実行することができます。Animatorを使用した経験がない場合、『入門書』の「NetExpress 統合開発環境」の章で NetExpress Animator を使用した「具体的な手順」を参照してください。

インターネット アプリケーションのアニメート方法は、次のとおりです。

1. Form Designer により、アニメート設定が完了しているため、Solo Web サーバーを使用してアプリケーションをアニメートすることができます。[アニメート] メニューの [設定] をクリックすると、「アニメーションの起動」フィールドにサーバー側プログラムの URL が設定されていることを確認できます。

```
http://127.0.0.1/cgi-bin/program.exe
```

program.exe は、サーバー側プログラムの名前です。別の Web サーバーを使用する場合には、URL を次のように変更してください。

```
http://machine.domainname/cgi-bin/program.exe
```

別のサーバーを使用する場合には、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [プログラミング]、[インターネット アプリケーションのプログラミング]、[インターネット アプリケーションのデバッグ]、[方法]、[インターネット アプリケーションのアニメート] の順に選択し、その説明にしたがってください。

2. Web サーバーを実行していない場合には、起動します。Solo を使用している場合は、NetExpress が自動的に Solo を起動するため、このステップを省略することができます。
3. [アニメート] メニューの [アニメート開始] をクリックします。

この操作により、サーバー側プログラムの URL で Web ブラウザが起動します。ブラウザは、Web サーバーに対してプログラムの実行要求を出します。プログラムを起動すると Animator も起動します。この段階で、サーバー側プログラムをステップ実行することができます。

EXEC HTML 文をステップ実行するたびに、プログラムはページをブラウザに出力します。

注記: 上記の説明は、対称アプリケーションの場合です。非対称アプリケーションの場合、「アニメーションの起動」フィールドには、アプリケーションの入力ページが設定されます。この場合、入力ページの [送信] ボタンをクリックすると、デバッガに COBOL コードが表示されるだけです。

4.4.4 例

飲料に関するインターネット アプリケーションのサーバー側のスケルトン アプリケーションを実行すると、問題がないか確認できます。ここでも、フォームのペイントに関して、手順ごとの詳しい説明は行いません。NetExpress IDE や Solo Web サーバーを使用した経験がない場合には、『入門書』の「具体的な手順」を参照してください。

サンプルをビルドし、実行する方法は、次のとおりです。

1. NetExpress からプロジェクトをリビルドします。
2. [アニメート] メニューで [実行] をクリックします。

デバッガでコードの実行を確認する方法は、次のとおりです。

1. [アニメート] メニューの [アニメート開始] をクリックします。

これにより、Web ブラウザが起動し、Web ブラウザにプログラムの URL が設定されます。また、NetExpress Animator でプログラムの実行が開始されます。Web ブラウザにフォームがすぐに表示され、NetExpress Animator でコードの実行が開始されたことが確認できない場合、ブラウザの [再読み込み] または [最新表示] ボタンをクリックしてください。

2. Animator の強力なデバッグ機能を使用して、プログラムのコードをステップ実行することができます。

Animator の使用方法の詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [開発環境]、[プログラムのデバッグとアニメート] の順に選択してください。

4.5 非対称サーバー側プログラムの作成

「インターネット プログラミングの概要」の章では、サーバー側プログラムを下図のように対称と非対称に分類しました。

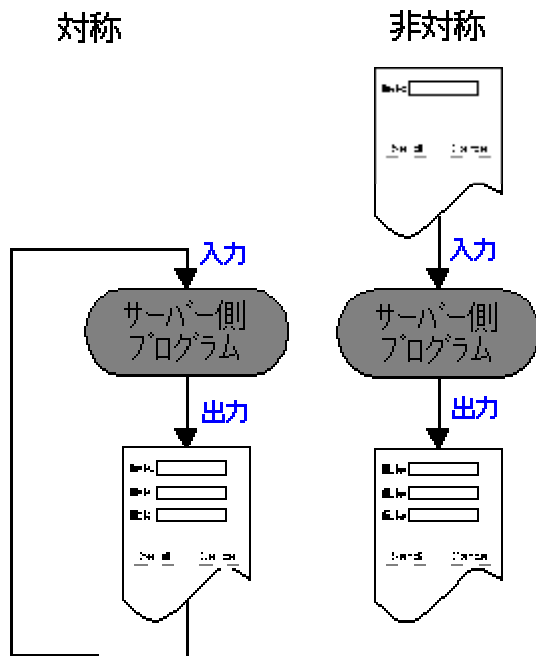


図 4-4 対称アプリケーションと非対称アプリケーション

ここまでは、対称プログラムの開発方法について説明してきましたが、この後は、Form Designer を使用して非対称プログラムを作成する方法について解説します。非対称プログラムを開発する場合、入力フォーム、出力フォーム、これらのフォームを結合するサーバー側プログラムを作成する必要があります。

非対称アプリケーションの開発手順は、次のとおりです。

1. ページ ウィザードを使用して入力フォームを作成します。
2. ページ ウィザードを使用して出力フォームを作成します。
3. インターネット アプリケーション ウィザードを実行し、サーバー プログラムを選択します。ウィザードで入力フォームと出力フォームを選択します。

ページ ウィザードで [完了] をクリックすると、サーバー側のスケルトン プログラムが自動的に生成され、プロジェクトに追加されます。このプログラムには、入力フォームからの入力を受け付け、出力フォームを返すためのコードが記述されています。

4.5.1 例

対称アプリケーションと非対称アプリケーションの違いを示すために、飲料に関するアプリケーションの 2 つ目のフォームを作成します。この 2 つ目のフォームは、入力フォームで選択された内容をまとめて表示する出力フォームを指します。この項では、出力フォームをペイントします。また、この章の最後の部分では、入力データを処理して出力フォーム用にそのデータをまとめるコードを、生成されたアプリケーションに追加します。

出力フォームは、次のように表示されます。

The image shows a simple web form on a light gray background. It consists of three vertically stacked input fields. The first field is labeled 'Name' and is empty. The second field is labeled 'Beverage' and contains a single vertical bar character '|'. The third field is labeled 'Options' and is empty.

図 4-5 飲料に関するアプリケーションの出力フォーム例

別のサーバー側プログラムには接続しないので、このフォームにはプッシュボタンがありません。

出力フォームのペイント方法は、次のとおりです。

1. NetExpress を起動し、この章で作成したプロジェクト bevord.app を読み込みます。
2. [ファイル] メニューの [新規作成] をクリックして、「新規作成」ダイアログ ボックスから [HTML ページ] を選択し、ページ ウィザードを起動します。bevsum_h というファイル名を持つ位置フォームを作成します。
3. フォームに「名前」、「飲料」および「オプション」などのラベルを追加します ([フォーム要素] メニューの [テキスト] を使用して、テキストを挿入するためのスパンをフォームに追加します)。
4. フォームに 3 つの入力テキスト コントロールを追加し、プロパティを次のように設定します。
 - a. Name を customerout に、COBOLPicture を X(60) に設定します。
 - b. Name を beverageout に、COBOLPicture を X(60) に設定します。
 - c. Name を optionsout に、COBOLPicture を X(60) に設定します。

サーバー側プログラムを再生成する場合、データ項目 CUSTOMEROUT、BEVERAGEOUT および OPTIONSOOUT は自動的に宣言されます。

5. フォームを保存します。
6. インターネット アプリケーション ウィザードを起動します。
7. 次のように、ウィザードに詳細を記入します。
 - サーバー プログラム
 - 入力ファイル bevord_h.htm
 - 入力フォーム FORM1

- 出力フォーム bevsum_h.htm
- プログラム ファイル名 bev_h.cbl

ウィザードの [完了] をクリックすると、サーバー側の新しいスケルトン プログラムが生成されます。既存のサーバー側プログラムにビジネス ロジックを追加しようとする、既存のサーバー側プログラムを上書きしてよいかどうかを確認するダイアログが表示されます。このとき、既存のプログラムを保存する場合は、別の名前で新しいプログラム ファイルを作成することができます。

4.6 サーバー側のスケルトン プログラムへの機能追加

この項では、インターネット アプリケーション ウィザードにより生成されたサーバー側のスケルトン プログラムの構造について説明します。対称アプリケーションの場合、サーバー側のスケルトン プログラムは、エンドユーザーが入力したデータを含むフォームをエコーバックするだけです。サーバー側のスケルトン プログラムを実用可能なアプリケーションにするには、機能を追加する必要があります。

インターネット アプリケーション ウィザードを実行すると、次のファイルが自動的に作成されます。

- *program.cbl*

サーバー側プログラムのスケルトン コードをもつ COBOL ソース ファイルです。これは、唯一編集する必要のある生成ファイルです。

- *program.cpy*

HTML コントロールの COBOL Picture プロパティで指定されたピクチャ文字列をもつ変数を含む COBOL コピーファイルです。これらのデータ項目名は、Form Designer で設定された名前と同じです。

非対称アプリケーションのプログラムを生成した場合、このコピーファイルで宣言されたデータ項目は、入力フォームと出力フォームに対するリンク データ項目をすべて統合したものです。ただし、重複した名前を共有するコントロールに対して作成される共有データ項目は 1 つだけです。

- *program.cpf*

フォームとプログラム間でデータを転送するデータ項目を宣言する COBOL コピーファイルです。これらのデータ項目は、すべて PIC X(n) 型で、フォームのコントロールの名前プロパティで命名されます。

非対称アプリケーションのプログラムを生成した場合、このコピーファイルで宣言されたデータ項目は入力フォームと出力フォームに対するデータ項目を統合したものになります。

- *program.cpv*

フォームから送信されるデータのフォーマット (.cpf ファイルのデータ項目) と、各コントロールの COBOL Picture プロパティでユーザーが指定したフォーマット (.cpy ファイル) の間でデータを変換するコ

ードを含む COBOL コピーファイルです。

入力フォームを指定するフォームからプログラムを生成した場合、入力変換は入力フォームのデータに対して行われ、出力変換は出力フォームのデータに対して行われます。

生成した *program.cbl* を修正すると、機能を追加することができます。サーバー側プログラムに関連付けられたフォームを保存するたびに、上記のコピーファイルはすべて再生成されます。そのため、サーバー側プログラムには、常に、使用するフォームと通信するために必要なデータ項目がすべて含まれることになります。

主となる .cbl ファイルは更新されないため、プログラム自体に行った変更が保存されます。ただし、上記のコピーファイルは変更しないでください。上記のコピーファイルを編集した場合、サーバー側プログラムに関連付けたフォームを次に編集して保存したときにその編集データが失われます。

4.6.1 例

ここまでの段階で、入力フォームと出力フォームを作成し、サーバー側プログラムを生成しました。この項では、入力フォームの入力データをすべて受け付けて要約し、出力フォームにその要約を送信するコードを、生成されたサーバー側プログラムに追加します。この作業は、非常に簡単ですが、サーバー側プログラムの編集箇所と編集方法について説明します。より複雑なアプリケーションについても同じ法則を適用できます。

プログラムに機能を追加する方法は、次のとおりです。

1. NetExpress IDE の「プロジェクト」ウィンドウでファイル *bev.cbl* をダブルクリックし、編集を開始します。
2. PROCESS-BUSINESS-LOGIC 節を検索し、テキスト カーソルを次のコメントの下の行に合わせます。

```
*> Add application business logic here
```

3. 次の文を追加します。

```
*> 顧客名を出力します。
```

```
move CustomerName to CustomerOut
```

```
*> 飲料の種類を出力します。
```

```
if BeverageType = "TEASELECTED"
```

```
    move "Tea" & x"0" to BeverageOut
```

```
end-if
```

```
if BeverageType = "COFFEESELECTED"
```

```
    move "Coffee" & x"0" to BeverageOut
```



```
end-if  *> ミルクと砂糖に関する選択をまとめます。

if MilkRequest = 1

    if SugarRequest = 1

        move "Milk and suger" to OptionsOut

    else

        move "Milk" to OptionsOut

    end-if

else

    if SugarRequest = 1

        move "Suger" to OptionsOut

    end-if

end-if
```

4. 変更を保存し、アプリケーションをリビルドします。
5. 入力フォームの URL でアプリケーションを起動し、再実行します。

データを入力し、フォームの送信ボタンをクリックすると、入力内容をまとめた出力フォームが表示されます。

第5章 データ アクセス アプリケーション

この章では、インターネット アプリケーション ウィザードを使用して、実用可能な SQL アプリケーションを簡単に作成する方法について説明します。また、アプリケーションの作成後に、その機能を拡張する方法についても説明します。

5.1 概要

インターネットアプリケーションウィザードでは、SQL データベースにアクセスするための HTML フォームとサーバー側コードをすべて生成することができます。このウィザードは、Open ESQL Assistant を使用して、SQL クエリコードを生成し、サーバー側プログラムに埋め込みます。インターネット アプリケーション ウィザードにより生成されるデータ アクセス アプリケーションでは、エンドユーザーは次のデータベース関数にアクセスできます。

- Query
- Update
- Add
- Delete

アプリケーションの作成中に使用する関数を決定します。

データベースは、次の 2 種類のビューで表示することができます。

- 単一レコード ビュー

エン트리 フィールドにクエリーの各列を表示するフォームです。エンドユーザーがデータベースの情報を実際に更新できるのは、このビューだけです。

- リスト ビュー

画面に表形式でレコードのグループを表示します。表示するレコード数は、アプリケーションの生成中に決定します。デフォルトでは、10 レコードです。

各ビューは、別々のフォームとサーバー側プログラムで処理されます。生成されるサーバー側プログラムは、入力と出力に同じフォームを使用する対称プログラムです。

単一レコード ビューとリスト ビューの両方をもつアプリケーションを生成する場合、これら 2 つのビューは互いにリンクされます。単一レコードのフォームには、リスト ビューを表示するためのプッシュボタンがあります。一方、リスト ビューに表示された各レコードには、単一レコード ビューを表示するためのハイパーテキスト リンクが含まれています。

COBOL アプリケーションと SQL アプリケーションの詳細については、オンライン マニュアル『データベース アクセス』を参照してください。

5.2 データ アクセス アプリケーションの作成

インターネット アプリケーション ウィザードでデータ アクセス アプリケーションを作成する方法を最も簡単に理解するには、実際に試してみることです。ここでは、インターネット アプリケーション ウィザードを使用して 2 つのプログラムを作成します。一方のプログラムでは、顧客情報テーブルに顧客の詳細情報を問い合わせ、もう一方のプログラムでは、注文テーブルに注文の詳細情報を問い合わせます。

データ アクセスに必要な ESQL コードをすべて使用して 2 つの基本プログラムを作成したら、これらを修正して注文処理システムを作成します。この作業では、顧客レコードとその顧客の注文を直接リンクします。そのため、ここでは、インターネット アプリケーション ウィザードを使用して簡単なデータベース アクセス アプリケーションを生成する方法だけでなく、生成されたアプリケーションをカスタマイズし、拡張する方法についても説明します。

5.2.1 アプリケーションの生成

この項では、2つの簡単なアプリケーションを作成します。

- 顧客の問い合わせと更新
- 注文の問い合わせと更新

顧客の問い合わせでは、エンドユーザーがデータベースの顧客情報を問い合わせることができます。注文のクエリーと更新では、エンドユーザーが注文を検索し、更新することができます。

顧客のクエリー

まずはじめに、NetExpress で現在開いているプロジェクトを閉じます (開いているプロジェクトがある場合)。インターネット アプリケーション ウィザードでは、生成されたファイルはすべて現在の NetExpress プロジェクトに追加されるためです。ここでは、このアプリケーションのファイルをすべて 1 つのプロジェクトに格納します。開いているプロジェクトがない場合は、インターネット アプリケーション ウィザードにより、新しいプロジェクトの名前と場所を指定するようにプロンプトが表示されます。

この章では、「ここで [OK] ボタンをクリックしてください。」というような詳しい説明は行わず、全体的な手順について説明します。はじめる前に Form Designer と NetExpress の概略を理解するには、『入門書』の「具体的な手順」を実行してください。

顧客のクエリーを生成する方法は、次のとおりです。

1. インターネット アプリケーション ウィザードを起動します ([ファイル] メニューの [新規作成] をクリックし、「新規作成」ダイアログ ボックスから [インターネット アプリケーション] を選択します)。

2. 新規プロジェクトを作成するために次の情報を記入し、[次へ] ボタンをクリックします。

プロジェクト名 orders
プロジェクトのパス ¥netexpress¥base¥workarea¥orders

3. 「ウィザードへの入力」から [SQL データベース] を選択し、[次へ] を押します。
4. ウィザードで次の情報を記入し、[次へ] ボタンをクリックします。

ファイルの基本名 customer
生成フォームのタイトル 顧客の詳細情報

インターネット アプリケーション ウィザードにより、アプリケーションが生成されます。ウィザードは、作成するファイルがすでに存在する場合にはプロンプトを表示するので、そのファイルを保存するか、上書きするかを選択することができます。

5. 「データの選択」ダイアログ ボックスで [NetExpress Sample2] をダブルクリックします。次に、[Customer] テーブルをダブルクリックして選択します。[Customer] テーブルを右クリックし、「すべてのカラムを選択」を選択します。

クエリーの結果を確認するには、[クエリーの実行] ボタンをクリックします。

6. [次へ] ボタンをクリックして、「アプリケーション スタイル」ダイアログ ボックスを表示します。このダイアログ ボックスから、データを表示するビューの種類を選択することができます。

7. 次のオプションを選択し、[次へ] ボタンをクリックします。

- 単一レコード ビューでデータを表示
- テーブル ビューでデータを表示
 - リスト中の単一レコード ビューへのホットリンク項目
 - 1 ページに 10 レコードずつページング

8. 「フォームの編集」ダイアログ ボックスでは、アプリケーションのユーザーがデータベースを更新できるようにするかどうかを決定します。オプション ボタン [はい] をクリックし、「挿入」、「更新」および「削除」のチェック ボックスを選択します。[次へ] ボタンをクリックします。

9. 最後のダイアログ ボックスで、チェックボックス「設定を省略値とする」と「NetExpress プロジェクトにファイルを追加する」の両方が選択されていることを確認し、[完了] ボタンをクリックします。

インターネット アプリケーション ウィザードでは、次のファイルが生成されます。

- フォーム ビュー
 - customerform.cbl - SQL コードを含むソース ファイル
 - customerform.cpf - フォームで使用するコントロールの変数を含むコピーファイル
 - customerform.cpy - ビジネス ロジックの変数を含むコピーファイル
 - customerform.cpv - ビジネス ロジックとフォームで使用するコントロールの変数の間で変換するためのコードを含むコピーファイル
 - customerform.htm - フォームの HTML ファイル
 - customerform.mff - フォームを編集するための Form Designer ファイル

- リスト ビュー
 - customerlist.cbl - SQL コードを含むソース ファイル
 - customerlist.cpf - フォームで使用するコントロールの変数を含むコピーファイル
 - customerlist.cpy - ビジネス ロジックの変数を含むコピーファイル
 - customerlist.cpv - ビジネス ロジックとフォームで使用するコントロールの変数の間で変換するためのコードを含むコピーファイル
 - customerlist.htm - フォームの HTML ファイル
 - customerlist.mff - フォームを編集するための Form Designer ファイル

注記: その他に 2 つのファイル (sqlca.cpy と sqlda.cpy) がプロジェクトに追加されます。これらのファイルは埋め込み SQL の標準的なコピーファイルなので、編集する必要はありません。詳細については、NetExpress のオンライン ヘルプの索引から「sqlca」と「sqlda」を参照してください。

この段階で、データベースのクエリーを実行できる完全なアプリケーションが完成したことになります。

注記: 別の SQL クエリーを含むアプリケーションを再生成する場合、まず .cbl ファイルを削除します。ビジネス ロジックに加えた変更内容の損失を防ぐために、ジェネレータは既存の .cbl ファイルを上書きしません。.cbl ファイルが存在しない場合には、スクラッチから再生成します。

5.2.2 基本アプリケーションの実行

この段階では、アプリケーションをビルドし、実行することができます。

1. [プロジェクト] メニューの [リビルド] をクリックします。
2. [アニメート] メニューの [実行] をクリックします。
3. 「アニメーションの起動」ダイアログ ボックスで、実行可能プログラムの名前を次のように変更します。

HTTP://127.0.0.1/cgi-bin/customerlist.exe

この操作により、顧客リストのサーバー側プログラムが実行され、顧客データベースの最初の 10 レコードが表示されます。 < または > ボタンをクリックすると、一度に 10 レコードずつページングすることができます。 << または >> ボタンをクリックすると、先頭または末尾をページングすることができます。

顧客レコードを単一で表示するには、左側の列にある CustID をクリックします。これにより、フォーム ビューが表示されます。フォーム ビューでは、データベースの修正やフィルタ処理オプションの変更が可能です。

たとえば、ニューヨーク州の全顧客を確認する方法は、次のとおりです。

1. [画面をクリア] ボタンをクリックします。
2. 「Region」フィールドに「NY」と入力します。
3. [フィルタ条件] リストをドロップダウンし、[Region] を選択します。
4. [クエリー] ボタンをクリックして、ニューヨーク州の顧客の最初のレコードを表示します。

一度に 1 レコードずつページングするか、テーブル ビューをクリックしてリスト形式で表示します。テーブル ビューを使用すると、地域ごとのフィルタ処理結果を表示するためにステータス行が変更されていることがわかります。

レコードの追加方法は、次のとおりです。

1. フォーム ビューで [画面をクリア] ボタンをクリックします。
2. 顧客の詳細情報を入力します。
3. [レコードの挿入] ボタンをクリックします。

フォームが再表示され、レコードが正常に挿入されたことがステータス行に示されます。

レコードの更新方法は、次のとおりです。

1. フォームのデータ (たとえば、住所など) を変更します。

2. [レコードの更新] ボタンをクリックします。

フォームが再表示され、レコードが正常に更新されたことがステータス行に示されます。

レコードの削除方法は、次のとおりです。

1. フォーム ビューで [レコードの削除] ボタンをクリックします。

5.2.3 第 2 のフォーム セットの生成

今度は、第 2 のフォーム セットを生成し、注文レコードを問い合わせ、更新してみましょう。

1. インターネット アプリケーション ウィザードを起動します。
2. [ウィザードへの入力] で [SQL データベース] を選択し、[次へ] を押します。
3. ウィザードで次の情報を記入し、[次へ] ボタンをクリックします。

ファイルの基本名 order

生成フォームのタイトル 注文の詳細情報

4. 「データの選択」ダイアログ ボックスの [NetExpress Sample2] をダブルクリックし、「Orders」テーブルを選択してダブルクリックします。「Orders」テーブルを右クリックし、[すべてのカラムを選択] を選択してから、[次へ] をクリックします。
5. [アプリケーション スタイル] ダイアログ ボックスで、単一レコード フォームとテーブル ビュー フォームの両方を選択し、ページング オプションを 1 ページあたり 10 レコードに設定します (前回と同様)。
6. 「フォームの編集」ダイアログ ボックスで、前回と同様、レコードの追加、更新、および、削除に関する完全なアクセス権をユーザーに与えます。
7. ウィザードの最後のページで [完了] を押します。

インターネット アプリケーション ウィザードにより、注文情報を表示するための第 2 のファイル セットが生成されます。

8. アプリケーションをリビルドします。
9. このアプリケーションに対してフォーム ビューを実行します。
 - [フィルタ条件] として CustID を設定します。
 - [画面をクリア] ボタンをクリックします。
 - 「CustID」フィールドに [MERRG] と入力し、[クエリー] をクリックします。

- テーブル ビューをクリックして、この顧客の注文リストを表示します。

次の項では、Customer と Order のクエリーフォームを互いに結合するために、このアプリケーションを修正します。

5.2.4 アプリケーションの修正

データベースにアクセスするサーバー側プログラムは COBOL で作成されているため、編集して機能を追加することができます。HTML フォームの編集には、Form Designer を使用することができます。この項では、アプリケーション例を修正する方法について説明します。

現在、2 対のフォームがあります。1 対は顧客情報のデータベースに問い合わせるフォームで、もう 1 対は注文情報のデータベースに問い合わせるフォームです。顧客情報のクエリーフォームに、この顧客の注文リストへ直接リンクするプッシュボタンを追加します。

新しいプッシュボタンは、orderform.exe というサーバー側プログラムを起動し、現在表示されている顧客の CustID をこのプログラムに渡します。

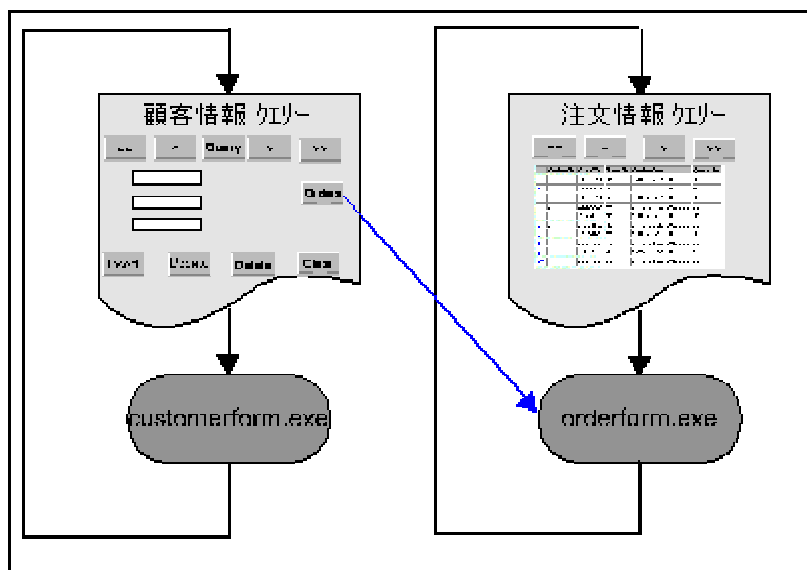


図 5-1 顧客情報のクエリーを注文リストへリンクさせた状態

この新機能を追加するには、フォームとサーバー側プログラムを修正する必要があります。次の 2 項では、この方法について説明します。

5.2.4.1 フォームの変更

サーバー側プログラム orderlist.exe を実行するリンクを「顧客情報のクエリー」フォームに追加する必要があります。また、orderlist.exe に次の情報を渡すことも必要です。

- 現在、フォームに表示されている Customer ID
- Customer ID でフィルタ処理するための命令

情報は、サーバー側プログラムを起動するリンクを通して、サーバー側プログラムに渡すことができます。この情報は、フォームのコントロールからのデータと同じように、名前と値の組み合わせとして渡されます。情報を渡すために必要な変更は、すべて Form Designer で customerform.htm と orderlist.htm を編集して行うことができます。

必要な変更内容は、次のとおりです。

- フォーム customerform.htm にプッシュボタンを追加し、スクリプト アシスタントを使用して、ハイパーリンクを通して orderlist.exe を起動する JavaScript を追加します。

顧客ごとにフィルタ処理するための Customer ID と命令は、このリンクにより OrderList に送信されます。

- Customer ID を受け付けるためのダミーのエントリ フィールドを orderlist.htm に追加します。

orderlist.htm を保存すると、Form Designer は orderlist.cbl が使用するフォームの変数とコードをすべて再生成し、フォームからのデータ受け付けを可能にします。データは、名前と値の組み合わせとしてサーバー側プログラムに渡されるため、サーバー側プログラムは、フォームからのデータ送信と同様にハイパーリンクによって起動されたときに、送信データを受け付けます。コマンド行により送信されたパラメータと同じ名前をもつダミー エントリ フィールドを追加すると、顧客 ID を受け付けるために必要なすべてのコードを自動生成することができます。ダミー フィールドには、フォームに表示されない非表示フィールドを使用します。

customerform.htm の編集方法は、次のとおりです。

1. NetExpress のプロジェクト ウィンドウで customerform.htm をダブルクリックして、Form Designer を起動し、フォームを読み込みます。
2. フォームに入力ボタンをドロップします ([入力ボタン] を使用します。customerform.exe を常に再実行する送信ボタンは使用しません。)
3. ボタンのキャプションを [Orders] に変更します。
4. 入力ボタンを右クリックし、コンテキスト メニューで [イベント] をクリックします。
5. [Orders] 入力ボタンに対して onclick イベントを選択し、[ハンドラの新規作成] をクリックします。
6. 最下部のペインで中括弧の間に次の文字列を入力します。

```
window.location.href="orderlist.exe?CustID=:A-CustID&Action=cquery"
```

注記: コントロール名、CustID、Action では、大文字と小文字が区別されます。ここに記載されているとおり、正確に入力してください。

上記のように記述すると、プッシュボタンが押されるたびに orderlist.exe?CustID=:f-A-CustID&Action=cquery というハイパーテキスト リンクを設定するのと同じことになります。リンクの最初の部分である orderlist.exe は、サーバーに対してリソース orderlist.exe を要求していることを示します。疑問符は、名前

と値の組み合わせリストが後に続くことを示します。アンパサンド (&) は、名前と値の組み合わせの間の区切り文字です。

- CustID=:A-CustID は、CustID という名前と :f-A-CustID という値を持つパラメータにより送信することを示します。このフォームは、EHTML を使用してサーバー側プログラム customerform.exe により出力されます。:f-A-CustID は、フォームの出力時に COBOL データ項目 f-A-CustID の値に置換される EHTML 代入マーカーです。COBOL データ項目 f-A-CustID は、「CusterID」フィールドの値を設定するために COBOL プログラムで使用されます。つまり、COBOL データ項目 f-A-CustID は、CustID という名前の HTML パラメータに Customer ID の値を設定します。

「フォーム」フィールドの値を設定する COBOL 変数を識別するには、フォームの「CustID」フィールドをクリックします。「CustID」フィールドの Name プロパティを、A_CustID に設定します。これにより、サーバー側プログラムで :A-CustID と f-A-CustID の 2 つの変数が生成されます (アンダスコアは COBOL データ名に使用できないため、ハイフンに変換されます)。コントロールと同じ名前をもつ変数は、COBOL プログラムでの操作対象となり、コントロールの COBOLPicture プロパティで設定したピクチャ文字列を格納します。

"f." が先頭につく変数は、フォームと COBOL プログラム間のデータ転記に使用され、常に PIC X(n) ピクチャ文字列を格納します。これは、フォーム データが常に文字列リテラルであるためです。生成されたコードには、入力用と出力用の 2 つのデータ項目間でデータを転記するためのルーチン セットと必要な変換が常に含まれています。

- Action=cquery は、Action という名前の HTML パラメータに値 "cquery" を設定します。

orderlist.htm のすべての送信ボタンでは、Group name プロパティは Action に設定されています。変数 Action により、プログラムを実行するためにクリックした送信ボタンの値がサーバー側プログラムに渡されます。orderlist.exe は、このリンクにより起動されると、COBOL データ項目 Action の値を調べ、押されたボタンを確認します。orderlist.exe は、Action=cquery というこのコードをもとに、フォームで "cquery" という値をもつボタンが押されたことを認識します。orderlist.cbl にコードを追加して、アクションが "cquery" に設定されたときに Customer ID でフィルタ処理を行うクエリーを実行することができます。

7. [OK] をクリックしてスクリプト アシスタントを閉じます。
8. 変更を保存します。

orderlist.htm の修正方法は、次のとおりです。

1. orderlist.htm を Form Designer に読み込みます。
2. フォームで希望の場所に非表示フィールド コントロールをドロップし、Name プロパティを CustID に設定します。COBOLPicture プロパティを PIC X(5) に設定します。これが、customerform.htm に含まれる「CustID」フィールドのピクチャ プロパティになります。

フォームを保存すると、CustID というデータ項目と、CustID という変数から値を読み込むためのコードが設定されます。

3. フォームへの変更を保存します。

この操作により、フォームだけでなく、フォームのデータ入力と出力に関連付けられたサーバー側プログラム (orderlist.cbl) のコードもすべて再生成され、CustID の値がある場合は、COBOL 変数 CustID に読み込まれるようになります。

5.2.4.2 サーバー側プログラムの修正

今度は、サーバー側プログラム orderlist.cbl を変更する必要があります。このプログラムには、次の 2 点を追加する必要があります。

- アクション "cquery" を識別するための、EVALUATE 文に対する追加条件
- CustID でフィルタ処理する SQL クエリーに必要な変数を設定する新しい節

通常、この 2 点は、orderform.exe というサーバー側プログラムにより設定され、状態ファイルに格納されます。ただし、orderlist.exe が customerform により起動された場合、このデータは設定されません。

サーバー側プログラムの編集方法は、次のとおりです。

1. 「テキスト編集」ペインで orderlist.cbl を開きます。
2. PROCEDURE DIVISION ヘッダとその数行下にある EVALUATE 文を検索します。
3. WHEN OTHER 句のすぐ上にある EVALUATE 文に新しい条件を追加します。

```
when "cquery"
```

```
perform DoQueryByCustomer
```

4. CustID でフィルタ処理するようにプログラムを設定するコードを追加します。プログラムの重要な節の後に次の節を追加してください。

```
DoQueryByCustomer section.
```

```
move "A.CustID" to s-filter-field *> アプリケーション ロジックでは、
```

```
*> 大文字と小文字を区別します。
```

```
*> "A.CustID" を大文字小文字を変えずに
```

```
*> 入力してください。
```

```

move "=" to s-filter-op

move custID to filter-A-Custid

move "???" to search-op

move low-values to sort-spec

perform Do-SQL-Query

perform SQL-Open

perform Next-DataTable

if no-data

    perform setup-status

    string

        " - no data to display" delimited size

        into frmesStatus pointer stat-index

end-if

exit.

```

5. 変更を保存します。

上記のコードでは、データベースを「CustID」フィールドでフィルタ処理して検索するクエリーを Do-SQL-Query によりビルドできるように、変数が設定されます。変数には次のような意味があります。

変数	説明
s-filter-field	フィルタ処理する列。値は "A.columnname" です。先頭の "A." という文字は、将来、機能を拡張し、テーブルを結合するクエリーを作成できることを示します。
s-filter-op	照合条件。次のどれかを設定することができます。"=", ">=", ">", "<=""<", または "!="。
filter-A-CustID	フィルタ処理条件となる CustID の値。データ ウィザードで設定したクエリーの各列に対応する「フィルタ」フィールドがあります。「フィルタ」フィールド名は、"filter-A-columnname" です。
search-op	検索方向。"???" は、最初の該当データを検索するための命令です。"?>" と "?<" は、次と前の該当データを検索するための命令です。次と前の該当データは、検索順序の列と、アプリケーションの状態レコードの内容により定義されます。

sort-spec

テーブルの順序を指定します。DESC または空白文字を設定します。

詳細については、『生成 CGI コード リファレンス』を参照してください。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、次に [目次] タブをクリックします。[参照]、[生成 CGI コード]、[データ アクセス CGI] の順に選択します。

5.2.4.3 新規アプリケーションの実行

この段階では、修正されたアプリケーションを実行することができます。

1. [プロジェクト] メニューの [リビルド] をクリックして、変更したプログラムをリコンパイルします。
2. [アニメート] メニューの [実行] をクリックします。
3. 「アニメーションの起動」ダイアログ ボックスのフィールドを次のように変更し、[OK] をクリックします。

`http://127.0.0.1/cgi-bin/customerform.exe`

Web ブラウザには、顧客のフォーム ビューが表示されます。

4. 前回のように、このフォームを使用して顧客の詳細情報を問い合わせ、更新することができます。
5. 特定の顧客の注文を確認する場合は、[注文] ボタンをクリックします。

この操作により、選択した顧客の注文リストが表示されます。

第6章 レガシー コードの再使用

この章では、既存の COBOL サブルーチンを再使用してインターネット アプリケーションを作成する方法を説明します。

6.1 概要

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、すぐに既存の COBOL サブルーチンをインターネット アプリケーションに変換できます。パラメータを受け付けて結果を返す連絡節を使用してコードを作成した COBOL プログラムは、ほとんど使用できます。インターネット アプリケーション ウィザードでは、フォームからの入力を受け付け、サブルーチンを呼び出す結合プログラムが生成されます。この結合プログラムは、プログラムから返された結果を同じフォームまたは別のフォームで表示します。連絡節を通してアプリケーションの入力パラメータと出力パラメータにアクセスできるようにするには、レガシー アプリケーションを修正し、既存のユーザー インターフェイス ロジックを分離する必要がある場合があります。

インターネット アプリケーション ウィザードを実行し、「ウィザードへの入力」で「COBOL ソース ファイル」を選択した場合、インターネット アプリケーションの基礎として使用する COBOL プログラムを選択するようにメッセージが表示されます。その後、インターネット アプリケーション ウィザードにより、連絡節で宣言されたすべてのデータが表示されます。このデータは、手続き部 (PROCEDURE DIVISION) ヘッダの USING 句でも表示されます。これらのデータ項目を、入力パラメータ、出力パラメータ、またはその両方として選択することができます。

さらに、インターネット アプリケーション ウィザードでは、アプリケーションに対して、次のフォームやプログラムを生成します。

- 入力フォーム
- 出力フォーム (入力フォームと出力フォームが異なるように指定した場合)
- フォームとサブルーチン間の通信を処理する COBOL プログラム

クロスプラットフォームでの使用に適した HTML と、ダイナミック HTML (両者の違いについては、「フォームと HTML」を参照してください) のどちらかを選択してフォームを生成することができます。アプリケーションの基本フォームは、インターネット アプリケーション ウィザードにより作成されます。フォームは、Form Designer を使用して、希望の表示形式に編集することができます。

サブルーチンの使用方法は、次のとおりです。

1. 新しい NetExpress プロジェクトを作成し、サブルーチンをプログラム ファイルとして追加します。
2. インターネット アプリケーション ウィザードを起動し、「ウィザードへの入力」で [COBOL ソース ファイル] を選択します。

3. インターネット アプリケーション ウィザードにより表示されるメッセージにしたがって、必要な情報を入力してください。
4. ウィザードの「パラメータの割り当て」ページで、入力パラメータと出力パラメータを選択します (ウィザードによりプログラムの連絡節で指定されたパラメータのリストが表示されます)。各パラメータについて、パラメータを表わすコントロールの型と、コントロールを表示する入力フォーム、出力フォーム、または両方のフォームを選択することができます。
5. ウィザードの最後のページで、[完了] を押します。
この手順では、アプリケーションのファイルを作成し、NetExpress プロジェクトに追加します。
6. オプションとして、インターネット アプリケーション ウィザードで作成されたフォームの表示状態を、Form Designer を使用して修正することができます。
7. オプションとして、生成されたコードを編集し、アプリケーションを改良したり、手順 6 でフォームに追加された新しいコントロールの処理を行うことができます。
8. アプリケーションをリビルドします。

この章では、インターネット アプリケーション ウィザードを使用してプログラムを作成する方法やそれらの作成されたプログラムを編集する方法について手順と例を紹介します。この章では、「ここで [OK] ボタンをクリックします。」というような詳しい説明は行わず、全体的な過程について説明します。インターネット アプリケーション ウィザードや NetExpress に関する手順ごとの詳細説明については、オンライン ヘルプを参照してください。はじめにインターネット アプリケーション ウィザードと NetExpress の概略を理解するには、『入門書』のチュートリアルを参照してください。

6.2 新しいインターネット アプリケーションの作成

インターネット アプリケーションを作成する前に、次の点について決定する必要があります。

- クロスプラットフォーム出力とダイナミック HTML のどちらを使用してフォームを生成するか。
- 入力と出力に対して同じフォームを使用するか (対称アプリケーション)、または、別のフォームを使用するか (非対照アプリケーション)。

対照アプリケーションと非対照アプリケーションについては、「インターネット プログラミングの概要」の章を参照してください。

次の説明は、上級者を対象としています。手順ごとの詳しい説明が必要な場合には、NetExpress のオンライン ヘルプを参照してください。インターネット アプリケーション ウィザードや NetExpress を使用した経験がない場合には、『入門書』の「具体的な手順」を参照してください。

インターネット アプリケーション ウィザードでアプリケーションを作成する方法は、次のとおりです。

1. 使用するレガシー サブルーチンのソース ファイルを含む NetExpress プロジェクトを作成します。
2. インターネット アプリケーション ウィザードを起動します。
3. 「ウィザードへの入力」で [COBOL ソース ファイル] を選択します。
4. ウィザードは、アプリケーションの基礎となるサブルーチンのファイル名を入力するようにメッセージを表示します。ファイル名を直接入力するか、[参照] を押して該当するファイル名を選択することができます。
5. 入力と出力に対して同じフォームを使用するか (対称アプリケーション)、別のフォームを使用するか (非対称アプリケーション) を選択します。また、クロスプラットフォームとダイナミック HTML どちらを使用するかを選択します。
6. ウィザードは、入力フォームと出力フォームのタイトルを入力するようにメッセージを表示します。また、アプリケーションに対して生成するファイルのデフォルトのファイル名も表示します。このデフォルトは、変更することができます。
7. ウィザードは、プログラムのパラメータを、フォームを生成するための HTML コントロールにマップするようにメッセージを表示します。

この手順については、次項の「データの選択」でより詳しく説明します。

8. [完了] をクリックしてフォームを生成すると、COBOL サーバー側プログラムは元の COBOL プログラムにフォームをリンクします。

6.3 データの選択

インターネット アプリケーション ウィザードの「パラメータの割り当て」ページは、次のように表示されます。

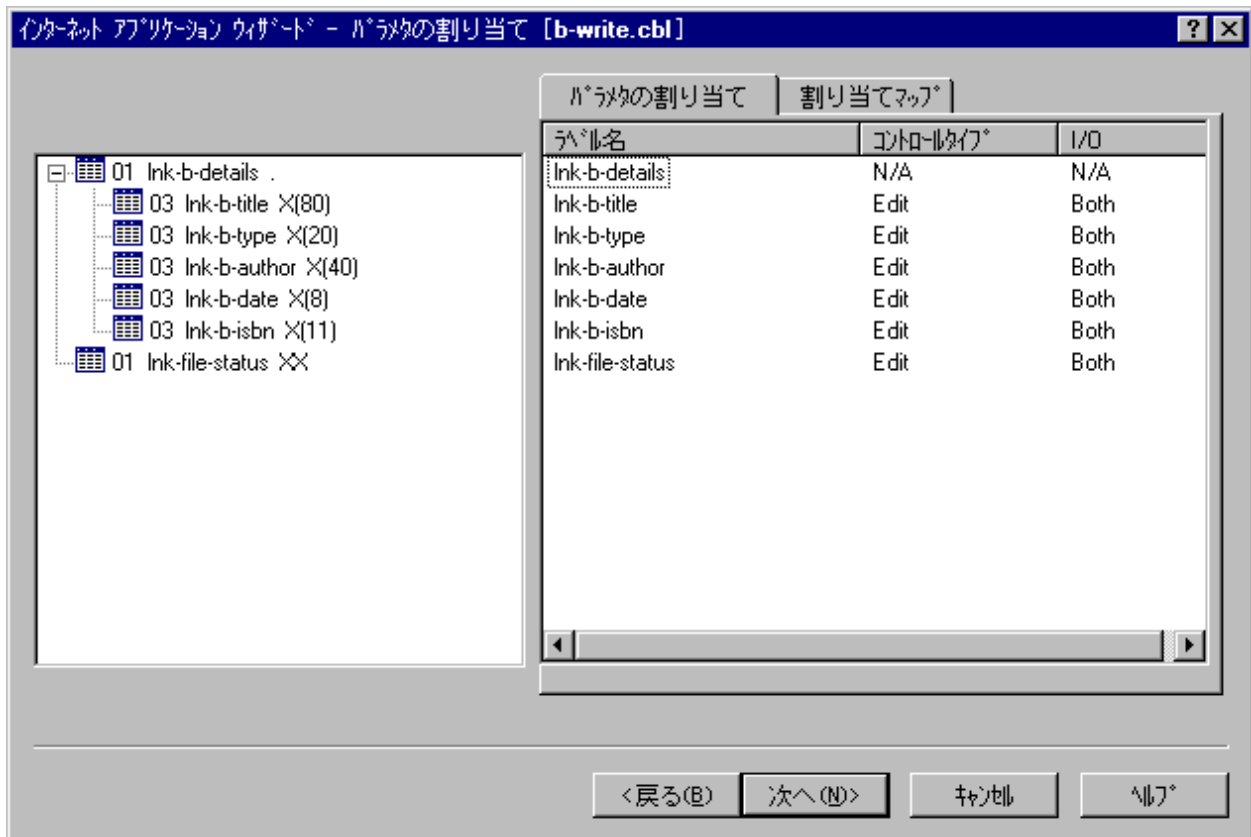


図 6-1 「パラメータの割り当て」ページ

選択したプログラムで手続き部 (PROCEDURE DIVISION) ヘッダの USING 句で使用される、連絡節のデータ項目は、次の情報とともに左側のツリービューに表示されます。

フィールド	説明
レベル番号	データ項目のレベル番号。ここに表示されるレベル番号は必ずしもソースコードのレベル番号と一致するとは限りません。グループ項目は、プラス記号で示されます。 グループ項目の横にあるプラス記号 (+) をクリックすると、グループの副項目のツリービューが展開されます。 たとえば、レベル 01、02、04 のグループ項目は、レベル 01、03、05 のグループ項目として表示されます。
フィールド	表示しているサブルーチンの COBOL データ項目の名前。
データ型	データ項目の COBOL ピクチャ文字列。 テーブルのデータ項目は、Occurs n で表します。 n は、項目の OCCURS 値です。再定義されるデータ項目は、Redefined という単

語で表します。他のデータ項目を再定義するデータ項目は、Redefines
という単語で表します。

右側の「パラメータの割り当て」ペインには、データをフォームのコントロールにマップする方法に関する情報が表示されます。

ラベル名	このデータ項目のフォームに表示されるキャプション。
コントロール型	フォームでこのデータ項目を表すために使用するコントロールの型。 インターネット アプリケーション ウィザードでは、テキスト エントリ フィールド (Edit として表示されます)、オプション ボタン (RadioButton として表示されます)、チェック ボックス (CheckBox として表示されます) をサポートしています。
データ方向 (I/O)	データは、フォームからプログラムへ (入力パラメータ)、プログラムからフォームへ (結果)、または、両方向に送信することができます。I/O エントリをクリックすると、次のリストが表示されるので、この中から選択することができます。 N/A データ項目がフォームで使用されていません。 In 入力パラメータ (入力と出力に別々のフォームを使用するアプリケーションで、入力フォームにだけこのフィールドが表示されます。) Out 結果 (入力と出力に別々のフォームを使用するアプリケーションで、出力フォームにだけこのフィールドが表示されます。) Both データ項目は、入力として送信され、結果として返されます (入力と出力に別々のフォームを使用するアプリケーションで、両方のフォームにこのフィールドが表示されます。)

最初は、すべてのパラメータのデータ方向は、Both に設定されています (グループの場合は N/A)。アプリケーションを作成するには、ユーザーがフォームで表示や変更を行う各データ項目について、ラベル、コントロール型、データ方向を指定する必要があります。

選択内容は、割り当てファイル (拡張子 .aht) に保存することができます。アプリケーションの生成後に選択を修正

する必要がある場合、同じ .cbl ファイルを使用してインターネット アプリケーション ウィザードを再起動し、.aht ファイルを再読み込みすると、前回終了した場所から作業を続けることができます。

6.3.1 エントリ フィールドへのフィールド割り当て

テキスト エントリ フィールドは、非常に簡単に使用できます。データがフォームからプログラムに送信されると、エントリ フィールドの内容が対応する COBOL データ項目に転記されます。インターネット アプリケーション ウィザードは、生成されたコードに基本的な確認手順を挿入します。受け取り側データ項目が数字 COBOL データ項目である場合に、エンドユーザーが入力したデータの変換が失敗すると、プログラムはエンドユーザーの Web ブラウザにエラー メッセージを返します。

データがプログラムからフォームに送信されると、データ項目の内容がエントリ フィールドに表示されます。数字データを、エントリ フィールドに表示できるリテラルに変換するためのコードは、インターネット アプリケーション ウィザードにより生成されます。

パスワード フィールドは、テキスト エントリ フィールドとまったく同じように処理します。唯一の違いは、パスワード フィールドは、ユーザーによるテキスト入力をアスタリスク (*) として表示する点です。ラベルと無効化された編集フィールドも、エントリ フィールドと同様に処理しますが、ユーザーがその内容を変更することはできません。すべてのブラウザが無効化された編集フィールドをサポートしているわけではありません。また、非表示フィールドという、ユーザー側のブラウザで表示されないフィールドもあります。

6.3.2 オプション ボタンへのフィールド割り当て

1 つの COBOL データ項目にオプション ボタンのグループを割り当てることができます。各オプション ボタンには、値が関連付けられています。データがフォームからプログラムに送信されると、現在選択されているオプション ボタンの値が COBOL データ項目に転記されます。データがプログラムからフォームに送信されると、COBOL データ項目の値と一致するオプション ボタンが選択されます。オプション ボタンと値は、「コントロール型」フィールドで [Selects] を選択したときに表示される [コントロール マップ] タブで設定します。

データを送受信するデータ項目には、オプション ボタンの有効な値を選択してください。たとえば、数字データ項目に "X"、"Y" および "Z" の値を設定した場合、エンドユーザーがどのオプション ボタンを選択しても、ランタイム エラー メッセージが返されます。

6.3.3 チェック ボックスへのフィールド割り当て

COBOL データ項目に関連付けられた値をチェックボックスに割り当てることができます。データがフォームからプログラムに送信されるときに、チェック ボックスが選択されていると、値は COBOL データ項目に転記されます。データがプログラムからフォームに送信されるときに、COBOL データ項目とチェック ボックスの値が一致すると、チェック ボックスが選択された状態が表示されます。「コントロール型」フィールドで [Checkbox] を選択したときに表示される [コントロール マップ] タブでオプション ボタンと値を設定します。

6.3.4 COBOL テーブルの割り当て

OCCURS 句で宣言された COBOL データ項目にテキスト エントリ フィールドだけ (パスワード フィールド、非表示フィールド、ラベル フィールド、無効フィールドもテキスト エントリ フィールドと見なされます) を割り当てることができます。OCCURS データ項目がグループ項目である場合、グループの副項目に対しても同じコントロールのサブセットを使用することができます。

COBOL テーブルの各コントロールは、HTML テーブルに配置され、OCCURS 句の値にしたがってエントリ フィールドを反復するように設定されます。

6.3.5 選択コントロールへのフィールド割り当て

COBOL データ項目に選択コントロールを割り当てることができます。このとき、コントロールにラベルと値の組み合わせを追加します。ブラウザでは、選択コントロールの中にラベルが表示されます。フォームが送信されるときに、選択されたラベルに対応する値が COBOL データ項目に渡されます。

6.4 アプリケーションの生成

アプリケーションに対してデータを選択すると、アプリケーションのファイルを生成することができます。インターネット アプリケーション ウィザードの最後のページで [完了] をクリックすると、ウィザードにより下表に示すファイルが生成されます。

この表中の名前は、インターネット アプリケーション ウィザードによるデフォルトの名前です。これらの名前は、ウィザードの途中にあるページで変更することができます。

ファイル	説明
<i>program_server.cbl</i>	フォームと元のサブルーチンをリンクするプログラム
<i>program_server.cpf</i>	フォームと送受信するデータのコピーファイル
<i>program_server.cpl</i>	サブルーチンのパラメータ データ
<i>program_server.cpv</i>	ブラウザとサブルーチン間のデータ変換ルーチン
<i>program_server.cpy</i>	アプリケーションのデータ
<i>program_input.htm</i>	アプリケーションの入力フォーム
<i>program_input.mff</i>	入力フォームの Form Designer ファイル。このファイルを Form Designer に読み込むと、インターネット アプリケーション ウィザードにより提示されるフォーム レイアウトを変更することができます。

<code>program_output.htm</code>	アプリケーションの出力フォーム (個別入出力フォーム アプリケーションを選択した場合のみ)
<code>program_output.mff</code>	出力フォームの Form Designer ファイル (個別入出力フォーム アプリケーションを選択した場合のみ)。このファイルを Form Designer に読み込むと、インターネット アプリケーション ウィザードにより提示されるフォーム レイアウトを変更できます。

インターネット アプリケーション ウィザードによりファイルが生成されると、実用可能なアプリケーションが完成します。この後は、Form Designer を使用して、生成されたフォームの修正や編集を行うことができます。

6.5 アプリケーション例の作成

この項では、2 つのサブルーチンに基づき、インターネット アプリケーション ウィザードで小さいアプリケーションを作成します。最初のサブルーチン `b-write.cbl` では、索引ファイルにレコードを書き込みます。2 番目のサブルーチン `b-read.cbl` では、キーによりレコードを検索します。レコードは、本のカタログに関するエントリです。各エントリには、次のフィールドがあります。

- Title (副キー)
- Author (副キー)
- Type
- Reprinting Date
- ISBN (主キー)

この例では、「ここで [OK] ボタンをクリックします。」というような詳しい説明は行わず、全体的な過程について説明します。インターネット アプリケーション ウィザードや NetExpress に関する手順ごとの詳細説明については、オンライン ヘルプを参照してください。はじめにインターネット アプリケーション ウィザードと NetExpress の概略を理解するには、『入門書』のチュートリアルを参照してください。

2 つのサブルーチンは、`¥netexpress¥base¥demo¥bookapp` に格納されています。まず、ファイルを書き込むためのアプリケーションとフォームを作成します。

1. NetExpress を起動し、フォルダ `¥netexpress¥base¥demo¥bookapp` からプロジェクト `bookwrit.app` を読み込みます。
2. 新しいインターネット アプリケーションを作成し、インターネット アプリケーション ウィザードで次のオプションを選択します。
 - COBOL ソース ファイル
 - COBOL ソース プログラムとして `b-write.cbl`

- 単一入出力フォーム
 - ダイナミック HTML
 - 入力タイトルとして「本レコードの追加」
3. インターネット アプリケーション ウィザードの「パラメタの割り当て」ページには、2 つの 01 データ項目が表示されます。



図 6-2 インターネット アプリケーション ウィザードで表示された連絡節のデータ項目

4. LNK-FILE-STATUS のラベルを File Status に変更し、データ方向を Out に設定します。
5. ツリー ビューで LNK-B-DETAILS の横にある + をクリックして、副項目を表示します。
6. [パラメタの割り当て] タブのフィールド ラベルを上から下に向かって設定します。
 - Title
 - Type
 - Author
 - Reprinting Date
 - ISBN

7. すべてのフィールドのデータ方向を In に設定します。
8. ツリー ビューのエントリを右クリックし、ポップアップ メニューで [割り当てファイルの保存] を選択します。[OK] をクリックします。

これで、データ割り当てがすべて保存されます。後でアプリケーションを再生成する場合は、COBOL サブルーチンを読み込んだ後で割り当てファイルを再度読み込みます。インターネット アプリケーション ウィザードは、割り当てファイル名がサブルーチン ファイル名と一致した場合に割り当てファイルを自動的に再読み込みします。

9. ウィザードの最後のページで [完了] をクリックします。

インターネット アプリケーション ウィザードによりファイルが生成され、プロジェクトに追加されます。

10. NetExpress の [プロジェクト] メニューで [リビルド] をクリックし、アプリケーションをビルドします。

今度は、ファイルを読み込むアプリケーションを作成します。

1. フォルダ %netexpress%base%demo%bookapp からプロジェクト bookread.app を読み込みます。
2. 新しいインターネット アプリケーションを作成し、インターネット アプリケーション ウィザードで次のオプションを選択します。

- COBOL ソース ファイル
- COBOL ソース プログラムとして b-read.cbl
- 単一入出力フォーム
- ダイナミック HTML
- 入力タイトルとして「本レコードの読み取り」

3. LNK-FILE-STATUS のラベルを File Status に変更し、データ方向を Out に設定します。
4. ツリー ビューの LNK-B-DETAILS の横にある + をクリックして、その副項目を表示します。
5. [パラメタの割り当て] タブのフィールド ラベルを次のように設定します。

- Title
- Type
- Author
- Reprinting Date
- ISBN

6. すべてのフィールドのデータ方向を [Both] に設定します (1 つのフィールドを検索キーとして入力すると、プログラムはすべてのフィールドのデータを返します)。
7. 割り当てを b-read.aht として保存します。
8. [完了] をクリックします。
9. NetExpress でアプリケーションをリビルドします。

これで、本レコード ファイルの更新と読み取りを行う 2 つの簡単なアプリケーションが完成しました。

6.5.1 アプリケーション例のテスト

この項では、アプリケーション例をテストし、レコードの追加と読み取りに使用します。

1. Web サーバーと Web ブラウザを起動して Web ブラウザにアプリケーションの URL を入力するか、NetExpress Animator を使用すると、アプリケーションを起動することができます。
 - 個々の構成要素を起動して、アプリケーションを起動する方法は、次のとおりです。
 - a. Solo Web サーバーを起動します。

別のサーバーを使用する場合は、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [プログラミング]、[インターネット アプリケーションのプログラミング]、[インターネット アプリケーションのデバッグ]、[方法]、[インターネット アプリケーションのアニメート] をクリックし、説明にしがいます。
 - b. Webブラウザのアドレス フィールドに次の URL を入力します。

`http://127.0.0.1/cgi-bin/b-write_server.exe`
 - NetExpress Animator によりアプリケーションを起動するには、[アニメート] メニューの [実行] をクリックします。ソース コードにブレークポイントがある場合、アプリケーションはブレークポイントで停止するため、コードのデバッグができます。

アプリケーションをどちらの方法で起動しても、Web ブラウザには次のようなフォームが表示されます。

図 6-3 「本レコードの追加」フォーム

2. フォームに詳細を入力し、[Submit] ボタンをクリックします。

Title	Moby Dick
Type	Fiction
Author	Melville
Reprinting Date	8feb96
ISBN	1

[Submit] ボタンをクリックすると、しばらくして、すべての入力内容が大文字に変換されたフォームが返されます。b-write サブルーチンは、b-read サブルーチンが簡単にキーを照合できるように、すべてのフィールドを大文字に変換します。レコードが正常に追加された場合、ファイル状態フィールドに「00」と表示されます。

3. さらに数冊に関して情報を作成し、レコードとして追加します。

データを入力したので、今度は、レコードを検索します。

1. Web ブラウザのアドレス フィールドに次の URL を入力します。

`http://127.0.0.1/cgi-bin/b-read_server.exe`

この操作により、レコード追加用のフォームに似たフォームが表示されます。

2. 「ISBN」フィールドに 1 を入力し、[Submit] ボタンをクリックします。

この操作により、*Moby Dick* のレコードが検索されます。

6.6 アプリケーションの拡張

ここまでで、フォームから直接取得したデータを 1 つのサブルーチンで処理する基本的なアプリケーションの作成方法を説明してきました。この項では、生成したフォームと COBOL プログラムを編集し、インターネット アプリケーション ウィザードで作成したアプリケーションの機能を拡張する方法について説明します。

6.6.1 COBOL プログラムの編集

インターネット アプリケーション ウィザードでは、アプリケーションの生成時に次の COBOL ソース コード ファイルが作成されます。

ファイル	説明
<i>program_server.cbl</i>	フォームと元のサブルーチンをリンクするプログラム
<i>program_server.cpf</i>	フォームと送受信するデータのコピーファイル
<i>program_server.cpl</i>	サブルーチンのパラメータ データ
<i>program_server.cpv</i>	ブラウザとサブルーチン間のデータ変換ルーチン
<i>program_server.cpy</i>	アプリケーションのデータ

編集する必要があるファイルは、*program_server.cbl* だけです。他のファイルは、すべて、インターネット アプリケーション ウィザードでアプリケーションを再生成するたびに自動的に上書きされます。ファイル *program_server.cbl* の内容は、次のとおりです。

- フォームとサブルーチンに関する全データの宣言
- フォームの入力データを受け付けるロジック
- フォームの入力データを英数字文字列からレガシー サブルーチンの形式に変換するロジック
- サブルーチンの呼び出し (CALL)
- サブルーチンの出力を表示可能な英数字に変換するロジック
- フォームを出力するロジック

プログラムでは、サブルーチン呼び出し (CALL) の前後に独自のアプリケーション ロジックを追加することができます。このとき、インターネット アプリケーション ウィザードにより、これらの場所を示すコードに "TO DO" コメントが挿入されます。

6.6.2 フォームの編集

Form Designer に生成されたフォームを読み込み、インターネット アプリケーション ウィザードにより作成されたデフォルト レイアウトを変更することができます。また、フォームに別のコントロールを追加することもできます。この場合、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 COBOL プログラムにロジックを追加する必要があります。

フォームを編集するためには、NetExpress の「プロジェクト」ウィンドウで .htm ファイルをダブルクリックします。これにより、Form Designer が起動し、フォームが読み込まれます。フォームの変更を保存すると、プログラムのコピーファイルは新規コントロールのデータ項目に更新されます。

6.6.3 アプリケーション例の拡張

この章でビルドしたアプリケーション例では、ファイル状態が 2 文字のコードとしてエントリ フィールドに返されます。ただし、ファイル状態コードがわからないと、役に立ちません。この項では、次のような状態を報告するメッセージを返すように、レコード読み取り用アプリケーションを変更します。

- レコードが見つかりました。
- レコードが見つかりません。
- ファイル操作に失敗しました。(レコードが見つかった場合と見つからなかった場合以外のファイル状態)

変更は、次の 3 段階で行います。

1. インターネット アプリケーション ウィザードによるサブルーチン b-read の割り当てから LNK-FILE-STATUS フィールドを削除し、変更します。

アプリケーションを再生成すると、このフィールドはフォームに存在しなくなります。

2. b_read_server.htm を Form Designer に読み込み、メッセージ用の新しいフィールドを追加します。
3. サブルーチンから返されるファイル状態コードをテストし、フォームに適切なメッセージを戻すコード b_read_server.cbl を追加します。

インターネット アプリケーション ウィザードによる割り当ての変更

b-read.cbl に対するインターネット アプリケーション ウィザードによる割り当ての変更手順は、次のとおりです。

1. NetExpress を起動し、フォルダ %netexpress%base%demo%bookapp からプロジェクト bookread.app を読み込みます。
2. 新しいインターネット アプリケーションを作成し、インターネット アプリケーション ウィザードで次のオプションを選択します。

- COBOL ソース ファイル
- COBOL ソース プログラムとして b-read.cbl
- 単一入出力フォーム
- ダイナミック HTML
- 入力タイトルとして「本レコードの読み取り」を入力

3. 「パラメタの割り当て」ページのツリー ビューで項目を右クリックします。ポップアップ メニューで [割り当てファイルを読み込む] をクリックし、「ファイルを開く」ダイアログ ボックスで b-read.aht を選択します。

この操作により、アプリケーションを最初に作成したときの割り当てをすべて再読み込みします。

4. LNK-FILE-STATUS の I/O 割り当てを N/A に変更します。

5. ウィザードの最後のページで [完了] をクリックします。

インターネット アプリケーション ウィザードにより、「LNK-FILE-STATUS」フィールドが省略された状態でこのアプリケーションのコピーファイルとフォームが再生成されます。

メッセージ フィールドの追加

フォームに新しいフィールドを追加する方法は、次のとおりです。

1. NetExpress の「プロジェクト」ウィンドウで b-read_server.htm をダブルクリックし、Form Designer を起動します。
2. フォームの最下部にテキスト ボックスを追加します。
3. 構成要素ツリー ビューで ID を resultfield に変更します。
4. プロパティを次のように設定します。
 - MaxLength を 30 に設定します。
 - COBOL Picture を X(30) に設定します。
5. 変更を保存し、Form Designer を閉じます。

プログラム例の変更

プログラムを変更して、新しいフィールドにエラー メッセージを返す方法は、次のとおりです。

1. NetExpress の「プロジェクト」ウィンドウで b-read_server.cbl をダブルクリックし、編集を開始します。

2. コメント "Add post-call application business logic here" を検索します。
3. 上記のコメントの後に、次のコードを追加します。

```
evaluate lnk-file-status

  when "00"

    move "読み取りが成功しました。" to resultfield

  when "23"

    move "見つかりませんでした。" to resultfield

  when other

    move "操作が失敗しました。" to resultfield

end-evaluate
```

4. プロジェクトをリビルドします。
5. アプリケーションを再実行します。

これで、アプリケーションは、ファイル状態を示すメッセージを返すようになります。

第7章 サーバー側のプログラミング

この章では、インターネット アプリケーションのサーバー側プログラムを作成する方法について説明します。この説明は、ISAPI、NSAPI、CGI のプログラミングに同じように適用できます。

7.1 概要

NetExpress を使用すると、COBOL の機能が拡張され、他の言語よりもインターネットのサーバー側プログラムを簡単に作成できます。ACCEPT 動詞を使用すると、フォームからデータを受け取ることができます。また、DISPLAY 動詞か、埋め込み HTML (EHTML)を使用すると、Web ブラウザに結果を返すことができます。EHTML では、COBOL プログラムの中から簡単に HTML ページを作成することができます。この章では、主に次の 4 分野について説明します。

- リソースの競合
- サーバー側プログラムへの入力
- サーバー側プログラムからの出力
- アプリケーション状態の維持

Form Designer を使用して HTML をペイントすると、インターネット アプリケーション ウィザードによりデータを読み取って結果を返すサーバー側プログラムのスケルトンが生成されます。この段階で、データを処理するための追加コードを書き込むことができます。つまり、アプリケーションの作成を開始するのに、入力を受け取って出力を返すための詳細な構文に関する知識は必要ありません。ただし、このような知識は、より高度なアプリケーションを作成する場合には便利です。

インターネット アプリケーション ウィザードにより生成されるサーバー側プログラムでは、フォームからのデータ受け取りには ACCEPT を使用し、フォームの表示には EHTML を使用します。

7.2 リソースの競合

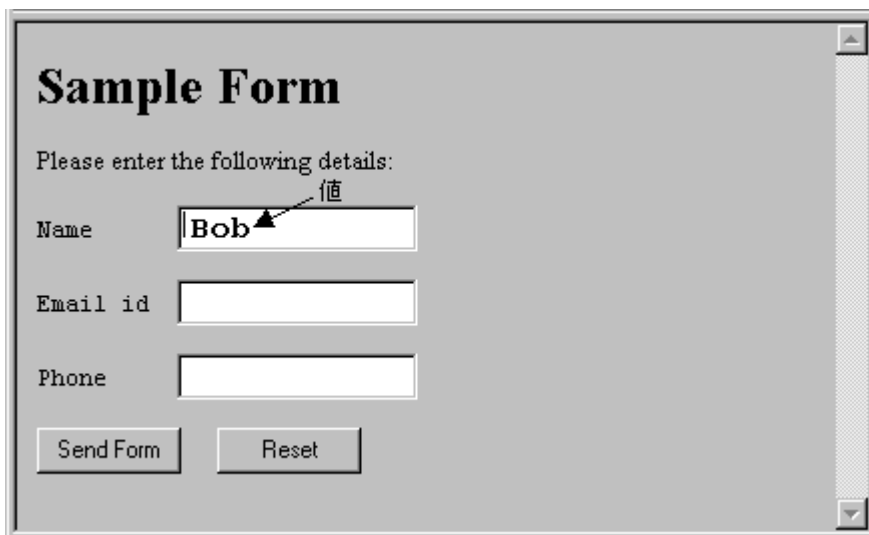
サーバー側プログラムは、別々のユーザーに対して同時に複数回、実行することができます。サーバー側プログラムが CGI プログラムである場合、プログラムが実行されるたびに、別のプロセスとして実行されます。サーバー側プログラムが ISAPI または NSAPI プログラムである場合、プログラムが実行されるたびに、別のスレッドで実行されます。サーバー側プログラムが共有リソースにアクセスする必要がある場合 (たとえば、ファイルやデータベースなど)、リソース競合を処理するためのロジックを記述することが必要です。

COBOL には、共有ファイルに対する関数セットが豊富にあります。詳細については、『ファイル処理』の「ファイルの共有」を参照してください。

7.3 サーバー側プログラムへの入力

フォームの各コントロールには、名前と値があります。「フォームと HTML」の章で説明したように、エンドユーザーがフォームを送信すると、フォームの情報は、名前と値の組み合わせとしてサーバー側プログラムに送信されます。COBOL 構文の拡張機能を使用すると、フォームのコントロールの名前を、サーバー側プログラムの COBOL データ項目に直接関連付けることができます。フォームのデータがサーバー側プログラムにポストされると、データ項目はフォームのコントロールの値に設定されます。

次のフォームには、「名前」という名前のフィールドがあります。この場合、エンドユーザーは、値 "Bob" を入力しています。



Working-Storage 節で次のように宣言すると、サーバー側プログラムを各名前付きコントロールに結合できます。

```
01 inputdata is external-form.
```

```
03 name-field      pic x(30) identified by "name".
```

```
03 email          pic x(15) identified by "emailid".
```

```
03 phone-no      pic x(30) identified by "phone".
```

エンドユーザーが [フォームを送信] ボタン (この例では、[送信] コントロール) をクリックし、サーバー側プログラムを実行すると、次の文が実行されたときに値 "Bob" がデータ項目 NAME-FIELD に転送されます。

```
accept input-form
```

Form Designer でフォームを作成すると、サーバー側のスケルトン アプリケーションが自動生成されます。その場合、フォームの各コントロールと照合するデータ項目を宣言するコピーファイルも生成されます。

7.3.1 構文

COBOL データ項目を CGI 入力にマップするには、次の構文を使用します。

```
level-number data-name-1 IS EXTERNAL-FORM.
```

この宣言では、グループ項目 *data-name-1* とその従属フィールドに HTML フォームからの入力により値を設定できることを示しています。基本データ項目を NAME 属性にマップする方法は、次のとおりです。

```
sub-level-number data-name-2 picture-string IDENTIFIED BY name.
```

パラメータの内容は、次のとおりです。

<i>sub-level-number</i>	COBOL データ項目のレベル番号
<i>data-name-2</i>	COBOL データ名
<i>picture-string</i>	COBOL ピクチャ文字列。PIC X(<i>n</i>)、PIC 9(<i>n</i>) および数字編集フィールドを使用できます。 <i>n</i> 文字よりも長い文字列は右端を切り捨てられ、 <i>n</i> 文字よりも短い文字列は左側から空白文字を埋められます。
<i>name</i>	フォームのコントロールに対応する Name プロパティまたは Groupname プロパティの値

7.3.2 例

この例は、フォームのフィールドにマップされるデータ項目セットを示します。

```
01 input-form is external-form.  
  
    03 name-field pic x(30) identified by "Name".  
  
    03 phone-no   pic x(30) identified by "Phone".  
  
    03 email      pic x(15) identified by "EmailID".
```

7.4 サーバー側プログラムからの出力

NetExpress では、COBOL プログラムから Web ブラウザに HTML を出力する方法が 2 種類あります。どちらの方法でも、COBOL 変数の値を HTML 出力に置き換えることができます。

- 埋め込み HTML (EHTML)

EHTML を使用すると、COBOL プログラムで EXEC HTML と END-EXEC の区切り文字の間に HTML を記述することができます。HTML 文は、コードをソース プログラムに直接作成するか、コピーファイルとしてインクルードすることができます。

EXEC HTML 機構は、HTML の出力に対してプログラムを完全に制御します。HTML ソースは、実際にはサーバー側プログラムにリンクされるため、HTML コピーファイルを変更すると、プログラムをリコンパイルしてから、再リンクする必要があります。

- DISPLAY

COBOL の DISPLAY 動詞を使用すると、外部ファイルに保持された HTML ページを表示することができます。

7.4.1 EHTML の使用方法

埋め込み HTML (EHTML) では、COBOL プログラムから直接、HTML を出力することができます。インターネット アプリケーション ウィザードにより生成されるサーバー側プログラムも EHTML を使用してエンドユーザーに結果を返します。

EHTML では、完全な HTML ページをコピーファイルとしてプログラム中にインクルードすることができるうえに、特別なデータ宣言を必要としないという利点があります。また、部分的な HTML ページをコピーファイルとして使用したり、個別の HTML 文を出力し、完全なプログラム制御のもとで完全なページをビルドすることもできます。このような機能により、柔軟性がたいへん高くなります。

EXEC HTML 文を使用して、COBOL プログラムに HTML を埋め込みます。この部分は、EHTML プリプロセッサ `htmlpp` により翻訳されます。

キーワード EXEC HTML と END-EXEC を使用すると、HTML を直接 COBOL ソース コードに埋め込むことができます。キーワード EXEC と HTML は、同じ行に記述する必要があります。その他は、書式の制約はありません。

次の書式は、一般的な書式です。

```
EXEC HTML
```

```
    [htmloutput]
```

```
    [copy "file.htm".]
```

```
END-EXEC
```

パラメータ内容は、次のとおりです。

htmloutput Web ブラウザへ出力する HTML マークアップ

file.htm HTML マークアップを含むファイル

EHTML プリプロセッサは、HTML マークアップの検証や解析は行いません。CGI プログラムを起動する Web サーバーに直接 HTML マークアップを出力するだけです。

固定フォーマットの COBOL ソース コードでは、EHTML プリプロセッサは、埋め込み HTML を出力するときに、列 8 を仮想列 1 として処理します。そのため、HTML の <PRE> タグを簡単に使用できます。ただし、拡張子が .htm のコピーファイルをインクルードした場合、プリプロセッサはコピーファイルを書式のないソースとして処理します。この動作を無効にするには、プリプロセッサへの指令 NOAUTOFORMAT を指定します。

7.4.1.1 代入マーカー

代入マーカーを使用すると、COBOL プログラムからの変数データを、直接 HTML 出力に代入することができます。代入マーカーは、次のように、コロン (:) が先頭に付く COBOL データ名です。

```
:data-name
```

表示項目は、常に、EHTML 中で使用する必要があります。バイナリのデータ項目を使用する場合、フォームに表示されたデータは判読できません。

コロンの後に空白文字または区切り文字マーカーが続く場合は、常にコロンとして出力され、代入マーカーとして処理されません。また、コロンの前に ¥ を付けても、コロンをコロンとして出力させることができます。

例

```
working-storage section.  
  
01 acct-code    pic 9(8).  
  
...  
  
procedure division.  
  
...
```

- *> 最初のコロンの後に空白文字が続くため、
- *> EXEC HTML では代入マーカーとして処理されません。
- *> 2 番目のコロンの後にはデータ名が続くので
- *> 代入マーカーとして処理されます。
- *> 3 番目のコロンの前には ¥ が付くため、
- *> コロンとして処理されます。

```
EXEC HTML
```

```
Account Code: :acct-code <BR>
```

```
    ¥:acct-code
```

```
END-EXEC
```

代入マーカーを修飾するには、グループ項目のデータ名と基本項目のデータ名の間にはピリオド (.) を使用します。たとえば、次のようになります。

```
working-storage section.
```

```
01 customer.
```

```
    03 name          pic x(80).
```

```
    03 acct-code    pic 9(8).
```

```
...
```

```
procedure division.
```

```
...
```

*> EXEC HTML ブロック中にある customer.acct-code は、

*> COBOL ソース中にある顧客の acct-code に相当します。

```
EXEC HTML
```

```
    Account Code: :customer.acct-code <BR>
```

```
END-EXEC
```

また、代入マーカーの参照を修正することもできます。

```
working-storage section.
```

```
01 acct-code    pic 9(8).
```

```
...
```

```
procedure division.
```

```
...
```

*> 参照を修正し、acct-code の

*> 最初の 4 文字を出力します。

```
EXEC HTML
```

Account Code: :acct-code(1:4)

END-EXEC

注記: *data-name* 参照が特定のテキスト文字列の後に続けて記述されている場合、代入マーカーとして処理されません。このようなテキスト文字列は多数あります。これらのテキスト文字列のリストは、オンライン ヘルプに示されています。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[埋め込み HTML]、[代入マーカー] の順に選択します。

7.4.1.2 EHTML プリプロセッサ指令

EHTML プリプロセッサを実行するには、次のコンパイラ指令を設定する必要があります。

```
preprocess(htmlpp) [preprocessor-directives] endp
```

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すべてのプログラムは、EHTML プリプロセッサを実行するための \$SET 文で起動します。

```
$SET preprocess(htmlpp) endp
```

また、htmlpp.dir と呼ばれる ASCII ファイルを作成し、NetExpress システムの \$COBDIR パスを与える方法でも、EHTML プリプロセッサ指令を設定することができます。

EHTML プリプロセッサ指令の説明については、オンライン リファレンスを参照してください。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[埋め込み HTML] の順に選択します。

7.4.2 DISPLAY 文の使用法

DISPLAY 動詞を使用すると、Web ブラウザに HTML ページを送信することができます。HTML ページには、プログラムから提供される変数データを格納する代入マーカーを含めることができます。代入マーカーは、標準 HTML ではありません。そのため、COBOL システムは、代入マーカーを変数データに置換してから、Web ブラウザにページを送信します。代入マーカーは、次の 2 つのうち、どちらかの形式をとります。

```
%%name%%
```

または

```
%%!s name%%
```

name は、データ宣言の IDENTIFIED BY 句により COBOL データ項目にマップされる名前です。データ項目をそのまま出力する場合は、`%%name%%` を使用します。データ項目から後続の空白を削除する場合は、`%%!s name%%` を使用します。標準テキストを出力する場合、Web ブラウザは、通常自動的に後続の空白文字を削除しますが、DISPLAY 動詞を使用してフォーム要素に値を入れる場合、この機能を使用して、後続の空白文字を削除することが必要です。

次の構文は、COBOL のグループ データ項目を出力ページにマップするための構文です。

```
level-number data-name-1 IS EXTERNAL-FORM identified by "pagefile.htm".
```

level-number グループ項目の COBOL データ項目レベル番号

data-name-1 COBOL データ名

pagefile.htm 出力する HTML ページのファイル名

この宣言では、HTML フォーム *pagefile.htm* にグループ項目 *data-name-1* とその従属フィールドからのデータ セットを使用するように指定しています。基本データ項目をフォーム名にマップする方法は、次のとおりです。

```
sub-level-number data-name-2 picture-string IDENTIFIED BY name.
```

sub-level-number COBOL データ項目のレベル番号

data-name-2 COBOL データ名

picture-string COBOL ピクチャ文字列。PIC X(*n*)、PIC 9(*n*) および数字編集フィールドを使用できます。*n* 文字よりも長い文字列は右端を切り捨てられ、*n* 文字よりも短い文字列は左側から空白文字で埋められます

name 出力する HTML ページの代入マーカの名前

7.4.2.1 例

この例は、フォームのフィールドにマップされるデータ項目セットを示します。

```
01 output-form is external-form identified by "outpage1.htm".
```

```
    03 name-field pic x(30) identified by "Name".
```

```
    03 phone-no   pic x(30) identified by "Phone".
```

```
    03 email      pic x(15) identified by "EmailID".
```

次のコードは、上記の名前に対して代入マーカを設定した HTML ページの例です。

```
<HTML><BODY>
```

```
<H1>これは、output-form です。</H1>
```

<P> %%Name%% 様</P>

<P> あなたが入力した情報は、次のとおりです。 </P>

 電話番号 - %%Phone%%

 電子メール ID - %%EmailID%%

</BODY></HTML>

7.5 アプリケーションの状態の維持

Web ベースのアプリケーションで問題となるのは、アプリケーションの状態を維持することです。CGI プログラムは、実行時に、前回の処理内容や現在プログラムを使用しているクライアントを記憶していません。プログラムを使用する各クライアントについてアプリケーションの状態を維持するには、次の 2 つの機構を使用します。

- 非表示フィールド

状態情報は、エンドユーザーのブラウザに送信されるフォームの、ブラウザに表示されないフィールドに保存されます。非表示フィールドは、フォームをサポートするブラウザならどれでも機能します。ただし、状態をセッション間で維持することはできません。これは、エンドユーザーのマシンからフォームがなくなると、すぐに、すべての状態情報が失われるためです。

- Cookie

エンドユーザーが自分のブラウザを他のページやサイトに送信してから、アプリケーションに戻る場合でも、状態情報が維持されます。Cookie は、セッション間でも状態を維持できます。ただし、エンドユーザーは cookie をサポートするブラウザを使用する必要があります。

これらの機構を使用する場合、2 つの問題があります。

- ネットワークを介して大量の状態情報を送信すると、アプリケーションの実行速度が遅くなる場合があります。ダイヤルアップ回線で接続されたインターネット クライアントの場合は、特に顕著です。
- セキュリティやその他の理由で、イントラネットやインターネットで詳細情報を送信したくない場合があります。

NetExpress には、すべての状態情報をサーバーに記憶させ、サーバー側プログラムによりこの情報に素早くアクセスできる追加機構があります。ランタイム システムの call-by-name ルーチンを使用すると、サーバー側プログラムは、接続する各クライアントに一意的クライアント ID を要求し、索引ファイルにクライアントの状態データを格

納します。

下図は、クライアント ID を含む cookie とフォームをクライアントのブラウザに送信するサーバー側プログラムを示します。クライアントのブラウザから次に Web サーバーに要求を出すときに cookie を戻します。サーバー側プログラム（最初のプログラムである場合と別のプログラムである場合があります）は、クライアント ID を使用して状態データを格納したレコードを検索します。

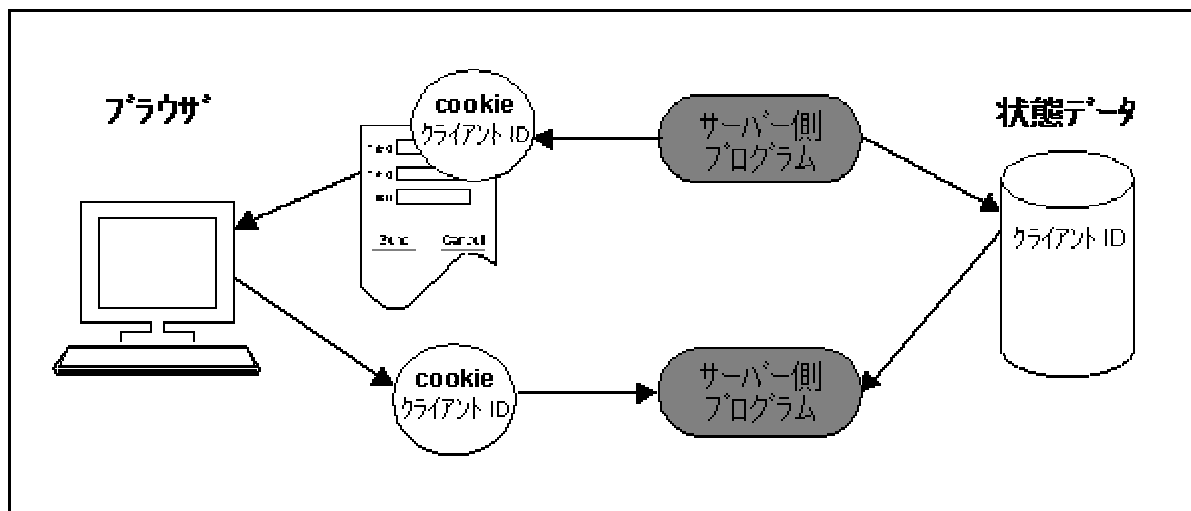


図 7-1 サーバー側の状態ファイル

注記: 状態ファイルに重要な情報を格納する場合（たとえば、クレジットカードの明細など）、暗号化のようなセキュリティ手段を追加で実装するか、セキュリティが確保されたネットワーク リンクだけを介してアプリケーションにアクセスするすることが必要です。悪意のあるユーザーは、フォームを再送信する前に別のクライアント ID を捏造し、他人の状態レコードにアクセスする可能性があります。その場合に、悪意のユーザーが実際の状態情報を表示できるかどうかは、アプリケーションの設計や Web ブラウザに返される情報の種類によって決まります。

次の 3 項では、次の内容を説明します。

- 非表示フィールドの使用方法
- cookie の使用方法
- NetExpress のサーバー側状態機構の使用方法

7.5.1 非表示フィールド

HTML フォームでは、非表示フィールドを使用することができます。HTML フォームで非表示フィールドを追加してみてください。非表示フィールドは、HTML 入力（エン트리 フィールド）と同様に機能します。ただし、フォームを読み込むエンドユーザー側では表示されません。

フォームを入力フォームや出力フォームとして使用するサーバー側プログラムを生成すると、エントリ フィールドの場合と同様に、非表示フィールドに対応する COBOL データ項目が生成されます。フォームの出力時に非表示フィールドにアプリケーションの状態情報を格納し、フォームが別のサーバー側プログラムに送信されたときに読み返すことができます。

7.5.2 cookie

cookie は、HTML ページと一緒に Web ブラウザに送信される名前と値の組み合わせです。cookie にアプリケーションの状態情報を記録し、次に特定のクライアントがサーバー側プログラムにアクセスするときにこの情報を検索することができます。Netscape Navigator 2.0 以降のバージョンと Microsoft Internet Explorer 3.0 以降のバージョンは、ともに cookie をサポートしています。他のブラウザでは、サポートしていない場合があります。

cookie の完全な規格は、Netscape サイトに掲載されています。ここをクリックしてください。

デフォルトでは、エンドユーザーがブラウザを閉じるまで、cookie は Web ブラウザに存在します。cookie には有効期限を設定することができます。その場合、ブラウザによりキャッシュに記憶された情報が有効期限まで維持されます。

Form Designer で HTML ページを設計する場合、[ページ] メニューの [cookie] オプションを使用して、cookie をページに関連付けることができます。cookie には、名前、値、COBOL ピクチャ、有効期限、パス、ドメイン、セキュリティを設定することができます。ドメイン属性とパス属性を使用してページ間で cookie を共有することもできます。「cookie」ダイアログ ボックスで [OK] をクリックすると、Form Designer により、サーバー側とクライアント側の両方で cookie の設定と読み取りに必要なコードがほぼすべて作成されます。

- クライアント側では、ページの読み込み時やアンロード時に呼び出される JavaScript コードが生成されます。このコードは、読み込み時に cookie の値を取得し、JavaScript 変数に格納して、アンロード時に記憶します。
- サーバー側では、「cookie」ダイアログボックスで指定した COBOL ピクチャ データ項目が各 cookie に対して生成されます。また、cookie を読み込むための ACCEPT 文が作成されます。HTML ページがサーバーで受け付けられると、cookie がデータ項目に代入されます。出力ページが Web ブラウザに送信されると、データ項目が cookie に代入されます。

Form Designer の Cookie Editor に関する詳細は、次の箇所を参照してください。まず、[ヘルプ] メニューの [ヘルプ トピック] をクリックし、[キーワード] タブで [cookie] と入力して、「検索結果」ダイアログ ボックスで [cookie を追加する] を選択します。

ページの表示中に cookie に新しい値を入力するためには、JavaScript コードをページに追加して、cookie に値を割り当てる必要があります。JavaScript コード作成の詳細については、「クライアント側のプログラミング」の章を参照してください。

7.5.2.1 cookie の例

この例では、HTML ページを作成し、cookie と関連付けます。ここでは、cookie をページに対するアクセス回数を保存するために使用します。サーバー側プログラムは、カウンタの値を増やします。この例では、「ここで [OK] ボタンをクリックします。」というような詳しい説明は行わず、全体的な過程について説明します。インターネット アプリケーション ウィザードや NetExpress に関する手順ごとの詳細説明については、オンライン ヘルプを参照してください。はじめにインターネット アプリケーション ウィザードと NetExpress の概略を理解するには、『入門書』のチュートリアルを参照してください。

cookie 例を作成し、設定する方法は、次のとおりです。

1. PAGECNTR.app という新しいプロジェクトを作成します。
2. 空白のテンプレートを選択して、HTML ページを新規作成します。
3. 見出しを作成し、ユーザーにページのヒット回数を知らせるフォームをペイントします。そのとき、送信ボタンもフォームに含めます。下図で「page」と「times」の間にあるフィールドは、テキスト入力フィールドです。このフィールドに "hitcounter" という名前を付けます。フォームは、図 7-2 のように作成します。

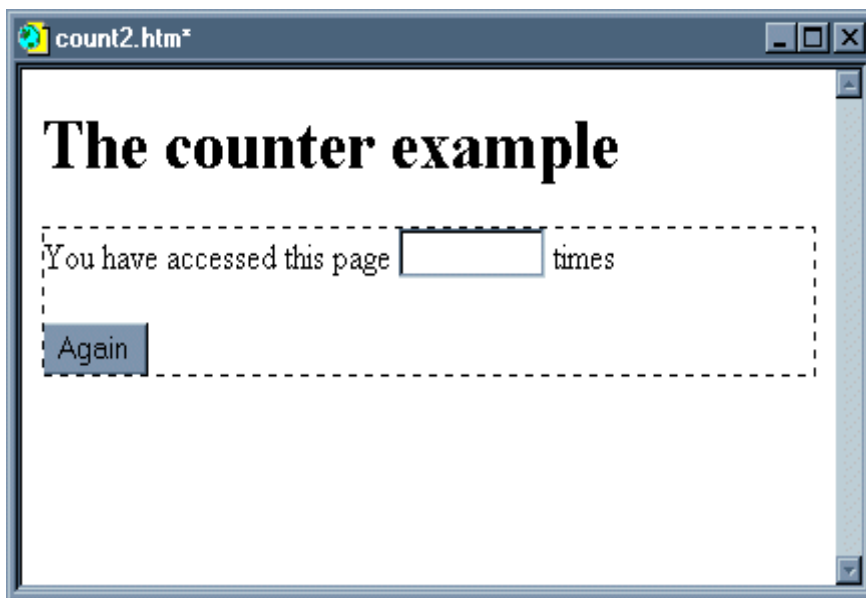


図 7-2 Cookie の例

4. [ページ] メニューの [Cookies] オプションを使用して、名前に cookiecounter を、値に 0 を、COBOL ピクチャに 9999 を設定した cookie をページに追加します。
5. [ファイル] メニューの [新規作成] をクリックし、インターネット アプリケーションを選択して、インターネット アプリケーション ウィザードを起動します。
6. 最初のページでサーバー プログラムを選択します。
7. 「サーバー プログラムの生成」ページで、「入力ファイル/フォーム」と「出力フォーム」のリストで両方

の HTML ページを選択します。

8. [次へ] をクリックし、最後のページで [完了] を選択します。インターネット アプリケーション ウィザードにより、サーバー側プログラムと、関連付けられたフォーム データ用コピーファイルが生成されます。
9. COBOL プログラムを開き、process-business-logic 節を検索します。
10. 既存のコードの前に、次のコードを入力します。

```
add 1 to cookiecounter  
move cookiecounter to hitcounter
```

11. COBOL ファイルを保存し、プロジェクトをリビルドします。
12. アプリケーションを実行します。Solo を使用している場合は、[アニメート] メニューで [実行] をクリックするだけです。上図の [再送信] をクリックするたびに、hitcounter フィールドに 1 ずつ加えられます。(アプリケーションの実行の詳細については、「新規アプリケーションの作成」の章の「アプリケーションの実行」を参照してください。)

7.5.3 サーバー側の状態機構

サーバー側の状態機構を使用すると、定義したレコード形式でアプリケーションの状態に関する情報を保存することができます。この機構を使用すると、常にサーバー側プログラムによりアプリケーションの最初のフォームを出力し、セッションの開始時にクライアント ID を割り当てる必要があります。

cookie を使用してクライアント ID を保存する場合、状態情報がユーザーの現在のセッションよりも長い間保存されるように有効期限情報を設定することができます。つまり、アプリケーションを永続化させることができます。この場合、ユーザーは、アプリケーションを翌日に再起動し、前回作業していた場所に戻ることができます。永続的なアプリケーションに対して作成されたプログラムは、クライアント ID を持つ cookie をテストして、初めて実行されるものであるかどうかを判断します。クライアント ID の値が空白文字である場合、ID を割り当て、新しいセッションを開始する必要があります。それ以外の場合は、状態情報を復元して、適切なフォームを表示する必要があります。

注記: サーバー側の状態機構は、sstate と呼ばれるモジュールにより実装されます。sstate のソース コードは提供されるので、暗号化のような機能を追加する場合は、ソース コードのサブルーチンを修正します。

ソース コード sstate.cbl は、¥netexpress¥base¥demo¥sstate フォルダにあります。変更を加える場合、新しい NetExpress プロジェクトを作成し、sstate.cbl をコピーします。アプリケーションで新しい sstate モジュールを使用するには、sstate.obj バージョンと sstate.gnt バージョンをビルドします。sstate.obj を ¥netexpress¥base¥lib に、sstate.gnt を ¥netexpress¥base¥bin にコピーします。

データ アクセス ウィザードで生成されたアプリケーションは、sstate に完全に依存しています。インターフェイス

は sstate に変更しないでください。インターフェイスを sstate に変更すると、データ アクセス ウィザードで作成したアプリケーションでは、予期しない障害が発生することがあります。すべての sstate 呼び出しへのインターフェイスについては、オンライン ヘルプを参照してください。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[ライブラリ ルーチン]、[関数ごとのライブラリ ルーチン] を選択します。最後に [状態の維持] へのリンクをクリックしてください。

7.5.3.1 クライアント状態レコードの保存と検索

次のフローチャートは、サーバー側の状態機構を使用するためのロジックを示します。各手順の詳しい実行方法については、下図のボックスをクリックしてください。

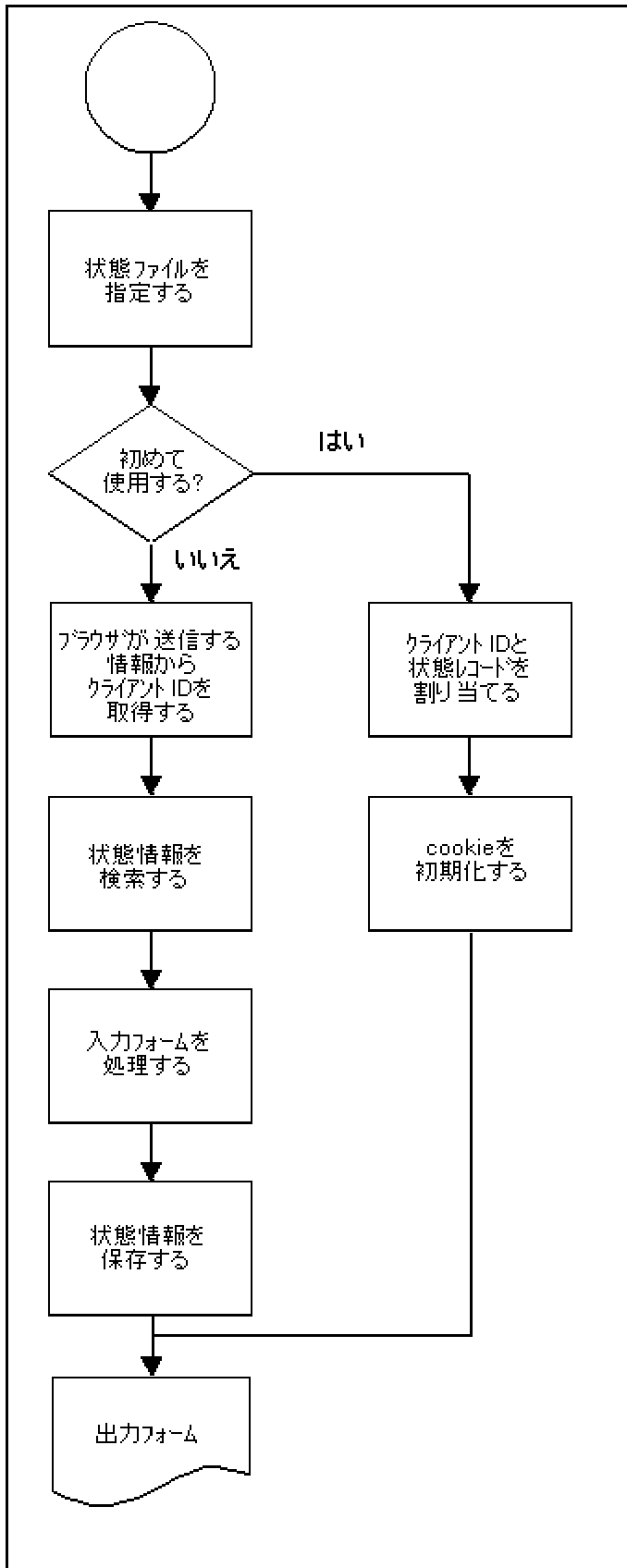


図 7-3 サーバー側状態機構の使用方法

次の説明は、上記のフローチャートに記載されている情報を要約したものです。

1. 状態ファイルの名前を指定します。

"MF_CLIENT_STATE_FILE"ルーチンを呼び出します。

状態ファイルを指定するには、ルーチン MF_CLIENT_STATE_FILE を指定します。たとえば、次のように記述します。

```
working-storage section.  
  
...  
  
01 state-status          pic x comp-x.  
  
01 state-filename       pic x(255)  
  
           value "MF-STATE-SAVE.DAT".  
  
...  
  
procedure division.  
  
...  
  
call "MF_CLIENT_STATE_FILE" using state-filename  
  
           server-status  
  
...
```

2. 初めてサーバー側状態機構を使用する場合、"MF_CLIENT_STATE_ALLOCATE"ルーチンを呼び出し、新しいクライアント ID を割り当てます。次に、cookie または非表示コントロールを初期化し、手順 6 に進みます。

クライアント ID を割り当てるには、ルーチン MF_CLIENT_STATE_ALLOCATE を使用します。たとえば、次のように記述します。

```
working-storage section.  
  
...  
  
01 client-id            pic x(30).  
  
01 client-length       pic xxxx comp-x.
```

```

01 state-status          pic x comp-x.

...

procedure division.

...

call "MF_CLIENT_STATE_ALLOCATE"

    using client-id client-length state-status

...

```

ルーチン MF_CLIENT_STATE_ALLOCATE の実行内容は、次のとおりです。

- 状態情報に対して、CLIENT_LENGTH で長さを指定した新しい空レコードを割り当てます。
- CLIENT-ID に渡された値が空白文字に設定されている場合、新しいクライアント ID を返します。CLIENT-ID に有効なクライアント ID が設定されている場合、この呼び出しは、state-status にエラーコードを返します。

ユーザーのマシンにクライアント ID を保存するために cookie を使用している場合、クライアント ID をもつ cookie の出力に使用しているデータ項目を初期化します。cookie は、プログラムによりフォーム出力と共に送信されます。

以前にサーバー側状態機構を使用している場合は、ブラウザが返す cookie または非表示コントロールからクライアント ID を読み取り、手順 3 に進みます。

ユーザーのブラウザによりサーバーに送信されたデータからクライアント ID を読み取ります。クライアント ID は、アプリケーション設計に応じて、非表示コントロール、または cookie に格納されています。どちらの技法が使用された場合でも、ACCEPT 動詞を使用してデータを取得することができます。たとえば、次のように記述します。

```

01 inputdata is external input-form.

    03 name-field        pic x(30) identified by "name".

    03 phone-no         pic x(30) identified by "phone".

    03 email            pic x(15) identified by "emailid".

    03 client-id       pic x(30) identified by "clientid".

```

```
...  
procedure division.
```

```
...
```

```
accept inputdata
```

```
...
```

このコードは、"clientid" という非表示フィールドまたは cookie からクライアント ID を取得します。

3. クライアント ID のレコードを読み取ります。

"MF_CLIENT_STATE_RESTORE" ルーチン呼び出します。

状態情報を検索するには、ルーチン MF_CLIENT_STATE_RESTORE を呼び出します。たとえば、次のように記述します。

```
working-storage section.
```

```
...
```

```
01 client-id          pic x(30).
```

```
01 client-length     pic xxxx comp-x.
```

```
01 state-status      pic x comp-x.
```

```
01 client-state.
```

```
03 ...*> クライアント状態レコードには、どのような
```

```
    *> 形式を定義してもかまいません。client-length
```

```
    *> フィールドの長さを保存します。
```

```
...
```

```
procedure division.
```

```
...
```

```
call "MF_CLIENT_STATE_RESTORE"
```

```
    using client-id client-state
```

client-length state-status

...

4. 必要に応じて状態情報を使用して、入力フォームを読み取り、処理します。

この段階では、フォームからの全情報と状態維持ファイルから検索した状態情報を取得していることとなります。そのため、ビジネス ロジックは、この情報を処理できるようになります。

5. クライアントの状態を保存します。

"MF_CLIENT_STATE_SAVE" ルーチン呼び出します。

ルーチン MF_CLIENT_STATE_SAVE を使用して状態ファイルの情報を更新することができます。たとえば、次のように記述します。

```
working-storage section.
```

```
...
```

```
01 client-id          pic x(30).
```

```
01 client-length     pic xxxx comp-x.
```

```
01 state-status      pic x comp-x.
```

```
01 client-state.
```

```
03 ...*> クライアント状態レコードには、どのような
```

```
    *> 形式を定義してもかまいません。client-length
```

```
    *> フィールドの長さを保存します。
```

```
...
```

```
procedure division.
```

```
...
```

```
call "MF_CLIENT_STATE_SAVE"
```

```
    using client-id client-state
```



```
client-length state-status
```

...

6. ブラウザに次のフォームを返します。

アプリケーションの次のフォームを出力します。クライアント ID をフォームの非表示コントロールに格納している場合、この情報によりフォームを初期化する必要があります。クライアント ID を cookie に格納している場合は、ユーザーに cookie を再送信する必要はありません。ユーザー側で cookie が存続する期間は、最低でも cookie が送信されてからユーザーがブラウザを閉じるまでの間です。cookie に有効期限を設定すると、より長く存続させることができます。この方法については、「Cookie」の項で説明しています。

これらのルーチンすべてについては、オンライン リファレンスを参照してください。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[ライブラリ ルーチン]、[関数ごとのライブラリ ルーチン] の順に選択します。最後に [状態の維持] をクリックしてください。次の例は、状態維持ルーチンを使用したプログラムのスケルトンです。

```
working-storage section.
```

...

```
01 state-filename          pic x(255) value
                            "MF-STATE-SAVE.DAT".
```

```
01 client-length          pic xxxx comp-x.
```

```
01 client-state.
```

```
    03 accessCount        pic 9(9).
```

```
01 state-status           pic x comp-x.
```

```
01 browserInput is external-form.
```

```
    03 client-id          pic x(30) identified by "clientid". *> cookie
```

...

```
procedure division.
```

*> クライアント状態ファイルを開きます。

```
    call "MF_CLIENT_STATE_FILE" using state-filename
```

```

...

accept browserInput      *> クライアント ID を持つ

                           *> cookie を読み取ります。

move length of client-state to client-length

  if client-id = spaces

*>      cookie は、空白です。 - cookie を受け取るのは初めてです。

        call "MF_CLIENT_STATE_ALLOCATE"

                using client-id client-length state-status

        ... *> その他は、初めて処理します。

  else

*>      クライアント ID を使用して、状態を復元します。

        call "MF_CLIENT_STATE_RESTORE" using client-id

                client-state

                client-length

                state-status

  end-if

...

*> データを処理し、クライアントに cookie を含むフォームを返します。

...

*> クライアント状態を保存します

        call "MF_CLIENT_STATE_SAVE" using client-id client-state

                client-length state-status

```

7.5.3.2 クライアント状態レコードの削除

クライアント状態レコードが不要になり、削除する場合、2つのルーチンが関係します。一方のルーチンは、1つのレコードを削除するもので、もう一方のルーチンは、特定の日数が経過したレコードを削除します。

1つのレコードを削除するには、"MF_CLIENT_STATE_DELETE" を呼び出します。たとえば、次のように記述します。

```
working-storage section.  
  
...  
  
01 state-status          pic x comp-x.  
  
01 client-id             pic x(30).  
  
01 state-filename        pic x(255) value  
                          "MF-STATE-SAVE.DAT".
```

```
procedure division.  
  
...
```

*> クライアント状態ファイルを開きます。

```
call "MF_CLIENT_STATE_FILE" using state-filename  
  
...  
  
call "MF_CLIENT_STATE_DELETE " using client-id  
  
server-status
```

特定の日数が経過したレコードをすべて削除するには、"MF_CLIENT_STATE_PURGE" を呼び出します。たとえば、次のように記述します。

```
working-storage section.  
  
...  
  
01 state-status          pic x comp-x.  
  
01 age-in-days           pic x(4) comp-x.  
  
01 state-filename        pic x(255) value
```

```
"MF-STATE-SAVE.DAT".
```

```
procedure division.
```

```
...
```

*> クライアント状態ファイルを開きます。

```
call "MF_CLIENT_STATE_FILE" using state-filename
```

```
...
```

move 5 to age-in-days *> 6 日以上経過したレコードをすべて削除します。

```
call "MF_CLIENT_STATE_PURGE " using age-in-days
```

```
server-status
```

これらのルーチンについては、オンライン リファレンスで詳細に説明しています。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[ライブラリ ルーチン]、[関数ごとのライブラリ ルーチン] の順に選択します。最後に [状態の維持] へのリンクをクリックしてください。

第8章 CGI、ISAPI、および NSAPI プログラム

この章では、サーバー側プログラムに対する 3 種類の API の相違点と、コンパイラ指令を変更するだけでこれらの API を切り替える方法について説明します。

8.1 概要

サーバー側プログラムの実行に使用できる API は、現在 3 種類です。

- CGI (Common Gateway Interface)

これは最も古い API で、すべての Web サーバーによりサポートされています。また、デバッグには最も簡単に使用できるため、サーバー側プログラムはすべて CGI を使用して開発します。

- ISAPI (Internet Server API)

ISAPI は、Microsoft 社の API で、CGI プログラムをさらに高速化したものです。ISAPI は、Microsoft Internet Servers やその他のベンダの Web サーバーでサポートされています。

- NSAPI (Netscape Server API)

NSAPI は、Netscape 社の API で、CGI プログラムを高速化したものです。NSAPI は、Netscape Web サーバーやその他のベンダの Web サーバーでサポートされています。

ISAPI と NSAPI のサーバー側プログラムは、次の 2 つの理由で、標準的な CGI アプリケーションよりも高速に起動します。

- ISAPI や NSAPI を使用すると、サーバー側プログラムは、Web サーバーにより個別のスレッドとして起動されます。標準的な CGI プログラムは、個別のプロセスとして起動されます。スレッドは、新規プロセスよりも高速に起動することができます。
- サーバー側プログラムは、ダイナミック リンク ライブラリ (.dll ファイル) としてコンパイルされます。この場合、プログラムが一度実行された後でもサーバーは読み込まれたままになります。そのため、その後の再起動はより高速になります。

ISAPI プログラムと NSAPI プログラムは、個別のプロセスではなく、個別のスレッドとして実行されるため、マルチスレッド プログラムであることが必要です。REENTRANT(2) コンパイラ指令を設定すると、COBOL プログラムをマルチスレッド化することができます。コンパイラ指令設定の詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[コンパイラ指令] の順に選択して、表示される説明を参照してください。

8.2 ISAPI および NSAPI サーバー側プログラムの作成

ISAPI アプリケーションと NSAPI アプリケーションは、入出力に ACCEPT/DISPLAY と EHTML を使用し、CGI アプリケーションと同じ方法で作成することができます。サーバー側プログラムは、まず、CGI プログラムとして開発してデバッグし、正常に機能することを確認した段階で ISAPI プログラム、または NSAPI プログラムに変更することをお勧めします。ISAPI を実行する Web サーバーは、バグに強くありません。ISAPI プログラムが破損すると、Web サーバー ソフトウェアがロックされ、強制的に再起動する必要がある場合があります。

1. サーバー側プログラムを CGI プログラムとして作成します。データ ファイルなどに対してリソースの競合を認めることを忘れないでください。リソースの競合は、すべてのサーバー側プログラムに影響します。

ISAPI プログラムは、Web サーバーのサービス プロセスの一部として実行されます。ISAPI プログラムは、サービスの中で実行されるため、Windows NT のシェルにはアクセスしないでください。たとえば、ウィンドウやダイアログボックスなどを作成しないでください。

ISAPI プログラム、または NSAPI プログラムでは、STOP RUN を使用しないでください。これらのプログラムは、web サーバー プロセスの中で実行されるため、STOP RUN によってサーバーが停止してしまうことがあります。ISAPI プログラムや NSAPI プログラムから制御を返す場合は、常に EXIT PROGRAM または GOBACK を使用します。

2. プログラムを CGI プログラムとしてビルドし、デバッグします。
3. プログラムで保護違反の原因となるエラーが発生しないことを確認できたら、後述の説明にしたがって、ISAPI プログラム、または、NSAPI プログラムとしてリビルドします。
4. 適用する API をサポートする Web サーバーでプログラムをテストします。
5. プログラムを実装します。

次の 3 項では、CGI プログラムを ISAPI プログラム、または、NSAPI プログラムとしてリビルドするための COBOL コンパイラ指令の変更方法と NetExpress のビルド設定について説明します。

8.2.1 ISAPI 用コンパイラ指令の設定

次のコンパイラ指令を設定してから、ISAPI プログラムをリビルドします。

- WEBSERVER(ISAPI)

このプログラムが ISAPI アプリケーションであることをコンパイラに伝えます。

- CASE

外部シンボル (呼び出されたプログラムの名前を含む) が大文字に変換されないようにします。この操作を行わないと、エンドユーザーが ISAPI を実行しようとしたときに、Web ブラウザに「見つかりません。」

というメッセージが表示されます。

- REENTRANT(2)

このプログラムの複数のコピーを確実に安全に実行できるようにします。

アプリケーションで Open ESQL を使用する場合、次の指令も設定する必要があります。

- SQL(THREAD=ISOLATE)

各スレッドの SQL リソースやトランザクションを他のスレッドと共有しないようにします。

インターネット アプリケーション ウィザードでビルドされたデータ アクセス アプリケーションは、すべて Open ESQL を使用します。

プログラムの冒頭に \$SET 文を記述すると、プログラムをコンパイルするたびにこれらの指令を必ず設定することができます。たとえば、次のように記述します。

```
$set webserver(isapi) case reentrant(2)
```

ドル記号 (\$) は、必ずソース コードの 7 列目に記述します。ただし、記述する列の指定を解除する指令 SOURCEFORMAT"FREE" を設定した場合を除きます。コンパイラ指令設定の詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[コンパイラ指令] の順に選択して、表示される説明を参照してください。

8.2.2 NSAPI 用コンパイラ指令の設定

次のコンパイラ指令を設定してから、NSAPI プログラムをリビルドします。

- WEBSERVER(NSAPI,entry-point-name)

このプログラムが NSAPI アプリケーションであることをコンパイラに伝えます。値 *entry_point_name* はユーザー定義の名前です。コンパイラは、*entry_point_name* で非表示のエントリ ポイントを作成し、NSAPI Web サーバーがプログラムを起動できるようにします。

このアプリケーションを NSAPI サーバーで実装する場合、サーバーの obj.conf ファイルを修正するときに *entry_point_name* を Init fn の funcs 属性の値として使用します。「アプリケーションの実装」の章の「NSAPI サーバー構成ファイルの修正」の項を参照してください。

- CASE

外部シンボル (呼び出されたプログラムの名前を含む) が大文字に変換されないようにします。この操作を行わないと、エンドユーザーが NSAPI を実行しようとしたときに、Web ブラウザに「見つかりません。」というメッセージが表示されます。

- REENTRANT(2)

このプログラムの複数のコピーを確実に安全に実行できるようにします。

アプリケーションで Open ESQL を使用する場合、次の指令も設定する必要があります。

- SQL(THREAD=ISOLATE)

各スレッドの SQL リソースやトランザクションを他のスレッドと共有しないようにします。

インターネット アプリケーション ウィザードでビルドされたデータ アクセス アプリケーションは、すべて Open ESQL を使用します。

プログラムの冒頭に \$SET 文を記述すると、プログラムをコンパイルするたびにこれらの指令を必ず設定することができます。たとえば、次のように記述します。

```
$set webserver(nsapi,run_update_1) case reentrant(2)
```

ドル記号 (\$) は、必ずソース コードの 7 列目に記述します。ただし、記述する列の指定を解除する指令 SOURCEFORMAT"FREE" を設定した場合を除きます。コンパイラ指令設定の詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[コンパイラ指令] の順に選択して、表示される説明を参照してください。

8.2.3 ISAPI および NSAPI プログラムのリンク

CGI プログラムをビルドする場合は、.exe ファイルとしてビルドします。ISAPI プログラムと NSAPI プログラムは、.dll ファイル (ダイナミック リンク ライブラリ) としてビルドします。詳細については、「アプリケーションの実装」の章の「共有ランタイム システムを使用する ISAPI または NSAPI プログラムのビルド」の項で詳しく説明します。

CGI サーバー側プログラムを ISAPI または NSAPI の .dll ファイルとしてリビルドすると、そのプログラムを参照する Web ページまたはフォームのすべての URL を変更する必要があります。プログラムが CGI の場合、参照する URL は次のようになります。

```
/share-name/program.exe
```

ここでは、次の URL を参照するように変更します。

```
/share-name/program.dll
```


第9章 Form Designer による出力の編集

この章では、HTML ページの作成時に Form Designer により出力されるファイルと、Form Designer を他の HTML 編集ツールと併用する方法について説明します。この章では、HTML については説明しません。

9.1 概要

Form Designer では、HTML ページのすべての部分を編集したり、表示することができますが、他のツールでページを作成し、フォームの追加やクライアント側のスクリプト作成にだけ Form Designer を使用する方法もあります。

Form Designer に読み込むページは、次の 2 種類のファイル形式をとります。

- .htm ファイル

これは、ページのテキスト情報とレイアウト情報をすべて格納した HTML ファイルです。すべてのフォーム レイアウト情報は、2 つの標準的な HTML 形式 (テーブルとダイナミック HTML) のどちらかでページに記述されます (「フォームと HTML」の章を参照してください)。

- .mff ファイル

.mff は、Form Designer に固有のファイル形式です。このファイルには、インターネット アプリケーション ウィザードを使用してページのフォームから作成したサーバ側プログラムで COBOL データ項目を宣言するために必要な情報が記述されます。たとえば、ページの各 HTML コントロールに関する COBOLPicture プロパティは、このファイルに格納されます。

Form Designer と他の HTML エディタの間では、どちらの方向へもフォームを転記することができます。

9.1.1 Form Designer への HTML ページの転記

Form Designer では、有効な HTML ページを NetExpress のプロジェクト ディレクトリにコピーし、プロジェクトに追加して、ダブルクリックすると、このページを開くことができます。このページに該当する .mff ファイルがない場合、Form Designer により作成されます。この段階で、このページの変更や新しいフォームの追加が可能になります。ページに既にフォームが含まれている場合には、フォームのコントロールに関する COBOL プロパティを設定するだけで、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ページを準備することができます。

注記: コントロールのデータ名は、コントロールの HTML Name 属性から適用されます。Form Designer では、ページを開いたときに、Name 属性が COBOL の予約語に設定されているかどうかを確認しません。COBOL の予約語と同じ名前をもつコントロールがある場合、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このフォーム

から生成された COBOL アプリケーションには、構文エラーが発生します。その場合、コントロール名を修正してください。

9.1.2 Form Designer からの HTML ページの転記

Form Designer の .htm ファイルは、標準 HTML なので、情報を失うことなく、他の HTML エディタに読み込むことができます。ただし、他のエディタが固有のファイルに情報を格納する場合、フォームを他のエディタにインポートし、.htm ファイルの情報からその固有ファイルを更新する必要があります。

第10章 クライアント側のプログラミング

この章では、フォームのイベントを処理する方法やサーバー側プログラムを実行せずに他の処理を行う方法を説明します。

オンライン ヘルプでは、この章で説明されている機能の使用方法について、手順ごとの詳細説明を掲載しています。NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [プログラミング]、[インターネット アプリケーションのプログラミング]、[Form Designer]、[JavaScript とイベント処理]、[概要 - スクリプト アシスタント] の順で選択してください。

10.1 概要

このマニュアルでは、フォームのデータを処理するサーバー側プログラムを作成し、結果を返すことを中心に説明しています。ただし、フォーム自体に対話型コードを入れることもできます。このようなコードを作成するには、Netscape 社の Web ブラウザでサポートされている簡単なスクリプト言語である JavaScript を使用するか、Microsoft 社の Web ブラウザでサポートされている JavaScript の変形である JScript を使用します。フォームに対話型コードを追加する場合は、JavaScript または JScript の関数を作成し、これをフォームの イベント にリンクします。イベントにリンクされた関数をイベント ハンドラと呼びます。たとえば、エンドユーザーが特定のチェック ボックスをクリックしたときに画像を表示するようなイベント ハンドラを作成することができます。

フォームに対話型コードを使用する例としては、入力データの確認があげられます。ただし、NetExpresss には、必要に応じて呼び出すことができる組み込みの確認関数があるため、多くの一般的な確認については、JavaScript や JScript でコードを作成する必要はありません。詳細については、「フォームの確認」の章で説明します。

Form Designer のスクリプト アシスタントでは、マウス ポインタを合わせてクリックするだけで、イベント ハンドラを作成することができます。JavaScript や JScript の知識がなくても、このように便利なイベント ハンドラをフォームに追加することができます。たとえば、フォームにデフォルトのデータを設定するコード、または、エンドユーザーを別の場所へ送信するためのコードを追加することができます。

スクリプト アシスタントにより生成されたコードは、JavaScript と JScript の構文規則にしたがっていますが、Microsoft インターネット エクスプローラで機能するコードは、Netscape Navigator で機能しないことがあります。また、反対に Netscape Navigator で機能するコードが、Microsoft インターネット エクスプローラで機能しないこともあります。これは、2 つのブラウザが異なる「ドキュメント オブジェクト モデル」をもつためです。

イベント ハンドラのコードは、たいへん複雑である場合があります。たとえば、条件文やループを含む場合です。ただし、より高度なコードを記述する場合は、この章で説明する以上に JavaScript や JScript について学ぶ必要があります。主にインターネット エクスプローラのユーザーを対象にコードを作成する場合は、Microsoft 社の Web サイトにある「Microsoft JScript Documentation」を参照してください。また、主に Netscape 社のブラウザのユーザーを対象にコードを作成する場合は、Netscape 社 Web サイトにある「Netscape JavaScript Documentation」を参照してください。ただし、クロスプラットフォーム互換性に関する問題に注意してください。

この章で説明する JavaScript リファレンスは、JScript にも同様に適用されます。

10.2 JavaScript の概要

JavaScript は、オブジェクト指向型言語で、その文はメソッド、プロパティ、スタイルを指定することができるオブジェクトで動作します。オブジェクトには、フォームのコントロールだけでなく、すべてのテキスト要素、文書自体、表示用ウィンドウも含まれます。メソッドは、「フォーカスを失う」、「選択される」などの、オブジェクトが実行できる動作です。プロパティとスタイルは、オブジェクトの属性です。ダイナミック HTML の出現により、新しくウィンドウを呼び出して簡単に追加の HTML 要素を作成し、フィールド内容の設定からテキストや背景の配色変更まで、広範に表示内容や動作を変更することができるようになりました。

HTML コントロールと ActiveX コントロールの両方に使用できるイベント ハンドラを作成することもできます。HTML コントロールのイベント ハンドラを作成すると、HTML 文書の 2 箇所 JavaScript が表示されます。

- JavaScript の関数は、文書の <HEAD> 部分で <SCRIPT> タグと </SCRIPT> タグの間にまとめて表示されます。
- 関数呼び出しは、フォームに含まれるコントロールの属性値で、文書の <BODY> 部分に表示されます。

ActiveX コントロールのイベント ハンドラを作成すると、JavaScript は、文書の <HEAD> 部分で <SCRIPT> と </SCRIPT> の間にだけ表示されます。

Java アプレットのイベント ハンドラを作成することもできます。メソッドを呼び出し、アプレットに属するスタイルやプロパティを取得し、設定することができます。アプレットの JavaScript コードは、文書の <HEAD> 部分で <SCRIPT> と </SCRIPT> タグの間にだけ表示されます。

HTML 文書の中に JavaScript コードを記述するかわりに、拡張子 .js を付けた別のファイルに JavaScript コードを格納することができます。別のファイルを使用する利点は、コードを複数のフォームから呼び出すことができることです。1 つのフォームで、別の .js ファイルから関数を呼び出すことも、埋め込み関数を記述することもできます。

スクリプト アシスタントを使用すると、文書の右側に JavaScript コードが自動生成されます。

10.3 スクリプト アシスタント

Form Designer のスクリプト アシスタントを使用すると、次のようなイベント ハンドラを簡単に設定することができます。

- メソッドを呼び出す。
- プロパティを取得し、設定する。
- スタイルを取得し、設定する

スクリプト アシスタントには、次の 5 つのタブがあります。

- [イベント] タブ
- [メソッド] タブ
- [プロパティ] タブ
- [スタイル] タブ
- [スクリプト] タブ

[イベント] タブ、[メソッド] タブ、[プロパティ] タブ、[スタイル] タブでは、マウス ポインタを合わせてクリックするだけでイベント ハンドラを作成することができます。一方、[スクリプト] タブでは、直接、コードを入力し、編集することができます。

タブの内容と スクリプト アシスタント の動作は、HTML コントロールと ActiveX コントロールのどちらを使用するかによって異なります。

10.3.1 [イベント] タブ、[メソッド] タブ、[プロパティ] タブ、および [スタイル] タブ

[イベント] タブ、[メソッド] タブ、[プロパティ] タブ、[スタイル] タブの表示内容は、似ており、4 つのペインを持っています。そのうち、次の 3 つのペインは、これら 4 つのタブすべてで同じです。

- オブジェクト ビューを含む左上端のペイン
- 右側の中央にあるヘルプ ペイン
- イベント ハンドラのコードを表示する最下部のペイン

4 番目のペインは右上端にありますが、選択したタブに応じて、イベント、メソッド、プロパティ、スタイルのビューが表示されます。ヘルプ ペインには、選択したイベント、メソッド、プロパティ、または、スタイルに関する情報が表示されます。

各タブには、タブと対話できる 1 つまたは複数のコントロールがあります。

図 10-3 には、HTML コントロールを含むフォームの [イベント] タブを示します。

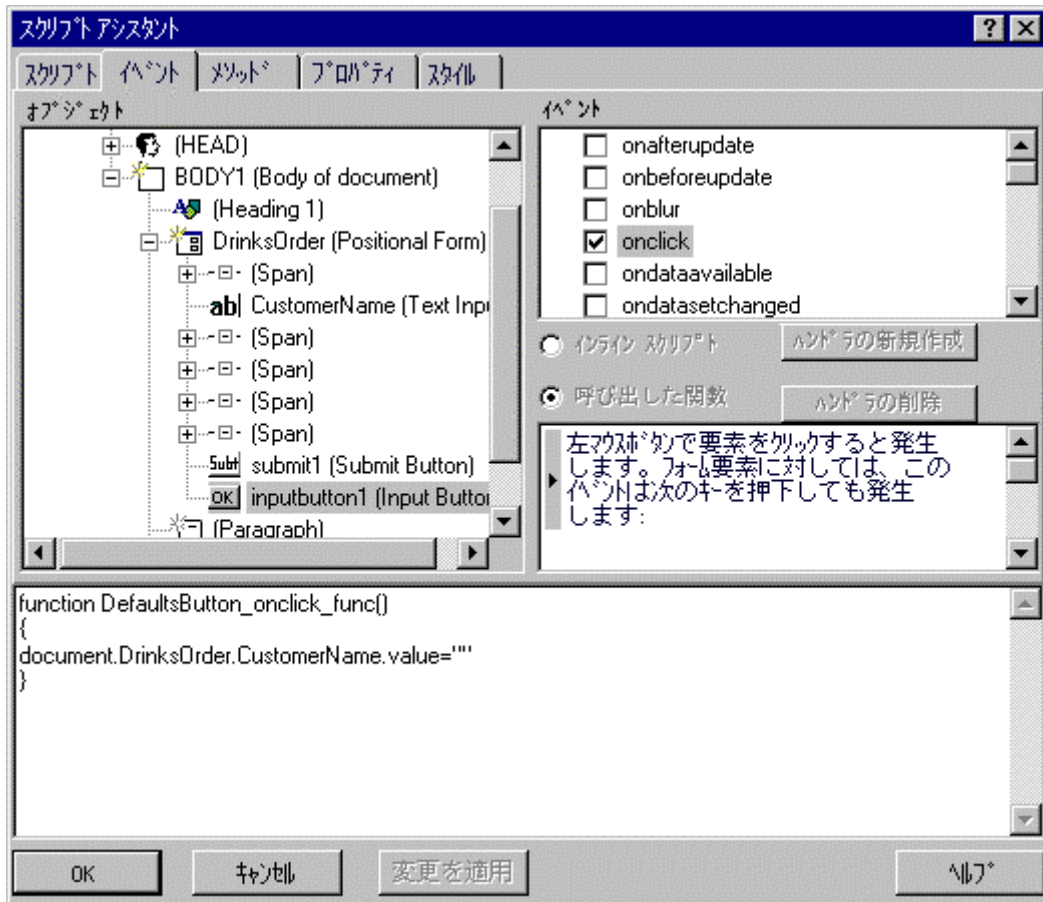


図 10-1 [イベント] タブ

また、[イベント] タブには、次のようなコントロールがあります。

- 新しいイベント ハンドラを作成するコントロール
- イベント ハンドラを削除するコントロール。関数だけ、または、関数と関数を呼び出すコードであるインライン スクリプトの両方を、削除することができます。
- 関数、または、インライン スクリプトを表示するコントロール

[イベント] タブを最初に表示したときは、これらのコントロールは淡色表示されています。[ハンドラの新規作成] を選択する前に、オブジェクトとイベントを選択する必要があります。

下図は、[メソッド] タブを示します。

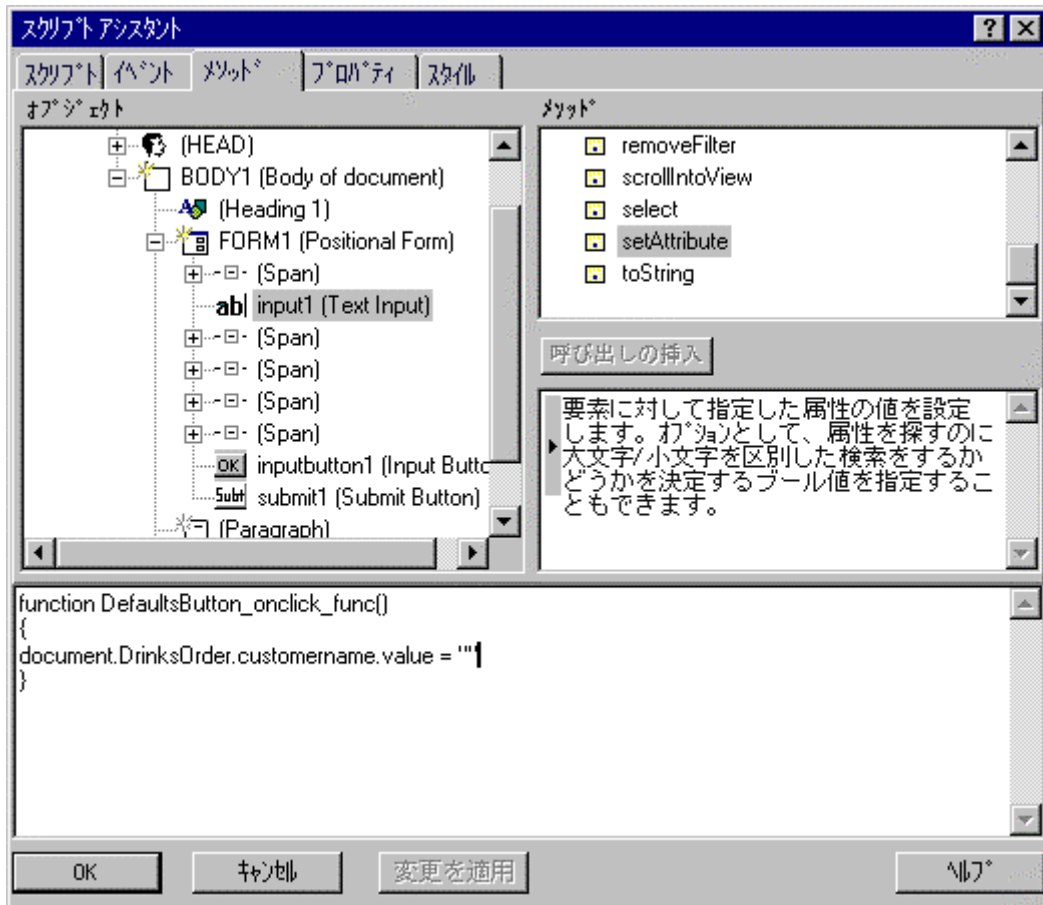


図 10-2 [メソッド] タブ

[メソッド] タブには、メソッド呼び出しを挿入するコントロールがあります。このコントロールは、メソッドを選択するまで淡色表示されています。

下図は、[プロパティ] タブを示します。[スタイル] タブは、右上のペインにプロパティではなく、スタイルが一覧表示される点を除くと、[プロパティ] タブと同じです。

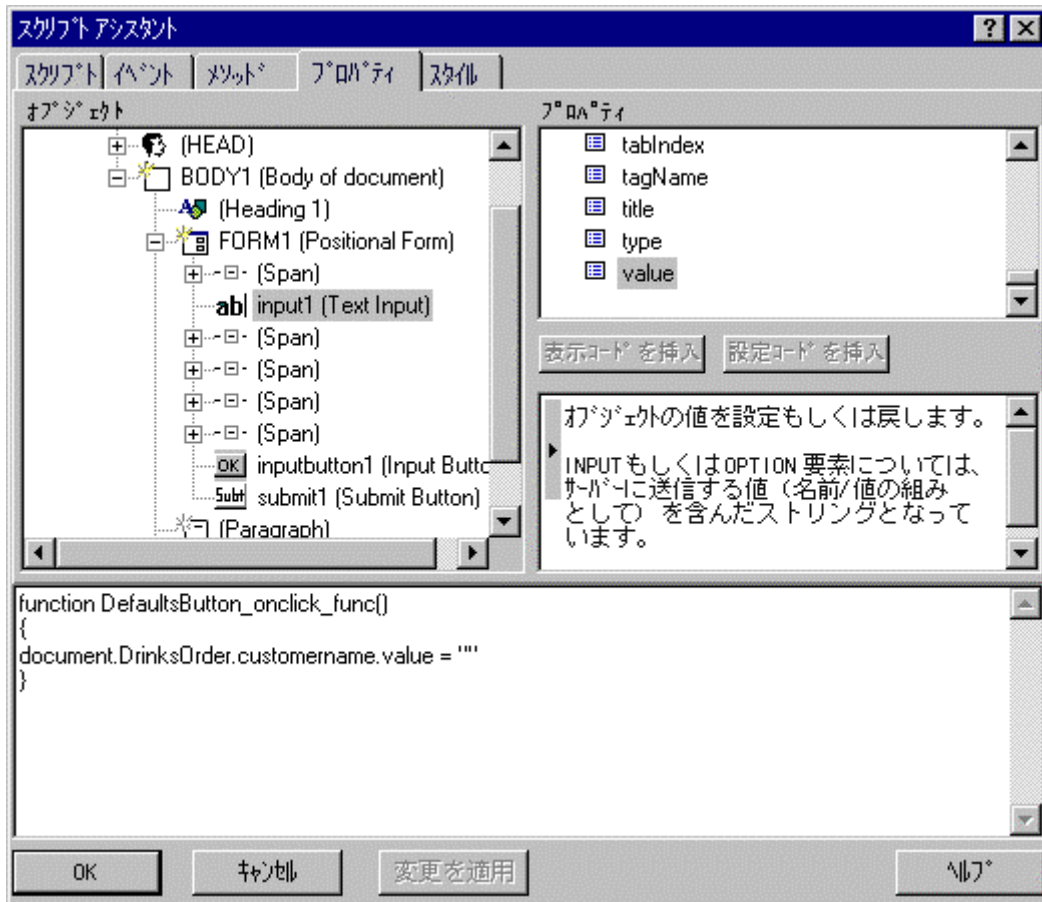


図 10-3 [プロパティ] タブ

[プロパティ] タブと [スタイル] タブには、次のようなコントロールがあります。

- プロパティまたはスタイルを取得するコードを挿入するコントロール
- プロパティまたはスタイルを設定するコードを挿入するコントロール

これらのコントロールは、プロパティまたはスタイルを選択するまで淡色表示されています。設定できないプロパティもあります。

これ以降の項では、次のビューの詳細について説明します。

- オブジェクト ビュー
- イベント ビュー
- メソッド ビュー
- プロパティ ビュー
- スタイル ビュー

- イベント ハンドラ コード ビュー

10.3.1.1 オブジェクト ビュー

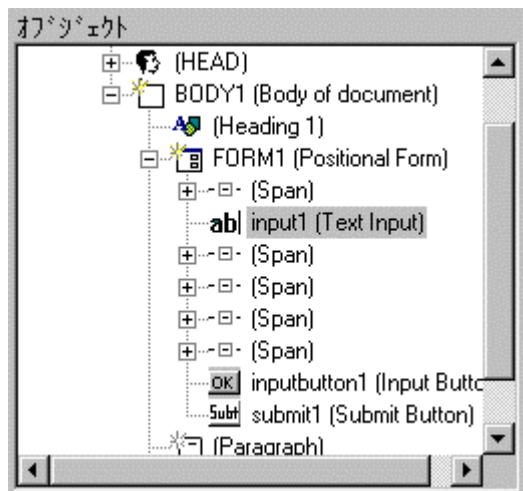


図 10-4 オブジェクト ビュー

オブジェクト ビューでは、HTML 文書のすべての要素がツリー表示されます。要素の隣にあるプラス記号 (+) をクリックすると、その要素に従属する要素が表示されます。

10.3.1.2 イベント ビュー

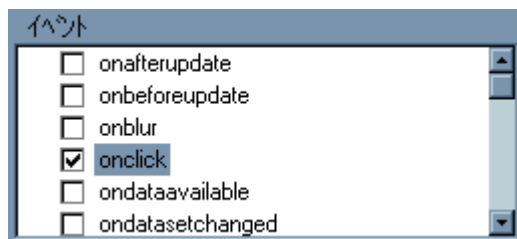


図 10-5 イベント ビュー

イベント ビューでは、オブジェクト ビューで選択されたフォームのコントロールで使用可能なすべてのイベントが一覧表示されます。上図は、HTML のオプション ボタンに対するイベントを最初からいくつか表示したものです。

10.3.1.3 メソッド ビュー

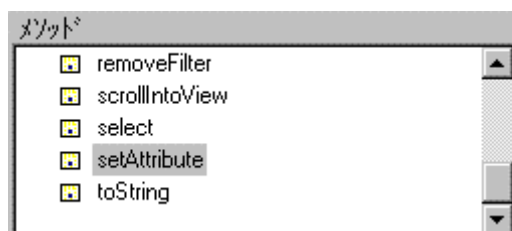


図 10-6 メソッド ビュー

メソッド ビューには、オブジェクト ビューで選択された HTML 要素で使用可能なすべてのメソッドが一覧表示されます。上図は、オプション ボタンのメソッドを最初からいくつか表示したものです。

コントロール上でメソッドを呼び出すと、何らかの動作が発生します。たとえば、入力テキスト フィールドで blur メソッドを呼び出すと、フォーカスを失います。

10.3.1.4 プロパティ ビュー

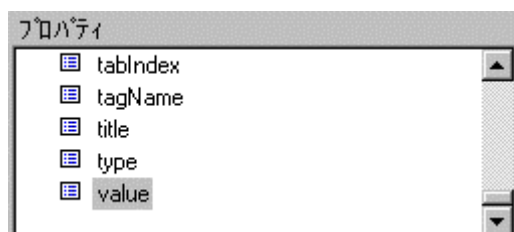


図 10-7 プロパティ ビュー

プロパティには、コントロールの状態に関する情報が含まれます。コントロールの状態を変更するためにプロパティを設定したり、コントロールの状態を調べるためにプロパティを取得することができます。たとえば、ほとんどの ActiveX コントロールと HTML コントロールには、Value プロパティがあります。テキスト フィールドの内容を検索するには、Value プロパティを取得します。別の内容を設定するには、Value プロパティを設定します。上図は、オプション ボタンのプロパティを最初からいくつか表示したものです。

10.3.1.5 スタイル ビュー

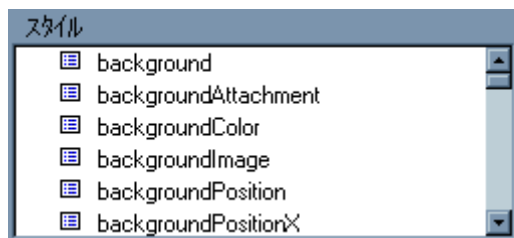


図 10-8 スタイル ビュー

スタイルには、要素の表示形式に関する情報が含まれます。プロパティと同様に、要素の表示形式を変更するためにスタイルを設定したり、表示形式を調べるためにスタイルを取得することができます。たとえば、ほとんどの ActiveX コントロールと HTML コントロールには、背景色と呼ばれるスタイルがあります。背景色を調べるには、backGround 色スタイルを取得します。配色を設定するには、背景色スタイルを設定します。上図は、HTML プッシュボタン コントロールのスタイルを最初からいくつか表示したものです。

プロパティと、スタイルの大半を設定する場合、ダイアログ ボックスに文字列を入力して、設定します。配色スタイルを設定する場合は、48 の基本色と 16 のカスタム色が表示されます。つまり、新しいカスタム色を定義することができます。ただし、JavaScript でサポートされている色は限定されており、サポートされていないカスタムカラーを定義すると、サポートされている色の中で最も似た色が使用されます。サポートされている色のリストについては、Netscape またはインターネット エクスプローラのリファレンスを参照してください。

10.3.1.6 イベント ハンドラ ビュー

```
function DefaultsButton_onclick_func()
{
document.all.FORM1.CustomerName.value = ""
}
```

図 10-9 イベント ハンドラ ビュー

イベント ハンドラ ビューには、イベント ビューで現在選択されているイベントのイベント ハンドラ コードが表示されます。上図は、プッシュボタンの Click イベントに関する JavaScript イベント ハンドラを示します。このコードは、別のコントロール (入力テキスト フィールド) の Value プロパティを空白文字に設定します。

オブジェクト ビューのイベントをクリックすると、イベント ハンドラ ビューには、そのイベントを処理するために現在設定中のコードが表示されます。コードは、次の 2 種類の方法で更新することができます。

- ペインのコードを直接編集する。
- 適切なビューで、メソッド、プロパティ、スタイルのどれかをダブルクリックして、メソッドを呼び出すコード、プロパティやスタイルを取得または設定するためのコードなどを追加することができます。

10.3.1.6.1 HTML コントロールのイベント ハンドラ

HTML コントロールのイベントを処理するコードは、関数自体とその関数呼び出しで構成されます。これらの 2 つのコード ビューは、切り替えることができます。

関数の書式は、次のとおりです。

```
function controlname_event_func()
{JavaScript code
}
```

controlname は、イベントに従属するコントロールの名前です。

event は、対象になるイベントです。

JavaScript code は、メソッドの呼び出し、プロパティの取得と設定、他の処理の実行などの目的で作成するコードです。

たとえば、次のように記述します。

```
function DefaultsButton_onclick_func()
{CustomerName.value=""
```

}

カッコ (()) 内には、イベントに応じて、パラメータを記述することができます。スクリプト アシスタントには、パラメータのプレースホルダがあり、プレースホルダを固有の値に置き換える必要があります。

10.3.1.6.2 ActiveX コントロールのイベント ハンドラ

ユーザーがコードを入力した場合、ActiveX コントロールのイベントを処理するコードは、メソッド呼び出し、プロパティやスタイルの取得や設定、他の処理の実行などを目的としたコードだけで構成されます。関数または関数呼び出しで始まる行はありません。

注記:

1. タブ間の切り替えと対象となるオブジェクトの選択の順序は、重要です。まず、イベント ハンドラを追加するコントロールとイベントを選択します (次に [ハンドラの新規作成] をクリックします)。適宜、[メソッド] タブ、[プロパティ] タブ、または [スタイル] タブに切り替えます。さらに、イベント ハンドラを設定するオブジェクトと、イベント ハンドラの呼び出し、取得、設定などを行うメソッド、プロパティ、スタイルを選択します。タブを切り替える前にイベント ハンドラを設定するオブジェクトを選択すると、イベント ハンドラのコードが抹消され、新しいコードを作成できなくなります。
2. コントロールのイベント ハンドラを作成した後で、名前を変更しないでください。この名前はコードで使用されているため、変更してしまうと、コードに障害が発生します。名前を変更するには、イベント ハンドラを削除し、名前を変更してから、イベント ハンドラを再作成します。

10.3.2 [スクリプト] タブ

[スクリプト] タブは、JavaScript 関数に直接アクセスするためのものです。[スクリプト] タブは、次の目的で使用することができます。

- 他のタブを使用して作成したコードを表示する。
- 新しいコードを入力して、関数を追加する。
- 関数を削除する。
- スクリプトを追加および削除する。

JavaScript を初めて使用する場合、他のタブで、マウス ポインタを合わせ、クリックするだけでコードを作成する方法を使用することをお勧めします。

[スクリプト] タブは、下図のように表示されます。



図 10-10 [スクリプト] タブ

[スクリプト] タブには、4 つのペインがあります。

- スクリプト ビューを含む左上端のペイン。このビューでは、文書のスクリプトと関数をすべてツリー形式で表示します。文書の隣のプラス記号 (+) をクリックすると、文書のスクリプトが表示されます。一方、スクリプトをクリックすると、スクリプトの関数が表示されます。
- 選択したスクリプトの属性を一覧表示する右上端のペイン
- 右側の中央にあり、選択した属性に関するヘルプを表示するヘルプ ペイン
- 選択したスクリプトの関数を表示する最下部のペイン

注記: 組み込みの確認機能呼び出した場合、「フォームの確認」の章で説明するように、呼び出された関数もスクリプト ビューに表示されます。関数は、独立した多くの .js ファイルに格納されており、mfcommon.js ファイルに格納された多くの共通関数を利用します。

10.4 クロスプラットフォーム互換性

Netscape 社と Microsoft 社の両方の Webブラウザで機能するコードを作成する場合、次の点では、両者の互換性がないことに注意する必要があります。

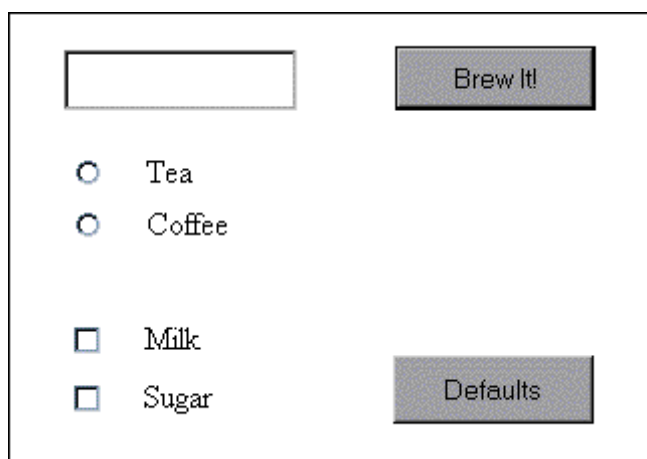
- ActiveX コントロールは、インターネット エクスプローラでしか機能しません。
- アプレットのイベントは、インターネット エクスプローラでしか機能しません。
- インターネット エクスプローラは、文書内のフォームとフォーム内のコントロールを連番により識別することができます。一方、Netscape 社のブラウザは、名前の付いたフォームやコントロールしか処理することができません。
- スクリプト アシスタントで表示されたイベントは、ブラウザによっては機能しないことがあります。

ダイナミック HTML とドキュメント オブジェクト モデルをサポートしていない旧バージョンの Web ブラウザを使用しているユーザーがいる可能性も考慮する必要があります。

10.5 例

10.5.1 例 1 プロパティを設定するイベント ハンドラの追加

この項では、「第 4 章 新規アプリケーションの作成」で例として使用されている飲料の注文フォームに簡単なイベント ハンドラを追加する方法について説明します。まず、2 つ目のプッシュボタンをフォームに追加し、ボタンをクリックしたときにフォームにデフォルト値を設定するイベント ハンドラを追加します。新しいフォームは、次のようになります。



The image shows a web form for ordering a beverage. It contains a text input field at the top left, a 'Brew It!' button at the top right, two radio buttons for 'Tea' and 'Coffee', two checkboxes for 'Milk' and 'Sugar', and a 'Defaults' button at the bottom right.

図 10-11 新しいプッシュボタンを追加した飲料注文フォーム

デフォルト設定は、次のとおりです。

- 名前フィールドを空白にする。

- Coffeeのオプション ボタンを選択した状態にする。
- Milkのチェック ボックスを選択した状態にする。
- Sugerのチェックボックスを選択しない状態にする。

この例では、HTML フォームにイベント ハンドラを追加する方法を示します。「*新規アプリケーションの作成*」の章で説明する例について操作練習をしていない場合は、¥netexpress¥base¥demo¥bevord_h¥bev_h.app を読み込むと、アプリケーションの HTML バージョンを参照することができます。

新規ボタンを追加する手順は、次のとおりです。

1. NetExpress を起動し、プロジェクト bevord_h.app (「*新規アプリケーションの作成*」の章で作成したプロジェクトまたは上記のプロジェクト) を開きます。
2. Form Designer を起動し、フォーム bevord_h.htm を開きます (NetExpress の「プロジェクト」ウィンドウで bevord_h.htm をダブルクリックすると簡単に開くことができます)。
3. フォームにコマンド ボタンを追加し、ID プロパティと Name プロパティを DefaultsButton に設定します。ボタンにテキスト「デフォルト」を入力します。
4. スクリプト アシスタントを起動し、オブジェクト ビューから DefaultsButton を選択します。
5. イベント ビューで、onclick イベントをクリックし、[ハンドラの新規作成] をクリックします。イベント ハンドラ ビューに次の JavaScript コードが表示されます。

```
function DefaultsButton_onclick_func1()
{
}
```

カーソルは、閉かっこの直前に置かれます。

6. DefaultsButton コントロールを選択したまま、[プロパティ] タブに切り替えます。
7. オブジェクト ビューで CustomerName コントロールを選択します。
8. Value プロパティをクリックし、[設定コードを挿入] をクリックします。
9. スクリプト アシスタントにより、文字列の入力を促すダイアログ ボックスがポップアップ表示されます。
10. フィールドを空白に設定するために、何も入力せず、[OK] ボタンだけをクリックします。次の JavaScript コードがイベント ハンドラ ビューのカーソル位置に表示されます。

```
document.DrinksOrder.CustomerName.value = ""
```

エンドユーザーが [デフォルト] ボタンをクリックするたびに、このコードが実行され、CustomerName フ

フィールドがクリアされます。

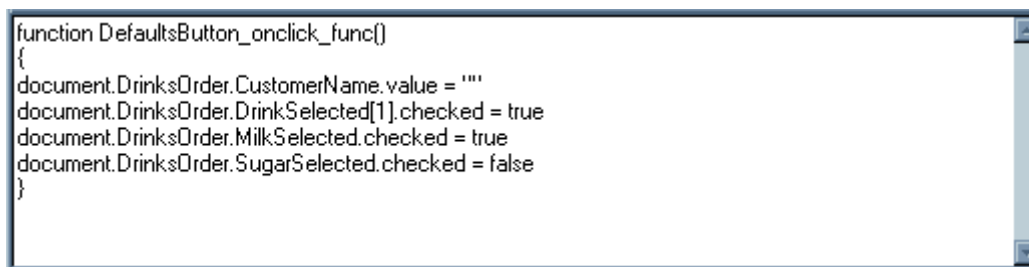
- 次に、オブジェクト ビューで CoffeeSelected オプション ボタンを選択します。
- Checked プロパティをクリックしてから、[設定コードを挿入] をクリックします。
- スクリプト アシスタントにより、真か偽のどちらかにプロパティを設定するよう促すダイアログ ボックスがポップアップ表示されます。「真」をクリックし、[OK] ボタンをクリックします。次の JavaScript コードがイベント ハンドラ ビューの現在のカーソル位置にある関数に追加されます。

```
document.DrinksOrder.DrinkSelected(1).Checked = True
```

エンドユーザーが [デフォルト] ボタンをクリックするたびに、このコードが実行され、[コーヒー] オプション ボタンが選択されるように設定されます。[紅茶] ボタンを「偽」に設定するためにコードを追加する必要はありません。これらのオプション ボタンは同じグループ内にあるためです。つまり、一方を選択すると、自動的に他方の選択が解除されます。


- 「MilkSelected」チェックボックスについても同じ手順を繰り返します。Checked プロパティを「真」に設定します。
- 「SugarSelected」チェックボックスについても同じ手順を繰り返し、Checked プロパティだけを「偽」に設定します。

この時点でのイベント ハンドラ ビューは、次のようになります。



```
function DefaultsButton_onclick_func()
{
document.DrinksOrder.CustomerName.value = ""
document.DrinksOrder.DrinkSelected[1].checked = true
document.DrinksOrder.MilkSelected.checked = true
document.DrinksOrder.SugarSelected.checked = false
}
```

図 10-12 コードを表示するイベント ハンドラ ビュー

- [OK] ボタンをクリックして、スクリプト アシスタントを閉じて、フォームを保存します。アプリケーションを再実行するか、Form Designer ツールバーの [テスト] ボタン  をクリックして、Web ブラウザにフォームを表示します。[デフォルト] ボタンをクリックするたびに、フォームのコントロールはデフォルト値に設定されます。

10.5.2 例 2 プロパティを取得するためのイベント ハンドラの追加

この項では、プロパティを取得するために簡単なイベント ハンドラを追加し、別のコントロールのプロパティを同じ値に設定する方法について説明します。最初に、2 つの入力フィールドとプッシュボタンをもつ新しいフォームを

作成し、次にイベント ハンドラを追加します。新しいフォームは次のようになります。

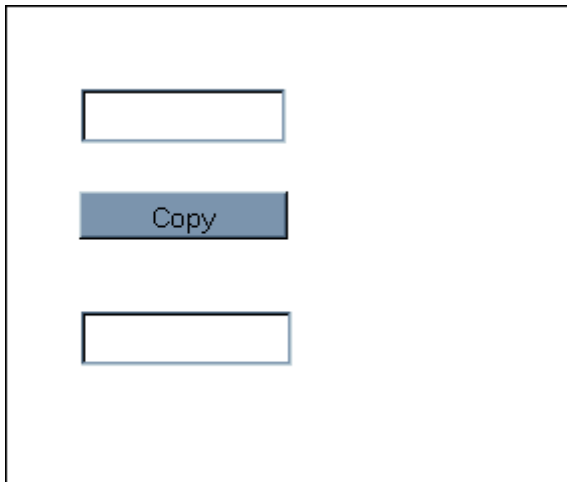


図 10-13 プロパティを取得するためのフォーム例

フォームとイベント ハンドラを作成する方法は、次のとおりです。

1. NetExpress を起動し、新規プロジェクトを作成します。
2. 新規 HTML ページを作成します。
3. .mff ファイルを開き、位置フォームを作成します。
4. 位置フォームにテキスト入力フィールドを追加します。テキスト入力フィールド InputName の名前を変更します。
5. プッシュボタンを追加し、このボタン名を CopyButton に変更します。ボタンのテキストを [Copy] に変更します。
6. 2 番目のテキスト入力フィールドを追加し、このフィールド名を CopyName に変更します。
7. スクリプト アシスタントを起動し、オブジェクト ビューから CopyButton を選択します。
8. イベント ビューで、onclick イベントをクリックし、[ハンドラの新規作成] をクリックします。イベント ハンドラ ビューに次の JavaScript コードが表示されます。

```
function CopyButton_onclick_func1()  
{  
}  
}
```

カーソルは閉かっこの直前に置かれます。


9. CopyButton コントロールを選択したまま、[プロパティ] タブに切り替えます。
10. オブジェクト ビューで CopyName コントロールを選択します。

11. Value プロパティをクリックしてから、[設定コードを挿入] をクリックします。スクリプト アシスタントにより、文字列の入力を促すダイアログ ボックスがポップアップ表示されます。 デフォルトの空白文字を受け付けます。次の JavaScript コードは、イベント ハンドラ ビューのカーソル位置に表示されます。

```
document.all.FORM1.CopyName.value = ""
```

12. デフォルト値 "" を選択し、削除します。カーソルは、値のあった位置に合わせたままにします。
13. オブジェクト ビューで InputName コントロールを選択します。
14. Value プロパティをクリックし、[表示コードを挿入] をクリックします。コードを取得するための JavaScript コードがカーソル位置に表示され、その結果、CopyName の値を入力名の値に設定する文が作成されます。

```
document.all.FORM1.CopyName.value = document.all.FORM1.InputName.value
```

15. [OK] ボタンをクリックし、スクリプト アシスタントを閉じて、フォームを保存します。アプリケーションを再実行するか、Form Designer ツールバーの [テスト] ボタン  をクリックして、Web ブラウザにフォームを表示します。最初のテキスト フィールドに名前を入力します。[コピー] ボタンをクリックすると、名前が 2 番目のテキスト フィールドに表示されます。

10.5.3 例 3 配色を設定するためのイベント ハンドラの追加

この項では、簡単なイベント ハンドラを追加して、ユーザーがボタンにマウス ポインタを合わせたときや離れたときに、ボタンの色を変化させる方法について説明します。まず、上記の例で使用した CopyButton コントロールの最初の背景色を設定し、イベント ハンドラを追加します。

フォームとイベント ハンドラを更新する方法は、次のとおりです。

1. NetExpress を起動し、前の例で作成したプロジェクトと .htm ファイルを開きます。
2. CopyButton コントロールを選択し、その背景色スタイルを Pale Green に変更します。
3. スクリプト アシスタントを起動し、オブジェクト ビューから CopyButton を選択します。
4. イベント ビューで、onmouseover イベントをクリックし、[ハンドラの新規作成] をクリックします。その結果、イベント ハンドラ ビューに次の JavaScript コードが表示されます。

```
function CopyButton_onmouseover_func()  
{  
}
```

カーソルは、閉かっこの直前に置かれます。

5. CopyButton コントロールを選択したまま、[スタイル] タブに切り替えます。

6. backgroundColor スタイルをクリックし、[設定コードを挿入] をクリックします。「色の設定」ダイアログ ボックスが表示されます。

7. 3 行目の 3 番目の明るい緑のセルを選択します。[OK] をクリックします。その結果、次の JavaScript コードがイベント ハンドラ ビューのカーソル位置に表示されます。


```
document.all.FORM1.CopyButton.style.backgroundColor = 0x0000FF00
```

8. イベント ビューで、onmouseout イベントをクリックし、[ハンドラの新規作成] をクリックします。その結果、イベント ハンドラ ビューに次の JavaScript コードが表示されます。

```
function CopyButton_onmouseout_func()  
{  
}  
}
```

カーソルは、閉かっこの直前に置かれます。

9. CopyButton コントロールを選択した状態で、[スタイル] タブに切り替えます。
10. backgroundColor スタイルをクリックし、[設定コードを挿入] をクリックします。「色の設定」ダイアログ ボックスが表示されます。
11. [色の作成 >>] を選択します。
12. 着色された四角形の領域で緑色の部分をクリックします。右側にある細長い長方形の上方をクリックし、手順 2 で設定した色、Pale Green に似た青緑色を選択します。[OK] をクリックします。色を再設定するための JavaScript コードが、イベント ハンドラ ビューのカーソル位置に表示されます。
13. [OK] ボタンをクリックし、スクリプト アシスタントを閉じて、フォームを保存します。アプリケーション

を再実行するか、Form Designer のツールバーで [テスト] ボタン  をクリックして、Web ブラウザでフォームを表示します。マウスを [コピー] ボタンの上で移動すると、緑の影が濃くなります。マウスを [コピー] ボタンから離すと、緑の影は明るくなります。

第11章 フォームの確認

この章では、組み込みの JavaScript 関数を呼び出してフォームの入力フィールドに入力されたデータを確認する方法と、ユーザー作成の確認関数を追加する方法について説明します。

ヘルプトピックの [確認] では、この章で説明する機能の使用方法を手順ごとに詳しく解説しています。[ヘルプ] の [目次] タブで、[プログラミング]、[インターネット アプリケーションのプログラミング]、[Form Designer]、[フォームの確認]、[概要 - 確認] の順にクリックしてください。

11.1 組み込みの確認関数

組み込みの確認関数は、次の HTML コントロールで使用できます。

- テキスト入力
- テキスト領域
- パスワード

フィールドが必要であるか、ステータス行にプロンプトを表示するか、ポップアップにもプロンプトを表示するかなどを指定することができます。ステータス行のプロンプトは、フィールドにフォーカスがある間は表示されています。一方、ポップアップのプロンプトは、フィールドにフォーカスがない場合と、確認が失敗した場合にだけ表示されます。プロンプトのテキストは、変更することができます。

フィールドの確認関数は、2 回実行されます。

- フィールドの内容が変更されたとき
- フォームが送信されたとき

NetExpress には、次のような確認関数があります。

関数名	確認内容
American Express card number	American Express のカード番号の形式に準拠しています。
Credit card number	すべてのクレジット カード番号の形式 (American Express、Mastercard、および、Visa) に準拠しています。
e-mail address	電子メール アドレスの形式に準拠しています。
Mastercard card number	Mastercard のカード番号の形式に準拠しています。

Number	符号のない整数です。
Number (decimal)	小数点を含む数字です。符号をつけることができます。
Number (signed positive)	正の整数です。
Number (signed)	符号付きの整数です。
Phone number - international	国際電話番号の形式に準拠しています。
Visa card number	Visa カード番号の形式に準拠しています。

注記: クレジット カード番号 (American Express、Mastercard および Visa) は、特定の人に対して発行されているため、番号の有効性は確認されません。

11.2 確認例

この項では、確認関数を呼び出す例を 3 件説明します。これらの例は、確認関数を適用できるフィールドを含む .htm ファイルが開いていることを前提としています。確認関数を適用できるフィールドは、テキスト入力フィールド、テキスト領域フィールド、およびパスワード フィールドです。

「確認」ダイアログ ボックスは、次のように表示されます。

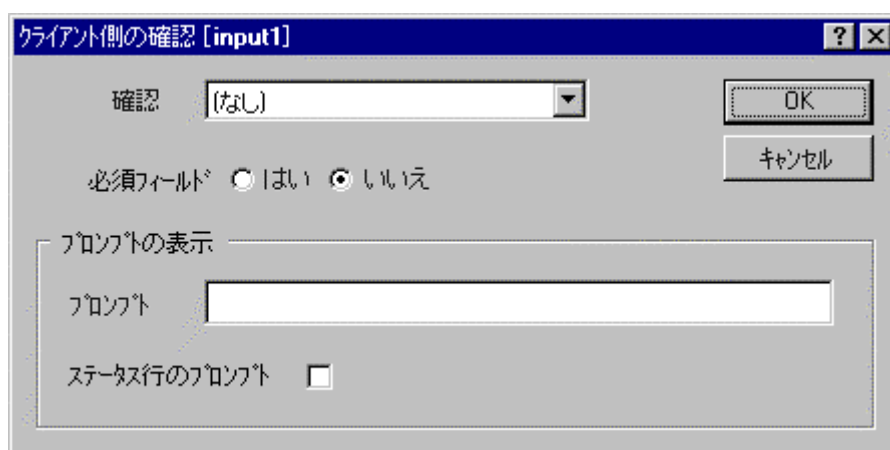


図 11-1 「確認」ダイアログ ボックス

11.2.1 例 1 オプションの小数を確認する方法

小数確認関数を呼び出し、フィールドをオプション フィールドにする方法を次に示します。

1. 確認するフィールドを選択します。
2. 右クリックし、ポップアップメニューから [確認] を選択します (または [編集] メニューから [確認] を選択します)。「確認」ダイアログボックスが表示されます。
3. 「確認」リストから [Number (decimal)] をクリックします。
4. 「必須フィールド」で「いいえ」オプション ボタンをクリックします。この操作により、このフィールドをオプションに指定できます。
5. 小数確認ルーチンに関連付けられたプロンプトのテキストが「プロンプト」フィールドに表示されます。必要に応じてテキストを編集することができます。プロンプトをステータス行に表示する場合には、「ステータス行のプロンプト表示」チェックボックスをクリックします。
6. [OK] をクリックします。

このルーチンが呼び出されると、エンドユーザーはフィールドを空白のまま残すことができます。ただし、小数以外の値を入力した場合には、フィールドに戻り、有効な値を入力するように促されます。

11.2.2 例 2 必須の Visa カード番号を確認する方法

Visa カード番号の確認関数を呼び出し、フィールドを必須フィールドにする方法は、次のとおりです。

1. 確認するフィールドを選択します。
2. 右クリックし、ポップアップメニューから [確認] を選択します (または [編集] メニューから [確認] を選択します)。「確認」ダイアログボックスが表示されます。
3. 「確認」リストから [Visa card number] をクリックします。
4. 「必須フィールド」で「はい」オプション ボタンをクリックします。この操作により、このフィールドは、必須フィールドになります。
5. Visa card number 確認関数に関連付けられたプロンプトのテキストが「プロンプト」に表示されます。必要に応じてテキストを編集することができます。プロンプトをステータス行に表示する場合には、「ステータス行のプロンプト表示」をクリックします。
6. [OK] をクリックします。

このルーチンが呼び出されると、エンドユーザーは有効な値を入力する必要があります。有効な値を入力しなかった場合には、フィールドに戻り、有効な値を入力するように促されます。

11.2.3 例 3 必須フィールドへのデータ入力確認

組み込みの確認関数がどれも適切でない場合でも、「確認」ダイアログボックスを使用して、フィールドを必須フ

フィールドに指定することができます。この方法は、次のとおりです。

1. 必須フィールドにするフィールドを選択します。
2. 右クリックし、ポップアップ メニューから [確認] を選択します (または [編集] メニューから [確認] を選択します)。「確認」ダイアログ ボックスが表示されます。
3. 「確認」リストから [なし] をクリックします。
4. 「必須フィールド」で「はい」オプション ボタンをクリックします。この操作により、このフィールドは必須フィールドになります。
5. 「確認」リストの「なし」に関連付けられてるプロンプトのテキストが「プロンプト」に表示されます。必要に応じて、テキストを編集することができます。プロンプトをステータス行に表示する場合には、「ステータス行のプロンプト表示」をクリックします。
6. [OK] をクリックします。

このルーチンが呼び出されると、エンドユーザーは値を入力する必要があります。入力しない場合、フィールドに戻り、値を入力するように促されます。

11.3 ユーザー定義確認関数の追加

組み込み関数が適切でない場合、テキスト入力、テキスト領域、パスワード フィールドに対してユーザー定義の確認関数を作成することができます。そのためには、JavaScript または JScript をよく理解する必要があります。詳細については、「クライアント側のプログラミング」の章を参照してください。

追加した確認関数をフォームに組み込むには、2 種類の方法があります。

- 他の関数を実行するためのコードと同様に、フォームに埋め込まれたイベント ハンドラをインクルードしてフィールドを確認することができます。通常は、onchange イベントと、フォームの onsubmit イベントで関数を呼び出します。確認関数を使用する回数が 1 回限りである場合、このアプローチは便利です。
- 確認関数を作成し、NetExpress に組み込まれた関数に追加することができます。このアプローチの利点は、その後、他の入力フォームのフィールドでもこの関数を使用できることです。

組み込みルーチンは、それぞれ .js ファイルに格納され、bin ディレクトリの validate.cpt という別のファイルから参照されます。そのため、コードを作成して .js ファイルに格納すると同時に、validate.cpt ファイルを更新する必要があります。テキスト エディタやスクリプト アシスタントを使用してコードを最初から作成することも、また、組み込みルーチンの 1 つをコピーし、編集して条件に合わせることもできます。ルーチンをコピーして編集する方法に関する手順ごとの説明については、オンライン ヘルプを参照してください。[ヘルプ] の [目次] で [プログラミング]、[インターネット アプリケーションのプログラミング]、[Form Designer]、[フォームの確認]、[方法]、[ユーザー定義確認関数の追加] の順にクリックします。

第12章 アプリケーションの実装

この章では、イントラネットやインターネットでプロダクション Web サーバーにアプリケーションを実装する方法について説明します。また、開発中に Solo 以外の Web サーバーを使用してデバッグする方法についても説明します。

12.1 概要

アプリケーションの実装とは、アプリケーションをインターネットやイントラネットで使用できるように Web サーバーに配置することです。Form Designer とインターネット アプリケーション ウィザードで NetExpress アプリケーションを開発中に、Web サーバーとして Solo を使用すると、Web サーバーの構成を変更せずにすべてが機能するように、環境、Web 共有リソース、および URL が設定されます。

アプリケーションをプロダクション Web サーバーに実装した場合、次の点を確認する必要があります。

- サーバーの Web 共有リソースのパスとアプリケーションの URL
- 送信するページ ファイル
- アプリケーションに関するプロジェクトのビルド設定

この章では、アプリケーションの実装手順について詳細に説明します。また、CGI プログラムを ISAPI プログラムまたは NSAPI プログラムに変換する手順についても解説します。すべてのサーバー側プログラムを CGI プログラムとして開発し、実装準備ができた段階で、ISAPI または NSAPI に変換することをお勧めします。

アプリケーションを Web サーバーに実装する前に、アプリケーションの配布に関する使用許諾条件を必ず読んでください。使用許諾条件は、標準のテキスト ファイル license.txt に収録されています。これを読むためには、Windows のエクスプローラを使用してフォルダ %netexpress%base%bin を開き、license.txt をダブルクリックします。この操作により、ファイルはメモ帳で開きます。

12.1.1 他の Web サーバーを使用したデバッグ

Solo 以外の Web サーバーを使用してプログラムをデバッグすることもできます。デバッグ中には、Web サーバーで COBOL と CGI-BIN の Web 共有リソースを設定することをお勧めします。これは、アプリケーションに変更を行っている間のデバッグを簡単にするためです。Solo 以外の Web サーバーを使用した場合の実装とデバッグの相違点については、「実装とデバッグに関するガイド」の項で説明します。

12.1.2 作業例

この章の内容をわかりやすくするために、付録「実装例とデバッグ例」に、別の Web サーバーを使用したデバッグと実装の作業例をいくつか掲載してあります。

12.2 サポートされているサーバー

NetExpress では、Web プログラムの CGI、ISAPI、または NSAPI インターフェイスに準拠したアプリケーションをビルドすることができます。つまり、これらのインターフェイスに完全に準拠した Web サーバーでは NetExpress アプリケーションを実行できることになります。NetExpress アプリケーションの発行時には、次のサーバーとオペレーティング システムでテストしてあります。

Windows NT

- Netscape Enterprise Server
- Microsoft Internet Information Server V4.0
- Netscape Fasttrack Server V 2.01
- MS Personal Web Server

UNIX

- Apache (Micro Focus Object COBOL V4.1 for UNIX を使用)

12.3 実装とデバッグに関するガイド

次の 3 項では、Solo 以外の Web サーバーで実行されるアプリケーションを取得するために必要な手順をまとめ、その後の項で各手順の詳細を説明します。説明の中には、プロダクション Web サーバーにアプリケーションを実装する場合にしか適用されない箇所や、アプリケーションのデバッグにしか適用されない箇所があります。このような特定の説明については、その旨明記してあります。アプリケーションをプロダクション Web サーバーで実装する場合と Solo 以外の Web サーバーでデバッグする場合の主な違いは、次のとおりです。

- 実装時には、デフォルトの NetExpress よりも柔軟な Web 共有リソースを使用した方がよい場合があります。デバッグ時には、デフォルトの COBOL と CGI-BIN で十分対応できます。
- 実装時には、開発に使用したマシンからアプリケーションを別のマシンにコピーします。デバッグ時には、NetExpress のライセンス コピーをもつ開発マシンを使用します。
- デバッグ時には、実装時には行わないサーバー構成を追加で変更する必要があります。

次の 3 項では、solo 以外の Web サーバーを使った実装手順またはデバッグ手順をまとめています。

- 実装手順ガイド(Windows サーバー)
- 実装手順ガイド(UNIX サーバー)
- デバッグ手順ガイド

12.3.1 実装手順ガイド (Windows サーバー)

この項では、プロダクション Web サーバーにアプリケーションを実装する場合の手順について概説します。ここでは、プロダクション Web サイトへの実装時に、NetExpress のデフォルトの Web 共有リソース COBOL と CGI-BIN 以外の Web 共有リソースを使用した方がよい場合を想定しています。

1. 使用するランタイム システムを選択します。

Form Designer とインターネット アプリケーション ウィザードで開発されたアプリケーションのデフォルトのランタイム システムは、動的に結合された共有シングル スレッド ランタイム システムです。アプリケーションの種類によっては、別のランタイム システムを使用した方がよい場合、また、使用する必要がある場合があります。

「ランタイム システムの選択」の項を参照してください。

2. Web サーバーの共有リソースとアクセス権限を設定します。

この手順は、Web サーバーと Web サイトの設定で一度しか行わない本質的な部分です。ただし、ここで選択した内容は、NetExpress で作成したアプリケーションをサーバーで実装する場合の作業に影響します。

「Web サーバーの設定」の項を参照してください。

3. NetExpress で作成したアプリケーションのファイルをすべて実装用にコピーします。

使用している Web サーバー、選択した Web 共有リソース名、およびアプリケーションのビルド方法によっては、COBOL ソース ファイル、および HTML ファイルにわずかな変更を行う必要がある場合があります。この場合、開発したソースとは別のコピーを変更することをお勧めします。

「実装用コピーの作成」の項を参照してください。

4. Web 共有リソース名に対して、必要に応じてアプリケーションの URL を変更します。

NetExpress では、自動生成されるすべてのコードに含まれる COBOL と CGI-BIN の Web 共有リソースを使用します。別の Web 共有リソース名を使用する場合、適宜、COBOL コードと HTML コードを更新する必要があります。

「アプリケーションの URL 変更」の項を参照してください。

5. 「サーバー側のプログラミング」の章で説明するサーバー側の状態機構をアプリケーションで使用する場合は、CGI に sstate モジュールを追加することが必要です。データ アクセス ウィザードで生成したアプリケーションは、すべてこの機構を使用しています。

「実装済みアプリケーションへの sstate の追加」の項を参照してください。

6. サーバーの API を選択し、ターゲット Web サーバー用にアプリケーションをリビルドします。

手順 4 で追加した変更を反映させるためにアプリケーションをリビルドする必要があります。また、アプリケーションをリビルドするこの段階では、CGI のかわりに、ISAPI または NSAPI サーバー API を使用するように選択することができます。

「アプリケーションのリビルド」の項を参照してください。

7. Web サーバーにアプリケーションをインストールします。

「アプリケーションの実装」の項を参照してください。

8. Object COBOL のオブジェクト指向機能を使用する ISAPI アプリケーションの場合、NetExpress ランタイム システムの構成を変更する必要があります。また、Windows NT のサーバー アクセス権を構成する必要があります。

「ISAPI に準拠した Object COBOL アプリケーションの実装」の項を参照してください。

9. NSAPI アプリケーションだけについては、Netscape サーバー構成ファイルの一部を修正する必要があります。

「NSAPI アプリケーションを実行する前に」の項を参照してください。

12.3.2 実装手順ガイド (UNIX サーバー)

この項では、UNIX で実行されるプロダクション Web サーバーにアプリケーションを実装する手順を概説します。ここでは、プロダクション Web サイトへの実装時に、NetExpress のデフォルトの Web 共有リソース COBOL と CGI-BIN 以外の Web 共有リソースを使用した方がよい場合を想定しています。

1. UNIX サーバーに Micro Focus COBOL for UNIX V4.1 以降のバージョンをインストールします。
2. UNIX サーバーに SCP をインストールします (詳細については、『UNIX オプション ユーザー ガイド』を参照してください)。
3. Web サーバーの共有リソースとアクセス権を設定します。

この手順は、Web サーバーと Web サイトの設定で一度しか行わない本質的な部分です。ただし、ここで選択した内容は、NetExpress で作成したアプリケーションをサーバーで実装する場合の作業に影響します。

「Web サーバーの設定」の項を参照してください。

4. NetExpress で作成したアプリケーションのファイルをすべて実装用にコピーします。

使用している Web サーバー、選択した Web 共有リソース名、およびアプリケーションのビルド方法によっては、COBOL ソース ファイル、および HTML ファイルにわずかな変更を行う必要がある場合があります。この場合、開発したソースとは別のコピーを変更することをお勧めします。

「実装用コピーの作成」の項を参照してください。

5. 「サーバー側のプログラミング」の章で説明するサーバー側の状態機構をアプリケーションで使用する場合は、CGI に sstate モジュールを追加することが必要です。

「実装済みアプリケーションへの sstate の追加」の項を参照してください。

6. Web 共有リソース名に対して、必要に応じてアプリケーションの URL を変更します。

NetExpress では、自動生成されるすべてのコードに含まれる COBOL と CGI-BIN の Web 共有リソースを使用します。別の Web 共有リソース名を使用する場合、適宜、COBOL コードと HTML コードを更新する必要があります。さらに、UNIX の実行可能ファイルでは拡張子が異なります。大文字と小文字の区別の問題も考慮しなければなりません。

「アプリケーションの URL 変更」の項を参照してください。

7. Web サーバーにアプリケーションをパブリッシュします。

「UNIX へのアプリケーションのパブリッシュ」の項を参照してください。

12.3.3 デバッグ手順ガイド

この項では、Solo 以外の Web サーバーでアプリケーションを開発する手順について概説します。アプリケーションの開発とデバッグに、NetExpress のデフォルトの Web 共有リソース COBOL と CGI-BIN を使用することを前提としています。これは、新しいフォームやプログラムを開発する場合に作業を削減することができるためです。

1. 使用するランタイム システムを選択します。

Form Designer とインターネット アプリケーション ウィザードで開発されたアプリケーションのデフォルトのランタイム システムは、シングルスレッド共有ランタイム システムです。アプリケーションの種類によっては、別のランタイム システムを使用した方がよい場合、また、使用する必要がある場合があります。

「ランタイム システムの選択」の項を参照してください。

2. Web サーバーの共有リソースとアクセス権を設定します。

フォームと HTML ページには COBOL、プログラムには CGI-BIN の 2 つの共有リソースが必要です。

「Web サーバーの設定」の項を参照してください。

3. サーバー API を選択し、ターゲット Web サーバー用にアプリケーションをリビルドします。

アプリケーションをリビルドするこの段階では、CGI のかわりに、ISAPI または NSAPI サーバー API を使用するよう選択することができます。これらの API のうちどれを使用しても、NetExpress でアプリケーションをデバッグできます。ISAPI または NSAPI で保護違反が起きた場合、通常 Web サーバーを閉じ

て再起動する必要があるため、初期開発は CGI で行うことをお勧めします。

「アプリケーションのリビルド」の項を参照してください。

4. Object COBOL の機能を使用する ISAPI アプリケーションについては、NetExpress のランタイム システムに構成変更を行う必要があります。また、Windows NT でサーバーのアクセス権を構成する必要もあります。

「ISAPI に準拠した Object COBOL アプリケーションの実装」の項を参照してください。

5. NSAPI アプリケーションだけについては、Netscape サーバー構成ファイルの一部を修正する必要があります。

「NSAPI アプリケーションを実行する前に」の項を参照してください。

この手順を一度実行した後で、Solo 以外の Web サーバーを使用してサーバー側プログラムを実行し、アプリケーションの編集、ビルド、デバッグのサイクルを再実行してみることもできます。

12.3.4 ランタイム システムの選択

Form Designer とインターネット アプリケーション ウィザードで作成されたアプリケーションでは、動的に結合された共有シングルスレッド ランタイム システムを使用するようにデフォルト設定されています。これは、実行可能ファイルが起動時に共有ランタイム システムに動的にリンクすることと、その実行可能プログラムがシングルスレッドであることを意味します。実行可能プログラムは、Windows のレジストリでランタイム システムを検索します。

次の場合には、実行可能ファイルのビルド方法を変更する必要があります。

- アプリケーションを ISAPI プログラムまたは NSAPI プログラムとして実装する場合。

ISAPI プログラムと NSAPI プログラムは、マルチスレッドあることが必要です。「共有ランタイム システムを使用する ISAPI または NSAPI プログラムのビルド」の項にしたがってプログラムをリンクします。

- スタンドアロンの実行可能プログラム (共有ランタイム システムを実行する必要がないもの) を作成する場合。1 台のサーバーに実装する COBOL CGI プログラムが 1、2 個で、COBOL ランタイム システムをインストールするオーバーヘッドを避ける場合以外は、この方法はお勧めできません。「静的ランタイム システムを使用する CGI プログラムのビルド」の説明にしたがってプログラムをリンクしてください。

注記: マルチスレッドを使用した方がよい場合は、CGI アプリケーションでマルチスレッドを使用できるようにすることができます。この場合、共有または静的マルチスレッド ランタイム システムとリンクします。このオプションについては、「共有ランタイム システムを使用する CGI プログラムのビルド」と「静的ランタイム システムを使用する CGI プログラムのビルド」の項で説明します。

12.3.5 Web サーバーの設定

この項では、インターネット アプリケーションを実行するための Web サーバーの設定方法を説明します。正確な方法は、サーバーによって異なるため、設定方法に関する正確な説明については、サーバーのマニュアルを参照してください。

実装とデバッグでは、サーバーの設定方法が異なるため、次の 2 項に分けて説明します。

- 実装用の Web サーバー設定
- デバッグ用の Web サーバー設定

実装用の Web サーバー設定

この項では、UNIX の Web サーバーについては説明しません。Web 共有リソースに関する情報はすべてのオペレーティング システムに適用されますが、Application Server に関する情報は、Windows NT と Windows 95 に固有の情報です。UNIX への実装の詳細については、『UNIX オプション ユーザー ガイド』を参照してください。

アプリケーションを実装するための Web サーバーの設定方法は、次のとおりです。

1. 共有ランタイム システムを使用してアプリケーションを実装する場合、ターゲット マシンに Application Server をインストールします。Application Server により、NetExpress で作成したアプリケーションをマシンで実行するためのランタイム システムが設定されます。

注記: UNIX サーバーでは、Object COBOL for UNIX V4.1 をインストールする必要があります。。

2. Web サーバー用の管理ツールを起動します (この方法については、Web サーバーのマニュアルを参照してください)。
3. 下表の Web 共有リソースを設定します。Web 共有リソースは、URL を Web サーバーの実際のディレクトリまたはフォルダにマップしたものです。

この表では、さまざまなリソースの共有リソースに対して NetExpress で使用されるデフォルトの名前を使用します。Web 共有リソースに対して NetExpress のデフォルト名以外の名前を使用する場合、「アプリケーションの URL 変更」の説明にしたがって、アプリケーションのファイルを編集する必要があります。ISAPI アプリケーションまたは NSAPI アプリケーションを実装している場合や、UNIX サーバーに実装している場合には、実行可能ファイル名が多少異なるため、アプリケーションのファイルを編集する必要があります。

Web 共有リソース	NetExpress のデフォルト名	アクセス権	説明
<i>bin-share</i>	/CGI-BIN/	実行	この共有リソースには、アプリケーションの実行可能ファイル (Windows の .exe ファイルと .dll ファイル) を格納します。
<i>form-share</i>	/COBOL/	読み取り専用	この共有リソースには、アプリケーションのフォームと HTML ページ (.htm ファイル) を格納します。
<i>image-share</i>	/COBOL/	読み取り専用	アプリケーションに多くのイメージがある場合には、独立した共有リソースに格納し、アプリケーションのメンテナンスを容易にすることができます。HTML イメージの Image Source プロパティや ActiveX イメージ コントロールの Picture Path プロパティを変更する必要があります。

注記:

- 異なる種類のアクセス権を設定した場合、サーバーによっては機能しないことがあります。たとえば、Netscape FastTrack では、CGI 共有リソースと Document 共有リソースの概念を使用します。FastTrack サーバーを管理するときには、実行可能ファイルに対して CGI 共有リソースを設定し、他のファイルに Document 共有リソースを使用します。FastTrack がインストールされている場合は、ドキュメントと CGI に対して、アプリケーションの実装に使用できるデフォルトの Web 共有リソースが設定されます。
- Web サーバー (例 Netscape) によっては、Web 共有リソース名を判読するときに大文字と小文字が区別されるので、その場合、/cgi-bin/ と /CGI-BIN/ は別のものとして処理されます。

NetExpress のデフォルトの Web 共有リソース名を使用し、ランタイム システム オプションを変更せず、実装時に ISAPI または NSAPI を使用しない場合は、「アプリケーションの実装」の項へとぶことができます。

デバッグ用の Web サーバー設定

NetExpress でデバッグするための Web サーバーの設定方法は、次のとおりです。

1. Web サーバー用の管理ツールを起動します (この方法については、Web サーバーのマニュアルを参照してください)。
2. 下表の Web 共有リソースを設定します。Web 共有リソースは、URL を Web サーバーの実際のディレクトリまたはフォルダにマップしたものです。

この表では、さまざまなリソースの共有リソースに対して NetExpress で使用されるデフォルトの名前を使用します。Web 共有リソースに対して NetExpress のデフォルト名以外の名前を使用する場合、「アプリケーションの URL 変更」の説明にしたがって、アプリケーションのファイルを編集する必要があります。

Web 共有リソース名	アクセス権	場所
/CGI-BIN/	実行	NetExpress プロジェクトの実行可能ファイルのディレクトリ (.exe ファイルと .dll ファイル)。これはソース ディレクトリのサブディレクトリです。ビルドの種類がデバッグとリリースの場合、NetExpress のデフォルトのサブディレクトリは debug と release になります。
/COBOL/	読み取り専用	プロジェクトのソース ディレクトリ。

3. Web サーバーが NT で実行され、サービス中である場合、Web サーバーへの匿名のログオンを排除します。ユーザー アカウントでログ オンし、Web サーバーを実行した場合、使用されているユーザー アカウントが管理者権限をもち、「サービスとしてログ オン」するアクセス権をもっていることを確認してください。

Web サーバーは、通常、匿名のログオンで実行されるようにデフォルト設定されているので、ユーザーは名前とパスワードを指定せずにイントラネットやインターネットを介してサーバーにアクセスすることができます。匿名のログオンでは、Web サーバーの実行権限は制限されており、デバッグを起動することはできません。

匿名のログオンを許可しない場合は、他のユーザーがユーザー名とパスワードを通知しないと Web サーバーに接続できないようにします。ただし、サーバーにローカルで接続するときは（サーバーが実行されている同じマシンから）、サーバーに同じユーザー権限を与えることになるので、Animator (NetExpress のデバッグ) を起動される可能性があります。

12.3.6 実装用コピーの作成

注記: アプリケーションをデバッグ用に設定している場合は、この手順をとばしてください。

次の項目に 1 つでも該当する場合は、コピーを作成する必要があります。

- NetExpress のデフォルトの Web 共有リソース名を使用していないため、アプリケーションを実装する前にアプリケーション ファイルを変更する必要がある場合（「Web サーバーの設定」の項を参照してください）
- 別のランタイム システムを使用する場合（「ランタイム システムの選択」の項を参照してください）
- UNIX サーバーに実装する場合（「実装手順ガイド (UNIX サーバー)」を参照してください）

- アプリケーションが「サーバー側のプログラミング」の章で説明するサーバー側状態機構を使用している場合

アプリケーションの開発に使用した元のファイルを変更するのではなく、そのファイルのコピーを作成し、コピーを編集することを強くお勧めします。実装用コピーに対して開発マシンに新しいディレクトリを作成し、プロジェクトディレクトリにあるすべてのファイルをそのディレクトリにコピーします。NetExpress でプロジェクトを開いている場合は、閉じます。.app ファイル (プロジェクト ファイル) は、NetExpress で開いていると、コピーすることができません。

12.3.7 アプリケーションの URL 変更

注記: デバッグ用にアプリケーションを設定している場合は、この手順をとばしてください。

NetExpress で生成されるアプリケーションについては、HTML ページ (.htmファイル) は Web 共有リソース COBOL にあり、実行可能ファイルは Web 共有リソース CGI-BIN にあることを前提としています。他の Web 共有リソース名を使用している場合には、参照を次のように変更する必要があります。

- すべての入力フォームで CGI の参照が実行されるように変更する。
- すべてのフォームで、イメージの参照を変更する。
- DISPLAY 動詞を使用して HTML ページを出力する COBOL ソース ファイルで、ページの参照を変更する。

インターネット アプリケーション ウィザードにより生成されたアプリケーションは、DISPLAY 動詞を使用しないため、「サーバー側のプログラミング」の章で説明するとおり、DISPLAY 動詞を使用している場合にだけこのような変更を行う必要があります。

CGI 参照を変更するには、CGI プログラムへの入力である各フォームに対して、次のような手順で作業を行います。

1. 「プロジェクト」ウィンドウの左側のペインで .htm ファイルを右クリックして、コンテキスト メニューの [編集] をクリックし、Form Designer にページをロードします。
2. Action プロパティをクリックして、次のように修正します。

```
/binshare/program.ext
```

パラメータの内容は、次のとおりです。

```
bin-share        実行可能ファイルの Web 共有リソース
```

<i>program</i>	プログラムのファイル名
<i>ext</i>	プログラムのファイル拡張子。Windows プラットフォームの CGI プログラムの場合は、.exe です。アプリケーションを ISAPI プログラムとして実装する場合は、拡張子は .dll です。NSAPI プログラムとして実装する場合は、拡張子をユーザーが定義できます。 UNIX サーバーでは、CGI はシェル スクリプトから実行され、その拡張子は .sh になります。

イメージ パスを変更するには、イメージを含む各ページを Form Designer に読み込み、各イメージに対して次の手順を実行します。

1. イメージをクリックします。
2. Image Source プロパティ (HTML イメージ) または Image Path プロパティ (ActiveX イメージ コントロール) をクリックし、次のように変更します。

```
/image-share/image.ext
```

パラメータの内容は、次のとおりです。

<i>image-share</i>	実行可能ファイルの Web 共有リソース
<i>image</i>	イメージのファイル名
<i>ext</i>	イメージのファイル拡張子

DISPLAY 動詞により出力される HTML フォームのパスを変更する方法は、次のとおりです。

1. COBOL ソース コードで IS EXTERNAL FORM IDENTIFIED BY 句をすべて検索します。
2. ファイル名を次のように修正します。

```
"fullpath¥htmlfile.htm"
```

この場合、*fullpath* は、.htm ファイルをインストールした Web サーバーの場所を指す完全な物理パスであり、Web 共有リソース名ではありません。

12.3.8 実装済みアプリケーションへの sstate の追加

「サーバー側のプログラミング」の章で説明するサーバー側のサポート機構では、sstate と呼ばれるモジュールを使用します。これは、COBOL ランタイム システムの一部ではありません (必要に応じて変更するためにソースが用意されています)。そのため、sstate を使用するアプリケーションを実装した場合、必ず CGI、ISAPI、または NSAPI の実行可能プログラムにリンクする必要があります。アプリケーションでこの機構を使用しない場合には、この項をとばしてください。

注記: 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すべてのデータ アクセス アプリケーションでは、この機構を使用しています。

この項では、Windows プラットフォームと UNIX プラットフォームでの作業を別々に説明します。

Windows

Windows では、`sstate.obj` を直接実行可能プログラムにリンクさせることができます。

1. NetExpress の「プロジェクト」ウィンドウで左側のペインにある CGI のメインの実行可能ファイル (.exe ファイル) を右クリックし、コンテキスト メニューで [ビルド設定] をクリックします。
2. [リンク] タブをクリックします。
3. [カテゴリ] ドロップダウンから [高度な設定] を選択します。
4. 「これらの OBJ とリンクする」フィールドに `sstate` を入力し、[OK] をクリックします。

UNIX

UNIX で実装するには、`sstate.int` を NetExpress のプロジェクトに追加し、UNIX オプションでパブリッシュすると、CGI に自動的にリンクされます。

1. [プロジェクト] メニューの [プロジェクトへのファイルの追加] をクリックします。
2. 「開く」ダイアログ ボックスで、フォルダ `¥netexpress¥unix¥cgi-sup` を選択します (このフォルダは、UNIX オプションをインストールしている場合にだけ使用することができます)。
3. 「ファイルの種類」ドロップダウンから [すべてのファイル] を選択し、リストから `sstate.int` を選択します。
4. メッセージ ボックスが表示され、このファイルがプロジェクト ディレクトリとは異なるディレクトリにあることを警告します。[OK] ボタンをクリックして、プロジェクト ディレクトリにコピーします。

12.3.9 アプリケーションのリビルド

この項では、ターゲット マシンでアプリケーションを実装するためにアプリケーションのプログラムをリビルドする方法について説明します。次の項目に 1 つでも該当する場合は、アプリケーションをリビルドする必要があります。

- ターゲット Web サーバーで別の共有リソース名を指定できるように URL を変更した場合

- ランタイム システムを変更する必要がある場合 (詳細については、「ランタイム システムの選択」の項を参照してください)
- ISAPI または NSAPI のアプリケーションをリビルドする場合
- マルチスレッド CGI を使用する場合。CGI プログラムをマルチスレッド アプリケーションとして作成した場合にだけ、CGI をマルチスレッド化する必要があります。詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] で [マルチスレッド] を検索してください。

次の項では、さまざまなアプリケーションのビルド方法について説明します。

- 静的ランタイム システムを使用する CGI プログラムのビルド
- 共有ランタイム システムを使用する CGI プログラムのビルド
- 共有ランタイム システムを使用する ISAPI または NSAPI プログラムのビルド

静的ランタイム システムを使用する CGI プログラムのビルド

Web サーバーに実装する COBOL CGI プログラムが 1、2 個で、サーバーで COBOL ランタイム システム ファイルのオーバーヘッドを避ける場合にだけ、静的ランタイム システムを使用することをお勧めします。

静的ランタイム システムで CGI プログラムをビルドする方法は、次のとおりです。

1. NetExpress の「プロジェクト」ウィンドウで左側のペインにある CGI のメインの実行可能ファイル (.exe ファイル) を右クリックし、コンテキスト メニューの [ビルド設定] をクリックします。
2. [リンク] タブをクリックします。
3. [静的] と [シングルスレッド] のオプション ボタンをクリックします。
4. アプリケーションをリビルドします。

アプリケーションをマルチスレッド ランタイム システムにリンクするには、上記の手順 3 で [マルチスレッド] オプション ボタンをクリックします。マルチスレッドを利用するように CGI アプリケーションを作成した場合にだけ、このように選択してください。詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] で [マルチスレッド] を検索してください。

共有ランタイム システムを使用する CGI プログラムのビルド

共有ランタイム システムとリンクされた CGI をビルドする場合には、アプリケーションを実装するマシンに、ランタイム システム ファイルを提供する NetExpress Application Server をインストールする必要があります。

Form Designer とインターネット アプリケーション ウィザードを使用してアプリケーションを作成する場合、それらのアプリケーションは、常に、動的に結合された共有ランタイム システムを使用してリンクされます。デフォルト

トのリンク設定を変更した場合は、「ビルド設定」ダイアログ ボックスを使用すると、リンク設定を元に戻すことができます。

1. NetExpress の「プロジェクト」ウィンドウで左側のペインにある CGI のメインの実行可能ファイル (.exe ファイル) を右クリックし、コンテキスト メニューの [ビルド設定] をクリックします。
2. [リンク] タブをクリックします。
3. [共有] と [シングルスレッド] のオプション ボタンをクリックします。

マルチスレッドを利用するように CGI アプリケーションを作成した場合、マルチスレッド ランタイム システムとリンクすることができます ([マルチスレッド] オプション ボタンをクリックします)。詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] の [マルチスレッド] を検索してください。

4. 「動的」チェック ボックスをチェックします。

この操作により、実行可能プログラムは Windows のレジストリで Application Server または NetExpress のランタイム システムを検出することができます。

5. アプリケーションをリビルドします。

共有ランタイム システムを使用する ISAPI または NSAPI プログラムのビルド

ISAPI プログラムまたは NSAPI プログラムをビルドするには、NetExpress のプロジェクトを次のように変更する必要があります。

- プログラムを .exe 実行可能ファイルとしてではなく、.dll 実行可能ファイルとしてビルドします。
- ISAPI プログラムまたは NSAPI プログラムに必要なエントリ ポイントと、プログラムをコンパイルするコンパイラ指令を追加し、共有マルチスレッド ランタイム システム (ISAPI プログラムと NSAPI プログラムはマルチスレッド プログラムです) にそのプログラムをリンクします。
- 追加ライブラリをリンクします (NSAPI のみ)

注記:

- ISAPI または NSAPI アプリケーションでリレーショナル データベースを使用している場合、データベースにアクセスするために使用する ODBC ドライバはスレッドセーフである必要があります。Microsoft Access はスレッドセーフでないため、ISAPI または NSAPI のサーバー側プログラムで使用することはお勧めできません。

- ISAPI または NSAPI Web サーバーで同時にロードしている 2 つ以上の COBOL.dll において、同じエントリ ポイント名を存在させることはできません。

プログラムを .dll 実行可能ファイルとしてビルドする方法は、次のとおりです。

1. 「プロジェクト」ウィンドウで左側のペインにある .exe を右クリックし、コンテキスト メニューで [ビルド タイプから削除] をクリックします。
2. 「ビルド構造から削除」ダイアログ ボックスで、「ビルド構造の全体を削除します」のチェックを解除し、「このターゲットをすべてのビルド タイプから削除します」を選択して、[削除] ボタンをクリックします。
3. 元の .exe プログラムをビルドするために、互いにリンクされたすべての .obj ファイルを選択します。次に、「プロジェクト」ウィンドウの左側のペインで右クリックし、コンテキスト メニューで [選択されたファイルをパッケージ化]、[ダイナミック リンク ライブラリ] の順にクリックします。
4. 「新規に定義 ダイナミック リンク ライブラリ」ダイアログ ボックスの [作成] ボタンをクリックします。

コンパイラとビルド設定を変更する方法は、次のとおりです。

1. 下表のとおり、ISAPI または NSAPI のコンパイラ指令を設定します。

プログラムの冒頭に \$SET 文を記述すると、プログラムがコンパイルされるたびに ISAPI または NSAPI のコンパイラ指令が確実に設定されます。

たとえば、次のように記述します。

```
$set webserver(nsapi,myentry) case reentrant(2)
```

ドル記号 (\$) は、必ずソース コードの 7 列目に記述します。ただし、記述する列の指定を解除する指令 SOURCEFORMAT(FREE) を設定した場合を除きます。コンパイラ指令設定の詳細については、NetExpress の [ヘルプ] メニューで [ヘルプ トピック] をクリックし、[目次] タブで [リファレンス]、[コンパイラ指令] の順に選択して、表示される説明を参照してください。

ISAPI プログラムと NSAPI プログラムの指令

指令	説明
WEBSERVER(ISAPI)	このプログラムが ISAPI アプリケーションであることをコンパイラに伝えます。

この指令は、アプリケーションを実行するために起動されるアプリ

ケーションのメインの .dll にだけ適用します。これは、メインの .dll に呼び出されるサブモジュールに適用しないでください。適用した場合は、ランタイム エラーが発生します。

WEBSERVER(NSAPI,
entry_point_name)

このプログラムが NSAPI アプリケーションであることをコンパイラに伝えます。値 *entry_point_name* は、ユーザー定義の名前で、任意の値に設定することができます。コンパイラは、*entry_point_name* をもつ非表示のエントリ ポイントを作成し、NSAPI web サーバーにプログラムを起動させます。
entry_point_name では、ハイフン (-) を使用しないでください。
entry_point_name は、プログラムの PROGRAM-ID ヘッダと異なる名前にする必要があります。

この指令は、アプリケーションを実行するために起動されるアプリケーションのメインの .dll にだけ適用します。この指令は、メインの .dll で呼び出されるサブモジュールに適用しないでください。適用した場合、ランタイム エラーが発生します。

CASE

外部シンボル (呼び出されたプログラムの名前を含む) が大文字に変換されないようにします。この操作を行わないと、エンドユーザーがプログラムを実行しようとしたときに、Web ブラウザに「見つかりません。」というメッセージが表示されます。

REENTRANT(2)

このプログラムの複数のコピーを確実に安全に実行できるようにします。

SQL(THREAD=ISOLATE)

Open ESQl アプリケーションにのみ必要です (インターネット アプリケーション ウィザードで作成されたデータ アクセス アプリケーションを含む)。スレッドのリソースとトランザクションを、スレッド間で共有しないようにします。

Open ESQl アプリケーション (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データ アクセス アプリケーションなど) については、次の指令を使用するとパフォーマンスを改善することができます。

2. プロジェクトのビルド設定を変更する方法は、次のとおりです。

- a. NetExpress の「プロジェクト」ウィンドウで左側のペインにある ISAPI 用または NSAPI 用の .dll ファイルを右クリックし、コンテキスト メニューで [ビルド設定] をクリックします。
- b. [リンク] タブをクリックします。
- c. [リンク ウィザード] ボタンをクリックします。ウィザードを使用する場合は、サーバー側のプログラムに対して選択肢を選びます。この場合、マルチスレッドを選択します。この操作により、プ

プログラムは、動的に結合された共有マルチスレッド ランタイム システムを使用するように設定されます。

ここで、プロジェクトをリビルドして、ISAPI または NSAPI の .dll プログラムを作成します。

注記: ISAPI プログラムの場合、ISAPI プログラムを起動するフォームの Action プロパティを、*program.exe* のかわりに *program.dll* を起動するように変更する必要があります（「アプリケーション の URL 変更」を参照してください）。NSAPI プログラムの Action プロパティも変更し、新しい MIME タイプを指定する必要があります。NSAPI プログラムに必要なその他の変更については、「NSAPI アプリケーションを実行する前に」の項で説明します。

12.3.10 NSAPI アプリケーションを実行する前に

この項では、NSAPI サーバー側プログラムの実装やデバッグの前に行う必要がある 2 つの追加手順について概説します。

- accnsapi モジュールの設定
- NSAPI サーバー構成ファイルの修正
- フォームの Action プロパティの変更

これらの手順を実施し、Web サーバー マシンでファイルを実装したら、サーバーをシャット ダウンし、再起動して NSAPI プログラムを読み込む必要があります。

accnsapi モジュールの設定

サーバー側プログラムが NSAPI Web サーバーとデータを送受信するには、accnsapi.dll モジュールが必要です。使用している Web サーバーを確認するには、accnsapi.dll モジュールをビルドする必要があります。これは、Netscape サーバーがサーバーごとに異なるエン트리 ポイント名を使用しているためです。

accnsapi を設定する方法は、次のとおりです。

1. accnsapi モジュールは、Netscape サーバーと通信できるようにエン트리 ポイント名セットを認識する必要があります。エン트리 ポイント名セットは、Netscape サーバーから提供される次の .lib ファイルに記述されています。
 - Netscape FastTrack の場合 libhttpd.lib
 - Netscape Enterprise Server の場合 libhttpd.lib
 - Netscape Commerce の場合 httpd.lib

これらのファイルは、Netscape サーバー ソフトウェアを含む %server%\nsapi\examples というサブフォルダにあります。適切なファイルを %netexpress%\base\lib ディレクトリにコピーします (この操作により、ファイルのリビルド時にリンクがファイルを確実に検出することができます)。

2. NetExpress には、Netscape 社のさまざまなサーバーと動作する別のバージョンの accnsapi.obj があります。

- FastTrack

FastTrack に対してデフォルト バージョンの accnsapi.obj が設定されるため、変更する必要はありません。

- Commerce Server

デフォルト バージョンの accnsapi.obj (%netexpress%\base\lib ディレクトリに格納されています) の名前を変更して、バックアップを作成します。その後で、accnscs.obj のファイル名を accnsapi.obj に変更します。

- Enterprise Server

デフォルト バージョンの accnsapi.obj (%netexpress%\base\lib ディレクトリに格納されています) の名前を変更して、バックアップを作成します。その後で、accnter.obj のファイル名を accnsapi.obj に変更します。

3. NetExpress コマンド プロンプトを起動します (Windows の [スタート] メニューから [プログラム]、[Micro Focus NetExpress 3.0J]、[NetExpress コマンド プロンプト] の順に選択します)。

4. %netexpress%\base\lib ディレクトリから、次のコマンドを実行します。

```
cbllink -d -j accnsapi.obj netscape.lib
```

この場合、*netscape.lib* は、*libhttpd.lib* または *httpd.lib* を指します (上記の手順 1 を参照してください)。

5. accnsapi.dll を %netexpress%\base\lib から %netexpress%\base\bin にコピーします。

12.3.10.1 NSAPI サーバー構成ファイルの修正

この項では、Netscape 社の Fasttrack サーバーについて修正します。別のサーバーを使用している場合は、その製品に添付されているマニュアルを参照してください。

次の設定を行うためには、修正が必要です。

- 物理的な実行可能ファイルとエントリ ポイントを、サーバーの初期化で読み込まれるモジュールとして定義する。
- NSAPI プログラムのエントリ ポイントをサービスとして定義し、新しい MIME タイプと関連付ける。

- 新しい MIME タイプを特定の拡張子と関連付ける。

次の構成ファイルを更新する必要があります。

- obj.conf
- mime.types

obj.conf を次のように変更してください。

1. 次の行を追加して、サーバーの起動時にサーバー側プログラムが読み込まれるようにします。

```
Init fn="load-modules" shlib="executable_file.dll"

    funcs="entry-point-name"
```

パラメータの内容は、次のとおりです。

executable_file サーバー側プログラムの物理パスと名前

注記: Windows プログラムで実装する場合でも、パスでは "¥" でなく、通常のスラッシュ "/" を使用してください。

entry-point-name プログラムのコンパイル時に指令 `WEBSERVER(NSAPI,entry-point-name)` で指定するエントリ ポイント名。「共有ランタイム システムを使用する ISAPI または NSAPI プログラムのビルド」の項を参照してください。

2. <OBJECT name="default"> と <OBJECT> タグの間に次の行を追加し、新しい MIME タイプのプログラムにエントリ ポイントを関連付けます。

```
Service fn="entry-point-name" method="(GET|POST)"

type="magnus-internal/new-type"
```

パラメータの内容は、次のとおりです。

new-type プログラムに関連付けられる新しい MIME タイプの名前

entry-point-name 上記のように指定されたエントリ ポイント名

mime.types を次のように変更してください。

1. 次の行を追加し、上記の説明で定義した新しい MIME タイプと拡張子を関連付けます。

```
type=magnus-internal/new-type exts=extension
```

パラメータの内容は、次のとおりです。

new-type 上記のプログラムと関連付けた MIME タイプ名

extension プログラムのファイル拡張子。拡張子が 3 文字に制限されないことに注意してください。

Action プロパティの変更

NSAPI プログラムは、前項で定義した新しい MIME タイプをブラウザが要求するたびに起動されます。このプログラムを呼び出して、新しい MIME タイプを要求するフォームの Action プロパティを変更する必要があります。

Action プロパティを変更する手順は、次のとおりです。

1. Form Designer にページを読み込みます。
2. ページで各フォームを選択し、Action プロパティをクリックして、次のように修正します。

```
program.extension
```

この場合、*program* は、プログラムのエントリ ポイントを指します。*extension* は、前項で定義した新しい MIME タイプの拡張子です。

3. ページを保存し、新しいバージョンの .htm ファイルを実装します。

12.3.11 ISAPI に準拠した Object COBOL アプリケーションの実装

Object COBOL を使用してオブジェクト指向の ISAPI アプリケーションを作成することができますが、このようなアプリケーションを実装する場合、追加手順が必要になります。また、OO 機能を使用すると、ISAPI アプリケーションで OLE オートメーションを使用することもできます。

Object COBOL による ISAPI アプリケーションを実行する前に、Object COBOL のデフォルトの例外処理機構を無効にする必要があります。デフォルトの例外ハンドラは、00 例外が発生すると、エラー メッセージを表示して、プロセスを閉じます。ISAPI アプリケーションは、Web サーバーと同じプロセスで実行されるので、このような対応は望ましくありません。デフォルトのハンドラを無効にする方法は、次のとおりです。

1. cobopt.cfg という ASCII テキスト ファイルを作成します。
2. 次の行をファイルに追加します。

```
set signal_regime(0)=1
```

3. Application Server のランタイム システム ファイル ディレクトリ (デフォルトでは、¥program files¥micro

focus¥appserver) にこのファイルを格納します。

OLE および ISAPI

OLE オートメーションの詳細については、オンライン マニュアル『分散コンピューティング』を参照してください。

ISAPI の .dll は、Windows NT でサービスとして実行され、特定のユーザーアカウントに適用される設定やユーザー アクセス権を含みません。これは、ISAPI の .dll から OLE を使用する場合に重要です。

インプロセス OLE サーバーを使用し、サーバーの .dll ファイルを読み込める場合、アプリケーションは .dll ファイルにアクセスすることができます。

OLE のローカル サーバーを使用する場合には、サーバーの実行可能ファイルにアクセスし、起動するための正しいアクセス権を持っていることを確認する必要があります。。

アクセス権の確認方法は、次のとおりです。

1. OLE サーバーのレジストリ エントリがあることを確認します。

OLEサーバーがサードパーティのソフトウェアで作成されている場合、インストール時に自動登録するか、サーバーの登録方法について製品マニュアルを参照してユーザーが登録します。

OLE サーバーが Object COBOL を使用して作成されている場合、ユーザー自身が登録する必要があります。NetExpress の [ツール] メニューで [OLE レジストリ ファイル ジェネレータ] をクリックし、「OCREG」ダイアログ ボックスの [ヘルプ] ボタンをクリックして、説明にしがいます。

2. regedit.exe (Windows 95) または regedt32.exe (Windows NT) を実行し、サーバーに APPID 設定を追加します。

- a. HKEY_CLASSES_ROOT¥CLSID に達するまでレジストリ ツリーを開きます。
- b. 左側のペインで CLSID を右クリックし、コンテキスト メニューで [新規作成]、[キー] の順に選択します。新規キーとしてサーバーに対して UUID を入力します。

UUIDは、OLE サーバーのレジストリ エントリを作成するときに生成される値です。上記の手順 1 で作成した .reg ファイルをテキスト エディタに読み込み、ファイルに繰り返し表示される long hex 型の値をコピーします (中かっこ "{}" も含みます)。キーの名前を変更して、クリップボードの値を貼り付けます。

- c. 新しいキーを右クリックし、コンテキスト メニューで [新規作成]、[文字列] の順に選択します。右側のペインの「名前」列で新しい名前として APPID と入力します。
- d. APPID エントリをダブルクリックし、値としてもう一度 UUID を貼り付けます。

サーバーを実行するためのアカウント アクセス権を設定することができます。

1. dcomcnfg.exe を実行し、サーバーを実行するアカウントの ID を設定します。
 - a. リストからサーバーアプリケーションを選択し、プロパティを表示します。
 - b. [ID] タブを選択します。
 - c. [ユーザー] オプション ボタンを選択します。
 - d. このアプリケーションを実行するアカウントのユーザー名とパスワードを入力します。

ユーザー名とパスワードにより、アプリケーションの実行中にアプリケーションがもつセキュリティ上のアクセス権を判断します。

2. dcomcnfg.exe を使用して、上記の手順で選択したユーザー アカウントがサーバーを起動するアクセス権を持っていることを確認します。
 - a. リストからアプリケーションを選択し、そのプロパティを表示します。
 - b. [セキュリティ] タブを選択し、上記の手順で選択したユーザー アカウント、またはユーザー アカウントが属するグループの 1 つがサーバーを起動する権限を持っているかどうかを確認します。

[デフォルト] オプション ボタンが選択されている場合、「アプリケーションのプロパティ」ウィンドウを閉じ、メイン ウィンドウの [デフォルトのセキュリティ] タブをクリックして、[デフォルトを編集] ボタンをクリックしてください。

デフォルト設定に必要なユーザーが含まれていない場合、ユーザーを追加します。まず、アプリケーションの [セキュリティ] タブに戻り、「独自の起動アクセス権を使用」オプション ボタンを選択します。次に、[編集] ボタンをクリックして、リストにユーザーを追加します。

3. サーバーを含むフォルダに、サーバーを実行するユーザーのアクセス権が正しく格納されていることを確認します。
 - a. Windows エクスプローラでフォルダを選択します。
 - b. フォルダの「プロパティ」ダイアログを表示します。
 - c. [セキュリティ] タブを選択し、[アクセス権] ボタンをクリックします。

アクセス権が正しく設定されたら、ISAPI の .dll ファイルのから OLE を通してローカル サーバーに接続することができます。

12.4 アプリケーションの実装

別のマシンの Web サーバーでアプリケーションを実装する場合は、この項を読んでください。

「実装とデバッグに関するガイド」の説明にしたがって Web サーバーの設定とアプリケーション変更を行うと、アクセスを別の Web サーバー マシンに実装することができます。

次の手順にしたがって、プロジェクトにファイルをコピーします。

- Web 共有リソース *form-share* にマップされたフォルダに入力フォームの .htm ファイルをコピーします。

ページが常に 1 つのサーバー側プログラムから EHTML により出力される場合、入力フォームの .htm ファイルは、プログラムの実行コードに埋め込まれているため、Web サーバーにコピーする必要はありません。インターネット アプリケーション ウィザードにより生成されたすべてのコードは、フォームを出力するために EHTML を使用します。
- Web 共有リソース *form-share* にマップされたフォルダに、入力と出力ページの .js ファイルをコピーします。

これらのファイルは、設定、cookies の受信、およびクライアント側の確認機能など、機能についての JavaScript を含みます。
- アプリケーションにイメージ ファイルがある場合、Web 共有リソース *image-share* にマップされたフォルダにこれらのイメージ ファイルをコピーします。
- Web 共有リソース *bin-share* にマップされたフォルダにアプリケーションの実行可能ファイルをコピーします。

Web 共有リソースは、「実装用の Web サーバー設定」の項で設定したものです。

次の項では、特定のアプリケーションの種類にだけ適用される特別な情報を説明します。

- ISAPI および NSAPI アプリケーションにおける従属関係の検索

ISAPI または NSAPI アプリケーションを実装している場合は、この項をお読みください。
- ODBC データ ソースの使用

ODBC データ ソース (これ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たデータ アクセス アプリケーションも含まれます) を使用している SQL アプリケーション実装している場合は、この項をお読みください。

12.4.1 ISAPI および NSAPI アプリケーションにおける従属関係の検索

この項では、これらの基準のいずれかを満たす ISAPI または NSAPI アプリケーションを実装している場合に必要な手順を説明します。

- 他の .dll ファイルでいくつかのサブプログラムを呼び出すメインの .dll ファイルを構成するアプリケーション

「ISAPI および NSAPI アプリケーションからのサブプログラムの検索」の項を参照してください。

- 「サーバー側のプログラミング」の章で説明されている拡張 DISPLAY 構文を通じて HTML ページを出力するアプリケーション

「ISAPI および NSAPI アプリケーションからの HTML ファイルの検索」の項を参照してください。

CGI プログラムとは異なり、単にメイン プログラム自体と同じディレクトリに ISAPI と NSAPI プログラムを置いた場合は、ISAPI と NSAPI プログラムは依存しているファイルを検索しません。ISAPI または NSAPI アプリケーションは Web サーバー処理内部のスレッドとして実行されるため、現在の作業ディレクトリは、メインの .dll プログラムが配置されているディレクトリではなく、常に Web サーバーの現在の作業ディレクトリに設定されます。

ISAPI/NSAPI アプリケーションからの サブプログラムの検索

ISAPI または NSAPI アプリケーションが呼び出す別の .dll ファイルを、ISAPI または NSAPI アプリケーションに検索させるようにするには、アプリケーションが実装されているそれぞれのマシンで次の手順を実行します。

1. 単一のディレクトリにすべての .dll ファイルをコピーします。

メインの ISAPI または NSAPI の .dll ファイルに使用しているディレクトリと同じディレクトリを使用できます。

2. マシンに NetExpress またはアプリケーション サーバー パスへのディレクトリ位置を追加します。

- a. コマンド プロンプトから、レジストリ エディタ (Windows 95 では regedit を、Windows NT では regedt32 を使用してください) を開始します。

- b. 次のキーを検索します。

HKEY_LOCAL_MACHINE/SOFTWARE/Micro Focus/3.0/COBOL/3.0/Environment

- c. PATH 値と .dll ディレクトリの位置を連結します。

ISAPI/NSAPI アプリケーションからの HTML ファイルの検索

ISAPI または NSAPI アプリケーションに、拡張 DISPLAY 構文を通じて ISAPI または NSAPI アプリケーションが出力した HTML ファイルを検索させるようにするには、アプリケーションが実装されているそれぞれのマシンで次の手順を実行します。

1. 単一のディレクトリにすべての HTML ファイルをコピーします。

2. マシンに NetExpress またはアプリケーション サーバー COBDATA パスへのディレクトリ位置を追加します。

- a. コマンド プロンプトから、レジストリ エディタ (Windows 95 では regedit を、Windows NT で

は regedt32 を使用してください) を開始します。

- b. 次のキーを検索します。

HKEY_LOCAL_MACHINE/SOFTWARE/Micro Focus/3.0/COBOL/3.0/Environment

- c. COBDATA 値が表示されない場合は、1 を追加します。
- d. 次の値を設定します。

```
;;%COBDIR%;html-dir
```

ここでは、*html-dir* は HTML ファイルがあるディレクトリです。最初にある 2 のセミコロンは重要であり、省略できません。

12.4.2 ODBC データ ソースの使用

アプリケーションで ODBC データ ソースを使用している場合は、サーバー側プログラムでソースが見えているか確認する必要があります。サーバー側プログラムは通常、それらを開始する Web サーバーと同じユーザー アクセス権を継承します。Web サーバーが ODBC データにアクセスできるかを確認するには、次の作業を行います。

- データ ソースをユーザー データ ソースではなく、システム データ ソースとして設定します。

Web サーバーはユーザー データ ソースにはアクセスできなくなります。

- データ ソースをユーザー DSN ではなく、システム DSN に割り当てます。

ユーザー DSN での作業が可能となりますが、正しいアクセス権の設定は難しくなります。

12.5 UNIX へのアプリケーションのパブリッシュ

「実装手順ガイド (UNIX サーバー)」の項で概説した設定とアプリケーションの変更を実行すると、UNIX サーバーにアプリケーションをパブリッシュすることができます。

1. NetExpress でアプリケーションの実装用コピーを開きます。
2. UNIX サーバーでは、アプリケーションのすべてのファイルを格納するディレクトリを作成します。
3. [UNIX] メニューで [設定] をクリックし、アプリケーションについてサーバー オプションを設定します。

詳細については、『UNIX オプション ユーザー ガイド』を参照してください。

4. [UNIX] メニューで [パブリッシュ] をクリックし、サーバーにプロジェクト ファイルを送信して、サーバーで実行可能プログラムをビルドします。
5. アプリケーションの CGI プログラムを実行するために、シェル スクリプトを作成します。

シェル スクリプトは、COBOLプログラムを実行する前に COBOL 環境を設定するために必要です。

- a. 次のようにスクリプトを作成します。

```
#!/bin/sh  
  
./mfenv.sh cginame
```

- b. このスクリプトを *cginame.sh* として保存します。

6. パブリッシュ先のディレクトリにあるファイルを次の手順にしたがってコピーします。

- Web 共有リソース *form-share* にマップされたディレクトリに入力フォームの .htm ファイルをコピーします。
- DISPLAY 動詞により CGI プログラムから出力されたフォームの .htm ファイルをサーバー側プログラムと同じディレクトリにコピーします。

常にサーバー側プログラムから EHTML により出力されるフォームは、コピーする必要がありません。EHTML では、プログラムの実行コードに出力フォームが埋め込まれています。インターネット アプリケーション ウィザードにより生成されたプログラムは、すべて、フォームを出力するために EHTML を使用します。

- アプリケーションにイメージ ファイルがある場合は、Web 共有リソース *image-share* にマップされたフォルダにこれらのイメージ ファイルをコピーします。
- Web 共有リソース *bin-share* にマップされるフォルダにアプリケーションの実行可能ファイルをコピーします。手順 5 で作成したシェル スクリプト *mfenv.sh* と CGI 実行可能ファイルも必ずコピーしてください。

Web 共有リソースは、「実装用の Web サーバー設定」の項で設定したものです。

12.6 アプリケーションのデバッグ

開発マシンで Solo 以外の Web サーバーを使用してアプリケーションをデバッグする場合は、この項を読んでください。

「実装とデバッグに関するガイド」の説明にしたがって Web サーバーの設定とアプリケーションの変更を行うと、アプリケーションのデバッグを開始することができます。CGI プログラムと ISAPI または NSAPI プログラムのデバッグ手順は、多少異なるため、次の 2 項に分けて説明します。

12.6.1 CGI プログラムのアニメート

この項では、「実装とデバッグに関するガイド」の説明にしたがって Web サーバーの設定とアプリケーションの変

更が実行されていることを前提としています。CGI プログラムは、Solo 以外の Web サーバーでアニメートする場合、最も簡単に処理できます。CGI プログラムをアニメートするには、NetExpress IDE でプロジェクトを多少変更する必要があります。変更後は、デバッグを開始することができます。

アニメーション用のプロジェクト設定手順は、次のとおりです。

1. CGI の .exe ファイルを右クリックし、コンテキスト メニューで [ビルド設定] をクリックします。[リンク] タブをクリックし、.exe が、グラフィック アプリケーションではなく、文字アプリケーションとしてビルドされることを確認します。必要に応じて、設定を変更し、リビルドします。
2. NetExpress の [オプション] メニューで [実行] をクリックし、「アニメータ実行オプション」ダイアログ ボックスを表示します。
3. 「CGI プロジェクトに Solo web サーバーを使用する」チェック ボックスのチェックを解除し、[OK] をクリックします。

注記: この設定は、このプロジェクトと設定後に読み込む他のプロジェクトに適用されます。Solo を再度使用するには、オプションを元に戻します。

4. NetExpress の [アニメート] メニューで [設定] をクリックし、「アニメート設定」ダイアログ ボックスを表示します。
5. 「アニメーションのアタッチメントを待つ」チェックボックスをチェックします。
6. 「アニメーションの起動」フィールドに URL を入力してアプリケーションを起動します。URLは、次の形式で入力します。

`http://machinename.domain/COBOL/inputform.htm`

または

`http://machinename.domain/CGI-BIN/program.exe`

machinename.domain イン트라ネットまたはインターネットでのマシンの位置を指定します。たとえば、`www.microfocus.com` または `dev1.slithy.com` のように指定します。

inputform アプリケーションを起動するフォームを含むページ名。

program アプリケーションを起動するプログラム名。

プログラムをアニメートする手順は、次のとおりです。

- NetExpress の [アニメート] メニューで [アニメート開始] または [ステップ実行] をクリックします。

この操作により、デフォルトの Web ブラウザが起動し、アプリケーションの起動ページまたはプログラムを読み込みます。CGI プログラムの実行が開始されると、NetExpress はアニメーションを開始し、強調表示された最初の行が実行されます。これで、プログラムをデバッグできるようになります。

12.6.2 ISAPI および NSAPI プログラムのアニメート

この項では、「実装とデバッグに関するガイド」の説明にしたがい、プログラムを ISAPI または NSAPI プログラムとしてリビルドするための変更を含めて、Web サーバーの設定とアプリケーションの変更が完了していることを前提としています。Solo を使用して ISAPI または NSAPI プログラムを実行したり、デバッグすることはできません。そのため、これらのプログラムをデバッグする場合は、ISAPI または NSAPI をサポートする Web サーバーを使用する必要があります。ISAPI および NSAPI プログラムをデバッグするには、NetExpress IDE の設定を変更し、各 ISAPI または NSAPI プログラムの先頭に文を追加する必要があります。

プロジェクトをアニメートするための設定手順は、次のとおりです。

1. NetExpress の [オプション] メニューで [実行] をクリックし、「アニメータ実行オプション」ダイアログ ボックスを表示します。
2. 「CGI プロジェクトに Solo web サーバーを使用する」チェック ボックスのチェックを解除し、[OK] ボタンをクリックします。

注記: この設定は、このプロジェクトと設定後に読み込む他のプロジェクトに適用されます。Solo を再度使用するには、オプションを元に戻します。

3. 各 ISAPI または NSAPI プログラムの先頭に次の文を追加します。

```
call "CBL_DEBUGBREAK"
```

この文を実行すると、NetExpress が起動し、デバッガも起動します。

4. プロジェクトをリビルドします。

Web サーバーがすでにアプリケーションの .dll ファイルを読み込んでいる場合、その .dll ファイルはロックされ、上書き処理できません。Web 共有リソースが、ファイルがビルドされたディレクトリを指すように設定されている場合、NetExpress でプロジェクトを適切にリビルドするには、Web サーバーをシャットダウンする必要があります。Web 共有リソースが別のディレクトリにある場合、リビルドした .dll ファイルにコピーするには、Web サーバーをシャットダウンする必要があります。

アプリケーションをアニメートする方法は、次のとおりです。

1. NSAPI Web サーバーを使用中である場合、そのサーバーを停止してから再起動し、NSAPI.dll プログラムを読み込みます。
2. Web ブラウザを起動し、アプリケーションを起動するページまたはプログラムの URL を入力します。

ISAPI プログラムが文 CALL"CBL_DEBUGBREAK" を実行すると、デバッガを起動するかどうかをたずねるメッセージ ボックスが表示されます。

3. [はい] ボタンをクリックします。

デバッガは、デバッグに必要な .idy ファイルの場所を検索できないため、「IDY ファイル エラー」ダイアログ ボックスが表示されます。

4. [参照] ボタンをクリックし、[開く] ダイアログ ボックスで、プログラムの .idy ファイルを検索します。実行可能ファイルは、通常、ソース ファイル (プロジェクト) ディレクトリの debug または release サブディレクトリ (ビルド タイプにより異なります) に格納されています。

これで、プログラムをデバッグできるようになります。

注記:

- NetExpress が ISAPI または NSAPI プログラムを実行している間に、アニメートを停止すると、Web サーバーも停止します。このような障害が発生した場合は、Web サーバーを再起動します。ISAPI または NSAPI プログラムの実行中またはアニメート中に、STOP RUN 文を入力すると、同じことが起こります。ISAPI または NSAPI プログラムでは、STOP RUN 文のかわりに、常に EXIT PROGRAM 文を使用してください。
- ISAPI または NSAPI の .dll ファイルが読み込まれると、Web サーバーが停止するまで読み込まれた状態になります。デバッガを起動するかどうかをたずねるメッセージに対して [いいえ] をクリックした場合 (上記の手順 2)、.dll ファイルを強制的に再読み込みするために、Web サーバーを強制的に停止および再起動させないかぎり、それ以降の実行時には、このプロンプトが表示されません。同様に、[はい] をクリックすると、プログラムが実行されるたびに、NetExpress の内部でアニメートされます。

付録A: WWW の概要

World-Wide Web(WWW) は、非常に短い間にコンピュータに対する考え方を变化させてしまいました。新しい企業や製品がほぼ一夜にして生まれ、コンピュータの様相を变化させています。しかし、Web による対話性が実現されたのは、ほんの最近のことです。

Web の原点は、1960年代と1970年代に米国政府が出資した研究プロジェクトにさかのぼります。このプロジェクトは、多くの災害に耐えうるネットワーク基盤を構築するためのものでした。研究の結果、コンピュータ同士をゆるく結び付けた、インターネットと呼ばれるネットワークが誕生しました。

インターネットは、1つの集中ホストを使用しない点、また、情報を転送する場合に定義済みの接続パスに依存しない点で、独創的なものでした。インターネットの特徴は、ネットワークを通じてソースからデスティネーションへ転送されるルートを「情報のパケット自体が見つける」ことでした。その結果、いくつかのノードに障害がある場合でも機能し続けるネットワークが生まれました。

1980年代から1990年代の初頭まで、インターネットは、主に大企業や大学で電子メールと文書を転送するために使用されていました。インターネットが本当に飛躍的に発達したのは、Web ブラウザと HTML と呼ばれる言語が開発されてからです。HTML (Hyper Text Markup Language) は、スイスで研究者として働いていた Tim Berners-Lee により開発されました。HTML の元来の目的は、科学者間での情報転送を迅速化および簡便化することでした。HTML の背景にある基本的な考え方は、標準の ASCII テキストにタグ (たとえば、など) を挿入し、ブラウザと呼ばれるソフトウェアでこれらのタグを解釈するというものです。

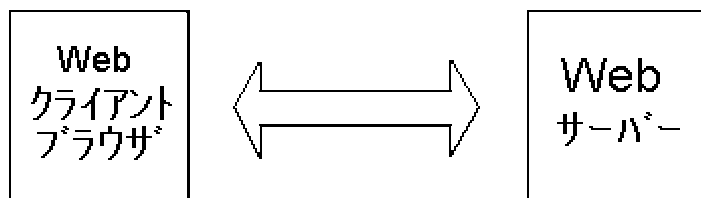


図 A-1 Web サーバーと Web ブラウザ

最初のブラウザ Mosaic は、すぐに成功を収めました。ユーザーは、特別に強調表示された単語やリンクをクリックすると、他のコンピュータにあるテキストやグラフィックスを表示することができました。HTML は、オーバーヘッドが少なく、簡単であるため、すべてのインターネット ユーザーにすぐに受け入れられました。その結果が、信頼性が高く、障害に強いネットワーク基盤を使用した簡単なデータ転送プロトコル、つまり、今日の World-Wide Web です。

A.1 Web と商取引

多くの人々にとって、個人で Web をサーフィンすることが日常生活の一部となりました。企業は、広告、マルチメディア、電子商取引の面で Web を活用するために、急速に Web の存在を確立しようとしています。その結果、Web は、情報転送やデータの調査に対する考え方を変えただけではなく、企業の営業方法も変えつつあります。Web の背景にある基本的な考え方を最近拡張したものが、イントラネットとエクストラネットです。

A.1.1 イン트라ネット

インターネットがメディアや公共の注目を集める一方で、企業は情報やアプリケーションを企業内に配布するための新しい方法を模索しているため、社内インターネット、つまりイントラネットが成長しつつあります。イントラネットでは、外部からの企業データに対するアクセスを防止するソフトウェア、ファイアウォールを使用しています。そのため、イントラネットは、企業全体への情報配布に使用され、退職規定から企業の重要な企画までのすべての最新情報を従業員に提供しています。イントラネットでは、インターネットと同様、企業のサーバーに接続された PC やワークステーションにインストールされた Web ブラウザを使用します。

Web ブラウザには、プログラミングしなくても、テキスト、グラフィックス、オーディオ、またはビデオを表示できるという利点があります。ブラウザは、マルチメディアに対応し、さまざまな標準ファイル形式を表示することができるようになりました。業務の改善を模索していた企業にとって、Web は、スケジューリング、レポート配布、会議などに加えて、日常的に社内や社外で発生した大量の情報をさまざまな技術的なプラットフォームで管理する活動を改善する有効な手段です。

A.1.2 エクストラネット

多くの企業では、すでにイントラネットを開発しており、現在は、社外の特定のサイトを含めるように拡張されつつあります。エクストラネットと呼ばれる新しいネットワーク構造は、イントラネットを拡張し、企業運営に不可欠なサイトまで含めたものです。たとえば、メーカーが、イントラネットを拡張して重要な商品や材料のサプライヤを含め、これらのサプライヤからの情報を企業の企画部署に提供するような場合を指します。具体的な例としては、GE が Trading Process Network (TPN) と呼ばれる独自のアプリケーションを作成しています。この TPN では、選定された下請け業者が GE のデータベースを検索し、GE のイントラネット Web サーバーに接続された安全なリンクを介して入札を行うことができます。Web 接続ではこのような方法を実現できるので、企業は顧客のニーズを満たすための新しい方法を開発することができます。

A.2 Web とクライアント サーバー

Web は、従来のクライアントサーバー型コンピューティングについて再考を促す結果となりました。1960年代と1970年代に、企業情報のリソースは、主にデータの取り込みと表示に使用されるダム端末と対話するメインフレームに格納されていました。PC とデスクトッププラットフォームが強力になるにつれて、ローカルエリアネットワーク (LAN) によりアプリケーションを分割し、メインフレームサーバーに集中していた負担を一部軽減できるようになりました。

その結果、強力なクライアント間で処理を分散するクライアント サーバー コンピューティング モデルが誕生しました。クライアントのプラットフォームが機能するにはより大きいメモリとディスク領域を必要とするため、これらのクライアントは、データの表示と入力だけを行う前世代のダム端末や Thin Client などと区別して、fat client と呼ばれるようになりました。

以前は、アプリケーションの汎用性が常に問題となり、情報システム管理者は、企業内でクライアント サーバー アプリケーションを実装するプラットフォームの種類について注意する必要がありました。今日では、完全なネットワーク環境に移行したため、プラットフォームに依存しないパートナー、同僚、顧客などと共有する機能が問題となっています。

A.3 メインフレームの復活

ネットワークを中心とした、クライアント サーバー コンピューティングの新しい図式が確立された結果、メインフレームが企業の重要なリソースとして復活しました。Thin Client では、データとプログラムの両方をサーバーから配布するため、馬力のあるサーバーをもつことが重要になります。そこで、一般的にはメインフレームが登場します。

Web ブラウザは、世界中の Web サーバーと接続できるため、Web ブラウザからサーバーへの接続が表面的には従来のクライアント サーバー接続に見える場合でも、Web に新しい工夫が追加されています。このように、従来のクライアント サーバーの定義を広げることにより、アプリケーションがイーサネット LAN や SNA 接続の制約のような従来の限界を超えることができるため、コンピュータ界が非常に活性化されています。また、多くの企業の中心にある膨大な COBOL プログラムを活用するよい機会でもあります。

A.4 Web と NetExpress

COBOL アプリケーションと COBOL プログラムをもつ組織では、Web を使用すると選択肢と機会が大きく広がります。レガシー COBOL アプリケーションに NetExpress で作成した簡単な文を追加して Web 接続を行うことにより、これらの COBOL プログラムに新しい命を吹き込むことができます。また、COBOL プログラムは、COBOL を使用して企業の重要なアプリケーション、データなどの間の接続をプログラムできます。COBOL を使用できる点は重要です。これにより、レガシー COBOL アプリケーションの価値が上がります。特に、Web で情報を受け付けて表示するための NetExpress の組み込み機能と併用すると、さらに価値が高まります。

また、NetExpress を使用すると、COBOL の新たな役割として、Web サーバー アプリケーションを作成することができるようになります。COBOL は、30 年以上にわたりメインフレーム アプリケーションの作成に使用された言語なので、サーバー アプリケーションを作成できるのは当然のことです。このように、既存の COBOL アプリケーションを新しい言語で再作成する危険を冒すより、NetExpress に組み込まれた Web アクセス言語機能を使用して、レガシー プログラムを拡張し、Web に対応させる方が便利です。NetExpress でこのような操作を行う方法について説明する前に、Web ブラウザと Web サーバーの間で情報を受け渡す方法について解説します。

A.5 Web が機能する原理

Web の背景にある基本的な原理は、簡単です。1 台のコンピュータにインストールされた Web ブラウザと呼ばれるプログラム (Netscape Navigator や Microsoft インターネット エクスプローラなど) が、Web サーバーから Web ページを要求します。



図 A-2 Web サーバーと対話する Web ブラウザ

Web ページを要求すると、Web サーバーからクライアントの Web ブラウザに文書が転送されます。World-Wide Web (WWW) は、次の構成要素で構成されていると考えることができます。

- **Web ブラウザ:** Netscape Navigator や Microsoft インターネット エクスプローラなどの Web ブラウザは、コンピュータ画面に Web ページを表示するためのソフトウェア プログラムです。Web ブラウザでは、一般的に `http://www.microfocus.com` などの URL (Uniform Resource Locators) を入力し、リモート コンピュータ サイトから受信した Web ページを表示します。Web ブラウザを使用するには、コンピュータをインターネットに接続する必要があります。そのためには、モデムやその他のネットワーク接続を必要とします。
- **Web サイト:** Web サイトは、組織や個人により維持されるコンピュータ システムであり、ここから Web ブラウザで Web ページをダウンロードすることができます。Web ブラウザのユーザーは、Web サイトに接続します。Web サイトは、商用、政府用、教育用などに分類されます。拡張子を確認すると、接続している Web サイトの種類を識別することができます。

エンティティ	拡張子	完全な Web サイト名
Corporate	com	<code>http://www.microfocus.com</code>
Government	gov	<code>http://www.whitehouse.gov</code>
Educational	edu	<code>http://www.seas.smu.edu</code>

- **Web ドキュメント:** Web ドキュメントは、Web サイトに常駐するファイルであり、Web ブラウザによりダウンロードします。Web ドキュメントは、通常、Web の他のドキュメントへのリンクを含みます。リンクをクリックすると、同じ Web サイトまたは別の Web サイトに接続され、他の Web ドキュメントが表示されます。Web リンクを使用して、ドキュメント間で移動することを通常、Web サーフィンと呼びます。

- *Web サーバー*: Web サーバーは、Web サイトで実行されるソフトウェア アプリケーションであり、Web ブラウザからの要求を処理します。サーバーは、Web ブラウザに Web ページを提供します。Web サーバーソフトウェアは、HTML でコード化された Web ページ (ピクチャ、オーディオ、場合によってはビデオなど) を Web ブラウザのクライアントに返します。さらに重要なことは、Web サーバーがユーザーから情報を取り込み、サーバーに常駐する他のプログラムを起動できることです。この機能により、Web から企業データベースやレガシー COBOL アプリケーションにアクセスすることができます。
- *ゲートウェイ プログラム*: ユーザーのかわりに、Web サーバーからの要求を受け付け、処理します。Visual Object COBOL では、Web の標準 CGI (Common Gateway Interface) をサポートしているため、COBOL を使用して、Web のゲートウェイ プログラムを作成することができます。ゲートウェイ プログラムを使用すると、ユーザーの要求に柔軟に対応できます。また、ゲートウェイ プログラムを使用すると、動的な Web ページを手軽に作成できるだけでなく、データベースへの問い合わせや、ユーザーに対する結果のパッケージ化が可能になります。

A.6 Web ブラウザ

Netscape Navigator や Microsoft インターネット エクスプローラなどの Web ブラウザを使用すると、Webサーバーからユーザーのコンピュータに転送された Web ページを表示できます。Web ブラウザごとに表示形態が異なることがあります。すべての Web ブラウザは、Web リソース、つまり、Web サーバーに常駐する情報ファイルを表示する機能を共有しています。Web に関して興味深いことは、テキストだけでなく、グラフィックス、サウンド、アニメーション、ビデオ、さらにはバーチャル リアリティなどのファイルも表示できることです。そのため、Web は、さまざまな種類の情報を配布できる本当の意味での汎用プラットフォームと言えます。実際に、Web ブラウザには、Web ページの表示機能だけでなく、組み込みの電子メール サポート機能もあり、Usenet ディスカッション グループに接続したり、ftp (ファイル転送プロトコル) を使用してロード ファイルをダウンロードすることもできます。これらすべてに共通する鍵は、URL です。

A.7 URL

インターネットに接続したコンピュータは、URL を使用して、Web ブラウザとターゲット Web サーバー間の接続を確立します。URL は、対象となるリソースを検索するディスクの場所を正確に Web サーバーに伝えるための情報です。

A.7.1 Web サイト

各 Web サイトには固有のネット アドレスがあり、URL にはこの情報を埋め込みます。たとえば、Micro Focus の Web アドレスは `www.microfocus.com` であり、URL に埋め込むと、次のようになります。

```
http://www.microfocus.com
```

Web ブラウザの最上部に上記の名前を入力すると、Micro Focus のホームページのメイン ページに進みます。URL は、企業名に対応していることが多いですが、必ずしもその必要はありません。

A.7.2 ドキュメント URL

企業名の後に複数の項目をもつ URL が表示されることがあります。たとえば、次のような場合です。

```
http://www.microfocus.com/netexpress/net.html
```

URL を使用すると、企業のホーム ページに接続できるだけでなく、Web サーバーに格納された個々のドキュメントに接続することができます。Web ページのリンクをクリックすると、Web サーバーにあるサブディレクトリ内の特定のドキュメントに移動することができます。Web ブラウザに完全な URL を入力すると、リンクを使用する必要がなく、ドキュメントに直接、進むことができます。

```
http://www.microfocus.com/netexpress/net.html
```

たとえば、上記の URL を使用すると、Micro Focus 社のサーバーにある、ネットワークと NetExpress に関する特定のドキュメントに移動します。

A.8 URL の分析

URL は、一般的に次のような形式に基づいています。

```
protocol://hostComputer/path
```

次の項では、URL の各構成要素について説明します。

A.8.1 プロトコル

プロトコルは、2 つのエンティティ間で通信するための法則です。US Department of State では、外国の高官と接する方法について規約書を保有しています。コンピュータ間の対話プロトコルも同様の目的を果たします。つまり、プロトコルでは、データの送受信に関する規則を定義しています。

Web で最も一般的なプロトコルは、HTTP (HyperText Transfer Protocol) です。HTTP は、Web ブラウザ (クライアント) と Web サーバー間の通信規約を指定します。HTTP は、最も広く使用されているプロトコルですが、Web では他のプロトコルも使用されています。他のプロトコルには、Web でデータ ファイルやプログラム ファイルを転送するための FTP (ファイル転送プロトコル)、ファイルのメニューを転送するための gopher、電子メールを転送するための mailto などがあります。

A.8.2 ホスト コンピュータ

次のプロトコル名と 2 つのスラッシュ (//) は、ホスト コンピュータの名前です。上記の例では、ホスト コンピュータ名は、www.microfocus.com で、通常、Web サーバーのドメイン名と呼ばれます。ドメイン名は、右から左へ次のように読みます。

- *com* は、コンピュータが企業によりサポートされていることを示します。

- *microfocus* は、企業名が Micro Focus であることを示します。
- *www* は、Web サーバー ソフトウェアを実行する、企業のマシン名を指します。

実際には、Web ブラウザから Web に要求が送信されると、テキストのドメイン名が Web サイトを検索するための数字 IP (インターネット プロトコル) アドレスに変換されます。インターネット プロトコルは、インターネットを介したコンピュータ間通信の基礎であり、TCP/IP と呼ばれる下位の通信プロトコルの一部です。IP アドレスは、3 桁の番号を 4 グループ並べ、グループ間をピリオドで区切ったものです。

`www.microfocus.com` の実際の IP アドレスは、`204.160.128.10` で、Micro Focus ホームページに接続する場合は、ブラウザ ウィンドウの最下部に表示されます。ブラウザでは、これら 2 つのアドレスは同じ内容ですが、英語で表記した方が便利で、覚えやすくなっています。

A.8.3 パス

ホスト コンピュータ名に続くテキストは、検索するドキュメントのパス名として解釈されます。上記の例では、パスは `netexpress/net.html` で、`netexpress` はファイル `net.html` を格納するディレクトリです。パスが指定されていない場合、`http` プロトコルのデフォルトにより、`public_html` という名前のディレクトリと `index.html` という名前のファイルが検索されます。

A.8.4 オプションのインターネット ポート

Web ブラウザから Web サーバーに接続した場合、Web に関する複数の作業を同時に行うことができます。たとえば、同じサーバーから Web ページをダウンロードし、`ftp` を使用してファイルを転送し、電子メールを確認することができます。つまり、サーバーは、アクティビティごとに別々のポートを使用します。より簡単に説明すると、プロトコル ベースの接続ごとにデフォルトのポートがあります。たとえば、次の URL を入力した場合、ポート番号は 80 であると考えられます。

`http://www.microfocus.com`

URL でポート番号を指定する必要がある場合もあります。ポート番号は、ホスト コンピュータ名の一部として記述し、次のようにコロンを前に付けます。

`http://www.microfocus.com:80`

プロトコル	説明	URL 形式
FTP	あるマシンから別のマシンにファイルを転送します。	<code>ftp://user:password@host/pathname</code>
Gopher	リモート コンピュータのディレクトリ構造を一覧表示します。	<code>gopher://host/pathname</code>

HTTP	Web ドキュメントを転送します。	http://host/path
Mailto	インターネット メールを送信します。	mailto:name@hostcomputer
News	USENET ニュース グループを読み込みます。	news:newsgroup
Telnet	別のコンピュータに接続します。	telnet://user:password@host/pathname

A.9 HTML

最も一般的なインターネット プロトコルは、HTTP です。HTTP は、HyperText Transfer Protocol の略です。HTTP の基礎は、HTML (Hyper Text Markup Language) です。

表示されるすべての Web ページのバックグラウンドには、HTML タグを使用してコード作成したテキスト ファイルがあります。Web ページは、実際には Web サーバーから Web ブラウザに送信されるタグ付きの ASCII テキストです。Web ブラウザは、タグを解釈し、タグとテキストに基づいて Web ページを表示します。

HTML には、Web ページの見出し、リスト、フォームなどの要素のコードがあります。HTML では Web ブラウザによる表示形式を指定しないことに注意してください。HTML は、単に、ドキュメントの一部が見出し、テキスト、リストまたはフォームであるということを伝えるだけです。HTML の解釈と表示は、Web ブラウザに任されています。そのため、Web サーバーと Web ブラウザ間のデータ転送が大幅に簡略化されます。ただし、ブラウザによってページが異なって表示されることがあります。Web ページを開発する場合は、さまざまなブラウザでページを表示し、問題が発生しないことを確認することが重要です。

A.9.1 HTML タグ

HTML タグは、Web ドキュメントに埋め込まれます。Web ブラウザがタグを発見すると、タグを解釈し、書式を指定します。たとえば、HTML タグ は、その後のテキストが強調表示されることを示します。ただし、強調表示の方法はブラウザに依存します。テキストは、ブラウザの設定ごとに、斜体、太字、色付きなどさまざまな方法で表示されます。このことから、Web ページをテストする場合、さまざまなブラウザを使用することが重要であるといえます。

A.9.2 HTML の概要

HTML ドキュメントのテキスト中に記述された HTML タグは、ページの構造やフォームに関する情報をブラウザに提供します。下表は、一般的な HTML タグをまとめたものです。

A.9.2.1 ドキュメント構造

<HTML>...</HTML>	HTML ドキュメント全体を囲みます。
------------------	---------------------

<HEAD>...</HEAD>	HTML ドキュメントの見出しを定義します。
<BODY>...</BODY>	HTML ドキュメントの本文を定義します。

A.9.3 タイトルと見出し

<TITLE>...</TITLE>	ドキュメントのタイトルを定義します。
<H1>	第 1 レベルの見出しです。
<H2>	第 2 レベルの見出しです。

A.9.3.1 段落

<P>...</P>	段落の先頭と末尾を定義します。
------------	-----------------

A.9.3.2 リスト

...	順番に並べられた (番号付きの) リストの先頭と末尾を示します。
...	番号なしリストの先頭と末尾を示します。
<MENU>...</MENU>	項目のメニュー リストの先頭と末尾を示します。
	、 または <MENU> と併用するリスト要素です。

A.10 Web のマルチメディア

Web ブラウザを使用すると、さまざまなマルチメディアを表示することができます。Web ブラウザは、テキスト、グラフィックス、ビデオ、オーディオ、さらにはバーチャル リアリティをも含めて、多くの種類の情報を表示することができます。Web ブラウザは、プラグインやアドオンを使用することで、多くのマルチメディア形式をサポートすることができます。プラグインやアドオンは、Web サーバーから送信されるさまざまな標準ファイルの表示方法についてブラウザに指示します。

たとえば、Web の開発時に、イメージ用の HTML タグ <IMG...> とイメージを格納したファイル名を指定するための内部コードを記述すると、グラフィック要素を Web ページに追加することができます。たとえば、次のように記述します。

```
<IMG SRC="mypicture.gif">
```

上記のコードでは、ファイル mypicture.gif に格納されたイメージを Web サーバーから読み込み、Web ブラウザで表示します。

以下の項では、インターネット、イントラネット、エクストラネットのアプリケーションで使用できる標準的なマルチメディア コンポーネントの種類について簡単に説明します。

A.10.1 グラフィックス

Web で使用される最も一般的なグラフィック イメージは、GIF (Graphics Interchange Format) と JPEG (Joint Photographic Experts Group) です。GIF ファイルは、最大 256 色をサポートし、簡単なグラフィックスと描画に最も適しています。JPEG ファイルは、24 ビットのグラフィックスを使用しているため、100 万色以上をサポートしており、写真に適しています。

A.10.2 オーディオ

Web オーディオは、8 ビット サンプリングに基づいており、ボイスと低品質の音楽に向いています。共通のオーディオ ファイル形式は、.wav と .au です。これらのファイルを処理できるようにブラウザを構成すると便利です。

A.10.3 ビデオ

Web のビデオ ファイル形式は、MPEG (Motion Pictures Expert Group) 形式、Apple 社の QuickTime 形式、または Intel 社の Indeo 形式になります。QuickTime と Indeo は、オーディオとビデオの同期をサポートしています。

A.10.4 アプレット

Web ベースのアニメーション (動画とオーディオの組み合わせ) は、Web サイトからダウンロードできるアプレットと呼ばれる小さなプログラムを使用して実現することができます。アプレットは、Sun Microsystems 社により開発された Java プログラミング言語で作成されます。最新のブラウザには、Java アプレットを実行する機能が装備されています。

A.10.5 3-D バーチャル リアリティ

3 次元の図形とランドスケープは、VRML (バーチャル リアリティ モデリング言語) 形式のファイルを使用して Web サーバーから Web ブラウザに転送することができます。VRML は、新しいモデリング方法とデータの表示方法を提示する言語と見なされています。

A.11 Java

Web ブラウザは、テキスト、グラフィックス、サウンドおよびビデオを合わせて表示できるという点で優れていますが、Sun Microsystems 社が開発したインターネット アプリケーションの新しいプログラミング言語である Java は、Web をさらに進化したものにしていきます。Java で作成されたプログラムは中間のバイトコード表現にコンパイルされ、Web ブラウザに送信されます。Web ブラウザは、バイトコード プログラムを解釈し、クライアント マシンで

そのプログラムを実行します。

この方法の利点は、Java で作成されたアプリケーションまたはアプレットが Java に対応したブラウザを使用するあらゆるプラットフォームにアクセスできる点です。基礎となるオペレーティング システムやハードウェアは関係ありません。企業がこの技術に期待している理由は、Web で簡単にソフトウェアや資料を配布できるためです。

A.11.1 COBOL および Java

Java は、NetExpress と同様、オブジェクト指向です。そのため、COBOL または Java で作成されたプログラムは、Microsoft 社の DCOM や CORBA 準拠のオブジェクト リクエスト ブローカ (ORB) のようなオブジェクト指向の基盤を使用すると、互いに簡単に通信することができます。これは、Web を中心としたコンピューティング環境で COBOL アプリケーションの新しい可能性を導きます。Web に接続できることにより、既存のプログラムの使用範囲が広がるだけでなく、世界中の汎用クライアントである Web ブラウザへ接続する新しく革新的な Web サーバーアプリケーションを COBOL により開発することができます。

A.12 HTML フォーム: レガシー プログラムへのゲートウェイ

最近まで、Web は、静的な媒体でした。サーバーに格納されたドキュメントは、Web ブラウザにダウンロードされ、表示されるだけでした。現在は、HTML フォームを使用して Web ユーザーが直接情報を取得することができるようになりました。フォームは、http://message にユーザーが入力した情報をパッケージ化することにより機能します。この情報は、サーバーで受信されたときに、サーバーの他のプログラムを起動することができます。これを可能にする鍵が、CGI (Common Gateway Interface) です。

A.13 Common Gateway Interface (CGI)

CGI (Common Gateway Interface) は、すべての Web サーバーでサポートされている標準的なインターフェイスであり、Web ブラウザと Web サーバー間での情報の受け渡しを可能にします。CGI の仕様を理解するサーバーのプログラムは、Web ブラウザを実行するユーザーからの情報を検索し、データベースやレガシー アプリケーションへの接続を取り込みます。その結果、ユーザーのニーズに合わせて個別に作成された動的な Web ページをビルドすることができます。

Common Gateway Interface (CGI) はクライアントとサーバーの概念を根本的に変えました。CGI ベースのゲートウェイ プログラムにより、Web はアプリケーションのための汎用ネットワークになりました。CGI ベースのプログラムでは、世界中で使用できるクライアントにより、ユーザーへのフォーム送信、入力受け付け、その入力を使用したデータベース検索またはトランザクションの開始、およびユーザーの要求に適したデータ画面のダウンロードが可能になります。

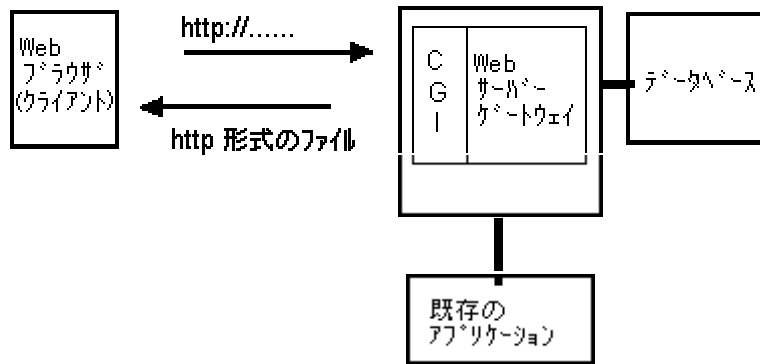


図 A-3 ゲートウェイとしての Web サーバー

A.14 NetExpress および CGI

NetExpress の最も重要な機能の 1 つとして、COBOL を使用して Web のプログラミングを行えるようにすることがあげられます。NetExpress が登場するまで、CGI プログラミングには、次の 2 つの作業が必要でした。1 つは、CGI インターフェイスの仕様について詳しく学習すること、もう 1 つは、Web ブラウザで受信された URL 文字列を解析するために新しい言語を学習することです。PERL は、文字列を解析するための暗号用言語ですが、現在は、受信する URL とデータ ストリームを解読するための一般的な言語になっています。

PERL は、Web サーバー プログラムをビルドするための一般的な言語ですが、大きな欠点があります。PERL のような言語は、既存の COBOL プログラムとうまく統合できないうえに、COBOL プログラムの技術やデータ操作言語としての COBOL の優位性を活用することができません。

NetExpress では、複雑な CGI の使用方法を簡便化し、慣れ親しんだ COBOL レコード構造を操作する場合と同じように Web プログラミングを簡易化してあるため、COBOL のデータ操作機能を最大限に活用できます。NetExpress では、External-Form という新しい句を導入し、COBOL 動詞である ACCEPT と DISPLAY の構文を拡張することによって、これを実現しています。COBOL の ACCEPT を理解していれば、HTML フォームのデータをあらかじめ定義されたデータレコードに ACCEPT することができます。同様に、慣れ親しんだ DISPLAY 文を使用すると、COBOL プログラムのデータ要素を直接クライアントの Web ページにマップできます。

これは、重要なことです。その理由は、次のとおりです。

- COBOL プログラムが Web アプリケーションを開発するための追加トレーニングは必要ありません。
- ACCEPT 文と DISPLAY 文を利用した既存のレガシー プログラムは、すぐに Web に対応していると見なされます。ACCEPT と DISPLAY を使用して端末からのプログラム入力と出力をビルドする対話型プログラムは、Web から同じ入出力を取得し、短時間で無制限のユーザー ベースまたは顧客ベースのプログラムに拡張されます。
- 管理者は、Perl スクリプトを作成するプログラマーに処理を託すリスクを負う必要はなくなります。たとえ

ば、http 文字列からストリート名を抽出し、印刷するための Perl 文は、次のとおりです。

```
if ($address =~ /^¥d+¥s+(¥w+)St¥.¥/) {  
  
    print "You live on $1.¥n";  
}
```

NetExpress を使用すると、Web クライアントからメインフレームにあるレガシー アプリケーションの全機能にアクセスすることができます。Web ブラウザと対話するゲートウェイ プログラムは、COBOL プログラムであることを覚えておいてください。この場合の COBOL プログラムは、プログラム自体が他のプログラム、データベース、またはトランザクション モニタと対話することができます。

A.15 詳細情報

World-Wide Web、インターネット、およびイントラネットの全容に関する書籍は、簡単なものから、たいへん詳細わたるものまで広範にあります。詳細については、近くの書店をたずねて、入手可能な情報を調べてください。主な分野は、Internet全般、HTML、CGI、ActiveX、および JavaScript の 5 分野です。CGI プログラミングに関する書籍は、CGI の概念を説明し、アプリケーションの設計方法をわかりやすく解説していることが必要です。HTML に関する書籍は、NetExpress で生成されるフォームを理解するのに役立ち、これらのフォームをインパクトのある表示形態に加工する方法について解説するものを選びます。ActiveX を使用しない限り、JavaScript に関して深く理解する必要はありません。

たとえば、Que 社により発行された Special Edition シリーズには、『*Using the Internet*』、『*Using HTML*』、『*Using CGI*』、『*Using ActiveX*』などのタイトルを持つ書籍があります。

JavaScript については、SAMS 社が発行した『*Teach Yourself JavaScript in a Week*』および『*JavaScript Unleashed*』を参照してください。

ActiveX については、上記の他に、SAMS 社が発行した『*Presenting ActiveX*』および『*Web Programming with ActiveX*』を参照してください。

インターネット、World-Wide Web、イントラネットに関しては、便利な情報を掲載しているインターネット サイトがあります。弊社の <http://www.microfocus.com/NetExpress/nxbooks/links.htm> にある NetExpress のページには、便利なサイトのリストを掲載しています。

付録B: 実装例とデバッグ例

この付録では、NetExpress で開発したアプリケーションを別の Web サーバーでデバッグし、実装するための作業例について説明します。

B.1 概要

この付録では、作業例を使用して、NetExpress で提供されるアプリケーション例を別の Web サーバーで実装し、デバッグする方法を説明します。これは、「アプリケーションの実装」の章の説明を補足するものです。

使用する Web サーバーがこの章で説明されていない場合でも、全般的な原則はすべてのサーバーに適用できるので、作業例を参照すると役に立つことがあります。

B.1.1 Microsoft Internet Server での ISAPI の実装

この項では、NetExpress で提供される Form Designer で作成したアプリケーション例を実装する方法について、作業例をもとに説明します。この例で説明する内容は、次のとおりです。

- 新しい Web 共有リソースに NetExpress のデフォルト名 COBOL と CGI-BIN 以外の名前を設定する方法
- 新しい Web 共有リソース名を使用するようにアプリケーションを修正する方法
- 共有ランタイム システムでアプリケーションを ISAPI としてリビルドする方法
- Web サーバーにアプリケーションをコピーする方法

アプリケーションの実装時には、より簡単な方法を使用することができます。たとえば、NetExpress のデフォルトの Web 共有リソース名を使用すると、アプリケーションで使用する共有リソース名を修正する必要はありません。さらに、ISAPI や共有ランタイム システムを使用しない場合は、アプリケーションをリビルドする必要はありません。

説明するアプリケーション例では、ActiveX レイアウト フォームを使用します。HTML フォームを使用した同様のアプリケーションを使用する場合は、この手順を参考にすることができます。

1. ディレクトリ netexpress¥base¥demo¥bevord_x ではなく、ディレクトリ netexpress¥base¥demo¥bevord_h のファイル例を使用します。
2. ActiveX や .alx ファイルに関する手順は、すべて省略します。

この例では、NetExpress で提供される ActiveX バージョンのアプリケーション例を Microsoft Internet Server で ISAPI プログラムとして実装する方法について説明します。

B.1.1.1 Internet Server での Web 共有リソースの設定

この項では、アプリケーションのすべての部分に別々の Web 共有リソースを作成します。

1. Windows のエクスプローラを使用し、netxapps という新しいフォルダを作成します。
2. netxapps に次のような 4 つのサブフォルダを作成します。
 - nx-bin
 - nx-html
 - nx-alx
 - nx-images
3. Internet Service Manager を起動します。
4. 「Microsoft Internet Service Manager」ウィンドウで [WWW] プロセスをダブルクリックします。
5. 「WWW サービスのプロパティ」ダイアログ ボックスの [ディレクトリ] タブをクリックします。
6. バイナリ ファイルを格納するために新しい Web 共有リソース nx-bin を作成します。
 - a. [追加] ボタンをクリックします。
 - b. 「ディレクトリ」フィールドに `x:\netxapps\nx-bin` を入力します。x は、手順 1 でフォルダを作成したドライブです。
 - c. 「エイリアス」フィールドに nx-bin を入力します。
 - d. 「アクセス」プロパティを「実行」のみに設定します（「読み取り」チェックボックスのチェックを解除し、「実行」チェックボックスだけをチェックします）。
 - e. [OK] ボタンをクリックします。
7. さらに 3 つの共有リソースを作成し、すべてに読み取りアクセス権だけを設定します。
 - エイリアス nx-html、ディレクトリ `x:\netxapps\nx-html`
 - nx-alx、ディレクトリ `x:\netxapps\nx-alx`

B.1.1.2 実装可能なバージョンのアプリケーション作成

まず、アプリケーション ソースを開発マシンの実装用ディレクトリにコピーします。

1. 入力フォームの Action プロパティを変更します。

- a. 「プロジェクト」ウィンドウで左側のペインにある bevard_x.htm を右クリックし、コンテキストメニューの [編集] をクリックします。
- b. ページにあるフォームを選択し、Action プロパティをクリックして、次のように修正します。

```
/nx-bin/bev_x.dll
```

2. bevard_x.htm を保存し、Form Designer を終了します。

3. .htm ファイル内の ALX パスを変更します。

- a. 「プロジェクト」ウィンドウで左側のペインにある bevard_x.htm を右クリックしてから、コンテキストメニューの [テキスト編集] をクリックします。
- b. ファイルの末尾に向かってテキスト <PARAM NAME を検索します。
- c. コードを変更します。

```
VALUE="/COBOL/bevard_x.alx"
```

上記のコードを次のように変更します。

```
VALUE="/nx-alx/bevard_x.alx"
```

- d. 変更を保存し、ファイルを閉じます。
- e. bevsum_x.htm についても上記の手順を繰り返し、パスを次のように変更します。

```
VALUE="/nx-alx/bevard_x.alx"
```

アプリケーションを ISAPI としてリビルドします。

1. 実行可能ファイルを .exe から .dll に変更します。

- a. 「プロジェクト」ウィンドウで左側のペインにある bev_x.exe を右クリックし、コンテキストメニューの [ビルドタイプから削除] をクリックします。
- b. 「ビルド構造から削除」ダイアログボックスで、「ビルド構造の全体を削除します。」チェックボックスのチェックを解除し、「このターゲットをすべてのビルドタイプから削除します」を選択してから、[削除] ボタンをクリックします。
- c. 「プロジェクト」ウィンドウで左側のペインにある bev_x.obj を右クリックし、コンテキストメニューで [選択されたファイルをパッケージ化]、[ダイナミックリンクライブラリ] の順にクリックします。
- d. 「新規に定義ダイナミックリンクライブラリ」ダイアログボックスの [作成] ボタンをクリックします。

2. プログラムのビルド設定を変更する方法は、次のとおりです。
 - a. 「プロジェクト」ウィンドウで左側のペインにある bev_x.dll を右クリックし、コンテキストメニューの [ビルド設定] をクリックします。
 - b. [リンク] タブをクリックし、[共有] と [マルチスレッド] オプション ボタンを選択し、「動的」チェックボックスがチェックされていることを確認します。
 - c. [閉じる] ボタンをクリックします。
3. このプログラムを ISAPI 用にコンパイルするために SET 文を追加します。
 - a. 「プロジェクト」ウィンドウで左側のペインにある bev_x.cbl を右クリックし、コンテキストメニューの [編集] をクリックします。
 - b. プログラムの先頭で \$SET PREPROCESS 指令の後に次の文を追加します (プリプロセッサ指令は、プログラムの最初の SET 文に記述する必要があります)。

```
$set webserver(isapi) case reentrant(2)
```

ドル記号 (\$) は、7 列目に記述する必要があります。
 - c. 変更を保存し、ファイルを閉じます。
4. プログラムをリビルドします。

B.1.1.3 アプリケーションの実装

この段階では、Web サーバーにアプリケーションを実装することができます。

1. .dll ファイルを x:\netxapps\nx-bin にコピーします。
2. bevord_x.htm と bevsum_x.htm を x:\netxapps\nx-html にコピーします。
3. bevord_x.alx と bevsum_x.alx を x:\netxapps\nx-alx にコピーします。

アプリケーションを実行します。

- Web ブラウザを起動し、URL を次のように設定します。

```
http://machine.domain/nx-html/bevord_x.htm
```

B.2 別の Web サーバーでのデバッグ

次の 3 項では、NetExpress に組み込まれたプログラム例を別の Web サーバーでアニメートする方法について、作業例をもとに説明します。この例で説明する内容は、次のとおりです。

- Microsoft Internet Server で CGI プログラムをアニメートする方法
- Microsoft Internet Server で ISAPI プログラムをアニメートする方法
- Netscape FastTrack で NSAPI プログラムをアニメートする方法

これらのプログラム例は、すべて、Windows NT サーバーをオペレーティング システムとして使用して作成されています。

B.2.1 Microsoft Internet Server での CGI プログラムのアニメート

次の説明では、NetExpress に組み込まれた飲料に関する HTML アプリケーションを、Windows NT サーバーで実行中の Microsoft Internet Server を使用してアニメートする方法を示します。

Internet Server の再構成方法は、次のとおりです。

1. Microsoft Internet Service Manager を起動します。
2. [WWW] サービスを選択し、[プロパティ] メニューで [サービスのプロパティ] をクリックして、「WWW サービスのプロパティ」ダイアログ ボックスを表示します。
3. [サービス] タブで、「匿名を認める」チェックボックスの選択を解除します。

注記: この操作により、ユーザーが他のマシンからこの Web サーバーにアクセスする場合は、パスワードとユーザー名を送信する必要があります。匿名のログオンを禁止して、Web サーバーに対してアニメータを実行するための十分なアクセス権を設定する必要があります。

オペレーティング システムと Web サーバーの設定方法によっては、「NT チャレンジ/レスポンス」または「基本 (テキストをクリア)」チェックボックスを選択する必要がある場合があります。Web ブラウザとして Microsoft インターネット エクスプローラを使用する場合は、「NT チャレンジ/レスポンス」を選択します。Web ブラウザとして Netscape Navigator を使用する場合は、「基本 (テキストをクリア)」を使用します。

上記の手順を行った後に CGI を実行してもアニメートされない場合は、次の説明にしたがいます。

- 「WWW サービスのプロパティ」ダイアログ ボックスの [ディレクトリ] タブをクリックします。
- 既存の共有リソース COBOL と CGI-BIN がある場合は、削除します (共有リソースをクリックし、[削除] ボタンをクリックします)。
- アプリケーションのソース ディレクトリを示す COBOL の共有リソースを追加します。
 - a. [追加] ボタンをクリックして、「ディレクトリのプロパティ」ダイアログ ボックスを表示します。

- b. 「ディレクトリ」フィールドに、次のように入力します。

x:¥netexpress¥base¥demo¥bevord_h

x: は、NetExpress をインストールしたドライブです。

- c. 「エイリアス」フィールドに、次のように入力します。

COBOL

- d. 「アクセス」権を「読み取り」に設定します。

- e. [OK] をクリックします。

- アプリケーションのソース ディレクトリを示す CGI-BIN の共有リソースを追加します。

- a. [追加] ボタンをクリックし、「ディレクトリのプロパティ」ダイアログ ボックスを表示します。

- b. 「ディレクトリ」フィールドで、次のように入力します。

x:¥netexpress¥base¥demo¥bevord_h¥debug

x: は、NetExpress をインストールしたドライブです。このプロジェクトを初めてビルドする場合、このディレクトリが存在しないため、これらの変更を適用するときにサーバー エラーが生じます。この場合、まず、Windows のエクスプローラを使用して、上記の場所に debug フォルダを作成します。その後で、下部にある [OK] ボタンをクリックし、これらの変更を適用します。

- c. 「エイリアス」フィールドに、次のように入力します。

CGI-BIN

- d. 「アクセス」権を「実行」に設定します。

- e. [OK] ボタンをクリックします。

- [OK] をクリックして、Web サーバーに変更を適用します。

アプリケーションをリビルドし、アニメートします。

1. NetExpress を起動し、bevord_h.app デモを読み込みます。

2. NetExpress で Solo を使用しないように再構成します。

- a. [オプション] メニューで [実行] をクリックします。

- b. [CGI プロジェクトに Solo web サーバーを使用する] チェックボックスを選択解除します。

- c. [OK] ボタンをクリックします。

3. アプリケーションを起動する URL を、Solo で使用した URL から、使用する Web サーバーに必要な URL に変更します。

- a. [プロジェクト] メニューの [プロパティ] をクリックします。
- b. 「アニメーションの起動」フィールドに、次のように入力します。

`http://servername/COBOL/bevord_h.htm`

`servername` は、この Web サーバーにアクセスするために使用する名前です。

- c. [OK] ボタンをクリックします。
4. [プロジェクト] メニューの [リビルド] をクリックして、アプリケーションをビルドします。
 5. [アニメート] メニューの [アニメート開始] をクリックして、アプリケーションを実行します。
 6. Web ブラウザに表示された [注ぐ] ボタンをクリックして、CGI を実行します。この操作によりアニメータが起動します。

B.2.2 Microsoft Internet Server での ISAPI プログラムのアニメート

次の説明では、NetExpress で提供される飲料に関する HTML アプリケーションの ISAPI バージョンを、Windows NT で実行中の Microsoft Internet Server V3.0 を使用してアニメートする方法を示します。この手順は、次の各段階に分けられます。

- Web サーバーを構成する。
- プログラムを ISAPI の .dll としてリビルドする。
- プログラムをアニメートする。

Web サーバーの構成方法は、次のとおりです。

1. Microsoft Internet Service Manager を起動します。
2. [WWW] サービスを選択し、[プロパティ] メニューの [サービスのプロパティ] をクリックし、「WWW サービスのプロパティ」ダイアログ ボックスを表示します。
3. [サービス] タブで、「匿名を認める」チェックボックスの選択を解除します。

この操作により、匿名のログオンができなくなります。別のマシンからこの Web サーバーにアクセスしようとするユーザーは、ユーザー名とパスワードを送信する必要があります。この設定は必ず必要です。この設定を行わないと、Web サーバーは NetExpress デバッガを起動することができません。

オペレーティング システムと Web サーバーの設定方法によっては、「NT チャレンジ/レスポンス」また

は「基本 (テキストをクリア)」チェックボックスを選択する必要がある場合があります。 Web ブラウザとして Microsoft インターネット エクスプローラを使用する場合には、「NT チャレンジ/レスポンス」を選択します。 Web ブラウザとして Netscape Navigator を使用する場合は、「基本」を使用します。

上記の手順を行った後に CGI を実行してもアニメートされない場合は、次の説明にしたがいます。

4. 「WWW サービスのプロパティ」ダイアログ ボックスの [ディレクトリ] タブをクリックします。
5. 既存の共有リソース COBOL と CGI-BIN がある場合は、削除します (共有リソースをクリックし、[削除] ボタンをクリックします)。

6. NetExpress のソース ディレクトリを COBOL という名前の Web 共有リソースとして設定します。

- a. [追加] ボタンをクリックし、「ディレクトリのプロパティ」ダイアログ ボックスを表示します。

- b. 「ディレクトリ」フィールドに、次のように入力します。

x:¥netexpress¥base¥demo¥bevord_h

x: は、NetExpress をインストールしたドライブです。

- c. 「エイリアス」フィールドに、次のように入力します。

COBOL

- d. 「アクセス」権を「読み取り」に設定します。

- e. [適用] ボタンをクリックします。

7. NetExpress の実行可能プログラムのディレクトリを CGI-BIN という名前の Web 共有リソースとして設定します。

- a. このプロジェクトを初めてビルドする場合、実行可能プログラムのディレクトリは存在しません。 Windows のエクスプローラを使用して、フォルダ x:¥netexpress¥base¥demo¥bevord_h¥ (x: は、NetExpress をインストールしたドライブ) を参照します。このディレクトリに debug という名前のサブフォルダが含まれない場合、Windows のエクスプローラを使用して debug という名前の新しいフォルダを作成します。

- b. [追加] ボタンをクリックし、「ディレクトリのプロパティ」ダイアログ ボックスを表示します。

- c. 「ディレクトリ」フィールドに、次のように入力します。

x:¥netexpress¥base¥demo¥bevord_h¥debug

- d. 「エイリアス」フィールドに、次のように入力します。

COBOL

- e. 「アクセス」権を「読み取り」に設定します。
- f. [適用] ボタンをクリックします。

プログラムを ISAPI の .dll としてリビルドする方法は、次のとおりです。

1. プロジェクト bevord_h.app を NetExpress の IDE に読み込みます。
2. プロジェクトのビルド情報を変更して、bev_h.exe のかわりに bev_h.dll を作成します。
 - a. 「プロジェクト」ウィンドウで左側のペインにある bev_h.exe をクリックし、コンテキスト メニューの [ビルド タイプから削除] をクリックします。
 - b. 「ビルド構造から削除」ダイアログ ボックスで、「ビルド構造の全体を削除します。」のチェックを解除し、「このターゲットをすべてのビルド タイプから削除します。」を選択して、[削除] ボタンをクリックします。
 - c. 「プロジェクト」ウィンドウで左側のペインにある bev_h.obj を右クリックし、コンテキスト メニューで [選択されたファイルをパッケージ化]、[ダイナミック リンク ライブラリ] の順にクリックします。
 - d. 「新規に定義 ダイナミック リンク ライブラリ」ダイアログ ボックスの [作成] ボタンをクリックし、プロジェクト リストに bev_h.dll を追加します。
3. .dll ファイルの [ビルド設定] を変更して、共有マルチスレッド ランタイム システムを使用するようにします。
 - a. NetExpress の「プロジェクト」ウィンドウで左側のペインにある bev_h.dll ファイルを右クリックし、コンテキスト メニューで [ビルド設定] をクリックします。
 - b. [リンク] タブをクリックします。
 - c. [共有] と [マルチスレッド] オプション ボタンをクリックして、[動的] チェックボックスがチェックされていることを確認します。
 - d. [閉じる] ボタンをクリックします。
4. このプログラムを ISAPI プログラムとしてビルドするためのコンパイラ指令を追加します。bev_h.cbl の先頭で、すでに存在する \$SET 文の後に次の文を追加します。

```
$set webserver(isapi) case reentrant(2)
```

ドル記号 (\$) は、ソース コードの 7 列目に記述する必要があります。

5. プログラムの実行時にデバッガを起動する文を追加します。

- a. 「プロジェクト」ウィンドウで bev_h.cbl をダブルクリックし、テキスト編集ペインに読み込みます。
- b. プログラムの手続き部を検索します。
- c. 次の文を手続き部の先頭に追加します。

```
call "CBL_DEBUGBREAK"
```

6. [プロジェクト] メニューの [リビルド] をクリックし、ISAPI プログラムをビルドします。
7. 入力フォームの Action プロパティを修正して、bev_h.exe のかわりに bev_h.dll を呼び出すようにします。
 - a. bevord_h.htm をダブルクリックして Form Designer を起動し、ページを読み込みます。
 - b. フォームを選択し、Action プロパティをクリックして、次のように修正します。

```
CGI-BIN/bev_h.dll
```
 - c. ページを保存し、Form Designer を終了します。
8. NetExpress IDE を終了します。

この段階で、プログラムをアニメートすることができます。

1. Web ブラウザを起動します。
2. アプリケーションの最初のページの URL を入力します。

```
http://machinename/cobol/bevord_h.htm
```

machinename は、Web サーバーが使用するマシンの名前です。

3. フォームの [注ぐ] ボタンをクリックします。

アプリケーションをデバッグするかどうかをたずねるダイアログ ボックスが表示されます。

4. [はい] ボタンをクリックします。

デバッグに必要な .idy ファイルを検索する場所をデバッガが認識しないため、[IDY ファイル エラー] ダイアログ ボックスが表示されます。

5. [参照] ボタンをクリックし、「開く」ダイアログ ボックスを使用して、プログラムの bev_h.idy ファイルを検索します。NetExpress では、bev_h.idy ファイルはアプリケーションの実行可能ファイルと同じディレクトリに作成されます。

これで、プログラムをデバッグすることができるようになります。

デバッガを停止すると、Web サーバーも停止するため、再起動する必要があります。EXIT PROGRAM 文をステップ実行すると、NetExpress IDE が開いた状態が続きますが、デバッガは、プログラムを再実行するまで処理を停止します。サーバーの実行中は、サーバーにより .dll ファイルが開いたままになるため、リビルドするには、Web サーバーをシャット ダウンする必要があります。

B.2.3 NSAPI プログラムのアニメート

次の説明では、NetExpress で提供される飲料に関する HTML アプリケーションの NSAPI バージョンを、Windows NT で実行中の Netscape FastTrack を使用してアニメートする方法を示します。この手順は、次の各段階に分けられます。

- Web サーバーを構成する。
- プログラムを NSAPI の .dll としてリビルドする。
- プログラムをアニメートする。

サーバーの構成方法は、次のとおりです。

1. Windows の [スタート] メニューで、[Netscape]、[Netscape サーバー管理] の順にクリックします。
2. Web サーバーに COBOL 共有リソースを追加します。
 - a. 最上部にあるフレームの [コンテンツ管理] をクリックします。
 - b. 左側のフレームの [追加ドキュメント ディレクトリ] をクリックします。

この操作により、「追加ドキュメント ディレクトリ」フォームが表示されます。
 - c. 「URL」フィールドに [COBOL] と入力します。
 - d. 「エイリアス」フィールドにプロジェクトのソース コード ディレクトリへの完全パスを入力します。
 - e. [OK] ボタンをクリックします。

この操作により、「変更の保存と適用」ページが表示されます。
 - f. 「変更の保存と適用」ページの [変更の保存と適用] ボタンをクリックします。

注記: NSAPI アプリケーションについては、検索する NSAPI プログラムの CGI-BIN 共有リソースを追加する必要はありません。NSAPI プログラムは、Web サーバーの起動時に、サーバー構成ファイルに設定された明示的なパス

を使用して読み込まれます。

-
- サーバーの構成ファイルを修正して、NSAPI プログラムを読み込み、新しい MIME タイプを定義します。

- a. Netscape のサーバー ファイル obj.conf を検索し、テキスト エディタに読み込みます。

このファイルは、`¥netscape¥server¥httpd-name¥config` に格納されています。*name* は、マシンのサーバー名です。

- b. 次の行を追加して、サーバーの起動時にサーバー側プログラムが読み込まれるようにします。

```
Init fn="load-modules"  
shlib="x:/netexpress/base/demo/debug/bev_h.dll"  
  
        funcs="beverage"
```

x は、NetExpress をインストールしたドライブです。

- c. `<OBJECT name="default">` タグと `<OBJECT>` タグの間に、次の行を追加して、プログラムのエントリ ポイントを `bev` という新しい MIME タイプに関連付けます。

```
Service fn="beverage" method="(GET|POST)" type="magnus-  
internal/bev"
```

- `mime.types` (obj.conf と同じディレクトリに格納されています) を編集して、定義したばかりの新しい MIME タイプを拡張子と関連付けます。次の行を `mime.types` に追加します。`magnus-internal/bev` の拡張子は `cobolcgi` として定義します。

```
type=magnus-internal/bev exts=cobolcgi
```

ブラウザが `beverage.cobolcgi` を要求すると、`bev_h.dll` が実行されます。

- Netscape サーバー管理ブラウザの最上部にあるフレームで [適用] ボタンをクリックします。

この操作により、「変更の適用」フィールドが表示されます。

- [構成ファイルの読み込み] ボタンをクリックします。

プログラムを NSAPI プログラムとしてリビルドする方法は、次のとおりです。

1. プロジェクト `bevord_h.app` を NetExpress IDE に読み込みます。
2. プロジェクトのビルド情報を変更して、`bev_h.exe` のかわりに `bev_h.dll` を作成します。
 - a. 「プロジェクト」ウィンドウで左側のペインにある `bev_h.exe` をクリックし、コンテキスト メニ

ューの [ビルド タイプから削除] をクリックします。

- b. 「ビルド構造から削除」ダイアログ ボックスで、「ビルド構造の全体を削除します。」のチェックを解除し、「このターゲットをすべてのビルド タイプから削除します。」を選択して、[削除] ボタンをクリックします。
 - c. 「プロジェクト」ウィンドウの左側のペインで bev_h.obj を右クリックし、コンテキスト メニューで [選択されたファイルをパッケージ化]、[ダイナミック リンク ライブラリ] の順にクリックします。
 - d. 「新規に定義 ダイナミック リンク ライブラリ」ダイアログ ボックスの [作成] ボタンをクリックして、プロジェクト リストに bev_h.dll を追加します。
3. プログラムの [ビルド設定] を変更して、共有マルチスレッド ランタイム システムを使用し、NSAPI に必要なモジュールやライブラリにリンクできるようにします。
- a. NetExpress の「プロジェクト」ウィンドウで左側のペインにある bev_h.dll ファイルを右クリックし、コンテキスト メニューで [ビルド設定] をクリックします。
 - b. [リンク] タブをクリックします。
 - c. [共有] と [マルチスレッド] オプション ボタンをクリックして、「動的」チェックボックスがチェックされていることを確認します。
 - d. [カテゴリ] ドロップダウンから [高度な設定] を選択します。
 - e. 「これらの OBJ とリンクする」フィールドに accnsapi.obj と入力します。

このファイルには、3 種類のバージョンがあります。Netscape サーバーの種類に対応したバージョンを使用する必要があります。

- FastTrack

デフォルト バージョンの accnsapi.obj は、FastTrack 用に設定されているため、変更する必要はありません。

- Commerce Server

デフォルト バージョンの accnsapi.obj (¥netexpress¥base¥lib ディレクトリに格納されています) の名前を変更して、バックアップを保存し、accnscs.obj の名前を accnsapi.obj に変更します。

- Enterprise Server

デフォルト バージョンの accnsapi.obj (¥netexpress¥base¥lib ディレクトリに格納されてい

ます) の名前を変更して、バックアップを保存し、accenter.obj の名前を accnsapi.obj に変更します。

- f. 「これらの LIB とリンクする」フィールドに libhttpd.lib と入力します (Commerce Server では、httpd.lib を使用します)。
4. NetExpress のコンパイラで検索できる場所に libhttpd.lib をコピーします。

このファイルは、Netscape サーバー ソフトウェアがインストールされているディレクトリのサブディレクトリ %server%\nsapi\examples に格納されています。このサブディレクトリから libhttpd.lib を x:\netexpress\base\bin にコピーします (x: は、NetExpress をインストールしたドライブです)。

5. このプログラムを NSAPI プログラムとしてビルドするためのコンパイラ指令を追加します。bev_h.cbl の先頭で、すでに存在する \$SET 文の後に次の文を追加します。

```
$set webserver(nsapi,beverage) case reentrant(2)
```

ドル記号 (\$) は、ソース コードの 7 列目に記述する必要があります。

6. プログラムの実行時にデバグガを起動する文を追加します。
 - a. 「プロジェクト」ウィンドウで bev_h.cbl をダブルクリックし、テキスト編集ペインに読み込みます。
 - b. プログラムの手続き部を検索します。
 - c. 次の文を手続き部の先頭に追加します。

```
call "CBL_DEBUGBREAK"
```

7. [プロジェクト] メニューの [リビルド] をクリックして、ISAPI プログラムをビルドします。
8. 入力フォームの Action プロパティを修正し、bev_h.exe のかわりに bev_h.dll を呼び出すようにします (サーバーの設定時に新しい MIME の拡張子として .dll を定義していることを前提とします)。

- a. bevord_h.htm をダブルクリックし、Form Designer を起動して、ページを読み込みます。
- b. フォームを選択し、Action プロパティをクリックして、次のように修正します。

```
beverage.cobolcgi
```

- c. フォームを保存し、Form Designer を終了します。
9. NetExpress IDE を終了します。
 10. Netscape サーバーにプログラムを読み込みます。そのためには、サーバーをシャット ダウンし、Netscape サ

サーバー管理から再起動します。

この段階で、プログラムをアニメートすることができます。

1. Web ブラウザを起動します。
2. アプリケーションの最初のページの URL を入力します。

`http://machinename/COBOL/bevord_h.htm`

machinename は、Web サーバーが使用するマシンの名前です。

3. フォームの [注ぐ] ボタンをクリックします。

アプリケーションをデバッグするかどうかをたずねるダイアログ ボックスが表示されます。

4. [はい] ボタンをクリックします。

デバッグに必要な .idy ファイルを検索する場所をデバッガが認識できないため、「IDY ファイル エラー」ダイアログ ボックスが表示されます。

5. [参照] ボタンをクリックし、「開く」ダイアログ ボックスで、プログラムの bev_h.idy ファイルを検索します。NetExpress では、bev_h.idy ファイルはアプリケーションの実行可能ファイルと同じディレクトリに作成されます。

これで、プログラムをデバッグすることができるようになります。

デバッガを停止すると、Web サーバーも停止するため、再起動する必要があります。EXIT PROGRAM 文をステップ実行すると、NetExpress IDE が開いた状態が続きますが、デバッガは、プログラムを再実行するまで処理を停止します。サーバーの実行中は、サーバーにより .dll ファイルが開いたままになるため、リビルドするには Web サーバーをシャット ダウンする必要があります。

索引

.aht ファイル.....	6-6	プリプロセッサ	7-4, 7-7
ACCEPT	7-2	END-EXEC	7-4
Access	12-14	ESQL Assistant.....	5-1
Action プロパティ、変更.....	12-10, 12-20	EXEC HTML	7-4
ActiveX コントロール.....	3-2, 10-2	EXTERNAL INPUT-FORM.....	7-2
イベント ハンドラ.....	10-10	Form Designer	2-1, 3-1, 4-1
ActiveX レイアウト フォーム	3-1	出力オプション	3-3
Application Server	12-7	出力の編集	9-1
CASE 指令	8-2, 8-3	HTML.....	9-1, 10-2
cbl ファイル.....	4-16, 5-4, 6-7	htmlpp.....	7-7
CGI.....	1-1, 7-1, 8-1	HTML コントロール.....	10-2
ビルド.....	4-11	イベント ハンドラ	10-9
Common Gateway Interface.....	8-1	HTML フィールドまたはラベル	6-5
cookie.....	7-9, 7-11	HTML フォーム.....	1-2, 2-1, 3-1, 6-1
cookie の例.....	7-12	htm ファイル.....	5-4, 6-8, 9-1
cpf ファイル.....	4-16, 5-4, 6-7	IDENTIFIED BY	7-2
cpv ファイル.....	4-16, 5-4, 6-7	idy ファイル.....	6-1
cpy ファイル.....	4-16, 5-4, 6-7	INPUT-FORM.....	7-2
DISPLAY.....	7-7	Internet Server	B-1
代入マーカー	7-7	Internet Server API.....	8-1
EHTML.....	7-4	ISAPI.....	1-1, 7-1, 8-1
指令.....	7-7	共有ランタイム システム	12-14
代入マーカー	7-5	コンパイラ指令	8-2

実装例.....	B-1	変換.....	12-1
ファイルの検索.....	12-23	obj.conf ファイル.....	12-19
プログラムの開発.....	8-2	ODBC.....	12-25
変換.....	12-1	REENTRANT 指令.....	8-3
JavaScript.....	10-1, 10-2, 11-4	SCRIPT タグ.....	10-2
関数の書式.....	10-9	Solo.....	2-2, 4-12
Java アプレット.....	3-2, 10-2, 10-12	SQL クエリー.....	5-1
js ファイル.....	10-2, 11-4	sstate.....	7-13, 12-11
JScript.....	10-1, 11-4	UNIX.....	1-1
Macintosh.....	1-1	URL.....	12-10
mff ファイル.....	5-4, 6-7, 9-1, 10-13, 10-15, 10-16	validate.cpt ファイル.....	11-4
Microsoft Access.....	12-14	Web サーバー.....	2-2
mime.types ファイル.....	12-19	設定.....	12-7
MIME タイプ.....	12-17	WEBSERVER 指令.....	8-2, 8-3
Netscape.....	10-1	Web 共有リソース.....	12-10
Netscape Server API.....	8-1	Web 生成.....	6-7
NOAUTOFORMAT.....	7-5	Web ブラウザ.....	1-1, 1-2
NSAPI.....	1-1, 7-1, 8-1	Windows NT.....	1-1
MIME タイプ.....	12-20	WWW.....	1-1
エントリ ポイント.....	8-3, 12-18	値.....	7-2
共有ランタイム システム.....	12-14	アニメート.....	4-12
コンパイラ指令.....	8-3	アプリケーション	
サーバー構成.....	12-17	作成.....	4-1
ファイルの検索.....	12-23	状態.....	7-9
プログラムの開発.....	8-2	生成ファイル.....	4-16

アプレット.....	10-2, 10-12	イントラネット	1-1
イベント.....	10-7	埋め込み SQL.....	5-4
イベント ハンドラ	10-1, 10-2, 10-9	エントリ フィールド	6-6
ActiveX コントロール	10-10	エントリ ポイント	
HTML コントロール.....	10-9	NSAPI	12-18
確認.....	11-4	エントリ ポイント、重複	12-15
クロスプラットフォーム互換性	10-12	オブジェクト ビュー	10-7
イベント ハンドラ ビュー	10-9	オプション ボタン	3-1, 6-6
イベント ビュー.....	10-7	確認.....	6-6, 10-1, 10-11, 11-1
イメージ パス.....	12-11	イベント ハンドラ	11-4
インターネット.....	1-1	組み込みルーチン	11-1
インターネット アプリケーション例		追加.....	11-4
機能の追加.....	4-17	プロンプト	11-1
実行.....	4-13	呼び出し	11-2
非対称アプリケーション	4-14	例.....	11-2
フォームのペイント.....	4-7	カラー、設定	10-8
インターネット アプリケーション	1-2	競合.....	7-1
作成.....	2-2	共有ランタイム システム	12-13
実行.....	4-11	共有リソース	7-1
インターネット アプリケーション ウィザード2-2, 3-3		クライアント サーバー	1-1
新規アプリケーション	4-1	クロスプラットフォーム	
データ アクセス アプリケーション	5-1	HTML.....	3-3
レガシー コード.....	6-1	アプリケーション	1-1
インターネット エクスプローラ	2-1, 10-1	構文	
		HTML への出力データのマップ	7-5, 7-7

コントロールへの入力データのマップ	7-3	インターネット アプリケーション	2-2
個別入出力フォーム	6-2	サーバー側プログラム	4-1
コントロール		非対称サーバー側プログラム	4-13
値	7-2	参照の修正	
型	6-5	代入マーカー	7-6
スタイル	10-8	辞書ファイル	6-1
名前	7-2	実行フロー	1-5
プロパティ	10-8	実装	12-1
メソッド	10-8	ISAPI	B-1
コンパイラ指令		Web サーバー	12-7
ISAPI プログラム	8-2	共有ランタイム システム	12-13
NSAPI プログラム	8-3	プロジェクトのコピー	12-9
サーバー側の状態機構	7-13, 12-11	まとめ	12-2
サーバー側プログラム	1-3, 5-1, 7-1	リビルド	12-12
作成	4-1	実装のためのリビルド	12-12
出力	7-3, 7-7	修飾	
対称	6-2	代入マーカー	7-6
対象	1-3	出力	
入力	7-2	サーバー側プログラム	7-3
非対称	6-2	サーバー側プログラムの構文	7-5, 7-7
非対象	1-3	状態	7-9
編集	4-16	処理シーケンス	1-5
再使用	2-2	指令	
作成		EHTML プリプロセッサ	7-7
アプリケーション	4-1	ISAPI プログラム	8-2

NSAPI プログラム	8-3
コンパイラ	8-2, 8-3
スクリプト アシスタント	10-1, 10-2
[スクリプト] タブ	10-10
スタイル	10-2, 10-8
[スタイル] タブ	10-5
スタイル ビュー	10-8
生成 CGI	
編集	4-16
生成ファイル	4-16, 5-4, 6-7
セキュリティ	7-10
選択ボックス	3-1
対称サーバー側プログラム	1-3, 6-2
ダイナミック HTML	3-3, 9-1, 10-2
代入マーカー	
DISPLAY	7-7
EHTML	7-5
単一入出力フォーム	6-2
単一レコード ビュー	5-1
チェック ボックス	3-1, 6-6
重複エントリ ポイント	12-15
データ	
確認	11-1
フォームのコントロールへの関連付け	7-2
データ アクセス アプリケーション	5-1

作成	5-2
実行	5-5, 5-12
修正	5-7
生成	5-2, 5-6
データ アクセス アプリケーション例	
実行	5-5, 5-12
修正	5-7
生成	5-2, 5-6
データ型	6-4
データ項目、リンク	4-4
データの選択	6-3
データベース アプリケーション	5-1
データ方向	6-5
テーブル	6-7
デバッグ	2-2, 4-12, 12-26
CGI	12-26
Web サーバーの設定	12-5, 12-8
名前	7-2
入力	
サーバー側プログラム	7-2
サーバー側プログラムの構文	7-3
入力フィールド	3-1
配置編集	4-4
パスワード フィールド	6-6
非対称サーバー側プログラム	1-3, 4-13, 6-2

例	4-14
非表示フィールド	6-6, 7-9, 7-10
ビルド	
CGI プログラム	4-11
ISAPI	8-4
NSAPI	8-4
サーバー側プログラム	4-11
ファイル	
.cbl	4-16, 5-4, 6-7
.cpf	4-16, 5-4, 6-7
.cpv	4-16, 5-4, 6-7
.cpy	4-16, 5-4, 6-7
.htm	5-4, 6-8
.idy	6-1
.js	10-2, 11-4
.mff	5-4, 6-7, 9-1, 10-13, 10-15, 10-16
.htm	9-1
ファイル、生成	4-16, 5-4
フィールド名	6-4
フォーム	3-1
Action プロパティ	12-10
位置指定	3-3
イベント	10-1
概要	1-2
確認	10-1, 11-1

スクリプト作成	10-1
ペイント	4-5
フォームのペイント	4-5
例	4-7
プッシュボタン	3-2
プリプロセッサ	
EHTML	7-4
プリプロセッサ、EHTML	
指令	7-7
フロー モード編集	4-4
プロジェクトのコピー	12-9
プロトコル	1-1
プロパティ	10-2, 10-8
[プロパティ] タブ	10-5
プロパティ ビュー	10-8
プロンプト、確認	11-1
ページプロパティ	3-3
編集	
HTML	9-1
サーバー側プログラム	4-16
生成 CGI	4-16
生成アプリケーション	6-13
フォーム	6-14, 9-1
マルチスレッド	12-6
無効化された編集	6-6

メソッド.....	10-2, 10-7	HTML への出力データのマップ	7-5
[メソッド] タブ.....	10-4	入力コントロールへのデータのマップ	7-3
メソッド ビュー.....	10-7	レガシー インターネット アプリケーション例.....	6-8
ユニット テスト.....	2-3	拡張.....	6-14
ラベル.....	6-6	実行.....	6-11
リスト ビュー.....	5-1	レガシー コード	
リソースの競合.....	7-1	拡張.....	6-13
リンク		再使用.....	2-2, 6-1
ISAPI.....	8-4	データの選択.....	6-3
NSAPI.....	8-4	ファイルの選択.....	6-3
リンク データ項目.....	4-4	レベル番号.....	6-4
例		割り当てファイル.....	6-5