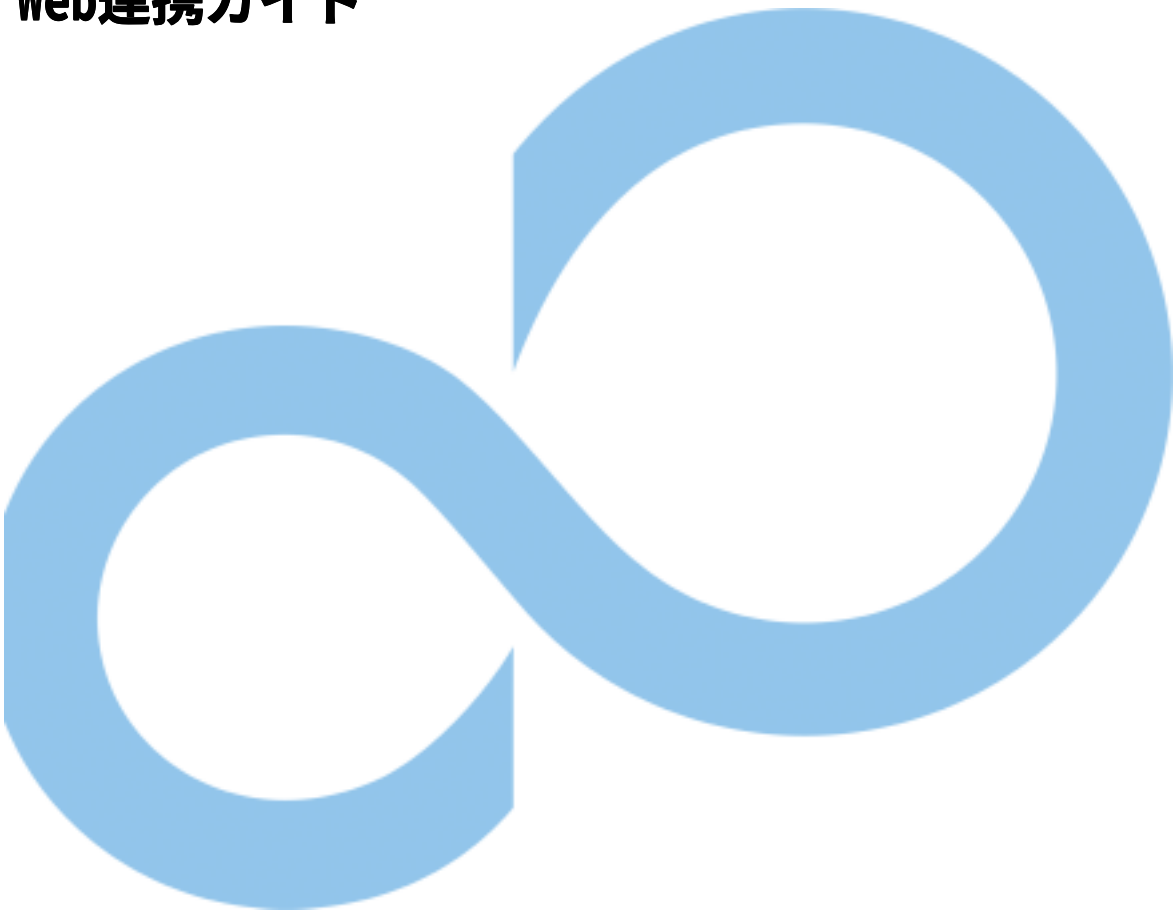
Microsoft® Windows® 95
Microsoft® Windows® 98
Microsoft® Windows® Me
Microsoft® Windows NT®
Microsoft® Windows® 2000
Microsoft® Windows® XP

B1JW-5381-01Z2

NetCOBOL for Windows V7.0

Web連携ガイド



NetCOBOL

FUJITSU

まえがき

COBOL Web連携機能は、以下に示すシステムの32Bitモードで動作し、WWWサーバで動作する32Bitアプリケーションの開発環境および運用環境を提供します。

- Microsoft(R) Windows NT(R) Server Network operating system Version 4.0
- Microsoft(R) Windows NT(R) Server Network operating system,Enterprise Edition Version 4.0
- Microsoft(R) Windows(R) 2000 Server operating system
- Microsoft(R) Windows(R) 2000 Advanced Server operating system

製品の呼び名について

本書に記載されている製品の名称を、以下のように略して表記します。

- 「Microsoft(R) Windows NT(R) Server Network operating system Version 4.0」
  → 「Windows NT(R)」または「Windows NT(R) 4.0」
- 「Microsoft(R) Windows NT(R) Server Network operating system,Enterprise Edition Version 4.0」
  → 「Windows NT(R)」または「Windows NT(R) 4.0」
- 「Microsoft(R) Windows(R) 2000 Server operating system」
  → 「Windows(R) 2000」
- 「Microsoft(R) Windows(R) 2000 Advanced Server operating system」
  → 「Windows(R) 2000」
- 「Microsoft(R) Internet Information Server 4.0」
  → 「IIS」または「IIS 4.0」
- 「Microsoft(R) Internet Information Server 5.0」
  → 「IIS」または「IIS 5.0」
- 「Microsoft(R) Internet Explorer」
  → 「IE」
- 「Netscape Enterprise Server」
  → 「NES」
- 「Fujitsu Interstage Application Server」
  → 「Interstage」
- 「Fujitsu Interstage Application Server InfoProvider Pro」
  → 「IPP」

本書の目的

本書は、COBOLによるWebアプリケーションの概要や各WWWサーバの特徴について説明しています。各Webサブルーチンの詳細な仕様については、“COBOL Webサブルーチン 使用手引書”を参照してください。

その他の注意事項

本書で使用している“Interstage”は、Fujitsu INTERSTAGE Application Server V4.0以前では“INTERSTAGE”に置き換えてお読みください。

登録商標について

本書に記載されている登録商標を以下に示します。

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

Netscape、Netscape Navigator、Netscape Enterprise Serverは、米国Netscape Communications Corporationの米国およびその他の国における商標または登録商標です。

Oracle、Web Request Brokerは、米国ORACLE Corporationの商標または登録商標です。

INTERSTAGE、Interstageは、富士通株式会社の登録商標です。

その他の会社名または製品名は、それぞれ各社の商標または登録商標です。

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

2002年7月

目次

第1章　概要

アプリケーションが動作するシステムの形態には、ホスト集中型とクライアント/サーバ型に加え、ネットワークコンピューティング型があります。
ネットワークコンピューティング型は、インターネットの急速な普及に伴い、情報公開としての役割だけではなく、業務アプリケーションの動作環境として利用されています。
ネットワークコンピューティング環境では、ネットワークを通じての業務連携およびWWWを使用した情報共有による業務効率の向上が可能となります。また、プラットホーム無依存な操作性を持つWWWブラウザの利用は、多種の業務連携を実現する上で欠かせないものといえます。
一方で、すでにホスト集中型およびクライアント/サーバ型のシステムでは、COBOLは業務システムの基幹系業務の領域での中心言語として多く使用されています。
このようなインターネット/イントラネット技術を基盤に、基幹系処理に適しているCOBOLを使用した業務アプリケーションを活用したいという要件は高まっています。

1.1　インターネット/イントラネット環境の活用

インターネット/イントラネット環境を有効に活用するために、WWWの特色とCOBOLを使用する利点を以下に示します。

a. WWWの特色

- 遠隔地
  ネットワーク上のあらゆる資産にアクセス可能となるため、遠隔地との接続も簡単に行えます。
- コストの削減
  WWWブラウザの利用により、クライアントマシンのセットアップ・メンテナンス作業を軽減できます。
- クライアントのシステム無依存
  WWWブラウザを使用するため、クライアントのシステムに依存しない業務システムを構築できます。モバイルコンピューティング環境でも利用可能です。
- サーバの負荷分散
  ネットワークで接続可能なサーバであれば、各種の業務システムを分散させることができます。
- セキュリティ
  WWWの特色として、クライアントから容易に接続することが可能な反面、セキュリティの問題があります。システムを構築する場合には、WWWサーバでクライアントが接続するドメインに対するアクセス制限を設けたり、アプリケーションでユーザ制限を行うといった、自己防衛手段を講じる必要があります。

b. COBOLを使用する利点

通常、WWW環境で動作するアプリケーションの開発言語を選択する場合、WWWブラウザとのデータ入出力を可能とするために、以下の特徴を考慮します。

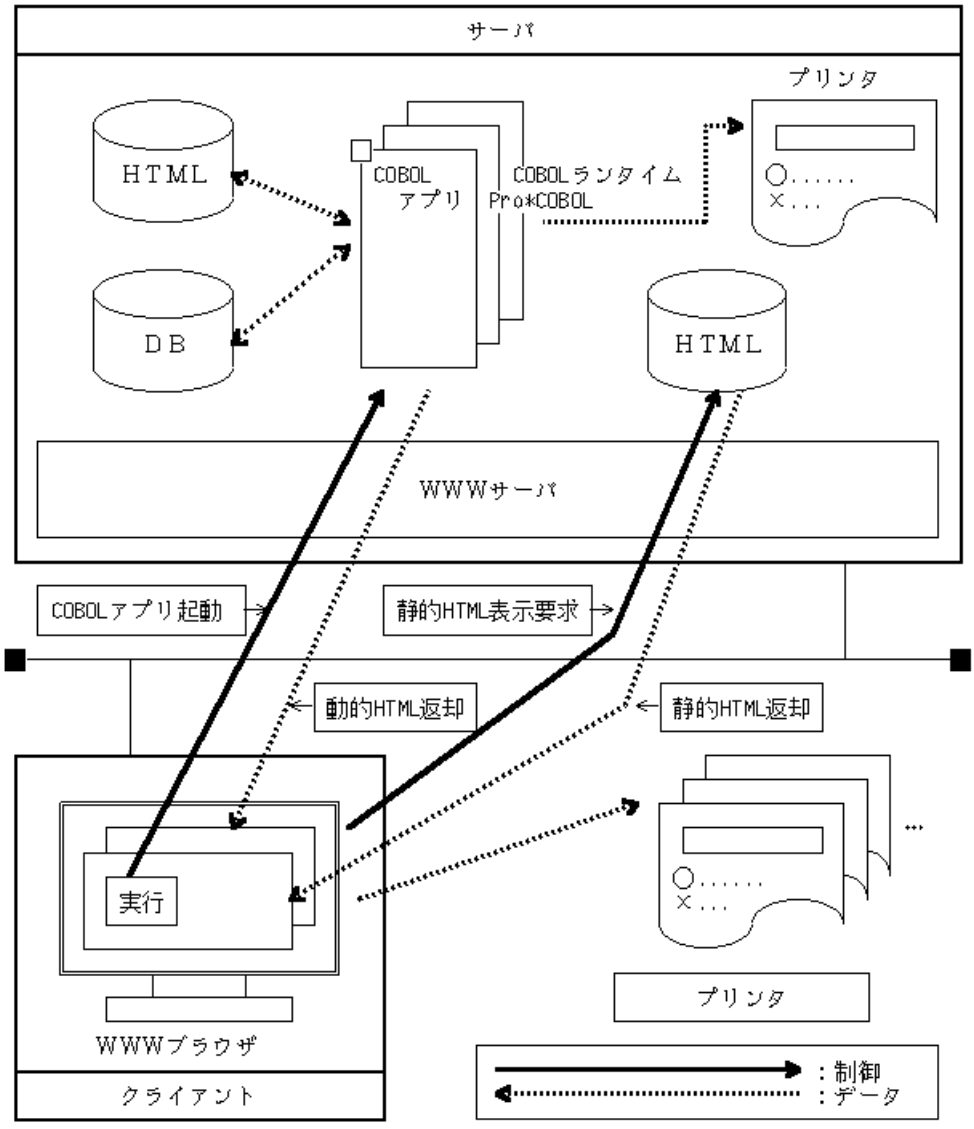
- データ操作
- ファイルやデータベースへのアクセス
- 環境変数操作

COBOLは、上記の特徴を満たすだけでなく、さらに以下の点からも効率のよい開発を行うことができます。

- 今まで培った知識やシステム構築ノウハウの活用
  COBOLを使用したアプリケーションの開発により培われてきたノウハウを活用できるため、開発のスピードアップを図ることができます。
- 既存資産の有効活用
  従来の環境での業務システムで使用していた資産を有効活用できます。

1.2 Webアプリケーション

HTML文書単独では静的な情報しか返せないため、更新される情報をリアルタイムに伝えられません。これに対し、CGIや各種WWWサーバ固有のAPIを、WWWブラウザから呼び出されるアプリケーションで利用し、WWWブラウザから入力した情報に対して、更新される情報を動的に返却することにより、リアルタイムに情報を提供することができます。また、情報の返却だけでなく、業務アプリケーションには欠かせないファイル入出力やデータベースアクセスなどのデータ処理手続きをWWWブラウザから依頼(実行)できます。
COBOLは、データ処理を中心とする事務処理言語であり、この特性を最大限に利用したインタラクティブな動的HTMLデータの作成が可能となります。



なお、Webアプリケーションは、WWWサーバ上でサービス配下のアプリケーションとして動作します。アプリケーション作成および動作の留意点については、“NetCOBOL 使用手引書”の“23.1.3 サービス配下で動作するプログラム”を参照してください。

1.3　Web連携の利用方法

COBOLでインターネット/イントラネット環境の業務システムを構築するには、以下の2つの方法があります。

- COBOL Webサブルーチンを利用したアプリケーションの開発
  WWWブラウザとのデータ入出力を使用したアプリケーション開発に必要なパラメタ解析や処理結果出力などの複雑な処理を意識することなく、COBOLの知識だけでアプリケーションを作成できます。
- MeFt/Web(注)またはMeFt/NET-SV(Web連携機能)の利用
  Windows上でMeFtを使用したCOBOLアプリケーションおよびクライアント/サーバ環境のMeFt/NET-SV(NET連携機能)を使用したCOBOLアプリケーションを、再翻訳することなく、そのままイントラネット環境の基幹業務として運用できます。

注

NetCOBOL Professional Edition、NetCOBOL Standard Editionに同梱されています。

第2章　COBOL Webサブルーチンを利用したアプリケーション

通常、Webアプリケーションを開発/メンテナンスするには、WWWブラウザとの間のインタフェースに関する知識が必要となります。しかし、COBOL Webサブルーチンを利用すれば、COBOLの知識だけでCOBOLプログラムとWWWブラウザとの間でデータの入出力を行うことができます。

2.1　Webアプリケーションが利用できるインタフェースの種類

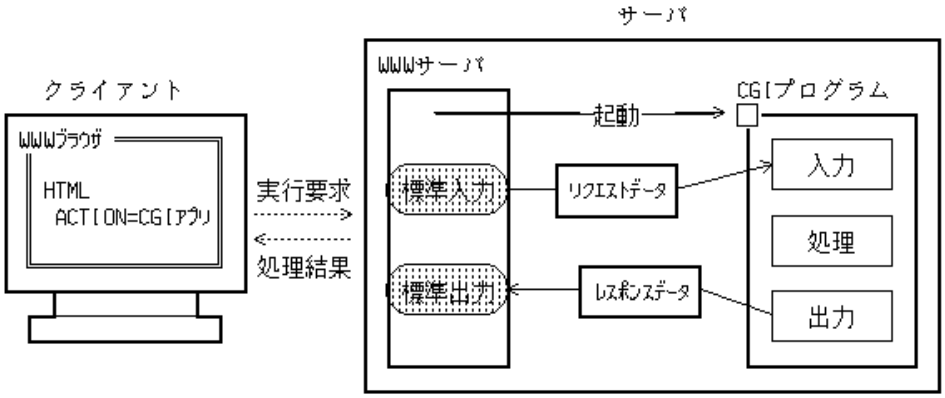
以下に示すように、WWWサーバのアプリケーションインタフェースは多数存在します。

- ISAPI(Internet Server API)
  ISAPIは、Microsoft(R)により提唱されたWWWサーバ上で動作する対話型アプリケーションのための拡張されたプログラミング・インタフェースです。ISAPIの実行コードは、Microsoft(R) Internet Information Server(以降では、IISと略します)などのISAPI準拠のWWWサーバに搭載されています。
  COBOLでは、ISAPIおよびその実行コードをCOBOLの言語仕様に合わせてカプセル化し、IIS上で動作するISAPIアプリケーション(一般的には、ISAと呼ばれます)をより簡単に開発するための機能として、ISAPIサブルーチンを提供しています。
  また、ISAPIは、WWWサーバ上で動作する対話型アプリケーションのプログラミング・インタフェースの点ではCGIとよく似ていますが、CPUやメモリなどの使用資源量およびパフォーマンスの面で相違があります。ISAPIとCGIの相違点については、“2.3 CGIとの相違点”を参照してください。
- NSAPI(Netscape Server API)
  NSAPIは、Netscape Enterprise Server(以降、NESと略します)などのWWWサーバの機能を拡張するためのプログラミング・インタフェースで、Netscape Communicationsが提供しています。
  COBOLでは、NSAPIおよびその実行コードをCOBOLの言語仕様に合わせてカプセル化し、NES上で動作するNSAPIアプリケーション(一般的には、SAFと呼ばれます)を簡単に開発するための機能として、COBOL SAFサブルーチンとCOBOL SAFディレクタを提供しています。
  NSAPIも、WWWサーバ上で動作する対話型アプリケーションのプログラミング・インタフェースの点ではCGIとよく似ていますが、CPUやメモリなどの使用資源量およびパフォーマンスの面で相違があります。NSAPIとCGIの相違点については、“2.3 CGIとの相違点”を参照してください。
- CGI(Common Gateway Interface)
  CGIは、WWWブラウザとのデータ入出力を行う最も標準的なインタフェースです。また、ほとんどのWWWサーバはこのCGIをサポートしています。
  COBOLでは、CGIを利用するアプリケーションをより簡単に開発するための機能として、COBOL CGIサブルーチンを提供しています。
- その他、各種WWWサーバ固有API
  前述のように、各アプリケーションインタフェースおよびWWWサーバはCGIで発生するオーバヘッドを解消するため、CGIに代わる独自の技術で高速処理を実現するためのAPIを提供しています。これにより、パフォーマンスの向上や、サーバ負荷の軽減を実現しています。
  以下に代表的なAPIを紹介します。
  - 拡張CGI
    拡張CGIは、IPPが提供するプログラミング・インタフェースです。拡張CGIでは、複数回のリクエストが継続できるセション管理型と、Webアプリケーションをメモリに常駐できる常駐型の2つのタイプの機構が提供されています。
    拡張CGIを利用するCOBOLアプリケーションをより簡単に開発するための機能として、Interstageでは、COBOL Webサブルーチンが提供されています。詳細については、Interstageの“Interstage Application Server WWWサーバ運用ガイド(InfoProvider Pro編)”および“COBOL Webサブルーチン ユーザーズガイド”を参照してください。
  - WRB(Web Request Broker) API
    WRB APIは、Oracle Application Serverの機能を利用するために提供されたプログラミング・インタフェースです。WRB APIを利用することにより、Oracle Application

Serverによって提供されているサービスにアクセスすることができます。提供されているサービスには、Loggerサービス、Inter-Cartridge Exchange(ICX)サービス、トランザクション・サービスなどがあります。
WRB APIを利用するCOBOLアプリケーションをより簡単に開発するための機能として、OracleからCOBOLカートリッジが提供されています。詳細については、Oracle COBOLカートリッジの“カートリッジ・ユーザーズ・ガイド”を参照してください。

2.2　Webアプリケーションの処理の流れ

以下に、WWWブラウザとWebアプリケーションの処理の流れをCGIを例に説明します。
CGIは、WWWブラウザとデータの入出力を行うために利用します。CGIとは、WWWブラウザ(クライアント)からWWWサーバ上のCGIプログラム(CGIスクリプトと呼ばれる)を動作させるための最も基本的なインタフェースであり、CGIプログラムの起動と標準入出力(環境変数経由によるデータ入力を含む)によるデータの入出力を提供しています。
CGIは、「データ入力(リクエスト)→処理→処理結果出力(レスポンス)」の流れで動作し、この流れを実現するのがCGIプログラムです。
ISAPIや他の各種APIも処理の流れは基本的に同じですが、CGIではデータの入出力が標準入出力を介して行われるのに対して、ISAPIや他の各種APIではデータ入出力専用のAPIが用意(APIの中に隠蔽)されています。この違いはCOBOL Webサブルーチンによって吸収されているため、COBOL Webサブルーチンを使用している限りCOBOLアプリケーション側で、このような違いを意識する必要はありません。



2.3 CGIとの相違点

CGIアプリケーションは、実行可能なモジュール形式である必要があります。このため、WWWサーバはクライアント(WWWブラウザ)からのリクエストごとにCGIアプリケーションの実行プロセスを個別に生成するので、サーバのCPUやメモリなどの資源が大量に消費されます。
これに対してISAPIアプリケーション(ISA)やNSAPIアプリケーション(SAF)は、ダイナミックリンクライブラリ形式(DLL)で作成し、WWWサーバ自身のプロセスにロードされ、一度ロードされたDLLはメモリ中に常駐したり一定時間を経過するとアンロードされるなどの制御が行われます。
CGIとこれら他アプリケーションインタフェースとの大きな違いは、CGIがリクエストごとに個別のプロセスで実行されるのに対して、他はリクエストごとにWWWサーバ自身のアドレス空間にスレッドを生成しマルチスレッド環境で実行される点です。このため、CGIにおけるプロセスの生成/消滅、メモリ使用量、タスク切り替えなどのオーバヘッドが解消され、パフォーマンスに優れたWebアプリケーションを構築することができます。

図2-1 ISAPI、NSAPIを使用した場合

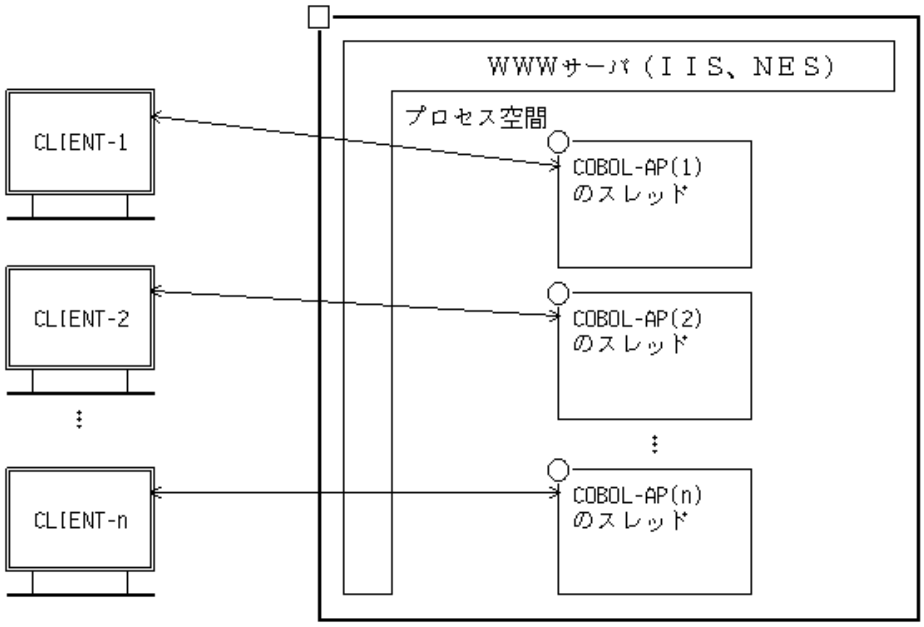
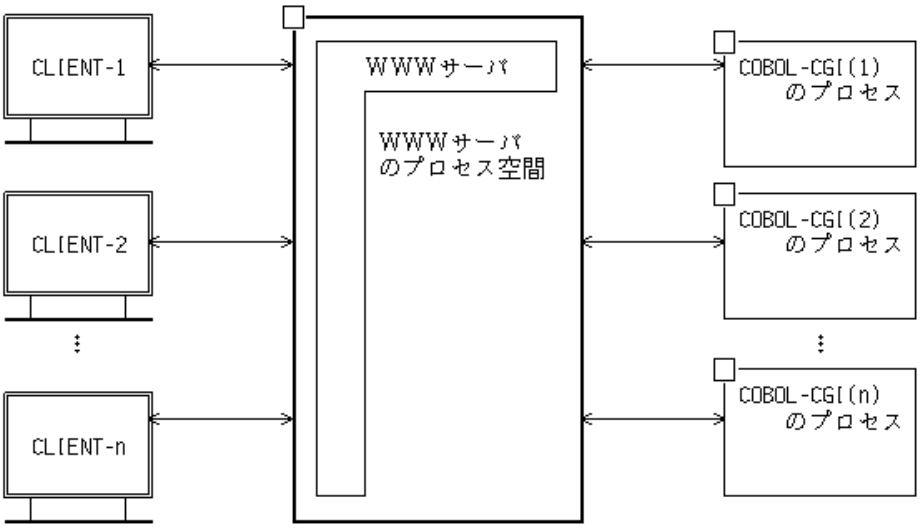


図2-2 CGIを使用した場合



2.4 COBOL Webサブルーチン利用の目的

COBOL Webサブルーチンは、Webアプリケーションを開発する上で必要となるWWWブラウザから入力したWebパラメタの解析や、処理結果の出力などの機能を提供します。これらの機能を使用することにより、特別な知識を必要とすることなく、COBOLだけで容易にWebアプリケーションを作成することが可能になります。

2.5　COBOL Webサブルーチン

ISAPIの利用

COBOL ISAPIサブルーチンを使用することにより、IIS上で動作するISAPIを利用したWebアプリケーションをCOBOLで作成することができます。

COBOL ISAPIサブルーチンは、COBOL CGIサブルーチンと共通インタフェースであるため、CGIからISAPIへの移行も容易に行えます(ほとんど同じ仕様ですが、一部ISAPI固有の記述を追加する必要があります)。

NSAPIの利用

COBOL SAFサブルーチンとCOBOL SAFディレクタを使用することにより、NES上で動作するNSAPIを利用したWebアプリケーションをCOBOLで作成することができます。

COBOL SAFサブルーチンは、COBOL CGIサブルーチンと共通インタフェースを持つサブルーチンに加えて、NES固有の拡張機能を利用できるサブルーチンを提供しています。このため、CGIからNSAPIへの移行も容易に行えます(ほとんど同じ仕様ですが、一部NSAPI固有の記述を追加する必要があります)。

CGIの利用

COBOL CGIサブルーチンを使用することにより、さまざまなWWWサーバ上で動作するCGIを利用したWebアプリケーションを作成することができます。

拡張CGI(セション管理型/常駐型)の利用

IPPが提供する拡張CGI(セション管理型/常駐型)を利用するためには、INTERSTAGE V1.0以降が必要となります。INTERSTAGE V1.0以降で提供されるCOBOL Webサブルーチンを使用することにより、拡張CGI(セション管理型/常駐型)を利用したWebアプリケーションを作成することができます。

第3章　MeFt/Web(注)を利用したアプリケーション

表示ファイル機能の画面入出力(画面定義体)を使用して開発した業務アプリケーションは、MeFt/WebまたはMeFt/NET-SV(Web連携機能)と連携することにより、再翻訳することなく、そのままインターネット/イントラネット環境の基幹業務に使用することができます。

注

MeFt/WebはMeFt/NET-SV(Web連携機能)と同等の機能を持つコンポーネントで、NetCOBOL Professional Edition、NetCOBOL Standard EditionまたはNetCOBOL Standard Edition サーバ運用パッケージに同梱されています。

3.1　MeFt/Web利用の目的

WWW環境では、データ入力のためにHTMLを使用しますが、HTMLはデータタイプの不足など、業務システムでの入力処理に使用するには、機能が不足する部分もあります。MeFt/WebまたはMeFt/NET-SV(Web連携機能)を使用すると、MeFtの入出力機能をそのまま利用することができるため、これらの機能不足を補うことが可能です。
また、以下のような利点も備えています。

- 既存資産の活用とノウハウの継承
  既存の表示ファイル(MeFt接続)を使用しているアプリケーションは、そのままWWWサーバ上で動作します。
- 資源の一元管理
  WWWブラウザを使用するため、クライアントアプリケーションの開発は不要になります。また、アプリケーションが使用する画面帳票定義体およびクライアント側で動作するMeFtもサーバ側に置くので、サーバでの資源の一元管理が可能です。
  [注意] WWWブラウザがIEの場合は、MeFtは実行時に自動的にサーバからダウンロードされます。WWWブラウザがNetscape Navigator(TM)の場合は、プラグインをインストールする必要があります。
- 帳票印刷機能の利用
  サーバ、クライアントのどちらにでも帳票印刷ができます。また、従来の帳票印刷の方法に加えて、帳票のスプールおよびプレビューが可能になります。
- WWWブラウザ機能の活用
  従来の画面機能に加えて、最小限の手直しで画面の3D化およびハイパーリンク機能を画面に追加することができます。

3.2 MeFt/Webの利用方法

アプリケーションは表示ファイル機能を利用するため、開発には従来のCOBOL開発環境を利用できます。表示ファイル機能としては、動作環境に依存せず、従来のMeFtを使用したアプリケーションと同様に開発できます。
詳細については、“ MeFt/Web 説明書 ”を参照してください。

第4章　COBOL Web連携機能の選択ポイント

4.1　各Web連携機能の特徴

COBOL Web連携の各機能における特徴を以下に示します。

連携機能 / 特徴機能	[SAP[サブルーチン	SAF サブルーチン	CG[サブルーチン	拡張CG[用 サブルーチン	MeFt/Web
動作可能 WWWサーバ	[[S	NES	CG[をサポートする全サーバ	[PP	[[S,[PP
WWW 利用技術	[SAP[	NSAP[	CG[	CG[および拡張CG[（セション管理/プロセス常駐機構）	Active対技術
WWWサーバが起動するファイルの形式	DLL	DLL	EXE	EXE	DLL/EXE
動作モデル	スレッド	スレッド	プロセス	プロセス	スレッド/プロセス
WWWサーバ上で実行されるCOBOLアプリケーションのローディング機構	該当アプリに対する初回リクエスト時にロードされ、以後、次回リクエストに備えメモリ内に常駐。	サーバ起動時または該当アプリに対する初回リクエスト時にロードされ、以後、次回リクエストに備えメモリ内に常駐。	該当アプリに対するリクエストごとにプロセスを起動し、処理終了後そのプロセスを破棄。	サーバ起動時に該当アプリのプロセスを事前起動し、リクエスト待機状態でメモリに常駐。	該当アプリに対する初回リクエスト時にメモリにロードされ、表示ファイルによる一連の画面対話が行われている間（ＯＰＥＮ～ＣＬＯＳＥ）メモリ内に常駐。
実行性能	マルチスレッド動作、ＤＬＬのプレロード機能等の仕組みにより、高速。	マルチスレッド動作、ＤＬＬのプレロード機能等の仕組みにより、高速。	プロセス起動・消滅を毎リクエストごとに行うオーバヘッドのため低速。	プロセス事前起動・待機の仕組みによりＣＧＩに比べ高速。	マルチスレッド動作により高速。
ユーザ/アプリケーション・インタフェース	・HTML ・Webサブルーチン				表示ファイル
業務アプリケーション作成の容易性	WWWサーバとのインタフェースは、COBOL言語仕様に則したWebサブルーチン内に隠蔽されているため、WWWに関する特別難しい専門知識は不要。				基本的には、表示ファイルおよびMeFt/Webの知識があれば良い。
既存資産の活用性	既存資産のビジネスロジックは、そのほとんどをそのまま有効活用することが可能。ただし、WWWサーバとのインタフェースが必要となるため、Webサブルーチンを使用するための記述を追加し、再翻訳する必要がある。				プロセスベースのアプリであれば、ソース修正および再翻訳・再リンクとも不要。スレッドベースであれば、再翻訳・再リンクが必要。
帳票印刷	サーバ側はアプリによるきめ細かい帳票印刷が可能。ただし、クライアント側はブラウザの印刷機能またはそれに代わるダウンロードアプリケーションの印刷機能を使用。				サーバ、クライアントともアプリによるＭｅＦｔを使用した帳票印刷が可能。また、クライアント側で印刷プレビュー機能も利用可能。
利点	マルチスレッドで、高性能かつ資源消費の少ないアプリを作成可能。		サーバに依存しない、独立性の高いアプリを作成可能。	サーバ側のセション管理により、簡単にセションを継続するアプリが作成可能。	既存の表示ファイル機能を使用したアプリを活用し、開発期間を短縮。マルチスレッド化も容易。

上記の表は、各Web連携機能の特徴を横並びで記すために挙げた大まかな概要情報です。これらの詳細情報、関連製品および注意事項などについては、各Web連携機能に対応するマニュアル(後述)を必ず参照してください。

4.2　各Web連携機能の詳細情報

ご使用になりたいCOBOL Web連携機能が決まったら、次は各連携機能ごとの詳細情報が記されたマニュアルを参照し、実際に業務アプリケーションの設計・開発を行います。
各Web連携機能に対応するマニュアルを以下に示します。

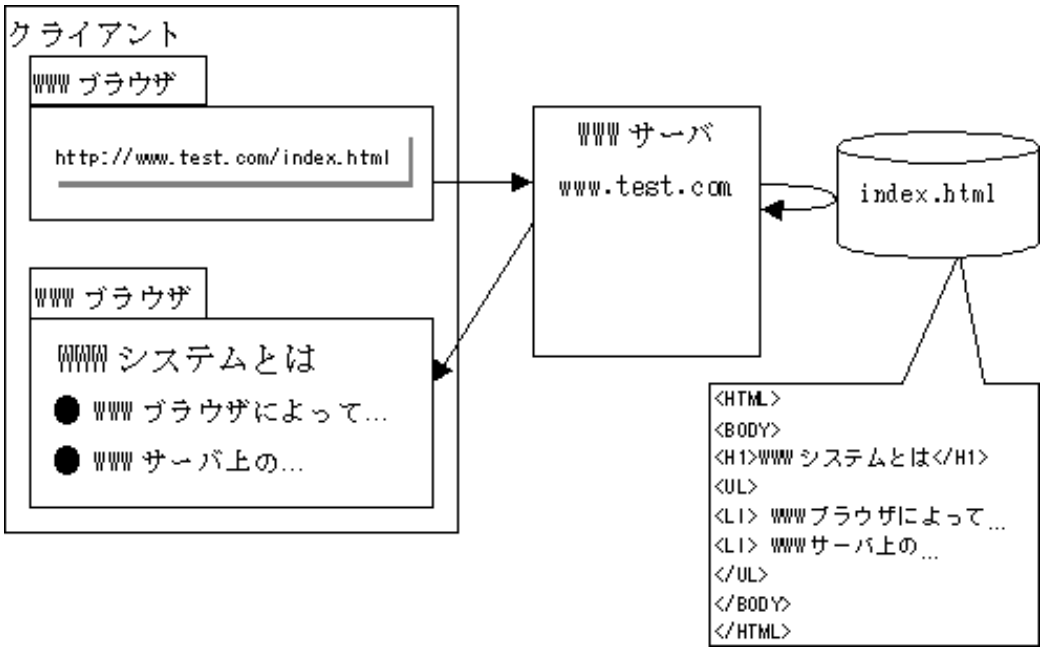
- COBOL ISAPIサブルーチンを使用する場合
  COBOL ISAPIサブルーチンの詳細については、“COBOL Webサブルーチン 使用手引書”を参照してください。
- COBOL SAFサブルーチンを使用する場合
  COBOL SAFサブルーチンの詳細については、“COBOL Webサブルーチン 使用手引書”を参照してください。
- COBOL CGIサブルーチンを使用する場合
  COBOL CGIサブルーチンの詳細については、“COBOL Webサブルーチン 使用手引書”を参照してください。
- すでにCOBOL CGIサブルーチンを使用して作成されたアプリケーションを、COBOL ISAPIサブルーチンを使用したアプリケーションに移行する場合
  CGIからISAPIへの移行については、“COBOL Webサブルーチン 使用手引書”の“CGIからISAPIへの移行の手引き”を参照してください。
- すでにCOBOL CGIサブルーチンを使用して作成されたアプリケーションを、COBOL SAFサブルーチンを使用したアプリケーションに移行する場合
  CGIからSAFへの移行については、“COBOL Webサブルーチン 使用手引書”の“CGIからSAFへの移行の手引き”を参照してください。
- 拡張CGI(セション管理型/常駐型)用COBOL Webサブルーチンを使用する場合
  拡張CGI(セション管理型/常駐型)用のCOBOL Webサブルーチンの詳細については、Interstageの“COBOL Webサブルーチン ユーザーズガイド”を参照してください。
- MeFt/WebまたはMeFt/NET-SV(Web連携機能)を使用する場合
  表示ファイル機能の詳細については、“NetCOBOL 使用手引書”の“9.2 表示ファイル(画面入出力)の使い方”、“8.5 表示ファイル(帳票印刷)の使い方”を参照してください。
  MeFt/WebまたはMeFt/NET-SV(Web連携機能)の詳細については、“MeFt/Web 説明書”を参照してください。

付録A Web上で動作するCOBOLアプリケーションをはじめて作成する方のために

ここでは、Webを利用したアプリケーションをはじめて構築する方または、はじめてCOBOL Webサブルーチンを利用する方のために基本的な事柄について説明します。

A.1 Webアプリケーションとは

インターネット/イントラネットなどで利用するWWW(World Wide Web)は、ネットワーク上のWWWサーバとWWWブラウザ(クライアント)の間をHTTP(HyperText Transfer Protocol)を使用して接続します。WWWブラウザはURL(Uniform Resource Locators)を指定することで、特定のWWWサーバ上の特定の資源にアクセスします。また、WWWサーバは指定されたリソースをWWWブラウザに送信します。通常はHTML(HyperText Markup Language)データを送信するので、WWWブラウザはその内容を適切な形式で表示します。



このようにして、WWWブラウザを使用して接続可能なサーバで公開されている情報を自由にアクセスすることが可能になります。現在では情報公開だけでなく、業務アプリケーションを実行するための環境としても利用されています。

HTML文書だけをサーバ上に置いた場合、静的な情報を公開することは可能です。しかし、静的な情報だけでは業務に欠かせないサーバ上のデータベースへのアクセスや、動的にHTML文書を生成するなどの処理ができません。WWWの主なインタフェースを以下に示します。

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

業務アプリケーションでは、これらインタフェースを利用してWWWサーバ上のアプリケーションを作成する必要があります。このWWWサーバ上で実行されるアプリケーションをWebアプリケーションと呼びます。

A.2 COBOL Webサブルーチンについて

COBOL Webサブルーチンは、COBOLを使用してWebアプリケーションを作成するためのインタフェ

ースと実行環境を提供します。CGIやISAPI、NSAPIなどの各種インタフェースを用いて構成されていますが、利用者はその違いをほとんど意識しないで利用できるため、PerlやC言語よりも簡単にWebアプリケーションを作成できます。
また、COBOL Webサブルーチンには次の特徴もあります。

- CALLインタフェースで実行できるためCOBOLで簡単に利用できます。
- HTMLの一部を変換して出力する機能を提供することで、プログラムとHTML文書を独立させることができ、生産性および保守性の高い開発が可能です。
- 利用者はCOBOL Webサブルーチンの提供するインタフェースを使用してプログラムを記述することで、さまざまなWWWサーバ上で動作するWebアプリケーションを作成することができます。

COBOL Webサブルーチンでは次のサービスを提供しています。

- WWWサーバからアプリケーション起動時に渡されるリクエストデータ(Webパラメタ)の受取りおよびCOBOLのデータ形式への変換
- 処理結果の出力(動的HTMLの出力機能など)
- ヘッダ(Content-Typeなど)の自動生成
- システムコマンドの実行機能
- リクエスト情報の取得
- Cookieデータの受取りおよび出力
- ファイルのアップロード

A.3　HTTPの基礎

WWWはHTTPの上に成り立っています。WWWサーバとWWWブラウザは、このHTTPを使用してデータの送受信を行います。WWWブラウザからWWWサーバへのデータの送信をリクエストと呼び、その逆をレスポンスと呼びます。通常は、リクエストに対してレスポンスが必ず存在し、リクエストで通信路を確保し、レスポンスが完了した時点で解放します。HTTPではリクエストとレスポンスは最初の行以外は同じ形式をしており、非常に簡単な構成になっています。

- リクエストデータ

メソッド　ＵＲＬ　バージョン
ＨＴＴＰヘッダ
メッセージボディ

- レスポンスデータ

バージョン　ステータスコード　理由
ＨＴＴＰヘッダ
メッセージボディ

それぞれの意味は以下のとおりです。

- メソッド
  WWWサーバへの要求内容を表し、GET、POST、HEADなどがあります。
- URL
  要求するリソースの場所を表します。
- バージョン
  使用するHTTPのバージョンを表します。
- ステータスコード
  3桁の数字でサーバおよびWebアプリケーションの処理状況を表します。
- 理由
  ステータスコードに対する理由を表します。
- HTTPヘッダ
  WWWサーバからWWWブラウザをコントロールしたり、WWWブラウザの情報やメソッドに対する条件付けを行うのに使用します。
- メッセージボディ
  WWWサーバとWWWブラウザ間で送受信されるデータを表します。

個々の詳細については、HTTPの仕様書または書籍などを参照してください。ここでは、メソッド、ステータスコードおよびHTTPヘッダのうち、Webアプリケーションが参照または設定するものについて簡単に説明します。

主なメソッドは、以下のとおりです。

- GET
  URLで指定した情報を取り出します。URLで指定したものがWebアプリケーションの場合、そのアプリケーションが出力するデータを受信します。
  なお、フォームに入力したデータはアプリケーションに渡されますが、データ量には上限があります。
- HEAD
  機能的にはGETと同じですが、HTTPヘッダだけを受信する点が異なります。URLで指定した情報の属性などを調べるときに使用します。
- POST
  フォームに入力したデータをWebアプリケーションへ送信し、そのアプリケーションが出力するデータを受信するために使用します。ファイルのアップロードを行う場合も、このメソッドを指定します。送信できるデータ量に上限はありません。

主なステータスコードは以下のとおりです。

ステータスコード	意味
200	正常終了しました。
302	要求されたドキュメントは一時的に移動しました。移動先は“Location”ヘッダを参照してください。
303	要求されたドキュメントは別のURL に移動されました。移動先は“Location”ヘッダを参照してください。
400	構文エラーです。
401	認証に失敗しました。
403	指定されたURL に対するアクセスが禁止されています。
404	要求されたURL に対するリソースが存在しません。
410	WWW ブラウザ、サーバ間の資源不一致です。
500	内部エラー。 Web アプリケーションからの応答がない場合など。
502	WWW サーバから不当なリクエストが返されました。
503	高負荷などの理由により、WWW サーバへのアクセスができません。

主要なHTTPヘッダは以下のとおりです。

HTTPヘッダ	意味
Accept	WWW ブラウザが受け取ることのできるMIMEタイプを表します。
Accept-Charset	WWW ブラウザが受け取ることのできる文字コードセットを表します。
Accept-Encoding	WWW ブラウザが受け取ることのできるエンコーディング方式。通常はデータ圧縮の形式を表します。
Accept-Language	WWW ブラウザが受け取ることのできる言語の種類を表します。日本語はja、英語はenとなります。
Content-Encoding	レスポンスデータのエンコーディング方式を示します。
Content-Language	レスポンスデータの言語の種類を示します。
Content-Type	レスポンスデータのMIMEタイプを示します。
Date	リクエストデータおよびレスポンスデータの作成さ

	れた日付を示します。
Expires	レスポンスデータの有効期限を示します。
Location	情報の正確な位置を示します。
Host	WWW サーバが稼動しているサーバのホスト名。
Referer	リクエスト元となったURL を表します。
User-Agent	WWW ブラウザの情報を表します。

A.4　HTMLの基礎

HTML(HyperText Markup Language)はハイパーテキストを表現するための言語で、文章の見栄えやリンクなどをタグと呼ばれる特定のキーワードを使用して表現します。タグは“<”と“>”に囲まれた文字列で表されます。また、この文字列をタグ名といいます。ほとんどの場合、タグは開始タグと終了タグを持ち、それぞれ“<タグ名>”、“</タグ名>”となります。この開始タグと終了タグで囲まれた文字列がWWWブラウザにとって特別な意味を持ちます。また、<タグ名>だけで意味を持つ単独タグと呼ばれるものもあります。また、タグは属性を持つ場合があります。ほとんどの場合、属性は属性名=属性値の形式で表現されますが、属性名だけのものもあります。属性を持つタグは<タグ名 属性名1=属性値 属性名2>のように表現されます。
ここでは、HTMLを記述するために最低限必要なタグと、Webアプリケーションを呼び出す際に必要なタグについてだけ紹介します。各タグの詳細や他のタグについては、HTMLについて解説している書籍やホームページなどを参照してください。
なお、現在使用されている代表的なWWWブラウザはIEとNetscape Navigator(TM)ですが、それぞれ独自のタグを定義していますので、すべてのタグが双方のWWWブラウザで使用できるとは限りません。また、共通のタグでも動作が異なるものがあります。さらに、WWWブラウザのバージョンによってもサポート範囲や動作が異なります。これらについても詳しくは書籍やホームページなどを参照してください。
以下に主なタグの説明とそのタグで使用できる属性を示します。

- <HTML>～</HTML>
  説明
  HTML文書であることを表します。
- <HEAD>～</HEAD>
  説明
  HTML文書のヘッダであることを表します。ヘッダには<TITLE>を記述します。他には<BASE>、<SCRIPT>、<STYLE>、<META>、<LINK>、<OBJECT>、<NEXTID>、<ISINDEX>などのタグが記述できます。
- <TITLE>～</TITLE>
  説明
  HTML文書のタイトルを表します。通常、タイトルはページ上には表示されません。
- <BODY>～</BODY>
  説明
  HTML文章の本文を表します。この中で、さまざまなタグを使用して文章の体裁を整えます。
  属性

属性名	意味
BGCOLOR= "色"	背景色を指定します。
BACKGROUND= "URL"	背景画像を指定します。
TEXT= "色"	テキストの色を指定します。TEXTは、通常のテキスト、LINKはリンクテキスト、VLINK はキャッシュ済みリンクテキスト、ALINK はマウスクリックしている間のリンクテキストを表します。
LINK= "色"	
VLINK= "色"	
ALINK= "色"	

● <Hn>～</Hn>

説明

章の見出しを指定します。nは章のレベルで、1から6まで指定できます。

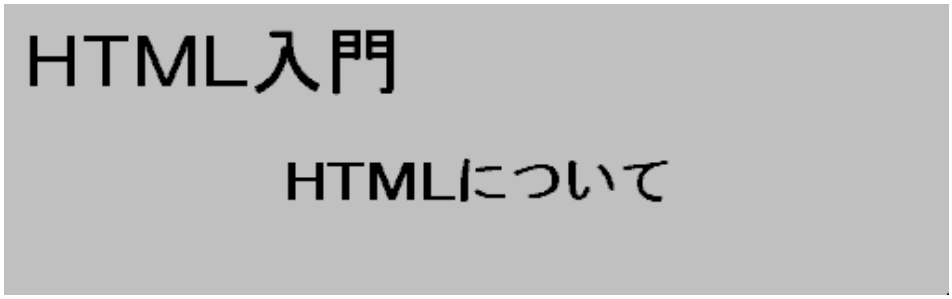
属性

属性名	意味
ALIGN= "位置"	表示位置を指定します。位置にはleft、center、rightが指定できます。

使用例

```
<H1>ＨＴＭＬ入門</H1>
<H2 ALIGN=center>ＨＴＭＬについて</H2>
```

表示結果



● <P>～</P>

説明

段落を表します。</P>は省略可能ですが、属性を指定する場合は必須です。

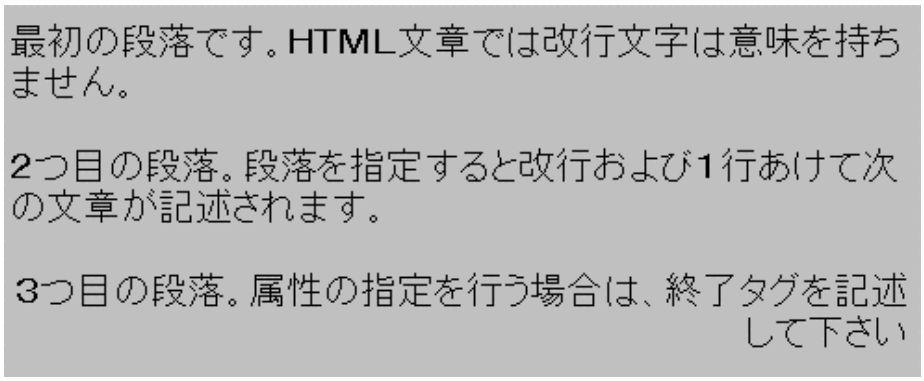
属性

属性名	意味
ALIGN= "位置"	表示位置を指定します。位置にはleft、center、rightが指定できます。

使用例

```
<P> 最初の段落です。
ＨＴＭＬ文章では改行文字は意味を持ちません。
<P> ２つ目の段落。
段落を指定すると改行および１行あけて次の文章が記述されます。
<P ALIGN=right> ３つ目の段落。
属性の指定を行う場合は、終了タグを記述して下さい</P>
```

表示結果



● <HR>
説明
横線を引きます。
属性

属性名	意味
ALIGN= "位置"	表示位置を指定します。位置にはleft、center、rightが指定できます。
SIZE= "高さ"	横線の高さをピクセル単位で指定します。
WIDTH= "幅"	横線の幅をピクセル単位で指定します。また、ブラウザの表示幅に対する割合（%）での指定も可能です。
NOSHADE	横線を立体的な影を持たない線にします。

使用例

```
最初の横線です。
<HR>
２つ目の横線です。
<HR SIZE=5 WIDTH=50% NOSHADE>
３つ目の横線です。
<HR ALIGN=left SIZE=8 WIDTH=20%>
```

表示結果

● <FORM>～</FORM>
説明
入力フォームを表します。フォーム中で<INPUT>や<SELECT>タグを使用することで入力部品を配置します。アクションを実行する際に<INPUT>や<SELECT>タグに対する入力データを渡すことができるので、Webアプリケーションの実行に使用されます。
属性

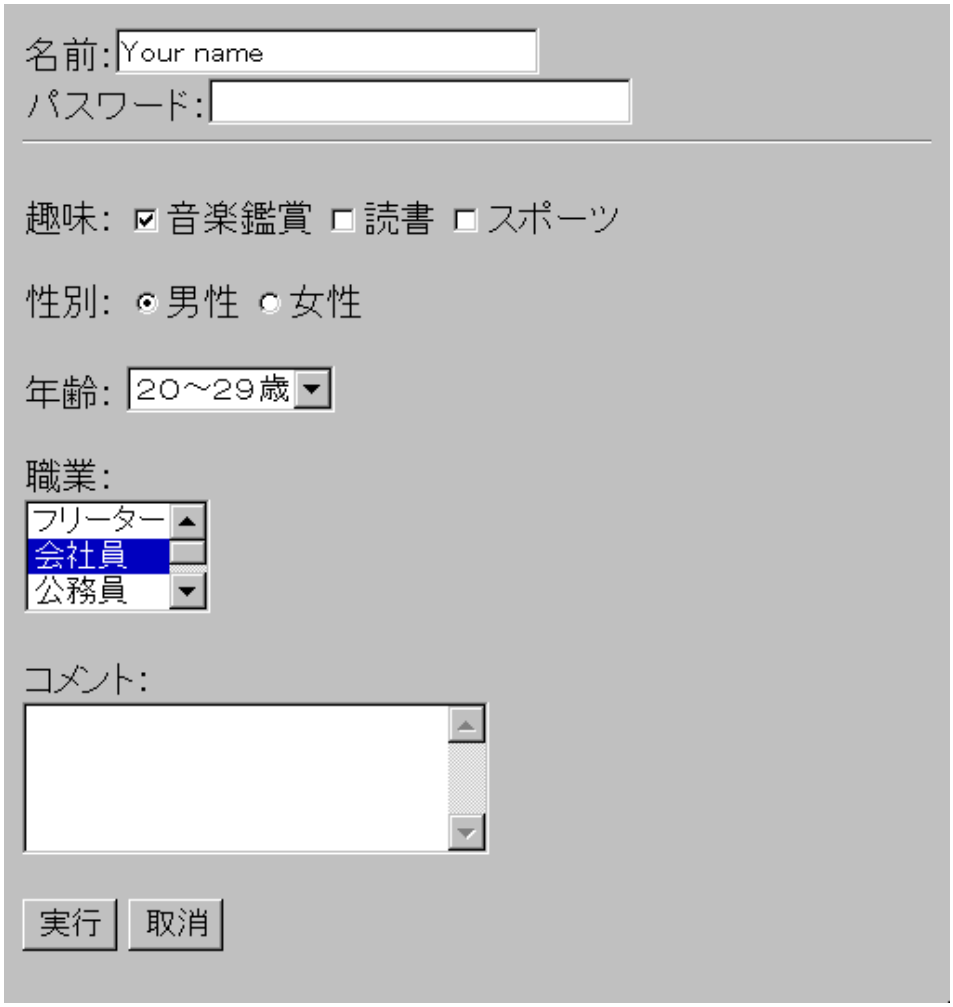
属性名	意味
ACTION= "アクション名"	実行（submit）ボタンを押下したときに実行するアクションを指定します。通常はWeb アプリケーションを指定します。
METHOD= "メソッド名"	フォームに入力した値をWeb アプリケーションに渡す方法を指定します。メソッド名がGET の場合は環境変数へ、POSTの場合は標準入力へ渡します。ファイルアップロードの機能を使用する場合は、POSTを指定します。
ENCTYPE= "エンコードタイプ名"	スクリプトにデータを受け渡す際のエンコード方法を指定します。通常は省略して構いませんが、ファイルアップロードの機能を使用する場合は、"multipart/form-data"を指定します。
TARGET= "ターゲット名"	アクションの実行結果を表示するウィンドウ（フレーム）を指定します。
NAME= "フォーム名"	フォームに名前を付けます。 スクリプトから使用します。
onSubmit= "スクリプト"	実行ボタンを押下したときに実行されるスクリプトを指定します。
onReset= "スクリプト"	取消しボタンを押下したときに実行されるスクリプトを指定します。

使用例

```
------------------------------------------------------------------------
  <FORM METHOD=POST ACTION="sample/action.script">
  名前：<INPUT TYPE=TEXT NAME="FULLNAME1" VALUE="Your name" SIZE=30><BR>
  パスワード：<INPUT TYPE=PASSWORD NAME="PASSWORD1" SIZE=30>

  <HR>
  <P>
  趣味：
  <INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="音楽鑑賞" CHECKED>音楽鑑賞
  <INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="読書">読書
  <INPUT TYPE=CHECKBOX NAME="CHECK1" VALUE="スポーツ">スポーツ

  <P>
  性別：
  <INPUT TYPE=radio NAME="RADIO1" VALUE="男性" CHECKED>男性
  <INPUT TYPE=radio NAME="RADIO1" VALUE="女性">女性

  <P>
  年齢：
  <SELECT NAME="DRDLIST1">
  <OPTION VALUE="１０代">１０～１９歳
  <OPTION VALUE="２０代" SELECTED> ２０～２９歳
  <OPTION VALUE="３０代">３０～３９歳
  <OPTION VALUE="４０代以降">４０歳～
  </SELECT>

  <P>
  職業：<BR>
```

```
<SELECT NAME="LIST1" SIZE=3>
<OPTION VALUE="フリー">フリーター
<OPTION VALUE="会社員" SELECTED> 会社員
<OPTION VALUE="公務員">公務員
<OPTION VALUE="自営業">自営業
</SELECT>

<P>
コメント：<BR>
<TEXTAREA NAME="TEXTAREA1" COLS=25 ROWS=4></TEXTAREA>
<P>
<INPUT TYPE=submit VALUE="実行">
<INPUT TYPE=reset VALUE="取消">
</FORM>
</BODY>
</HTML>
```

表示結果



- <INPUT>

  説明

  入力フォームにおける各種フォーム部品を表示します。表示する部品はTYPE属性によって決まります。

  共通属性

属性名	意味
TYPE= "タイプ名"	タイプ名に応じて表示する内容が異なります。 TEXT：テキスト PASSWORD：パスワード CHECKBOX：チェックボックス RADIO ：ラジオボタン HIDDEN：隠し領域 BUTTON：ボタン SUBMIT：サブミットボタン RESET ：リセットボタン FILE ：ファイルアップロード
NAME= "名前"	フォーム部品の名前を指定します

● <INPUT TYPE=TEXT>
説明
テキストを入力するためのフォーム部品です。
属性

属性名	意味
SIZE= "幅"	入力領域の横幅を指定します。
MAXLENGTH= "長さ"	最大入力文字数を指定します。
onChange= "スクリプト"	入力領域が変更されたときに呼び出されるスクリプトを指定します。
onSelect= "スクリプト"	入力領域内の文字列が選択されたときに呼び出されるスクリプトを指定します。
onFocus= "スクリプト"	入力領域にフォーカスが移ったときに呼び出されるスクリプトを指定します。
onBlur= "スクリプト"	入力領域からフォーカスが外れるときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=PASSWORD>
説明
パスワードを入力するためのフォーム部品です。入力文字が“*”で表示される以外はTYPE=TEXTと同じです。

● <INPUT TYPE=CHECKBOX>
説明
複数選択を行うためのフォーム部品です。
属性

属性名	意味
VALUE= "文字列"	この項目がチェックされたときに送信される値を指定します。
CHECKED	この項目の初期状態をチェックされた状態にします。
onClick= "スクリプト"	チェックされたときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=RADIO>
説明
択一選択を行うためのフォーム部品です。
属性

属性名	意味
VALUE= "文字列"	この項目がチェックされたときに送信される値を指定します。
CHECKED	この項目の初期状態をチェックされた状態にします。
onClick= "スクリプト"	チェックされたときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=HIDDEN>
説明
隠しフィールドです。ブラウザには表示されません。ブラウザを介してWebアプリケーション間でデータを引き継ぐときなどに使用します。
属性

属性名	意味
VALUE= "文字列"	フィールドの値を指定します。

● <INPUT TYPE=SUBMIT>
説明
FORMタグのACTION属性で指定したアクションを実行します。
属性

属性名	意味
VALUE= "文字列"	ボタンに表示する文字列を指定します。
onClick= "スクリプト"	チェックされたときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=RESET>
説明
FORM内のすべての入力項目が初期値に戻ります。
属性

属性名	意味
VALUE= "文字列"	ボタンに表示する文字列を指定します。
onClick= "スクリプト"	チェックされたときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=BUTTON>
説明
JavaScriptで使用します。したがって、JavaScript(またはJScript)が使用できないWWW

ブラウザでは表示されません。
属性

属性名	意味
VALUE= "文字列"	ボタンに表示する文字列を指定します。
onClick= "スクリプト"	チェックされたときに呼び出されるスクリプトを指定します。

● <INPUT TYPE=FILE>
説明
クライアント側からWWWサーバ側へアップロードするファイルを指定するためのフォーム部品です。
使用例

```
-------------------------------------------------------------------------
  <FORM METHOD="POST" ACTION="sample/action.script"
        ENCTYPE="multipart/form-data">
   <P>
   送信ファイル1：<INPUT TYPE="file" NAME="FILE1"><BR>
   送信ファイル2：<INPUT TYPE="file" NAME="FILE2"><BR>
   <INPUT TYPE="submit" VALUE="送信">
   <INPUT TYPE="reset"  VALUE="取消">
   </FORM>
-------------------------------------------------------------------------
```

表示結果

● <TEXTAREA>～</TEXTAREA>
説明
入力フォームにおける複数行入力フィールドを表示します。<TEXTAREA>～</TEXTAREA>までの間のテキストがフィールドに表示されます。
属性

属性名	意味
NAME= "文字列"	フィールドに名前を付けます。
ROWS= n	テキストエリアの行数を指定します。
COLS= n	テキストエリアの桁数を指定します。
onChange= "スクリプト"	入力領域が変更されたときに呼び出されるスクリプトを指定します。
onSelect= "スクリプト"	入力領域内の文字列が選択されたときに呼び出されるスクリプトを指定します。
onFocus= "スクリプト"	入力領域にフォーカスが移ったときに呼び出されるスクリプトを指定します。
onBlur= "スクリプト"	入力領域からフォーカスが外れるときに呼び出されるスクリプトを指定します。

● <SELECT>～</SELECT>
説明
入力フォームにおける選択フォーム部品を表示します。
属性

属性名	意味
NAME= "文字列"	フィールドに名前を付けます。
SIZE= n	選択フィールドの表示行数を指定します。
MULTIPLE	複数選択を可能にします。
onChange= "スクリプト"	入力領域が変更されたときに呼び出されるスクリプトを指定します。
onFocus= "スクリプト"	入力領域にフォーカスが移ったときに呼び出されるスクリプトを指定します。
onBlur= "スクリプト"	入力領域からフォーカスが外れるときに呼び出されるスクリプトを指定します。

● <OPTION>～</OPTION>
説明
SELECTタグにおける選択項目を表示します。</OPTION>は省略可能です。
属性

属性名	意味
SELECTED	初期状態として選択状態をとります。
VALUE= "文字列"	この項目がチェックされたときに送信される値を指定します。

A.5　WWWブラウザ操作上の注意点と対策

クライアント・サーバシステムでは、クライアント側で利用者が勝手にアプリケーションの制御を変更することはできませんでした。ところが、WWWブラウザを使用したインターネット/イントラネット型のシステムでは、利用者がWWWブラウザの状態を容易に変更できるため、操作によってはサーバ上のWebアプリケーション(業務)の整合性が損なわれる恐れがあります。
一般的には、以下に示す操作が問題視されています。

● WWWブラウザの「戻る」ボタンを使用する。
すでに行った処理を再度実行できるため、登録や更新系の処理では、業務に矛盾を引き起こす場合があります。

● ダブルクリックなどの操作によって、SUBMITボタン(INPUTタグのINPUT属性に”SUBMIT”を指定したボタン)を複数回押してしまう。
同じ処理が二重に実行されるため、登録や更新系の処理では、業務に矛盾を引き起こす場合があります。

● ブックマークやURLの直接入力によって、業務の途中から開始する。
認証なしで業務を実行できるため、セキュリティ上の問題があります。また、正しい順序で業務を実行しないため、ロジック上の問題が発生する可能性があります。

● キャッシュされたページの参照
WWWブラウザのキャッシュ機能を使用することで、すでに参照されたページをサーバへのアクセスなしに参照できるため、第三者によって売上げデータや個人情報などの機密情報が参照される危険性があります。

上記の問題に対して、WWWブラウザおよびサーバ上のWebアプリケーションがとれる対策の例を以

下に示します。

利用者の操作	対策例	
	WWWブラウザ	Webアプリケーション
WWW ブラウザの「戻る」ボタンを使用する。	JavaScriptなどを使用して、「戻る」ボタンをなくしたウィンドウを開く。	アプリケーションの実行順序を把握し、正しい順序で実行されていることを確認する。
ダブルクリックなどで、SUBMITボタンを複数回押してしまう。	JavaScriptなどを使用して、処理中かどうかを判断し、処理中の場合はSUBMITを無効にする。	二重に実行されても問題のない造りにしておく。
ブックマークやURL の直接入力によって、業務の途中から開始する。	JavaScriptなどを使用して、「ブックマーク」などをなくしたウィンドウを開く。	アプリケーションの実行順序を把握し、正しい順序で実行されていることを確認する。
キャッシュされたページを参照する。	WWW ブラウザのキャッシュを無効にする。	―

通常、業務の内容に応じて対策の内容も異なります。たとえば、機密度が低い業務では、キャッシュを無効にする必要はありません。また、上記以外にも業務内容によって問題となる操作がある場合は、それぞれの局面に応じた対策をとってください。

索引

FUJITSU

このマニュアルはエコマーク認定の再生紙を使用しています。