(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4015375号
(P4015375)
(45)発行日　平成19年11月28日(2007.11.28)　(24)登録日　平成19年9月21日(2007.9.21)

(51)Int.Cl.　　F I
G06F 15/00 (2006.01)　　G06F 15/00　３１０R
G06F 13/00 (2006.01)　　G06F 13/00　５５０A

請求項の数 12　外国語出願　(全 28 頁)

(21)出願番号　特願2001-129923(P2001-129923)
(22)出願日　平成13年4月26日(2001.4.26)
(65)公開番号　特開2002-49484(P2002-49484A)
(43)公開日　平成14年2月15日(2002.2.15)
審査請求日　平成15年12月4日(2003.12.4)
(31)優先権主張番号　09/573769
(32)優先日　平成12年5月18日(2000.5.18)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　バード、ゲイリー　エス.
アメリカ合衆国、９８０３３　ワシントン州、カークランド、エヌイー　１０３ド　ストリート　１１４１１

最終頁に続く

(54)【発明の名称】クライアント側ユーザインタフェース要素を処理するサーバ側制御オブジェクト

(57)【特許請求の範囲】
【請求項１】
ウエブページを表示するクライアントとウエブサーバとの間において、前記クライアント上に表示される前記ウェブページの中に組み込まれる少なくとも１つのクライアント側ユーザインタフェース要素を処理する方法であって、
前記サーバが、動的ウェブページ資源を参照しているリクエストを前記クライアントから受信するステップと、
前記サーバが、前記受信したリクエストにおいて指定された資源から宣言を読み出すステップであって、前記宣言は、前記動的ウェブページ資源において、前記少なくとも１つのクライアント側ユーザインタフェース要素を特定する情報および前記少なくとも１つのクライアント側ユーザインタフェースの機能を記述していることと、
前記サーバが、前記宣言に基づいて、前記クライアント側ユーザインタフェース要素と論理的に対応し、同時に存在する階層化された複数のサーバ側制御オブジェクトを生成するステップと、
前記サーバが、前記宣言において記述された前記クライアント側ユーザインタフェースの前記機能を実現するために、前記同時に存在する複数のサーバ側制御オブジェクトを用いて、前記クライアント側ユーザインタフェース要素の処理をするステップであって、前記処理には、
（ａ）　前記サーバ側制御オブジェクトを用いて、前記少なくとも１つのクライアント側ユーザインタフェース要素から受信したポストバックデータを処理することと、

（ｂ） 前記サーバ側制御オブジェクトを用いて、前記少なくとも１つのクライアント側ユーザインタフェース要素から受信したポストバックイヴェントを処理することと、

（ｃ） 前記サーバ側制御オブジェクトのビューステートを保存し、前記ビューステートを前記クライアントに送信することであって、前記ビューステートは、前記同時に存在する階層化された複数のサーバオブジェクトの状態を示すことと、

を含んでいることと、

前記サーバが、前記処理するステップの後で、前記同時に存在するサーバ側制御オブジェクトに基づいて、前記ウェブページに前記クライアント側ユーザインタフェース要素を組み込むオーサリング言語データを生成する処理を実行するステップと、

を備えることを特徴とするクライアント側ユーザインタフェース要素を処理する方法。

【請求項２】

処理をする前記ステップにおける前記処理は、

（ｄ） 前記サーバが、先のレスポンスにおいて送信した前記複数のサーバ側制御オブジェクトに対応する前記ビューステートを、前記クライアントから受信することと、

（ｅ） 前記サーバが、前記受信した前記ビューステートをロードすることと、

をさらに含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。

【請求項３】

処理をする前記ステップは、

前記サーバが、前記同時に存在する階層化された複数のサーバ側制御オブジェクト中であって、前記クライアント側ユーザインタフェース要素に対応する１つ以上の前記サーバ側制御オブジェクトに関連付けられたサーバ側制御オブジェクトイベントを立ち上げるステップと、

前記サーバが、前記サーバ側制御オブジェクトイベントを処理するように予め登録された、前記１つ以上の前記サーバ側制御オブジェクトとは異なるサーバ側制御オブジェクトを用いて、前記制御サーバ側オブジェクトイベントを処理するステップと

をさらに含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。

【請求項４】

処理をする前記ステップは、非ユーザインタフェースサーバコンポーネントを用いて前記サーバ側イベントを処理するステップをさらに含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。

【請求項５】

前記同時に存在する階層化された複数のサーバ側制御オブジェクトは、

前記１つ以上のクライアント側ユーザインタフェース要素に対応する１つ以上のサーバ側子オブジェクトであって、前記サーバ側子オブジェクトの各々は、前記クライアントから受信したポストバック入力を処理し、前記クライアント上でクライアント側ユーザインタフェース要素を表示するオーサリング言語データを生成するように適合されていることと、

前記ポストバック入力とともに前記クライアントから受信した、前記サーバ側子オブジェクトのうちの１つを特定する少なくとも１つの階層識別子と、

前記同時に存在する階層化された複数のサーバ側制御オブジェクトの階層の最上位に位置し、前記階層識別子に基づいて前記サーバ側子オブジェクトのうちの１つに分配される前記ポストバック入力を、受信するサーバ側ページオブジェクトと、

を含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。

【請求項６】

前記サーバ側ページオブジェクトは、前記クライアントからの前記リクエストに応答して作成され、前記オーサリング言語データが生成されるまで存在し続けることを特徴とする請求項５に記載のクライアント側ユーザインタフェース要素を処理する方法。

【請求項７】
前記サーバ側子オブジェクトの各々は、前記階層識別子によって特定されたサーバ側制御オブジェクトへのアクセスに応答して作成され、前記オーサリング言語データが生成されるまで存在し続けることを特徴とする請求項５に記載のクライアント側ユーザインタフェース要素を処理する方法。
【請求項８】
同時に存在する階層化された複数のサーバ側制御オブジェクトを生成する前記ステップは、
前記サーバが、前記ウェブページ上のクライアント側ユーザインタフェースコンテナ要素に対応するサーバ側コンテナ制御オブジェクトを生成するステップと、
前記サーバが、前記クライアント側ユーザインタフェースコンテナ要素に含まれた１つ以上のクライアント側ユーザインタフェース子要素に対応する１つ以上のサーバ側子制御オブジェクトを生成するステップと、
を含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。
【請求項９】
処理をする前記ステップは、
前記サーバが、前記階層化された複数のサーバ側制御オブジェクトの中の１つ以上のサーバ側制御オブジェクトを参照する一意の階層識別子を前記クライアントから受信するステップと、
前記サーバが、前記一意の階層識別子に関連付けられたポストバック入力情報を、前記クライアントから受信するステップと、
前記サーバが、前記参照されたサーバ側制御オブジェクトを識別するために、前記一意の階層識別子を解決するステップと、
前記サーバが、前記参照されたサーバ側制御オブジェクトへ、前記ポストバック入力情報を渡すステップと、
前記サーバが、前記参照されたサーバ側制御オブジェクトを用いて、前記ポストバック入力情報を処理するステップと、
を含むことを特徴とする請求項８に記載のクライアント側ユーザインタフェース要素を処理する方法。
【請求項１０】
処理をする前記ステップは、前記同時に存在する階層化された複数のサーバ側制御オブジェクトの中の１つ以上のサーバ側制御オブジェクトによる個々の処理を呼び出すために、前記同時に存在する階層化された複数のサーバ側制御オブジェクトをトラバースするステップを含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。
【請求項１１】
処理をする前記ステップは、
前記サーバが、前記同時に存在する複数のサーバ側制御オブジェクトの中の１つ以上の前記サーバ側制御オブジェクトに基づいて、制御オブジェクトイベントを生成するステップと、
前記サーバが、前記制御オブジェクトイベントを処理するように予め登録されたサーバ側制御オブジェクトを用いて、前記制御オブジェクトイベントを処理するステップと
を含むことを特徴とする請求項１に記載のクライアント側ユーザインタフェース要素を処理する方法。
【請求項１２】
コンピュータシステムによって読み取り可能であり、かつ、クライアント上で表示されるウェブページに組み込まれる少なくとも１つのクライアント側ユーザインタフェース要素を処理する、請求項１乃至１１のいずれかの方法の各ステップを実行処理するコンピュータプログラムをコード化しているコンピュータプログラム記憶媒体。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、一般にウェブサーバフレームワークに関し、特にウェブページのクライアント側ユーザインタフェース要素を処理するサーバ側制御オブジェクトに関する。
【０００２】
【従来の技術】
典型的なウェブブラウザは、クライアントシステムにおいて表示するウェブページの外観および基本動作を定義するウェブサーバからデータを受け取る。典型的な手順としては、ユーザが、ワールドワイドウェブ上の資源のグローバルアドレスであるユニホームリソース（資源）ロケータ（以下、「URL」という。）を特定し、所望のウェブサイトにアクセスする。一般に、用語「資源」とは、プログラムによってアクセスすることができるデータまたはルーチンのことである。
【０００３】
URLの一例は、“ HYPERLINK "http://www.microsoft.com/ms.htm" http://www.microsoft.com/ms.htm ”である。このURL例の第１の部分は、通信に用いる所与のプロトコル（例えば“http”）を示している。第２の部分は、資源の所在を示すドメイン名（例えば“ HYPERLINK "http://www.microsoft.com" www.microsoft.com ”）を特定している。第３の部分はドメイン内の資源（例えば“ms.htm”と呼ばれるファイル）を特定している。これに従い、ブラウザは、 HYPERLINK "http://www.microsoft.com" www.microsoft.com ドメイン内のms.htmファイルに関連するデータを取り出すためのURL例に関連するHTTP(ハイパーテキストトランスポートプロトコル)リクエストを生成する。www.microsoft.comサイトをホストしているウェブサーバはHTTPリクエストを受信し、要求されたウェブページまたは資源をクライアントシステムにHTTPレスポンスで戻し、ブラウザに表示する。
【０００４】
上記の例の"ms.htm"ファイルは、静的なHTML（ハイパーテキストマークアップ言語）コードを含んでいる。HTMLは、ワールドワイドウェブ上でのドキュメント（例えば、ウェブページ）作成に用いられるプレイン（平文）テキストオーサリング言語である。このようなものであるので、HTMLファイルは、ウェブサーバから取り出され、インターネットからの情報を表示している間、ユーザが期待するグラフィカルな体験を提供するために、ブラウザにウェブページとして表示することができる。
【０００５】
HTMLを用いて、開発者は、例えば、ブラウザに表示するフォーマット化されたテキスト、リスト、フォーム、テーブル、ハイパーテキストリンク、インライン画像および音声、ならびに背景グラフィックスを特定することができる。しかし、HTMLファイルは、ウェブページコンテンツの動的な生成を本質的にサポートしていない静的ファイルである。
【０００６】
また、ウェブページは、ブラウザに株価の変化や交通情報など動的コンテンツを表示する必要がある場合がある。このような状況では、典型的なサーバ側アプリケーションプログラムは、動的データを取得し、それを、ウェブページが更新されるとともにウェブページに表示するブラウザに送信されるHTML形式にフォーマットするように開発される。
【０００７】
さらに、データは厳密には動的でないが、静的なウェブページに表示する値が非常に多くの異なる値であるために、要求された数の静的なウェブページを作成することが実用的でないような状況において、これらの同様のサーバ側アプリケーションプログラムを用いることができる。例えば、旅行計画ページは、出発日用カレンダーと帰着日用カレンダーとの２種類のカレンダー表示をすることがある。考え得るすべてのカレンダーの組み合わせによる何百もの静的なページを開発するかわりに、サーバ側アプリケーションプログラムは、適切なカレンダーを表示した適切な静的なページを動的に生成することができる。
【０００８】

多くのウェブページにより、ユーザはページの視覚的要素を選択することによってブラウザに表示されたページと対話することができる。例えば、上記の旅行計画ページにおいて、カレンダーによって、ユーザは日付をクリックしてその日付を選ぶ、またはアイコンをクリックして月を進めたり戻ったりすることによってそのカレンダーと対話することができる。既存の解決法では、ブラウザは、サーバ側アプリケーションプログラムにHTTPリクエストを出す。このHTTPリクエストは、フォームポスト変数のような照会行列あるいはその他、クライアント側イベントまたはデータ(例えば、どのコントロールをユーザがクリックしたか)を記述するデータフォーマットでコード化されたパラメータを含むことができる。例えば、パラメータは、１つのカレンダーでユーザが選択したデータを他方のカレンダーで現在表示中のデータとともに含む場合がある。

【０００９】

サーバに戻すイベントおよびデータの通信は、「ポストバック」と呼ばれる。これは、典型的に、ブラウザが、HTTPポストリクエストを用いてリクエストを送信するからである。サーバ側アプリケーションプログラムは、HTTPリクエストを処理し、クライアントへHTTPレスポンスで送信するユーザのアクションを反映する新たに計算されたカレンダーを有するウェブページに適切なHTMLコードを生成する。その後、得られたドキュメントは、HTTPレスポンスでクライアントシステムに送信され、このドキュメントは、更新されたカレンダーを示すウェブページとしてブラウザに表示される。

【００１０】

サーバ側アプリケーションプログラムの開発は、ウェブページ設計に用いる通常のHTMLコード化に精通しているだけでなく、どのようにデータがブラウザとサーバとの間で送信されるか、または１つ以上のプログラム言語（例えば、C++, Perl, Visual Basic, またはJscript）およびHTTPプロトコルを含むプログラムベーシックスに精通していることが要求される複雑な作業である。しかし、ウェブページ設計者は、グラフィックデザイナーまたはエディタであることが多く、プログラム経験がない場合がある。さらに、複雑なウェブページ開発を単純化すると、いかなる開発者によっても新たなウェブコンテンツの開発の速度を上げることができる。

【００１１】

一般に、カスタムサーバ側アプリケーションプログラムの開発もまた、多大な努力が要求され、実際、開発者はしばしばそれを試みたくないと思う程である。開発者は、所望のウェブページを表示するために生成が要求されるHTMLコードを理解するだけでなく、ウェブページからのユーザ対話およびクライアントデータがどのようにしてポストバック処理になるのかを理解しなければならない。したがって、開発者が最小のプログラミングでウェブページを動的に作成および処理することができる開発フレームワークを提供することが望ましい。

【００１２】

【発明が解決しようとする課題】

動的ウェブページ生成のプログラム要件を最小にする１つの手段は、マイクロソフト社によって提供されるアクティブサーバページ（ASP）フレームワークである。ASPリソースは、典型的に、所望の資源としてASPリソースを特定するHTTPリクエストを処理し、その後、クライアントへのHTTPレスポンスでの結果HTMLコードを生成する、例えばVisual BasicまたはJscriptを含む。さらに、ASP資源は、所与のアプリケーションプログラミング努力を緩和するために、予め開発されたまたは第三者のクライアント側ライブラリコンポーネント（例えば、クライアント側"ACTIVEX"制御）を参照することができる。しかし、現在のサーバ側アプリケーションフレームワークにおいて、サーバ側アプリケーション内でクライアント側ユーザインタフェース要素（例えば、テキストボックス、リストボックス、ボタン、ハイパーリンク、画像、音声など）を動的に管理しなければならないプログラミングは、依然として高度なプログラミング技術と相当な努力を必要とする。未解決の課題は、ウェブページ開発者がウェブページの他のアスペクトに焦点を当てることができるように、ポストバックイベントをハンドリングすることなど、ユーザインタフェース要素を

処理することが要求されるプログラミングを適切にカプセル化することについてである。
【００１３】
【課題を解決するための手段】
本発明によると、上述および他の課題は、クライアント側ユーザインタフェース要素の処理および生成を管理するサーバ側制御オブジェクトフレームワークを提供することによって解決される。さらに、サーバ側制御オブジェクトの階層が、クライアントにウェブページを表示するための標準HTMLのような結果オーサリング言語コードを協働して生成することができる。クライアントは、例えば標準HTMLまたは別の結果オーサリング言語コードをサポートするものであればいかなるブラウザであってもよい。クライアント側ユーザインタフェース要素の処理は、１つ以上のポストバックイベントハンドリング処理、ポストバックデータハンドリング処理、データ結合処理またはサーバ側制御オブジェクトの状態に関する状態管理処理を含む。
【００１４】
本発明のある実施形態の大きな有効性は、クライアント側ユーザインタフェース要素（例えば、ユーザインタフェース要素からの受信入力およびユーザインタフェース要素の生成に用いる出力）および関連機能性のサーバ側処理のカプセル化の向上にある。１つ以上のサーバ側制御オブジェクトは、１つ以上のユーザインタフェース要素に論理的に対応するように生成される場合がある。例えば、カレンター表示において月を表すユーザインタフェース要素を考えると、サーバ側制御オブジェクトの階層は、カレンダー表示およびその種々のサブ要素に対応して生成される場合がある。ある構成では、「月」制御オブジェクトは、多数の「週」制御オブジェクトを階層的に含み、各「週」制御オブジェクトは、７つの「日」制御オブジェクトを階層的に含んでいる。
【００１５】
さらに、本発明のある実施形態において、サーバ側制御オブジェクトは、論理的に対応するユーザインタフェース要素を協働して処理することができる。この利点は、クライアントリクエスト処理およびレスポンスの生成中において多数のサーバ側制御オブジェクトが同時に存在することなどによりもたらされた結果である。例えば、クライアントから受信した第１のポストバックイベント（例えば、カレンダー表示上の「次の月」のボタン要素のクリック）を検知すると、１つの制御オブジェクトは、その後に検知され１つ以上の同時存在している制御オブジェクトによって処理される第２のイベント（例えば、「月」制御オブジェクトによる次の月の選択および表示）を立ち上げる。この協働により、サーバ側制御オブジェクト自体内で複雑な制御対話のカプセル化を行うことができ、これにより、ウェブページ開発者に要求されるカスタムイベントハンドリングを最小限にすることができる。
【００１６】
クライアント上で表示されるウェブページに組み込まれるクライアント側ユーザインタフェース要素を処理する方法およびコンピュータプログラムプロダクトが提供される。サーバ側宣言データ格納部を参照するリクエストを受信する。宣言が、サーバ側宣言データ格納部から入力される。サーバ側制御オブジェクトが、宣言に基いてユーザインタフェース要素の機能性を与えるように生成およびプログラムされる。サーバ側制御オブジェクトを用いて、ユーザインタフェース要素が処理される。オーサリング言語データが、ウェブページにユーザインタフェース要素を表示するサーバ側制御オブジェクトから生成される。
【００１７】
クライアント上に表示されるウェブページに組み込まれる１つ以上のクライアント側ユーザインタフェース要素を処理する、コンピュータで実行処理可能なサーバ側制御オブジェクトの階層が提供される。１つ以上のサーバ側子オブジェクトは、１つ以上のクライアント側ユーザインタフェース要素に対応する。各サーバ側子オブジェクトは、クライアントから受信した入力を扱い、クライアント上でクライアント側ユーザインタフェース要素を表示するためのオーサリング言語データを生成する。少なくとも１つの階層識別子が、サ

ーバ側子オブジェクトのうちの１つを特定している入力に関連するクライアントから受信される。サーバ側ページオブジェクトは、サーバ側子オブジェクトを含み、階層識別子に応じたサーバ側子オブジェクトのうちの１つへ分配する入力を受信する。

【００１８】

【発明の実施の形態】

本発明のある実施形態は、ウェブページ上に表示するクライアント側ユーザインタフェース要素を処理および生成するサーバ側制御オブジェクトを含む。さらに、サーバ側制御オブジェクトの階層が、クライアントにウェブページを表示するための標準HTMLのような結果オーサリング言語コードを協働して生成することができる。クライアントは、例えば標準HTMLまたは別の結果オーサリング言語コードをサポートするものであればいかなるブラウザであってもよい。本発明のある実施形態では、サーバ側制御オブジェクトは、クライアント側ユーザインタフェース要素と論理的に対応し、ウェブページを表示および処理するクライアント側ブラウザによって用いられるオーサリング言語コードをサーバで生成する。クライアント側ユーザインタフェース要素の処理は、１つ以上のポストバックイベントハンドリング処理、ポストバックデータハンドリング処理、データ結合処理および状態管理処理を含んでも良い。

【００１９】

図１は、本発明のある実施形態におけるクライアントに表示するウェブページコンテンツを動的に生成するウェブサーバを示す。クライアント１００は、クライアント１００の表示装置上におけるウェブページ１０４を表示するブラウザ１０２を実行する。クライアント１００は、ビデオモニタのような表示装置を有するクライアントコンピュータシステムを含んでもよい。マイクロソフト社によって販売されている"INTERNET EXPLORER"ブラウザは、本発明のある実施形態におけるブラウザ１０２の一例である。他のブラウザの例として、"NETSCAPE NAVIGATOR" および "MOSAIC"などがあるが、これに限らない。例示したウェブページ１０４には、テキストボックス制御１０６および２つのボタン制御１０８、１１０が組み込まれている。ブラウザ１０２は、HTTPレスポンス１１２でウェブサーバ１１６からHTMLコードを受け取ることができ、HTMLコードにより記述されたウェブページを表示する。ある実施形態を参照してHTMLについて説明するが、特に制限されるものではなく、SGML（Standard Generalized Markup Language）、XML（eXtensible Markup Language）および、XMLベースのマークアップ言語であって、ポケットベルおよび携帯電話などの狭帯域無線装置の内容およびユーザインタフェースを特定するように設計されたWML（Wireless Markup Language）を含む他のオーサリング言語も本発明の範囲内であると考えられている。さらに、標準HTML３．２が、本明細書中に主に開示されているが、HTMLのいかなるバージョンも本発明の範囲内に含まれ得る。

【００２０】

クライアント１００とウェブサーバ１１６との通信は、HTTPリクエスト１１４およびHTTPレスポンス１１２の一連の処理を用いて行うことができる。ある実施形態を参照してHTTPを説明するが、特に制限されるものではなく、S－HTTPを含めた他のトランスポートプロトコルも本発明の範囲内であると考えられている。ウェブサーバ１１６において、HTTPパイプラインモジュール１１８が、HTTPリクエスト１１４を受信し、URLを解析し、リクエストを処理する適切なハンドラを呼び出す。本発明のある実施形態において、異なるタイプの資源をハンドリングする複数のハンドラ１２０がウェブサーバ１１６に備わっている。

【００２１】

例えば、URLがHTMLファイルのような静的なコンテンツ資源１２２を特定する場合、ハンドラ１２０は、静的コンテンツ資源１２２にアクセスし、HTTPパイプライン１１８を介して静的コンテンツ資源１２２をHTTPレスポンス１１２でクライアント１００へ送る。あるいは、本発明のある実施形態において、URLがASP+リソースのような動的コンテンツ資源１２４を特定する場合、ハンドラ１２０は、動的コンテンツ資源１２４にアクセスし、動的コンテンツ資源１２４のコンテンツを処理し、ウェブページ１０４用の結果HTMLコード

を生成する。本発明のある実施形態において、結果HTMLコードは、標準HTML３．２コードを含む。一般に、動的コンテンツ資源は、クライアントに表示すべきウェブページを記述するオーサリング言語を動的に生成するのに用いることができるサーバ側宣言データ格納部（例えば、ASP+リソース）である。そして、ウェブページ用のHTMLコードは、HTTPパイプライン１１８を通過し、HTTPレスポンス１１２でクライアント１００へ送られる。

【００２２】

この処理の間、ハンドラ１２０はまた、開発労力を簡単化するために予め開発された、あるいは第３者コードのライブラリにアクセスすることができる。このようなライブラリの１つがサーバ側クラス制御ライブラリ１２６であり、ここから、ハンドラ１２０は、ユーザインタフェース要素を処理しウェブページに表示する結果HTMLデータを生成するサーバ側制御オブジェクトをインスタンス化することができる。本発明のある実施形態において、１つ以上のサーバ側制御オブジェクトは、動的コンテンツ資源１２４に記述されたウェブページ上に、可視的にまたは隠して、１つ以上のユーザインタフェース要素にマッピングする。

【００２３】

これに対し、第２のライブラリは、マイクロソフト社からの"ACTIVEX"コンポーネントを含むライブラリのようなクライアント側制御クラスライブラリ１２８である。"ACTIVEX"制御は、クライアントおよび他のコンポーネントとの対話の仕方において一定の標準に従うCOM（コンポーネントオブジェクトモデル）オブジェクトである。クライアント側"ACTIVEX"制御は、例えば、クライアントに自動的にダウンロードされ、クライアントのウェブブラウザによって実行処理され得るCOMベースのコンポーネントである。サーバ側ACTIVEXコンポーネント（図示せず）は、株価検索アプリケーションまたはデータベースコンポーネントのサーバ側機能性を提供するような多様なサーバ側機能を果たすためにサーバ上で実行され得るCOMベースのコンポーネントである。ACTIVEXについては、「ACTIVEXおよびOLEの理解」（デビッド・チャペル、マイクロソフトプレス、１９９６年）により詳細に記載されている。

【００２４】

"ACTIVEX"制御とは対照的に、動的コンテンツ資源１２４に特定される本発明の実施形態のサーバ側制御オブジェクトは、クライアント上の表示されるユーザインタフェース要素に論理的に対応する。サーバ側制御オブジェクトはまた、例えば、HTMLタグと所与のクライアント側"ACTIVEX"制御を参照するロケータを含み得る有効なHTMLコードを生成することができる。ブラウザが、すでに格納システム内にクライアント側"ACTIVEX"制御用コードを有している場合は、クライアント上のウェブページ内で"ACTIVEX"制御を実行処理する。そうでなれば、ブラウザは、ロケータによって特定された資源から"ACTIVEX"制御用コードをダウンロードし、そして、クライアント上のウェブページ内で"ACTIVEX"制御を実行処理する。本発明の実施形態におけるサーバ側制御オブジェクトはまた、サーバ上で株価検索アプリケーションを実行するのに用いられるサーバ側"ACTIVEX"オブジェクトに対しイベントを立ち上げることができる。

【００２５】

ハンドラ１２０はまた、ウェブサーバ１１６上または別のアクセス可能ウェブサーバ上で実行処理する１つ以上の非ユーザインタフェースサーバコンポーネント１３０にアクセスする。株価検索アプリケーションまたはデータベースコンポーネントのような非ユーザインタフェースサーバコンポーネント１３０は、ハンドラ１２０によって処理される動的コンテンツ資源１２４において参照され、またはそれに関連付けられる。動的コンテンツ資源１２４で宣言された制御オブジェクトによって立ち上げられたサーバ側イベントは、非ユーザインタフェースサーバコンポーネント１３０の適切なメソッドを呼び出すサーバ側コードによって処理され得る。その結果、サーバ側制御オブジェクトによって提供される処理は、ウェブページのユーザインタフェース要素の処理および生成をカプセル化することによって非ユーザインタフェースサーバコンポーネント１３０のプログラミングを簡単にし、これにより、非ユーザインタフェースサーバコンポーネント１３０の開発者は、ユ

ーザインタフェース問題ではなく、アプリケーション固有の機能の開発に集中することができる。

【００２６】

図２は、本発明のある実施形態におけるサーバ側制御オブジェクトを用いてのクライアント側ユーザインタフェース要素の処理および生成処理のフローチャートを示す。処理２００において、クライアントは、HTTPリクエストをサーバに送信する。HTTPリクエストは、ASP+リソースなどの資源を特定するURLを含む。処理２０２において、サーバは、HTTPリクエストを受信し、特定された資源を処理する適切なハンドラを呼び出す。ASP+リソースは処理２０３において読み出される。処理２０４は、特定された動的コンテンツ資源（例えば、ASP+リソース）の内容に基づきサーバ側制御オブジェクト階層を生成する。

【００２７】

処理２０６において、制御オブジェクト階層のサーバ側制御オブジェクトは、ポストバックイベントハンドリング、ポストバックデータハンドリング、状態管理およびデータ結合のうちの１つ以上の処理を行う。ユーザインタフェース要素からのポストバックイベントおよびデータ（まとめて「ポストバック入力」）は、クライアントからサーバへと送られ処理される。ポストバックイベントは、特に制限されるものではなく、例えば、クライアント側ボタン要素からの「マウスクリック」またはサーバに送られるクライアント側テキストボックス要素からの「データ変化」イベントを含んでもよい。ポストバックデータは、特に制限されるものではなくが、例えば、テキストボックス要素またはドロップダウンボックスから選ばれた項目のインデックスにユーザによって入力されたテキストを含んでもよい。しかし、ポストバック処理は、他のイベントによって実行されるものでも良く、単にユーザとの対話によって実行されるものとは限らない。

【００２８】

処理２０８において、階層内の各サーバ側制御オブジェクトが、クライアント側ユーザインタフェース要素のウェブページでの表示のためのHTMLコードのようなオーサリング言語データを生成（またはレンダリング）するために呼び出される。用語「レンダリング」は、ユーザインタフェース上にグラフィックスを表示する処理を意味することがあるが、本明細書において、用語「レンダリング」は、表示およびクライアント側機能のためのブラウザのようなクライアントアプリケーションによって解釈され得るオーサリング言語データの生成処理をも意味している。処理２０６およびレンダリング処理２０８のより詳細な説明は図６との関連で行われている。ある実施形態において、個々の制御オブジェクトにおけるrender（）メソッドの呼び出しは、ツリートラバーサルシーケンスを用いて行われる。すなわち、ページオブジェクトのrender（）メソッドの呼び出しは、階層内の適切なサーバ側制御オブジェクトにわたる反復トラバースになる。適切な制御オブジェクトのrender（）メソッドを呼び出す別の方法として、イベントシグナリングまたはオブジェクト登録手法などの方法を用いてもよい。括弧は、データ値と比較するなどのメソッドを示す「render（）」レベルを指定する。

【００２９】

本発明のある実施形態において、個々のサーバ側制御オブジェクトの実際の作成は、サーバ側制御オブジェクトが処理２０６または２０８においてアクセスされる（ポストバック入力のハンドリング、状態のロード、制御オブジェクトからHTMLコードのレンダリングなど）まで遅らせてもよい。サーバ側制御オブジェクトが所与のリクエストのためにアクセスされることがない場合、制御オブジェクトの作成を遅らせ、不要な制御オブジェクト作成処理を排除することによって、サーバ処理が最適化される。

【００３０】

処理２１０において、HTMLコードをHTTPレスポンスでクライアントに送信する。処理２１４において、クライアントは、表示されるべき新たなウェブページに関連するHTMLコードを受信する。処理２１６において、クライアントシステムは、HTTPレスポンスから受け取ったHTMLコードに応じて新しいページのユーザインタフェース要素を組み入れる（例えば、表示する）。しかし、ユーザインタフェース要素の組み入れは、音声または触覚出力を

提供する、メモリへの読出しおよび書込み、スクリプト処理の制御などの非表示処理を含んでもよいことを理解すべきである。処理２１２において、サーバ側制御オブジェクト階層を終了する。本発明のある実施形態において、階層内サーバ側制御オブジェクトは、関連のASP+レスポンスを参照するHTTPリクエストに応答して作成され、オーサリング言語データ（例えば、HTMLデータ）のレンダリングが終わると破壊される。別の実施形態において、処理２１２は、処理２０８の後、処理２１０の前に行ってもよい。

【００３１】

図３は、本発明のある実施形態において用いるウェブサーバでのモジュールの一例を示す。ウェブサーバ３００は、HTTPパイプライン３０４にHTTPリクエスト３０２を受け入れる。HTTPパイプライン３０４は、ウェブページ統計のロギング、ユーザ照合、ユーザアクセス権およびウェブページの出力キャッシュ化用モジュールなどの種々のモジュールを含んでもよい。ウェブサーバ３００によって受信される各入力HTTPリクエスト３０２は、最終的には、IHTTPハンドラクラス（ハンドラ３０６として図示）の特定のインスタンスによって処理される。ハンドラ３０６は、URLリクエストを分解し適切なハンドラファクトリ（例えば、ページファクトリモジュール３０８）を呼び出す。

【００３２】

図３において、ASP+リソース３１０に関連付けられたページファクトリ３０８が呼び出され、ASP+リソース３１０のインスタンス化および構成をハンドリングする。ある実施形態において、ASP+リソースは、ファイルに特定の接尾語（または".aspx"のようなファイル拡張部分）を指定することによって認識され得る。所与の".aspx"資源に対するリクエストがまず、ページファクトリモジュール３０８によって受信されると、ページファクトリモジュール３０８は、ファイルシステムを検索して適切なファイル（例えば、.aspxファイル３１０）を得る。このファイルは、リクエストを処理するサーバによって後に解釈またはアクセスされ得るテキスト（例えば、オーサリング言語データ）または別のフォーマットでのデータ（例えば、バイトコードデータまたはコード化されたデータ）を含んでもよい。物理的なファイルが存在する場合、ページファクトリモジュール３０８は、ファイルを開き、そのファイルをメモリに読み出す。ファイルが見つからない場合、ページファクトリモジュール３０８は、適切な「ファイルが見つからない」というエラーメッセージを戻す。

【００３３】

ASP+リソース３１０をメモリに読み出した後、ページファクトリモジュール３０８は、ファイル内容を処理し、ページのデータモデル（例えば、スクリプトブロックのリスト、ディレクティブ、静的テキスト領域、階層サーバ側制御オブジェクト、サーバ側制御プロパティなど）を構築する。データモデルは、ページベースのクラスを拡張させるCOM+（Component Object Model+）クラスのような新たなオブジェクトクラスのソースリストを生成するのに用いられる。ページベースクラスは、基本ページオブジェクトの構造、プロパティおよび機能を規定するコードを含んでいる。本発明のある実施形態において、次に、ソースリストは、中間言語に動的にコンパイルされ、のちにプラットフォームに固有の命令（例えば、X86、Alphaなど）に適時に（Just-In-Time）コンパイルされる。中間言語は、COM+ IL コード、Java バイトコード、Modula 3 コード、SmallTalk コードおよびVisual Basicコードなどの汎用またはカスタム指向言語コードを含んでもよい。別の実施形態において、中間言語処理を省き、ネイティブ命令をソースリストまたはソースファイル（例えば、ASP+リソース３１０）から直接生成してもよい。制御クラスライブラリ３１２は、制御オブジェクト階層の生成に用いられる予め定義されたサーバ側制御クラスを得るためにページファクトリモジュール３０８によってアクセスされ得る。

【００３４】

ページファクトリモジュール３０８は、図１のウェブページ１０４に相当するサーバ側制御オブジェクトであるページオブジェクト３１４を出力する。ページオブジェクト３１４およびその子オブジェクト（例えば、テキストボックスオブジェクト３１８、ボタンオブジェクト３２０および別のボタンオブジェクト３２２）が、制御オブジェクト階層３１６

の一例である。他の制御オブジェクトの例は、本発明に従い考えることができ、カスタム制御オブジェクトと同様に、特に制限されるものではなく、表１のHTML制御に対応するオブジェクトも含んでいる。ページオブジェクト３１４は、図１のウェブページ１０４に対応する。テキストボックスオブジェクト３１８は、図１のテキストボックス１０６に対応する。同様に、ボタンオブジェクト３２０は、図１の追加ボタン１０８に対応し、ボタンオブジェクト３２２は、図１の削除ボタン１１０に対応する。ページオブジェクト３１４は、サーバ上の他の制御オブジェクトと階層的に関連している。ある実施形態において、ページオブジェクトは、その子制御オブジェクトを階層的に含んでいるコンテナオブジェクトである。別の実施形態では、依存関係などの他の形態の階層関係を用いることができる。多数レベルの子オブジェクトを有するより複雑な制御オブジェクト階層において、１つの子オブジェクトが、他の子オブジェクトのコンテナオブジェクトであってもよい。

【００３５】

上記の実施形態において、制御オブジェクト階層３１６の制御オブジェクトは、サーバ３００上で作成および実行され、各サーバ側制御オブジェクトは、クライアント上の対応するユーザインタフェース要素と論理的に対応する。サーバ側制御オブジェクトはまた、協働して、HTTPリクエスト３０２からのポストバック入力をハンドリングし、サーバ側制御オブジェクトの状態を管理し、サーバ側データベースとのデータ結合を行い、クライアントでの結果ウェブページの表示に用いるオーサリング言語データ（例えば、HTMLコード）を生成する。結果オーサリング言語データは、サーバ側制御オブジェクト階層３１６から生成（すなわち、レンダリング）され、HTTPレスポンス３２４でクライアントに送信される。例えば、結果HTMLコードは、いかなる有効なHTML構成も具体化することができ、ACTIVEXタイプの制御、JAVAアプレット、スクリプトおよびその他、ブラウザによって処理されたとき、クライアント側ユーザインタフェース要素（例えば、制御ボタン、テキストボックスなど）を生み出すいかなるウェブ資源も参照することができる。

【００３６】

ASP+リソース３１０でなされた宣言によって、サーバ側制御オブジェクトは、非ユーザインタフェースサーバコンポーネント３３０とクライアント側ユーザインタフェース要素との対話のために、１つ以上の非ユーザインタフェースサーバコンポーネント３３０にアクセスすることができる。例えば、ポストバック入力に応答して、サーバ側制御オブジェクトは、サーバ側イベントをそれらのイベント用に登録された非ユーザインタフェースサーバコンポーネントに対し立ち上げることができる。このように、非ユーザインタフェースサーバコンポーネント３３０は、ユーザとの対話を、ユーザインタフェース要素を介して、これらの要素を表示および処理するのに必要なコードをプログラミングすることなく行うことができる。

【００３７】

図４は、本発明のある実施形態における動的コンテンツ資源の一例の内容を示す。例示された実施例において、ファイル４００は、ある動的コンテンツ資源フォーマット（例えば、ASP+）でプレインテキスト宣言を含んでいる。各宣言は、ファイル４００を読み出すページコンパイラに対し命令を与え、クライアント側ユーザインタフェース要素を処理するために適切なサーバ側制御オブジェクトの作成および呼び出しを行い、最終的にはクライアントへHTTPレスポンスで送信するレンダリングされたHTMLコードを組み合わせる。そして、宣言は、サーバ側制御オブジェクトによって実行されるクライアント側ユーザインタフェース要素の機能を記述または参照する。そして、サーバ側制御オブジェクトは、クライアント上のウェブページの新たなバージョンを定義するために用いるHTMLコードを生成する。

【００３８】

ファイル４００の第１ラインは、以下のフォーマットのディレクティブを含む：

<%@ directive [attribute=value] %>

ここで、directiveは、特に制限されるものではなく、"page"、"cache"または"import"を含んでもよい。ディレクティブは、バッファリングセマンテックス、セッション状態要件

、エラーハンドリングスキーム、スクリプティング言語、トランザクションセマンテックスおよびインポートディレクティブのような特性を決定するために、動的コンテンツ資源を処理するときにページコンパイラによって用いられる。ディレクティブは、ページファイル内のどこに存在してもよい。

【００３９】

第２ラインの<html>は、リテラルとして結果HTMLコードへ渡される（すなわち、結果HTMLコードをレンダリングするために追加的な処理を行わない）標準HTML開始タグである。HTML構文において、HTMLファイルの始まりを示し、これもまたリテラルであるライン２１の終了タグ</html>と対になっている。

【００４０】

コード宣言ブロックは、ファイル４００のライン３～１０に存在する。一般に、サーバ側コード宣言ブロックは、ページオブジェクトおよび制御オブジェクトメンバ変数ならびにサーバ上で実行処理されるメソッドを定義する。以下のフォーマットにおいて：

```
<script runat = "server" [language = "language"][src = "externalfile"]>
.....................................
</script>
```

ここで、言語およびsrcパラメータは任意である。本発明のある実施形態において、コード宣言ブロックは、"サーバ"に設定された値を有する"runat"属性を含む<script>タグを用いて定義される。任意には、"language"属性を内コードのシンタックスを特定するために用いてもよい。デフォルト言語は、ページ全体の言語構成を表すことができるが、コード宣言ブロックの"language"属性により、開発者は、例えば、Jscript およびPERL (Practical Extraction and Report Language)などの同じウェブページ実行内で異なる言語を用いることができる。<script>タグはまた、任意には"src"ファイルを特定することができ、"src"ファイルは、そこからページコンパイラによる処理のための動的コンテンツ資源にコードが挿入される外部のファイルである。開示されている文法が本実施形態において用いられているが、別の実施形態では、本発明の範囲内で異なる文法を用いることができることを理解すべきである。

【００４１】

図４において、２つのサブルーチン、AddButton#ClickおよびDeleteButton#Clickがコード宣言ブロック内においてVisual Basicフォーマットで宣言されている。いずれのサブルーチンも２つの入力パラメータ、"Source"および"E"をとり、クライアント側クリックイベントが対応のボタンに検知されるとHTTPリクエストへのレスポンスでサーバ上で実行処理される。AddButton#Clickサブルーチンにおいて、ユーザ名テキストボックスでのテキストは、単語"Add"に連結され、メッセージのテキストデータメンバにロードされる。DeleteButton#Clickサブルーチンにおいて、ユーザ名テキストボックスでのテキストは、単語"Delete"に連結され、メッセージのテキストデータメンバにロードされる。図４に示していないが、サーバ側制御オブジェクトのメンバ変数は、ファイル４００のコード宣言ブロックで宣言され得る。例えば、Visual Basicシンタックスを用いると、キーワードスペース"DIM"は、サーバ側制御オブジェクトのデータ変数を宣言する。

【００４２】

「コードレンダリングブロック」（図示せず）もまた、動的コンテンツ資源に含まれ得る。本発明のある実施形態において、コードレンダリングブロックは、ページレンダリング時に実行処理される１つの「レンダリング」メソッドで実行処理する。コードレンダリングブロックは、以下のフォーマット（他のフォーマットも別の実施形態において考えられるが）を満たす。

【００４３】

```
<% InlineCode %>
```

ここで“InlineCode”は、レンダリング時にサーバ上で実行処理する独立言語型コードブロックまたは制御フローブロックを含んでいる。

【００４４】

インライン表現もまた、以下のような例示的なシンタックスを用いてコードレンダリングブロック内で用いることができる。

【００４５】

<%= InlineExpression %>

ここで、“InlineExpression”ブロックに含まれる表現は、“InlineExpression”からの値を宣言内の適切な場所のホールダに書込むページオブジェクトの"Response.Write(InlineExpression)"への呼び出しによって最終的に包含される。例えば、“InlineExpression”は、以下のようなコードレンダリングブロックに含まれ得る。

【００４６】

<font size = "<%=x%>" > Hi <%=Name%>, you are <%=Age%>!

これは、値"x"に格納されるフォントで挨拶およびある人の年齢についての記述を出力する。その人の名前および年齢は、コード宣言ブロック（図示せず）においてストリングとして定義される。結果HTMLコードは、適切な場所に“InlineExpression”の値を含むようにサーバでレンダリングされHTTPレスポンスでクライアントへ送信される。すなわち、以下のようなものである。

【００４７】

<font size = "12" > Hi Bob, you are 35!

ファイル４００のライン１１において、<body>は、HTMLドキュメントの本文の始まりを規定するための標準HTMLタグである。フィイル４００のライン２０において、終了タグ</body>もまた示されている。本発明のある実施形態において、<body>および</body>のいずれもリテラルである。

【００４８】

HTMLフォームブロックの開始タグ<form>が、図４のファイル４００の本文セクション内、ライン１２に見られる。フォームブロックの終了タグ</form>は、HTMLファイル４００のライン１９に見られる。任意のパラメータ"id"もまた、所与の識別子をフォームブロックに関連付けるためにHTML制御タグに含むことができ、これによって、１つのHTMLファイルに多数のフォームブロックが含まれることが可能になる。

【００４９】

ファイル４００のライン１８において、「メッセージ」によって識別されたサーバ側ラベルが宣言される。「メッセージ」ラベルは、ウェブページ上にラベルを表示するために、ファイル４００のライン５および８で宣言されたコードで用いられる。

【００５０】

フォームブロック内に、図１のユーザインタフェース要素１０６、１０８および１１０に対応する３つのHTML制御タグ例が示されている。第１のユーザインタフェース要素がテキストボックスに相当するファイル４００のライン１３で宣言される。テキストリテラル"User Name"は、テキストボックスの左側に位置するラベルを宣言する。type="Text"を有する入力タグは、テキストボックスクライアント側ユーザインタフェース要素をレンダリングするサーバ側制御オブジェクトとして"UserName"という識別子を有するテキストボックスサーバ側制御オブジェクトを宣言する。ファイル４００のライン１５および１６は、それぞれ、図１のボタン１０８および１１０として示されるクライアント側ユーザインタフェース要素を宣言する。"OnServerClick"パラメータは、ファイル４００のコード宣言ブロックで宣言された適切なサブルーチンを特定する。そして、ファイル４００での宣言へのレスポンスで生成されたサーバ側ボタン制御オブジェクトが、クライアント側ボタンのHTMLコードおよびボタンクリックイベントを実行する関連のサーバ側コードをレンダリングする。

【００５１】

ファイル４００で宣言されたテキストボックスおよびボタンは、HTMLサーバ制御宣言の例である。初期状態では、ASP+リソース内のすべてのHTMLタグはリテラルテキストコンテンツとして取り扱われ、ページ開発者のプログラミングにおいてアクセスすることができない。しかし、ページ開発者は、"server"に設定された値を有する"runat"属性を用いて指

定することによって、HTMLタグは構文解析され、アクセス可能サーバ制御宣言として扱われるべきであることを示すことができる。任意には、各サーバ側制御オブジェクトは、対応する制御オブジェクトのプログラム参照を可能にする一意の"id"属性と関連付けることができる。サーバ側制御オブジェクト上のプロパティ引数およびイベント結合もまた、タグ要素における属性対である宣言名／値を用いて特定することができる（例えば、OnServerClick は"MyButton#Click"対に等しい）。

【００５２】

本発明のある実施形態において、HTML制御オブジェクトを宣言する一般的なシンタックスは以下のとおりである。

【００５３】

```
<HTMLTag id = "Optional Name" runat = server>
.....
</HTMLTag>
```

ここで、"Optional Name"は、サーバ側制御オブジェクトの一意の識別子である。現在サポートされているHTMLタグのリストならびに関連シンタックスおよびCOM+クラスを表１に示すが、他のHTMLタグも本発明の範囲内で考えることができる。

【００５４】

【表１】

HTML タグ名	例	COM+ クラス
<a>	 My Link 	AnchorButton
<img>	<img id = “MyImage” runat = server>	Image
		Label
	Some contents	Panel
<form>	<form id = “MyForm” runat = server> </form>	FormControl
<select>	<select id = “MyList” runat = server> <option>One</option> <option>Two</option> <option>Three</option> </select>	DropDownList
<input type = file>	<input id = “MyFile” type = file runat = server>	FileInput
<input type = text>	<input id = “MyTextBox” type = text>	TextBox
<input type = password>	<input id = “MyPassword” type = password>	TextBox
<input type = reset>	<input id = “MyReset” type = reset>	Button
<input type = radio>	<input id = “MyRadioButton” type = radio runat = server>	RadioButton
<input type = checkbox>	<input id = “MyCheck” type = checkbox runat = server>	CheckBox
<input type = hidden>	<input id =”MyHidden” type = hidden runat = server>	HiddenField
<input type = image>	<input type = image src = “foo.jpg” runat = server>	ImageButton
<input type = submit>	<input type = submit runat = server>	Button
<input type = button>	<input type = button runat = server>	Button
<button>	<button id = MyButton runat = server>	Button
<textarea>	<textarea id = “MyText” runat = server> This is some sample text </textarea>	TextArea

【００５５】

標準HTML制御タグに加えて、本発明のある実施形態によって、開発者は、HTMLタグセット

の外側に共通のプログラム機能についてカプセル化する再利用可能コンポーネントを作成することが可能になる。これらのカスタムサーバ側制御オブジェクトは、ページファイル内の宣言タグを用いて特定される。カスタムサーバ側制御オブジェクト宣言は、"server"に設定された値を有する"runat"属性を含む。任意には、カスタム制御オブジェクトのプログラム参照を可能するために、一意の"id"属性が特定される。さらに、タブ要素の宣言名／値属性対は、サーバ側制御オブジェクトのプロパティ引数およびイベント結合を特定する。インラインテンプレートパラメータもまた、適切な「テンプレート」接頭辞子要素を親サーバ制御オブジェクトに与えることによってサーバ側制御オブジェクトに結合されることがある。カスタムサーバ側制御オブジェクト宣言のフォーマットは、次のようになる。

【００５６】

<servercntrlclassname id="OptionalName" [propertyname="propval"] runat=server/>

ここで、“servercntrlclassname”は、アクセス可能制御クラスであり、“OptionalName”は、サーバ側制御オブジェクトの一意の識別子であり、“propval”は、制御オブジェクトの任意のプロパティ値を表す。

【００５７】

別の宣言文法を用いて、XMLタグ接頭語は、以下のフォーマットを用いることにより、ページ内にサーバ側制御オブジェクトを特定するためのより簡潔な表記法を提供するのに用いることができる。

【００５８】

<tagprefix:classname id = "OptionalName" runat = server/>

ここで、“tagprefix”は、所与の制御名スペースライブラリと関連し、“classname”は、関連名スペースライブラリでの制御の名前を表す。任意の“propertyvalue”もまたサポートされている。

【００５９】

要するに、本発明の実施形態は、クライアントに送るHTMLコードを生成するためにサーバ上で作成および実行処理されるサーバ側制御オブジェクトを含む。HTMLコードは、いかなる有効なHTML構成も具現化することができ、例えば、ACTIVEXタイプの制御、JAVA アプレット、スクリプト、その他クライアントでのユーザインタフェースボタンおよび他のユーザインタフェース要素を生成するいかなるウェブ資源も参照することができる。クライアントでのユーザは、サーバ側制御オブジェクトに論理的に対応するこれらのユーザインタフェース要素と対話し、リクエストをサーバに送り返すことができる。サーバ側制御オブジェクトは、サーバ上で再作成され、クライアントにレスポンスとして送信すべき次のラウンドのHTMLコードを生成するように、ユーザインタフェース要素のデータ、イベントおよび他の特性を処理する。

【００６０】

図５を参照すると、本発明の実施形態のコンピュータシステムの一例は、プロセッサユニット５０２、システムメモリ５０４およびシステムメモリ５０４を含む種々のシステムコンポーネントをプロセッサユニット５００に接続するシステムバス５０６を含んでいる従来のコンピュータシステム５００という形態の汎用コンピュータ装置を含んでいる。システムバス５０６は、メモリバスまたはメモリコントローラ、種々のバスアーキテクチャを用いるペリフェラルバスおよびローカルバスを含む幾つかのタイプのバス構造のいずれであってもよい。システムメモリは、再生専用メモリ（ROM）５０８およびランダムアクセスメモリ（RAM）５１０を含んでいる。コンピュータシステム５００内の要素間での情報の転送を助ける基本ルーチンを含んでいる基本入力／出力システム５１２（BIOS）は、ROM５０８に格納されている。

【００６１】

コンピュータシステム５００は、さらに、ハードディスクの読み出しおよび書き込みを行うハードディスクドライブ５１２、着脱可能な磁気ディスク５１６の読出しおよび書込みを行う磁気ディスクドライブ５１４およびCD ROM、DVDまたは他の光学媒体のような着脱

可能な光ディスク５１９の読出しおよび書込みを行う光ディスクドライブ５１８を含んでいる。ハードディスクドライブ５１２、磁気ディスクドライブ５１４および光ディスクドライブ５１８は、それぞれ、ハードディスクドライブインタフェース５２０、磁気ディスクドライブインタフェース５２２および光ディスクドライブインタフェース５２４によってシステムバス５０６接続されている。ドライブおよびその関連コンピュータ読み取り可能媒体が、コンピュータシステム５００のコンピュータ読み取り可能命令、データ構造、プログラムおよび他のデータの不揮発性記憶部を提供している。

【００６２】

本明細書に記載の上記環境例では、ハードディスク、着脱可能な磁気ディスク５１６および着脱可能な光ディスク５１９を用いているが、データ保存可能な他のタイプのコンピュータ読み取り可能媒体を上記システム例に用いることができる。上記動作環境例に用いることができるこれらの他のタイプのコンピュータ読み取り可能媒体は、例えば、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、ベルヌイ(Bernoulli)カートリッジ、ランダムアクセスメモリ（RAM）および再生専用メモリ（ROM）などがある。

【００６３】

多数のプログラムモジュールが、ハードディスク、磁気ディスク５１６、光ディスク５１９、ROM５０８またはRAM５１０に格納され、これらは、オペレーティングシステム５２６、１つ以上のアプリケーションプログラム５２８、他のプログラムモジュール５３０およびプログラムデータ５３２を含む。ユーザは、コマンドおよび情報をコンピュータシステム５００にキーボード５３４およびマウス５３６または他のポインティング装置などの入力装置によって入力することができる。他の入力装置としては、例えば、マイクロフォン、ジョイステック、ゲームパッド、サテライトディッシュおよびスキャナなどがある。これらおよび他の入力装置は、システムバス５０６に接続されているシリアルポートインタフェース５４０を介して処理装置５０２に接続されていることが多い。しかし、これらの入力装置はまた、パラレルポート、ゲームポートまたはユニバーサルシリアルバス（USB）などの他のインタフェースによって接続されていてもよい。モニタ５４２または他のタイプの表示装置もまた、ビデオアダプタ５４４などのインタフェースを介してシステムバス５０６と接続している。モニタ５４２に加えて、コンピュータシステムは、典型的には、スピーカおよびプリンタなどの他の周辺出力装置（図示せず）を含む。

【００６４】

コンピュータシステム５００は、リモートコンピュータ５４６のような１つ以上のリモートコンピュータへの論理接続を用いたネットワーク化された環境で動作することができる。リモートコンピュータ５４６は、コンピュータシステム、サーバ、ルータ、ネットワークPC、ピア（peer）装置、または他の共通ネットワークノードであり得、典型的にコンピュータシステム５００との関連で上述した要素の多くまたはすべてを含む。ネットワーク接続は、ローカルエリアネットワーク（LAN）５４８およびワイドエリアネットワーク（WAN）５５０を含む。このようなネットワーク環境は、オフィス、企業規模コンピュータネットワーク、イントラネットおよびインターネットにおいて珍しいものではない。

【００６５】

LANネットワーク環境で用いるとき、コンピュータシステム５００は、ネットワークインタフェースまたはアダプタ５５２を介してローカルネットワーク５４８に接続される。WANネットワーク環境で用いるとき、コンピュータシステム５００は、典型的に、インターネットのようなワイドエリアネットワーク５５０による通信を確立するためのモデム５５４または他の手段を含む。モデム５５４は内蔵または外付けのいずれでもよく、シリアルポートインタフェース５４０を介してシステムバス５０６と接続されている。ネットワーク化された環境において、コンピュータシステム５００に関連して述べたプログラムモジュールまたはその一部は、リモートメモリ記憶装置に記憶されてもよい。図示されたネットワーク接続は例であって、コンピュータ間の通信リンク確立のために他の手段を用いることができる。

【００６６】

本発明の実施形態において、コンピュータ５００は、ウェブサーバを表し、CPU５０２が、記憶媒体５１６、５１２、５１４、５１８、５１９またはメモリ５０４のうち少なくとも１つに記憶されたASP+リソース上でページファクトリモジュールを実行処理する。HTTPレスポンスおよびリクエストは、クライアントコンピュータ５４６に接続されたLAN５４８により通信される。

【００６７】

図６は、本発明のある実施形態におけるページオブジェクトおよび他の制御オブジェクトのサーバ側処理を表すプロセスフローチャートである。処理６００において、ページオブジェクトコンストラクタが、ページファクトリモジュール３０８によって呼び出される（図３参照）。その結果、ページオブジェクト（例えば、図３におけるページオブジェクト３１４を参照）は、クライアント上のウェブページユーザインタフェース要素と論理的に対応するように作成される。処理６０２において、ページファクトリモジュールは、クライアントから受け取ったHTTPリクエストの段階的な処理を始めるページオブジェクトのProcessRequest（）メンバ関数を呼び出す。本発明の一実施形態の第１の段階において、サーバ側作成処理(図示せず)は、ページオブジェクトの制御オブジェクト階層に含まれる子孫サーバ側制御オブジェクトの作成である。すなわち、子制御オブジェクトのコンストラクタがHTTPリクエスト処理の処理生存期間の間制御オブジェクトを作成するために繰り返し呼び出される。

【００６８】

しかし、別の実施形態において、子制御オブジェクトの作成は、制御オブジェクトが所与の処理ステップ（例えば、ポストバックイベントのハンドリング、ポストバックデータのハンドリング、ビューステートのローディングおよび保存、データ結合の分解または対応するユーザインタフェース要素のHTMLコードのレンダリング）に必要とされるまで遅らせることができる。「制御オブジェクト作成を遅らせる」後者の実施形態は、不必要なCPUおよびメモリの利用を減らすことができるので最適である。例えば、クライアントから受け取ったユーザ入力イベントが、全く異なるウェブページの作成ということになる場合がある。この場合、直ちに制御オブジェクト階層の終了させ、新たなページの新たな異なる制御オブジェクト階層をインスタンス化することになるイベントを処理するためだけに、以前のページの制御オブジェクト階層全体をインスタンス化する必要はない。

【００６９】

ページオブジェクトのProcessRequestメソッドのサーバ呼び出しに応答して、処理６０４～６２０は、所与のHTTPリクエストのデータなどに応じ、データページオブジェクトおよび個々の子孫制御オブジェクトによって実行処理することができる。本発明のある実施形態において処理６０４～６２０を図６の順序で各個々のオブジェクトに対し行う。しかし、１つのオブジェクトに対する所与の処理は、HTTPリクエストによっては、別のオブジェクトの所与の処理に関して順番に行われなかったり、全く処理が行われないこともある。例えば、第１のオブジェクトは、初期化処理６０４およびそのロード処理６０６を行い、ポストバックデータ処理６０８を開始し、その後、遅れた制御オブジェクト作成により子孫制御オブジェクトが、それ自体の初期化処理６０４およびロード処理６０６を行う。ページオブジェクトおよび子孫制御オブジェクトによる処理の順番は、これには限らないが、HTTPリクエストにおけるデータの性質、制御オブジェクト階層の構成、制御オブジェクトの現在の状態および制御オブジェクト作成が遅れて行われるかどうかを含む種々の要因による。

【００７０】

初期化処理６０４は、動的コンテンツ資源における初期化に関する任意のサーバ側コードを実行処理することによって制御オブジェクトが作成された後、制御オブジェクトを初期化する。このようにして、各サーバ側制御オブジェクトは、動的コンテンツ資源で宣言された特定のサーバ側機能でカスタマイズされ得る。本発明の実施形態において、動的コンテンツコードは、サーバ上のASP+リソースでページ開発者によって宣言されたベースページ制御クラスをカスタマイズまたは拡張するものである。ASP+リソースがコンパイルされ

ると、宣言されたコードは、適切な初期コード（例えば、ページオブジェクトおよび子孫制御オブジェクトの初期化（）メソッド）に含まれている。初期化処理６０４は、このコードを実行処理して、ページベースクラスおよび子孫制御オブジェクトのベースクラスをカスタマイズまたは拡張する。

【００７１】

本発明のある実施形態において、サーバ側制御オブジェクトの状態管理は、サーバ側制御オブジェクトを以前の状態に戻すことによってクライアント・サーバシステムの無状態モデルを収納するために搬送可能状態構造を用いるロード処理６０６および保存処理６１６においてサポートされる。ある実施形態では、状態は、一対のHTTPリクエスト／レスポンスの隠れた１つ以上のHTMLフィールドでサーバを往復するが、他の搬送可能状態構造も本発明の範囲内であると考えられる。

【００７２】

クライアントとサーバ間の現在のページに関する所与のリクエストおよびレスポンスによる一連の処理において、１つ以上の制御オブジェクトの状態は、先のリクエストの処理後に保存処理６１６によって搬送可能状態構造に記録される。本発明のある実施形態において、階層情報またはサーバが適切な制御オブジェクトと所与の状態とを関連付けることを可能にする制御オブジェクト識別子を含む追加の状態情報もまた、搬送可能状態構造に含まれる。次のHTTPリクエストにおいて、状態情報は、搬送可能状態構造でサーバに戻される。サーバは、受信した搬送可能状態構造から状態情報を抽出し、状態データを制御オブジェクト階層内の適切な制御オブジェクトにロードし、各制御オブジェクトを先のHTTPレスポンス以前に存在したような状態に戻す。現在のリクエストに対する処理後、１つ以上のサーバ側制御オブジェクトの状態は、再度保存処理６１６によって搬送可能状態構造に記録され、次のHTTPレスポンスで搬送可能状態構造をクライアントに戻す。

【００７３】

ロード処理６０６の結果、各サーバ側制御オブジェクトを先のHTTPリクエスト以前の状態と同じ状態にする。例えば、テキストボックス制御オブジェクトが、先のHTTPレスポンス以前に"JDoe"と同等のプロパティ値を含む場合、テキストストリング"JDoe"をそのプロパティ値にロードすることなどによって、ロード処理６０６は、同じ制御オブジェクトを先の状態に戻す。さらに、所与のオブジェクトの状態が記憶され回復されたかどうかについても構成可能である。

【００７４】

本発明の一実施形態を要約すると、１つ以上のサーバ側制御オブジェクトの状態が処理後「保存」される。保存された状態情報は、レスポンスによりクライアントに送信される。クライアントは、次のレスポンスで保存された状態情報をサーバ側に戻す。サーバは、階層の状態が以前の状態に戻るように、新たにインスタンス化されたサーバ側制御オブジェクト階層に状態情報をロードする。

【００７５】

別の実施形態では、サーバ上、またはサーバからクライアント、そしてサーバへ戻るという往復の間、サーバによってアクセス可能な幾つかの他のウェブロケーションに状態情報を保持できる。クライアントリクエストをサーバが受信した後、この状態情報は、サーバによって取り出され、制御オブジェクト階層内の適切なサーバ側制御オブジェクトにロードされ得る。

【００７６】

処理６０８において、HTTPリクエストから受け取ったポストバックデータが処理される。ポストバックデータは、対になったキー値、階層表示（例えば、XML）またはRDF（Resource Description Framework）のような他のデータ表示でのHTTPリクエストのペイロードに含まれ得る。処理６０８は、ペイロードを構文解析してサーバ側制御オブジェクトの一意の識別子を識別する。識別子（例えば、"page1:text1"）を見つけ、識別されたサーバ側制御オブジェクトが制御オブジェクト階層に存在する場合、対応するポストバックデータがその制御オブジェクトへと渡される。例えば、図１を参照すると、テキストボックス１

０６およびテキスト"JDoe"に関連する一意の識別子は、HTTPリクエスト１１４のペイロードでウェブサーバ１１６へ送られる。処理６０８はHTTPリクエスト１１４のペイロードを構文解析し、テキストボックス１０６の一意の識別子とその関連値（すなわち"JDoe"）を得る。それから処理６０８は、テキストボックス１０６の一意の識別子を分解し、対応するサーバ側制御オブジェクトを識別し、処理するオブジェクトに"JDoe"値を渡す。

【００７７】

ロード処理６０６に関して説明したように、サーバ側制御オブジェクトのプロパティ値は、以前の状態に戻され得る。ポストバックデータを受け取ると、サーバ側制御オブジェクトは、渡されたポストバック値が対応する先のプロパティ値を変化させたかどうかを判断する。変化した場合には、関連制御オブジェクトのデータ変化を示す変化リストにその変化をロギングする。すべてのポストバックデータが制御オブジェクト階層内で処理された後、制御オブジェクトメソッドが呼び出され、サーバ上で起動している株価検索アプリケーションのような１つ以上の非ユーザインタフェースサーバコンポーネントに対し、１つ以上のポストデータ変化イベントを提起し得る。ポストデータ変化イベントは、例えば、ポストバックデータがサーバ側制御オブジェクトのプロパティを変化させたということを示しているイベントである。例示的な実施形態において、このようなイベントは、システム提供イベント待ち行列に送られ、イベントを処理するよう登録されたサーバ側コードを呼び出すことができる。そして、サーバ側コードは、非ユーザインタフェースサーバコンポーネントのメソッドを呼び出し得る。このように、サーバ側非ユーザインタフェースサーバコンポーネントは、サーバ側制御オブジェクトのデータの変化によってトリガされたイベントに応答することができる。アプリケーション提供イベント待ち行列、ポーリング、および処理割込みを用いることを含む、イベントを実行する別の方法もまた本発明の範囲内で考えることができる。

【００７８】

処理６１０において、ポストバックイベントをハンドリングする。ポストバックイベントは、HTTPリクエストのペイロードで通信され得る。処理６１０は、そのイベントが向けられているサーバ側制御オブジェクトを識別する特定のイベントターゲット（例えば、本発明のある実施形態では、"##EVENTTARGET"とラベル付けされている）を構文解析する。さらに、処理６１０は、探し当てたイベント引数がある場合はそれを構文解析し、イベント引数（例えば、本発明のある実施形態では、"##EVENTARGUMENT"とラベル付けされている）を特定されたサーバ側制御オブジェクトに与える。制御オブジェクトは、動的コンテンツ資源に関連する非ユーザインタフェースサーバコンポーネント（例えば、サーバ側株価検索アプリケーション）のメソッドを呼び出すサーバ側コードによって処理するイベントを立ち上げる。

【００７９】

処理６１２は、サーバ側制御オブジェクトとサーバがアクセス可能な１つ以上のデータベースとの間のデータ結合関係を解明し、これにより、データベース値で制御オブジェクトプロパティにおいて更新し、かつ／または制御オブジェクトプロパティの値でデータベースフィールドを更新する。本発明のある実施形態では、サーバ側制御オブジェクトのプロパティは、サーバ側アプリケーションデータベースの表のような親データ結合コンテナのプロパティと関連付けられ（またはデータ結合され）得る。データ結合処理６１２の間に、ページフレームワークは、対応する親データ結合コンテナプロパティの値を有するデータ結合された制御オブジェクトプロパティを更新することができる。このように、次のレスポンスにおけるウェブページ上のユーザインタフェース要素は、更新されたプロパティ値を正確に反映する。なぜなら、ユーザインタフェース要素が対応する制御オブジェクトプロパティは、データ結合処理６１２の間に、自動的に更新されたからである。同様に、制御オブジェクトプロパティはまた、親データ結合コンテナフィールドに更新され得、これにより、サーバ側制御オブジェクトからのポストバック入力でサーバ側アプリケーションデータベースを更新する。

【００８０】

処理６１４は、制御オブジェクト状態が保存され出力がレンダリングされる以前に実行処理され得る多数の更新処理を行う。処理６１６は、制御オブジェクト階層内の１つ以上の制御オブジェクトから状態情報（すなわち、ビューステート）を要求し、状態情報を格納し、HTTPレスポンスペイロードでクライアントへ送られる搬送可能状態構造に挿入する。例えば、“grid”制御オブジェクトは、値のリストの現在のインデックスページを保存し、“grid”制御オブジェクトは、次のHTTPリクエスト（すなわち、処理６０６）の後この状態に戻ることができる。上記のように、ビューステート情報は、クライアントによる次のアクション以前の制御オブジェクト階層の状態を表す。ビューステート情報が戻ってくると、それは、制御オブジェクト階層を、クライアントポストバック入力処理またはデータ結合以前の先の状態にするのに用いられる。

【００８１】

レンダリング処理６１８は、HTTPレスポンスでクライアントへ送る適切なオーサリング言語出力（例えば、HTMLデータ）を生成する。レンダリングは、すべてのサーバ側制御オブジェクトのトップダウン階層ツリーウォークおよび埋め込まれたレンダリングコードにより行われる。処理６２０は、任意の最終のクリーンアップ作業（例えば、ファイルを閉じたりデータベースの接続）を行い、制御オブジェクト階層を終了させる。次いで、処理は６０２に戻り、処理６２２を行い、そこでページオブジェクトは、そのデストラクタを呼び出すことによって終了する。

【００８２】

図７は、本発明のある実施形態におけるサーバ側制御クラスの一例の表記を表している。サーバ側制御クラスは、本発明のある実施形態におけるすべてのサーバ側制御オブジェクトに共通のメソッド、プロパティおよびイベントを定義している。より具体的な制御クラス（例えば、ウェブページでのクライアント側ボタンに対応するサーバ側ボタン制御オブジェクト）は、この制御クラスから派生している。示された制御クラス７００は、プロパティ７０２およびメソッド７０４を格納するメモリを含む。データメンバとメソッドとの組み合わせが異なる他の制御クラスの実施形態もまた、本発明の範囲内で考えられる。

【００８３】

示された実施形態において、プロパティ７０２はパブリックである。プロパティ"ID"は、制御オブジェクト識別子を示している読み取りおよび書き込み可能なストリング値である。プロパティ"Visible"は、対応しているクライアント側ユーザインタフェース要素のオーサリング言語データをレンダリングすべきかどうかを示す読み取りおよび書き込み可能なブーリアン値である。プロパティ"MaintainState"は、制御オブジェクトが、現在のページリクエストの終了時に（すなわち、図６の保存処理６１６へのレスポンスで）そのビューステート（およびその子オブジェクトのビューステート）を保存すべきかどうかを示している読み取りおよび書き込み可能なブーリアン値である。プロパティ"Parent"は、制御オブジェクト階層の現在の制御オブジェクトを関連する制御コンテナ（図８参照）への読み取り可能なリファレンスである。プロパティ"Page"は、現在の制御オブジェクトがホストされているルートページオブジェクトへの読み取り可能なリファレンスである。プロパティ"Form"は、現在の制御オブジェクトがホストされているフォーム制御オブジェクトへの読み取り可能なリファレンスである。プロパティ"Trace"は、開発者のトレースログの書き込みを可能にする読み取り可能なリファレンスである。プロパティ“ BindingContainer ”は、制御オブジェクトの直接データ結合コンテナへの読み取り可能なリファレンスである。プロパティ"Bindings"は、制御オブジェクトデータ結合連関の集合への読み取り可能なリファレンスである。

【００８４】

メソッド７０４は、リクエストの処理および制御オブジェクトのデータメンバへのアクセス方法を含む。ある実施形態において、メソッドは、制御オブジェクトのメモリスペースに格納されるポインタによって参照される。この参照手段ははまた、コンテナ制御オブジェクト、制御コレクションオブジェクト、およびページオブジェクトを含む他のサーバ側オブジェクトの実施形態において用いられる。メソッド"Init()"は、子制御オブジェクト

が作成された後、これらを初期化するのに用いられる（図６の処理６０４を参照）。メソッド"Load()"は、先のHTTPリクエストからビューステート情報を回復するのに用いられる（図６の処理６０６を参照）。メソッド"Save()"は、後のHTTPリクエストで用いるビューステート情報を保存するのに用いられる（図６の処理６１６を参照）。メソッド"PreRender()"は、ビューステートの保存およびコンテンツのレンダリングに先立って必要なプレレンダリングステップを行うのに用いられる（図６の処理６１４を参照）。メソッド"Render(TextWriter output)"は、現在の制御オブジェクトに対応するユーザインタフェース要素のオーサリング言語コードを出力するのに用いられる（図６の処理６１８を参照）。コードは、クライアントへのレスポンスでコードを格納するために出力ストリームを通してやり取りされる（「出力」パラメータでそれを渡す）。メソッド"Dispose()"は、制御オブジェクトを終了する前に最終的なクリーンアップ作業を行うのに用いられる（図６の処理６２０を参照）。

【００８５】

メソッド"GetUniqueID()"は、制御オブジェクトの一意の、階層的に認定されたストリング識別子を得る。メソッド"GetControlWithID(String id)"は、与えられた識別子("id")を有する直接の子制御オブジェクトへのリファレンスを戻す。メソッド"GetControlUniqueWithID(String id)"は、一意の階層識別子("id")を有する子制御オブジェクトへのリファレンスを戻す。

【００８６】

メソッド"PushControlPropertyTwoBindingContainer (String prop Name)"は、ポストバックデータ値がサーバ側制御オブジェクト内で変化するとポストバックデータからの２方向データ結合の結合コンテナを更新するのに用いられる。メソッド"PushBindingContainerPropertyTwoControl(String prop Name)"は、現在の結合コンテナ値をもつサーバ側制御オブジェクトプロパティを更新するのに用いられる。メソッド"HasBindings()"は、制御オブジェクトが結合連関を有しているかどうかを示しているブーリアン値を戻す。これらの３つの機能は、サーバ側制御オブジェクトのプロパティとサーバ側データ格納部における属性との結合関係を解明するのに用いられる（図６のデータ結合処理６１２を参照）。

【００８７】

図８は、本発明のある実施形態におけるサーバ側コンテナ制御クラスの一例を示す。コンテナ制御クラスは、入れ子の子制御オブジェクトをサポートし、搬送可能状態構造へビューステート情報を自動的に直列化および非直列化する構成を提供する。コンテナ制御クラス８００は、プロパティ８０２およびメソッド８０４を格納するメモリを含んでいる。プロパティ"Controls"は、制御オブジェクト階層内の子制御オブジェクトの"ControlCollection"への読み取り可能なリファレンスである（図９を参照）。プロパティ"StateCollection"は、多数のページリクエストにわたって制御オブジェクトの状態を維持するのに用いられるビューステート情報の辞書への読み取り可能なリファレンスである。メソッド"HasControls()"は、制御オブジェクトが子制御オブジェクトを有しているかどうかを示しているブーリアン値を戻す。

【００８８】

別の実施形態において、コンテナ制御クラス８００のメンバプロパティおよびメソッドは、各制御オブジェクトがそれ自体、子オブジェクトをサポートできるように図７の制御クラス７００に組み込まれてもよい。しかし、このような実施形態の制御オブジェクトが、このような子オブジェクトを含まないなら、（制御コレクションタイプの）制御メンバは空っぽであるだろう（すなわち、子オブジェクトを含まない）。

【００８９】

図９は、本発明の実施形態におけるサーバ側制御コレクションクラスの一例を示す。制御コレクションクラス９００は、プロパティ９０２およびメソッド９０４を格納するメモリを含んでいる。プロパティ"All"は、インデックスによって命令された現在の制御オブジェクトのすべての子制御オブジェクトの読み取りおよび書き込み可能なスナップショットアレイである。プロパティ"this[index]"は、制御コレクション内の順序インデックスの

制御オブジェクトへの読み取り可能なリファレンスである。プロパティ"Count"は、コレクション内の子制御オブジェクトの数を示している読み取り可能値である。

【００９０】

メソッド"Add'Control child)"は、現在のコレクションに特定の制御オブジェクトを追加するのに用いられる。メソッド"IndexOf(Control child)"は、コレクション内の特定された子制御オブジェクトの順序インデックスを戻す。メソッド"GetEnumerator(boolAllowRemove)"は、コレクション内のすべての子制御オブジェクトのエニュメレータを戻す。メソッド"Remove(Control value)"は、現在のコレクションから特定の制御オブジェクトを取り除く。メソッド"Remove(int index)"は、与えられたインデックスに基づき現在のコレクションから特定の制御オブジェクトを取り除く。メソッド"Clear()"は、現在のコレクションからすべての制御オブジェクトを取り除く。

【００９１】

図１０は、本発明のある実施形態におけるサーバ側ページオブジェクトの一例を示す。ページベースクラスは、本発明の実施形態内でのすべてのサーバ側操作ページに共通のメソッド、プロパティおよびイベントを定義する。ページオブジェクト１０００は、プロパティ１００２およびメソッド１００４を格納しているメモリを含んでいる。プロパティ"ErrorPage"は、未処理ページ例外のイベントでレンダリングされる読み取りおよび書き込み可能なストリングである。このように、ページオブジェクトは、HTTPレスポンスでクライアントに読み取り可能なエラーページを戻す。プロパティ"Requests"は、HTTPRequestへの読み取り可能なリファレンスであり、HTTPRequestは、入ってくるHTTPリクエストデータにアクセスするための機能性を与えるウェブサーバフレームワークによって与えられる。

【００９２】

プロパティ"Response"は、クライアントへHTTPレスポンスデータを送信するための機能を提供するウェブサーバフレームワークによって与えられるHTTPレスポンスへの読み取り可能なリファレンスである。プロパティ"Application"は、ウェブサーバフレームワークによって与えられるHTTPApplicationへの読み取り可能なリファレンスである。プロパティ"Session"は、ウェブサーバフレームワークによって与えられるHTTPSessionへの読み取り可能なリファレンスである。プロパティ"Server"は、アプリケーションサーバページ互換性ユティリティオブジェクトであるサーバオブジェクトへの読み取り可能なリファレンスである。プロパティ"Context"は、ウェブサーバフレームワークによって与えられるウェブサーバオブジェクトのすべてへの読み取り可能なリファレンスであり、これにより、開発者は、さらなるパイプラインモジュール露出オブジェクトへのアクセスが可能になる。プロパティ"IsFirstLoad"は、ページオブジェクトが、初めて、またはクライアントポストバックリクエストへのレスポンスでロードおよびアクセスされているかどうかを示す読み取り可能なブーリアン値である。

【００９３】

プロパティ"Load()"は、ページオブジェクトを初期化し、先のページリクエストからビューステート情報を回復するのに用いられる。プロパティ"Save()"は、後のページリクエストに用いるビューステート情報を保存するのに用いられる。プロパティ"HandleError（Exception errorInfo）"は、ページ実行処理の間に起こる未処理エラーを扱うのに用いられる。このイベントにおいて、ベースクラス実行は、クライアントをデフォルトエラーウェブページを有するURLへ向けなおす。メソッド"GetPostbackScript(Control target, String name, String arg)"は、所与の制御オブジェクトと関連するクライアント側スクリプトメソッドを戻す。メソッド"RegisterClientScriptBlock(String key, String script)"は、クライアントへ送信されるクライアント側スクリプトコードの重複ブロックを排除するのに用いられる。重複ブロックは同じキー値を有するスクリプトである。メソッド"IHTTPHandler.ProcessRequest(HTTPContext Context)"は、ウェブリクエストを処理するのに用いられる。IHTTPHandler.ProcessRequestは、クライアントから受信されたHTTPリクエストの処理を初期化するために呼び出される（図６の処理６０２を参照）。メソッド"IHT

TPHandler.IsReusable()"は、ページオブジェクトが、多数のウェブリクエストを処理するのに再利用できるかどうかを示している。

【００９４】

本明細書に記載の本発明の実施形態は、１つ以上のコンピュータシステムの論理ステップとして実行される。本発明の論理処理は、（１）１つ以上のコンピュータシステムで実行処理されるプロセッサ実行ステップシーケンスとして、（２）１つ以上のコンピュータシステム内の相互接続された機械モジュールとして実行される。実行は選択の問題であり、本発明を実行するコンピュータシステムの性能要件に依存する。したがって、本明細書に記載の本発明の実施形態を構成する論理処理は、処理、ステップ、オブジェクトまたはモジュールと様々に表現することができる。

【００９５】

上記の明細書、実施例およびデータは、本発明の実施形態の構造および使用の完全な説明を提供している。本発明は、本発明の精神および範囲から逸脱することなく多数の形態で実施することができるので、本発明は添付の請求項にある。

【００９６】

【発明の効果】

以上のように本発明によれば、クライアント側ユーザインタフェース要素の処理および生成を管理するサーバ側制御オブジェクトフレームワークを提供することができることから、クライアント側ユーザインタフェース要素および関連機能に関するサーバ側処理のカプセル化を図ることができ、開発者が最小のプログラミングでウェブページを動的に作成および処理することができる開発フレームワークを提供することが可能となる。

【図面の簡単な説明】

【図１】　本発明のある実施形態におけるクライアントに表示するウェブページコンテンツを動的に生成するウェブサーバを示す。

【図２】　本発明のある実施形態におけるサーバ側制御オブジェクトを用いてクライアント側ユーザインタフェース要素の処理およびレンダリングのための処理のフローチャートを示す。

【図３】　本発明のある実施形態で用いるウェブサーバのモジュールの一例を示す。

【図４】　本発明のある実施形態における動的コンテンツリソース（例えば、ASP+リソース）の一例を示す。

【図５】　本発明のある実施形態の実行に有用なシステムの一例を示す。

【図６】　本発明のある実施形態におけるページオブジェクトの処理を表すフローチャートを示す。

【図７】　本発明のある実施形態におけるサーバ側制御クラスの一例を示す。

【図８】　本発明のある実施形態におけるサーバ側コンテナ制御クラスの一例を示す。

【図９】　本発明のある実施形態におけるサーバ側制御コレクションクラスの一例を示す。

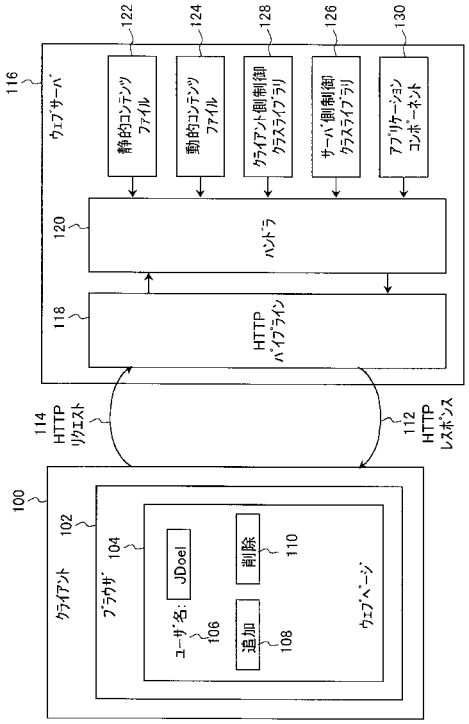
【図１０】　本発明のある実施形態におけるサーバ側ページクラスの一例を示す。

【符号の説明】

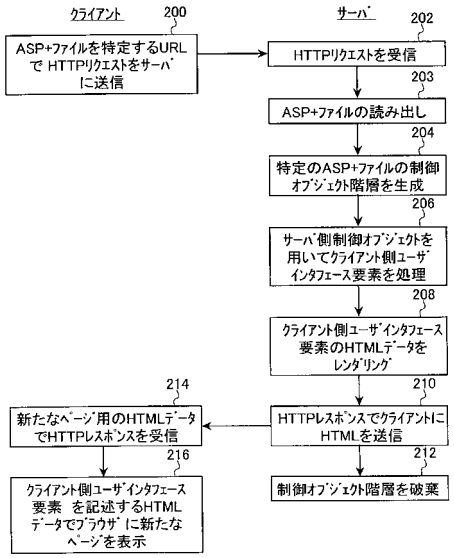
１００　クライアント
１０２　ブラウザ
１０４　ウェブページ
１０６　ユーザ名
１０８　追加
１１０　削除
１１２　HTTPレスポンス
１１４　HTTPリクエスト
１１６、３００　ウェブサーバ
１１８、３０４　HTTPパイプライン
１２０　ハンドラ

１２２　静的コンテンツファイル
１２４　動的コンテンツファイル
１２６　サーバ側制御クラスライブラリ
１２８　クライアント側制御クラスライブラリ
１３０、３３０　アプリケーションコンポーネント
３０８　ページファクトリ
３１０　ASP+ファイル
３１２　制御クラスライブラリ
３１４　ページオブジェクト
３１８　テキストボックスオブジェクト
３２０、３２２　ボタンオブジェクト
５００　コンピュータ
５０４　メモリ
５１２　ハードディスクドライブ
５１４　磁気ディスクドライブ
５１６　着脱可能記憶媒体
５１８　光ディスクドライブ
５１９　光ディスク
５２０、５２２、５２４　Ｉ／Ｆ
５２６　オペレーティングシステム
５２８　アプリケーションプログラム
５３０　プログラムモジュール
５３２　プログラムデータ
５３４　キーボード
５３６　マウス
５４０　シリアルポートインタフェース
５４２　モニタ
５４６　リモートコンピュータ
５５２　ネットワークアダプタ
５５４　モデム
７００　制御オブジェクト
７０２、８０２、９０２、１００２　プロパティ
７０４、８０４、９０４、１００４　メソッド
８００　コンテナ制御オブジェクト
９００　制御コレクションオブジェクト
１０００　ページオブジェクト

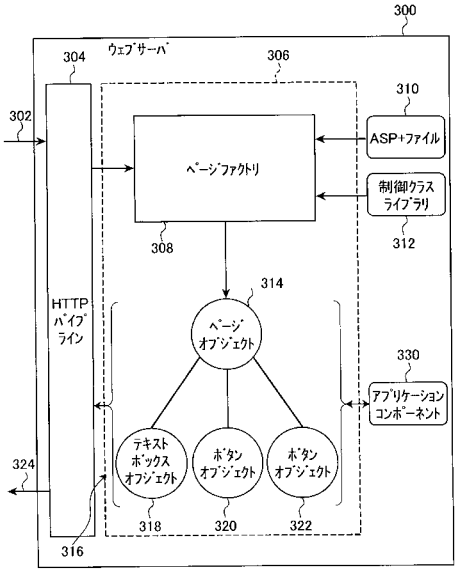
【図１】



【図２】



【図３】

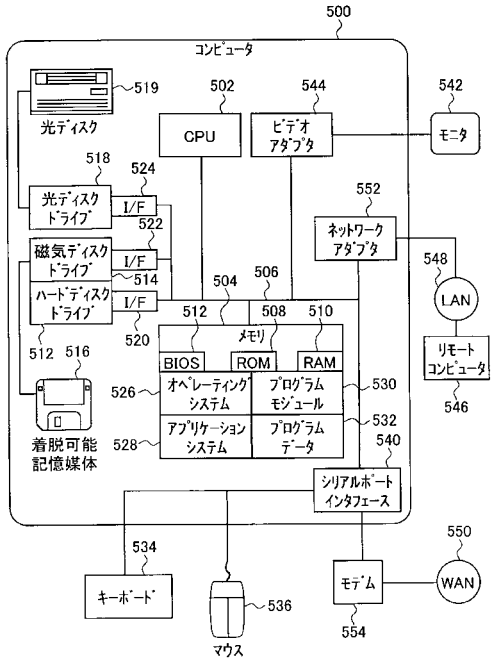


【図４】

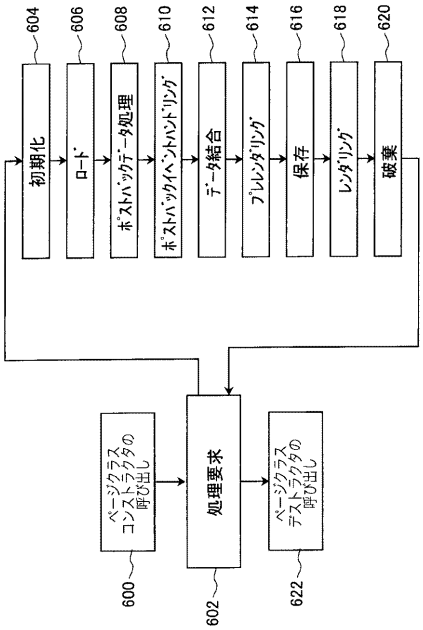
```
1  <%@ Page Language="VB" Description="Simple Sample Page" Errorpage="ErrorPage.aspx" %>
2  <html>
3      <script runat=server>
4          Sub AddButton_Click(ByVal Source as Object, By Val E as Event Args)
5              Message.Text = "Add" & UserName.Text
6          End Sub
7          Sub DeleteButton_Click(ByVal Source as Object, By Val E as Event Args)
8              Message.Text = "Delete" & UserName.Text
9          End Sub
10     </script>
11     <body>
12         <form runat="server">
13             User Name:     <input type="Text" id="UserName" runat=server>
14             <br>
15             <button id="AddButton" value="ADD" OnClick="AddButton_Click" runat=server>
16             <button id="DeleteButton" value="DELETE" OnClick="DeleteButton_Click" runat=server>
17             <br><br>
18             <span id="Message" runat=server> </span>
19         </form>
20     </body>
21 </html>
```

400

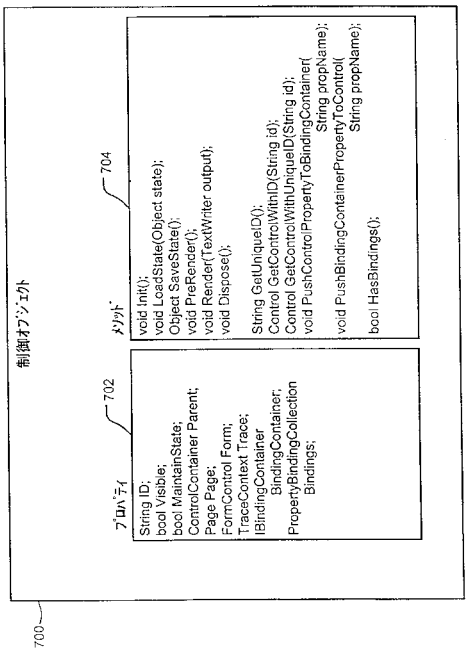
【図５】


500
コンピュータ
519
光ディスク
502
CPU
544
ビデオアダプタ
542
モニタ
518
光ディスクドライブ
524
I/F
552
ネットワークアダプタ
522
磁気ディスクドライブ
514
548
LAN
504
506
ハードディスクドライブ
520
512
508
510
メモリ
BIOS
ROM
RAM
リモートコンピュータ
546
512
516
着脱可能記憶媒体
526
オペレーティングシステム
530
プログラムモジュール
528
アプリケーションシステム
532
プログラムデータ
540
シリアルポートインタフェース
534
キーボード
536
マウス
モデム
554
550
WAN

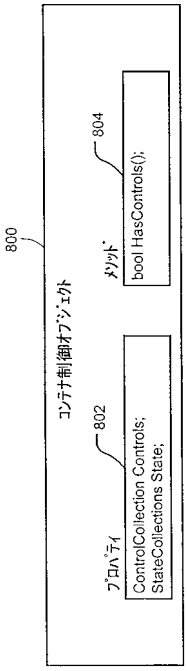
【図６】


604
606
608
610
612
614
616
618
620
初期化
ロード
ポストバックデータ処理
ポストバックイベントハンドリング
データ結合
プレレンダリング
保存
レンダリング
破棄
ページクラスコンストラクタの呼び出し
処理要求
ページクラスデストラクタの呼び出し
600
602
622

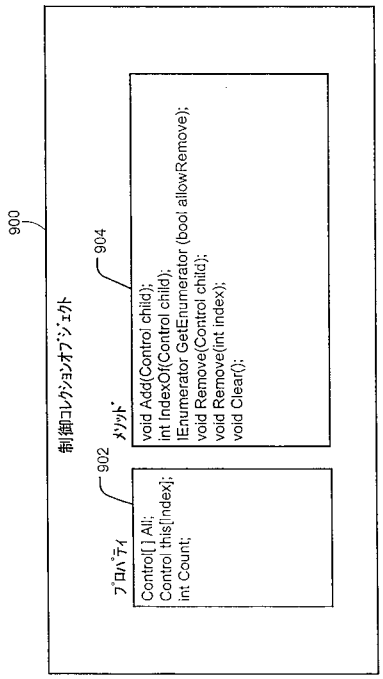
【図７】


制御オブジェクト
プロパティ
メソッド
String ID;
bool Visible;
bool MaintainState;
ControlContainer Parent;
Page Page;
FormControl Form;
TraceContext Trace;
IBindingContainer BindingContainer;
PropertyBindingCollection Bindings;
void Init();
void LoadState(Object state);
Object SaveState();
void PreRender();
void Render(TextWriter output);
void Dispose();
String GetUniqueID();
Control GetControlWithID(String id);
Control GetControlWithUniqueID(String id);
void PushControlPropertyToBindingContainer(String propName);
void PushBindingContainerPropertyToControl(String propName);
bool HasBindings();
700
702
704

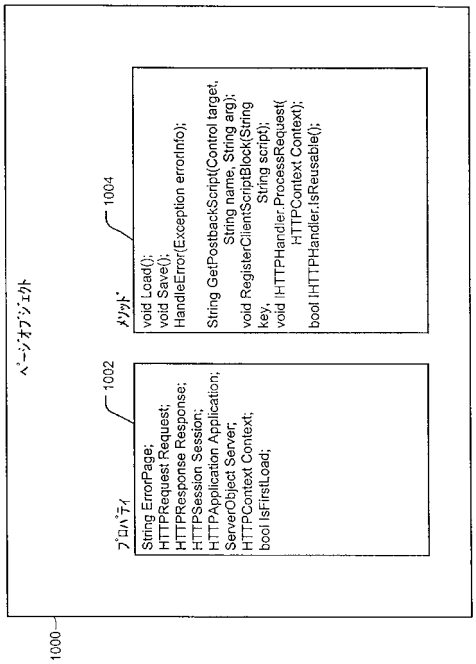
【図８】


800
コンテナ制御オブジェクト
プロパティ
メソッド
ControlCollection Controls;
StateCollections State;
bool HasControls();
802
804

【図９】


900
制御コレクションオブジェクト
902
プロパティ
Control[] All;
Control this[Index];
int Count;
904
メソッド
void Add(Control child);
int IndexOf(Control child);
IEnumerator GetEnumerator (bool allowRemove);
void Remove(Control child);
void Remove(int index);
void Clear();

【図１０】


1000
ページオブジェクト
1002
プロパティ
String ErrorPage;
HTTPRequest Request;
HTTPResponse Response;
HTTPSession Session;
HTTPApplication Application;
ServerObject Server;
HTTPContext Context;
bool IsFirstLoad;
1004
メソッド
void Load();
void Save();
HandleError(Exception errorInfo);
String GetPostbackScript(Control target,
String name, String arg);
void RegisterClientScriptBlock(String
String script);
void IHTTPHandler.ProcessRequest(
HTTPContext Context);
bool IHTTPHandler.IsReusable();

フロントページの続き

(72)発明者 クーパー、ケネス ビー.
アメリカ合衆国、９８１９９ ワシントン州、シアトル、ウエスト ブレイン ３４１０
(72)発明者 ギャスリー、スコット ディー.
アメリカ合衆国、９８０５２ ワシントン州、レッドモンド、アパートメント ユー２１０２、エヌイー ９０ス ストリート １７７８６
(72)発明者 エボ、デイビッド エス.
アメリカ合衆国、９８０５２ ワシントン州、レッドモンド、アパートメント ジェイ１３９、アボンデイル ロード １０９０９
(72)発明者 アンダース、マーク ティー.
アメリカ合衆国、９８００７ ワシントン州、ベルビュー、エヌイー １２ス プレイス １４４２５
(72)発明者 ピータース、テッド エー.
アメリカ合衆国、９８１１９ ワシントン州、シアトル、８ス アベニュー ウエスト ２４３１
(72)発明者 ミレット、スティーブン ジェイ.
アメリカ合衆国、９８０２６ ワシントン州、エドモンズ、オリンピック ビュー ドライブ ８４１２

審査官 多胡 滋

(56)参考文献 米国特許第０５９９１８０２（US，A）
秋月巌，Ｄｅｓｉｇｎ Ｔｉｍｅ Ｃｏｎｔｒｏｌその可能性と問題点（その2），DDJ，日本，株式会社翔泳社，１９９９年 2月 1日，第8巻、第2号，ｐｐ．１０８－１１４
Ken Miller，インサイド Ｍｉｃｒｏｓｏｆｔ Ｖｉｓｕａｌ ＩｎｔｅｒＤｅｖ，日本，日経ＢＰソフトプレス，１９９８年 4月16日，第1版，ｐｐ．２７－３０
Ken Spencer，ＡＳＰのＰａｇｅＯｂｊｅｃｔ，ＭＩＣＲＯＳＯＦＴ Ｉｎｔｅｒａｃｔｉｖｅ ＤＥＶＥＬＯＰＥＲ １９９９Ｊａｎｕａｒｙ，日本，株式会社アスキー，１９９９年 1月18日，Ｎｏ．１１，ｐｐ．１１６－１２０
Rick Dobson，Ｍｉｃｒｏｓｏｆｔ Ａｃｃｅｓｓ９７を使ってWebを駆使する，ＭＩＣＲＯＳＯＦＴ Ｉｎｔｅｒａｃｔｉｖｅ ＤＥＶＥＬＯＰＥＲ １９９８Ｊａｎｕａｒｙ，日本，株式会社アスキー，１９９８年 1月16日，Ｎｏ．５，ｐｐ．９２－１０４
新保政弘，ノーツが分かる人のための実践！Webアプリ入門 第6回，Ｎｏｔｅｓ／Ｄｏｍｉｎｏ Ｍａｇａｚｉｎｅ，日本，ソフトバンクパブリッシング株式会社，１９９９年12月 1日，第4巻、第１１号，ｐｐ．１００－１０４

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 15/00
G06F 13/00