

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2009-70372
(P2009-70372A)

(43) 公開日 平成21年4月2日(2009.4.2)

(51) Int.Cl.		F I			テーマコード (参考)
G06F 9/44	(2006.01)	G06F 9/44	530P		5B082
G06F 13/00	(2006.01)	G06F 13/00	560A		
G06F 12/00	(2006.01)	G06F 12/00	546A		

審査請求 有 請求項の数 20 O L 外国語出願 (全 33 頁)

(21) 出願番号	特願2008-199892 (P2008-199892)	(71) 出願人	500046438
(22) 出願日	平成20年8月1日 (2008.8.1)		マイクロソフト コーポレーション
(62) 分割の表示	特願2001-129924 (P2001-129924) の分割		アメリカ合衆国 ワシントン州 9805 2-6399 レッドモンド ワン マイ クロソフト ウエイ
原出願日	平成13年4月26日 (2001.4.26)	(74) 代理人	100077481
(31) 優先権主張番号	09/573768		弁理士 谷 義一
(32) 優先日	平成12年5月18日 (2000.5.18)	(74) 代理人	100088915
(33) 優先権主張国	米国 (US)		弁理士 阿部 和夫
(特許庁注：以下のものは登録商標)		(72) 発明者	バード、ゲイリー エス.
1. J A V A			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

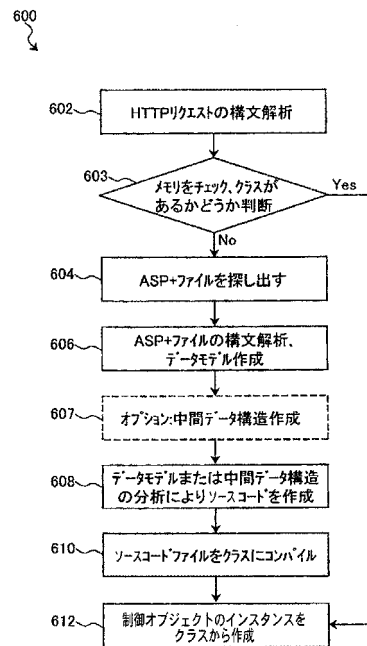
(54) 【発明の名称】 動的ウェブページコンテンツファイルからのサーバ側コード生成

(57) 【要約】

【課題】 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。

【解決手段】 サーバ側資源または動的ウェブページファイルから中間言語またはソースコードファイルを作成する方法であり、ソースコードは、クライアントレスポンスのレンダリングを含むサーバ側機能を行うウェブページ制御オブジェクトを素早く生成することを可能にする実行可能なクラスにコンパイルされる。本発明のコード生成スキームは、イベント処理および特定のオブジェクトへの属性の設定を扱うために、階層で連結された制御オブジェクトを作成することができる。

【選択図】 図6



【特許請求の範囲】**【請求項 1】**

メモリを有するサーバコンピュータシステムにおいて、メモリ内に、クライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示されるウェブページコンテンツを動的にレンダリングするためのサーバ側オブジェクトを作成するために、前記サーバコンピュータシステムによって用いられるクラスを作成する方法であって、

前記クライアントから動的なウェブページコンテンツファイルを特定するリクエストを受信する工程と、

前記動的なウェブページコンテンツファイルを処理して、前記ウェブページコンテンツファイルにおいて宣言された制御オブジェクトを表すソースコードを含むソースコードファイルを生成する工程と、

前記ソースコードファイルをコンパイルして、表示するウェブページを生成するウェブページオーサリング言語の作成のために 1 組の階層オブジェクトのインスタンスが作成され得るクラスを生成する工程と

を含むクラスを作成する方法。

【請求項 2】

前記動的なウェブページコンテンツファイルが、サーバ側宣言データ格納部である請求項 1 に記載のクラスを作成する方法。

【請求項 3】

前記クラスは、前記サーバコンピュータシステムのキャッシュメモリに格納され、前記動的なウェブページコンテンツファイルを特定する別のリクエストへのレスポンスでオブジェクトのインスタンスを作成するために用いることができる請求項 1 に記載のクラスを作成する方法。

【請求項 4】

前記クラスは、磁気記憶媒体に格納され、前記動的なウェブページコンテンツファイルを特定する別のリクエストへのレスポンスでオブジェクトのインスタンスを作成するために用いることができる請求項 1 に記載のクラスを作成する方法。

【請求項 5】

前記動的なウェブページコンテンツファイルの処理工程において、

前記動的なウェブページコンテンツファイルの一部を、階層的に関連付けられた複数のデータオブジェクトを含んでいるデータモデルへ格納するために、前記動的なウェブページコンテンツファイルの構文を解析する工程と、

第 1 フェーズ時に、前記データモデルの解析に基づいて宣言情報に関するソースコードを生成する工程と、

前記宣言情報に関するソースコードを前記ソースコードファイルへ書き込む工程と、

第 2 フェーズ時に前記データモデルの解析に基づいて制御オブジェクト情報に関するソースコードを生成する工程と、

第 2 フェーズ時に前記制御オブジェクト情報に関するソースコードを前記ソースコードファイルへ書き込む工程と

を含む請求項 1 に記載のクラスを作成する方法。

【請求項 6】

前記方法は、第 3 フェーズ時に前記データモデルの解析に基づいてレンダリング情報に関するソースコードを生成する工程と、

第 3 フェーズ時に前記レンダリング情報に関するソースコードを前記ソースコードファイルへ書き込む工程と

を含む請求項 5 に記載のクラスを作成する方法。

【請求項 7】

前記第 1 フェーズが実質的に完了すると前記第 2 フェーズが生じ、前記第 2 フェーズが実質的に完了すると前記第 3 フェーズが生じる請求項 6 に記載のクラスを作成する方法。

10

20

30

40

50

【請求項 8】

前記第 1 フェーズ、前記第 2 フェーズおよび前記第 3 フェーズが同時に生じる請求項 6 に記載のクラスを作成する方法。

【請求項 9】

前記動的ウェブページコンテンツファイルの処理工程の前に、前記受信したリクエストに関するクラスがコンパイルされメモリに格納されているかどうか判断する工程と、

前記クラスがコンパイルされメモリに格納されている場合、前記処理工程をスキップし、そうでない場合は、処理工程を行う工程と

をさらに含む請求項 1 に記載のクラスを作成する方法。

【請求項 10】

10

メモリを有するコンピュータシステムによって搬送波で具体化され、かつクライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示されるウェブページコンテンツを動的にレンダリングするためのサーバ側オブジェクトを作成するために、前記サーバコンピュータシステムによって用いられるクラスをメモリ内に作成するコンピュータプロセス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データ信号であって、

前記コンピュータプロセスが、前記クライアントから動的なウェブページコンテンツファイルを特定するリクエストを受信する工程と、

前記ウェブページコンテンツファイルにおいて宣言された制御オブジェクトを表すソースコードを含むソースコードファイルを生成するために前記動的なウェブページコンテンツファイルを処理する工程と、

20

表示するウェブページを生成するウェブページオーサリング言語を生成するために 1 組の階層オブジェクトのインスタンスが作成され得るクラスを生成するべく前記ソースコードファイルをコンパイルする工程と

を含むコンピュータデータ信号。

【請求項 11】

メモリを有するコンピュータシステムによって読み取り可能であり、かつクライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示され、ウェブページコンテンツを動的にレンダリングするためのサーバ側オブジェクトを作成するために前記サーバコンピュータシステムによって用いられるクラスをメモリ内に作成するコンピュータプロセス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であって、

30

前記コンピュータプロセスは、前記クライアントから動的なウェブページコンテンツファイルを特定するリクエストを受信する工程と、

前記ウェブページコンテンツファイルにおいて宣言された制御オブジェクトを表すソースコードを含むソースコードファイルを生成するために前記動的なウェブページコンテンツファイルを処理する工程と、

表示するウェブページを生成するウェブページオーサリング言語を生成するために 1 組の階層オブジェクトのインスタンスが作成され得るクラスを生成するべく前記ソースコードファイルをコンパイルする工程と

40

を含むコンピュータプログラム記憶媒体。

【請求項 12】

メモリを有するサーバコンピュータシステムにおいて、動的にレンダリングされたウェブページコンテンツを有し、1 つ以上のクライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示される複数のウェブページレスポンスを作成する方法であって、

動的ウェブページコンテンツファイルを識別するリクエストを、前記ウェブページのための前記クライアントコンピュータシステムから受信する工程と、

前記動的なウェブページコンテンツファイルの要素を格納するためのデータモデルを作成する工程と、

50

前記データモデルの評価に基づき、前記動的なウェブページコンテンツファイルに関するソースコードファイルを生成する工程と、

前記ソースコードファイルをコンパイルしてメモリにコンパイルされたクラスを作成する工程と、

前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と、

前記動的なウェブページコンテンツを前記クライアントコンピュータシステムに運ぶウェブページレスポンスヘンダリングする工程と、

前記ウェブページレスポンスを前記要求しているクライアントコンピュータシステムへ導く工程と、

動的ウェブページコンテンツファイルを識別する第2のリクエストを、前記ウェブページのために受信する工程と、

当該動的ウェブページコンテンツファイルのコンパイルされたクラスがメモリに存在するかどうかを判断する工程と、

前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と、

前記動的ウェブページコンテンツを第2のウェブページレスポンスヘンダリングする工程と、

前記第2のウェブページレスポンスを前記要求しているクライアントコンピュータシステムへ導く工程と

を含むウェブページレスポンスを作成する方法。

【請求項13】

メモリを有する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、動的にレンダリングされたウェブページコンテンツを有し、1つ以上のクライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示される、複数のウェブページレスポンスを作成するコンピュータプロセスを実行処理するコンピュータプログラムをコード化するコンピュータプログラム記憶媒体であって、

前記コンピュータプログラムは、動的ウェブページコンテンツファイルを識別するようなリクエストを、前記ウェブページのための前記クライアントコンピュータシステムから受信する工程と、

前記動的なウェブページコンテンツファイルの要素を格納するためのデータモデルを作成する工程と、

前記データモデルの評価に基づき、前記動的なウェブページコンテンツファイルに関するソースコードファイルを生成する工程と、

前記ソースコードファイルをコンパイルしてメモリにコンパイルされたクラスを作成する工程と、

前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と、

前記動的ウェブページコンテンツを前記クライアントコンピュータシステムに運ぶウェブページレスポンスヘンダリングする工程と、

前記ウェブページレスポンスを前記要求しているクライアント側コンピュータシステムへ導く工程と、

動的ウェブページコンテンツファイルを識別する第2のリクエストを、前記ウェブページのために受信する工程と、

10

20

30

40

50

当該動的ウェブページコンテンツファイルのコンパイルされたクラスがメモリに存在するかどうか判断する工程と、前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と、

前記動的ウェブページコンテンツを第2のウェブページレスポンスヘレンダリングする工程と、

前記第2のウェブページレスポンスを前記要求しているクライアントコンピュータシステムへ導く工程と

を含むコンピュータプログラム記録媒体。

10

【請求項14】

メモリを有するコンピュータシステムによって搬送波で具体化され、かつ、動的にレンダリングされたウェブページコンテンツを有し、1つ以上のクライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上でウェブページとして表示される複数のウェブページレスポンスを作成するコンピュータプロセスを実行処理するコンピュータプログラムをコード化するコンピュータデータ信号であって、

前記コンピュータプログラムは、動的ウェブページコンテンツファイルを識別するリクエストを、前記ウェブページのための前記クライアントコンピュータシステムから受信する工程と、

前記動的なウェブページコンテンツファイルの要素を格納するためのデータモデルを作成する工程と、

20

前記データモデルの評価に基づき、前記動的なウェブページコンテンツファイルに関するソースコードファイルを生成する工程と、

前記ソースコードファイルをコンパイルしてメモリにコンパイルされたクラスを作成する工程と、

前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と

、

前記動的ウェブページコンテンツを前記クライアントコンピュータシステムに運ぶウェブページレスポンスヘレンダリングする工程と、

30

前記ウェブページレスポンスを前記要求しているクライアントコンピュータシステムへ導く工程と、

動的ウェブページコンテンツファイルを識別する第2のリクエストを、前記ウェブページのために受信する工程と、

当該動的ウェブページコンテンツファイルのコンパイルされたクラスがメモリに存在するかどうか判断する工程と、

前記サーバコンピュータシステムにクラスリファレンスを戻す工程であって、これにより、前記サーバコンピュータシステムが当該クラスからサーバ側処理オブジェクトのインスタンスを作成することが可能になり、ウェブページコンテンツを動的に生成する工程と

40

、

前記動的ウェブページコンテンツを第2のウェブページレスポンスヘレンダリングする工程と、

前記第2のウェブページレスポンスを前記要求しているクライアントコンピュータシステムへ導く工程と

を含むコンピュータデータ信号。

【請求項15】

メモリ内に、クライアント側コンピュータシステムに運ばれ、前記クライアントコンピュータシステム上で処理されるオーサリング言語要素を動的にレンダリングするためのサーバ側オブジェクトを作成するために、サーバ側コンピュータシステムによって用いられる

50

クラスを作成するコンピュータプロセスをコンピュータシステムで実行処理するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって、前記プロセスは、

動的ウェブページ資源を識別するリクエストを、前記資源のための前記クライアントコンピュータシステムから受信する工程と、

前記資源に関するソースコードファイルを生成するために前記資源を処理する工程と、
前記ソースコードファイルをコンパイルして、コンパイルされたクラスをメモリに作成し、前記コンパイルされたクラスのオブジェクトのインスタンス作成を可能にする工程とを含むコンピュータプログラムプロダクト。

【請求項 16】

メモリ内にクラスを作成するコンピュータプロセスをコンピュータシステムで実行処理するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって、前記データモデル作成処理は、

前記資源を論理要素に分離し前記論理要素間の関係を識別するために前記資源の構文解析を行う工程と、

階層データモデルを形成する複数の階層的に関連するデータ構造を作成する工程と、

前記データ構成に前記資源の部分を格納する工程と

を含む請求項 15 に記載のコンピュータプログラムプロダクト。

【請求項 17】

メモリ内にクラスを作成するコンピュータプロセスをコンピュータシステムで実行処理するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって

、
前記処理工程において、可変の宣言情報に関連するソースコードを生成するために前記資源の第 1 の解析を行う工程と、

制御オブジェクト情報に関連するソースコードを生成するために前記資源の第 2 の解析を行う工程と、

レンダリング情報に関連するソースコードを生成するために前記資源の第 3 の解析を行う工程と、

前記ソースコードファイルに前記ソースコードを格納する工程と

を含む請求項 15 に記載のコンピュータプログラムプロダクト。

【請求項 18】

メモリ内にクラスを作成するコンピュータプロセスをコンピュータシステムで実行処理するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって、前記ソースコード生成処理工程において、中間データ構造を生成する工程をさらに含み、前記ソースコードが、前記中間データ構造から生成される請求項 16 に記載のコンピュータプログラムプロダクト。

【請求項 19】

メモリ内にクラスを作成するコンピュータプロセスをコンピュータシステムで実行処理するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって

、
前記中間データ構造の生成処理工程において、可変の宣言情報に関連する中間データ構造要素を生成するために前記資源の第 1 の解析を行う工程と、

制御オブジェクト情報に関連する中間データ構造要素を生成するために前記資源の第 2 の解析を行う工程と、

レンダリング情報に関連する中間データ構造要素を生成するために前記資源の第 3 の解析を行う工程と、

前記中間データ構造要素からソースコードを生成する工程と

を含む請求項 18 に記載のコンピュータプログラムプロダクト。

【請求項 20】

メモリ内にクラスを作成するコンピュータプロセスをコンピュータシステムで実行処理

10

20

30

40

50

するコンピュータプログラムをコード化するコンピュータプログラムプロダクトであって、前記中間データ構造は、複数のソースコード言語ファイルに翻訳され得る包括的な記述であって、少なくとも1つのソースコードファイルは、別のソースコード言語ファイルと異なることを特徴とする請求項20に記載のコンピュータプログラムプロダクト。

【発明の詳細な説明】

【技術分野】

【0001】

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制御（コントロール）オブジェクトを作成するサーバ側コード作成に関する。

10

【背景技術】

【0002】

典型的なウェブブラウザは、クライアントシステムに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（「URL」）を特定し、所望のウェブサイトにアクセスする。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リクエストを受け取り、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。

20

【0003】

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ウェブページファイルと対応する。HTMLは、ワールドワイドウェブ上のドキュメント（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、HTMLコードを実際の視覚的な画像または音声コンポーネントへ変換するクライアントブラウザによってウェブサーバから取り出され、ウェブページとして表示される。クライアントコンピュータシステムにおいて、このプロセスは、渡されたHTMLファイルに定義されたウェブページ内容を表示する。HTMLを用いて、開発者が、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。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。ウェブページコンテンツは、表示のためにクライアントに戻されるHTMLコードである。このような動的処理は、単にクライアントブラウザに所定のコードを送信するのではなく、処理ステップの結果、送信に先立ちHTMLコードを生成するサーバ側アプリケーションに関する。

30

40

【0004】

より複雑なクライアント・サーバ間の対話をハンドリングするために、サーバ側アプリケーションプログラムは、動的コンテンツ、例えば、変化している株価または交通情報を与えるような、より複雑なクライアント・サーバ間の対話をハンドリングするように開発されてきた。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、クライアントにHTTPレスポンスでクライアントへの送信に適切なHTMLコードを生成する。例えば、サーバ側アプリケーションプログラムは、クライアント側へのHTTPレスポンスにおける送信HTMLコードを動的に生成するために、HTTPリクエストにおいてクライアント側によって提供される照会ストリングまたはウェブベースのフォームからのデータを処理すること

50

ができる。本質的に、サーバ側アプリケーションは、クライアント側からのリクエストにおける情報に基づきカスタマイズされたHTML型ファイルを生成することができる。この場合、サーバ上に格納された静的なHTMLファイルは存在せず、HTMLファイルは、実行時に動的に作成される。サーバ側アプリケーションプログラムの一例として、メモリ機構への1つ以上のフォーマット化されたテキスト書き込み処理のシーケンスを用いて、HTMLコードを生成する場合がある。その後、得られたテキストは、HTTPレスポンスでクライアントシステムに送信され、そこでブラウザに表示される。

【0005】

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、1つ以上のプログラム言語（例えば、C++、Perl、Visual Basic、またはJscript）を含むプログラムベ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、残念なことに、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、人間的な感じを与えるが、プログラミング経験がない場合が多い。したがって、プログラミング経験が少ない人が、サーバ側アプリケーションとそのそれぞれのクライアントとのウェブページインタフェースを開発することができるような、ウェブページファイルを作成するための単純化されたウェブページ開発フレームワークを提供することが必要である。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れている。

【発明の開示】

【発明が解決しようとする課題】

【0006】

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フレームワークにより、開発者は、典型的に、Visual BasicまたはJscriptコードおよびその他のHTMLコードを含む"ASP"ウェブページファイルを作成できる。ASPファイルは、種々の機能を実行する宣言またはタグならびにVBスクリプトまたはJスクリプトコードを含む。一般に、実際のプログラミングコードを書き込むより、これらの宣言を書き込むほうが簡単である。

【0007】

処理中、HTTPリクエストは、所望の資源としてASPファイルを特定し、その後、ASPファイルを用いてクライアント側へのHTTPレスポンスでの結果HTMLコードを生成する。さらに、ASPファイルは、所与のアプリケーションプログラミング労力を減らすために、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）およびデータベースまたは他の第3者アプリケーションを参照することができる。

【0008】

単純化されたASPウェブページファイルは、実行時間でスクリプトエンジンにより解釈され得るスクリプトに変換されなければならない。スクリプトエンジンは、典型的に、所望の結果を達成するために、連続または同期的にASPファイルにおける種々の宣言タイプコマンドを実行する。実行処理可能なファイルとしてコンパイルされ格納されたファイルと比較して、一般に、スクリプトエンジンによって実行されたスクリプトファイルは時間がかかる。これは、スクリプトエンジンが単にファイルを実行するだけではなく、解釈機能を果たさなければならないからである。

【0009】

スクリプトファイルを実行処理可能なファイルにコンパイルする際のある1つの問題は、スクリプトファイルには、おそらく種々の言語の組み合わせがある、またはその可能性があるということである。例えば、スクリプトファイルは、HTMLで書かれたコンポーネントおよびVisual Basicで書かれた他のコンポーネントを含む場合がある。スクリプトエンジンは、実行時間にこれらの要素を解釈するために種々の処理を用いるが、異なる言語コ

10

20

30

40

50

ンポーネットを1つの言語、すなわち、1つのソースコードファイルに翻訳するためのコンパイラが存在しない。さらに、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）を動的に管理することを要求されるプログラミングは、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。これらのサーバ側プロセスが、より複雑になるにしたがって、スクリプトエンジンは、持続的にこの要求を満たし続けることができなくなる。

【0010】

これらおよび他の理由により、本発明がなされた。

【課題を解決するための手段】

10

【0011】

本発明は、中間言語またはソースコードファイルをサーバ側資源から作成するためのコード生成方法および装置に関するものであり、ソースコードファイルは、実行処理可能なクラスにコンパイルされる。実行処理可能なクラスにより、クライアントレスポンスのレンダリングを含むサーバ側機能を実行するウェブページ制御オブジェクトの素早い生成が可能になる。本発明のある実施形態において、コード生成スキームは、イベント処理および特定のオブジェクトへの属性の設定を扱うために階層内で接続された制御オブジェクトを作成することができる。さらに、コード生成方法はまた、テンプレートを用いて宣言されたオブジェクトを接続することもできる。

【0012】

20

より好ましくは、本発明は、サーバコンピュータシステムメモリにおけるクラスの作成方法に関する。クラスは、ウェブページコンテンツを動的にレンダリングするサーバ側オブジェクトを作成するために、サーバコンピュータシステムによって用いられ、ウェブページコンテンツは、クライアント側コンピュータシステムに送られ、クライアントコンピュータシステム上のウェブページとして表示される。処理において、サーバコンピュータシステムは、ウェブページのためのリクエストをクライアントコンピュータシステムから受信し、このリクエストが動的ウェブページコンテンツファイルを識別する。サーバコンピュータは、動的ウェブページコンテンツファイルの要素を格納するためのデータモデルを作成し、データモデルを評価し、データモデルの評価に基づき動的なウェブページコンテンツファイルに関するソースコードファイルを生成する。ソースコードが作成されると、ソースコードファイルがコンパイルされて、コンパイルされたクラスをメモリに作成する。一般に、プロセスは、サーバコンピュータシステムにクラス参照を戻してサーバコンピュータシステムがクラスを用いることが可能になって終了する。

30

【0013】

他の好ましい実施形態によると、方法は、サーバコンピュータシステム上のキャッシュメモリにクラスを格納する。一旦、キャッシュメモリに格納されると、多数のサーバ側ページオブジェクトが1つのコンパイルされたクラスからインスタンスを作成し、元々の資源は再度使用されない。ウェブページに対するリクエストを受信する度に、サーバコンピュータシステムは、その動的ウェブページコンテンツファイルのコンパイルされたクラスがメモリにあるかどうかを判断する。要求されたクラスが、メモリに存在しない場合は、それを作成する。クラスがあれば、サーバコンピュータシステムは、ウェブページコンテンツを動的に生成するために、そのクラスからサーバ側処理オブジェクトのインスタンスを作成する。次に、ウェブページコンテンツはレンダリングされクライアントコンピュータシステムに送られる。

40

【0014】

本発明のさらに別の実施形態によれば、データモデルを評価する方法ステップは、複数のパスでデータモデルの反復トラバースを伴う。各パス時にソースコードが生成され、そのパス時のデータモデルの評価に基づきソースコードファイルに書き込まれる。データモデルは、階層的に連結されたデータ構造を用いて構築される。

【0015】

50

本発明によるコンピュータプログラムプロダクトの実施形態は、コンピュータシステムによって読み取り可能であり、サーバコンピュータ上にメモリにコンパイルされたクラスを作成するコンピュータプロセスを実行処理するコンピュータプログラムをコード化するコンピュータプログラム記憶媒体を含む。コンパイルされたクラスは、クライアントコンピュータシステム上に表示される要求されたウェブページに対応するレスポンスをレンダリングするためにサーバ側処理オブジェクトのインスタンスを作成するのに用いられる。本発明によるコンピュータプログラムプロダクトの別の実施形態は、コンピュータシステムによる搬送波によって具体化され、サーバにコンパイルされたクラスを作成するためのコンピュータプログラムをコード化するコンピュータデータ信号を含む。

【発明を実施するための最良の形態】

【0016】

本発明の実施の形態は、動的ウェブページコンテンツ資源またはファイルによって定義された特定のウェブページのためのメモリにコンパイルされたクラスを生成する方法に関するものである。コンパイルされたクラスの作成は、ウェブページファイルからソースコードファイルの作成を伴う。次に、ソースコードファイルがクラスにコンパイルされる。コンパイルされたクラスが、メモリに存在すると、表示するクライアントに送り返されるレスポンスをレンダリングするためにページオブジェクトのインスタンスが作成され得る。一般に、ページオブジェクトは、ウェブページ上に表示される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のためのサーバ側制御オブジェクトを伴う。さらに、サーバ側制御オブジェクトの階層が、これらのオブジェクトが、クライアント上のウェブページの表示のためのHTMLのような結果オーサリング言語コードを最終的に協働して生成するウェブページファイルに宣言され得る。

【0017】

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の表示装置上にウェブページ104を表示するブラウザ102を実行処理する。クライアント100は、ビデオモニタ(図示せず)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む。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コードを受信し、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) およびXML (eXtensible 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。

【0018】

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のシーケンス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を受け取り、URLを解析し、リクエストを処理する適切なハンドラ120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扱う複数のハンドラ120がウェブサーバ116に備わっている。

【0019】

例えば、URLがHTMLファイルのような静的なコンテンツファイル122を特定する場合、ハンドラ120は、静的コンテンツファイル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ファイル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+ (Active Server Page+) ページのような動的コンテンツ資源またはファイル124を特定する場合、ハンドラ

10

20

30

40

50

120は、動的コンテンツファイル124にアクセスし、動的コンテンツファイル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。一般に、ファイル124のような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を通じて、HTTPレスポンス112でクライアント100へ送られる。

【0020】

この処理の間、ハンドラ120はまた、開発労力を簡素化するために予め開発された、または第三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のインスタンスを作成することができる。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、動的コンテンツファイル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。

10

【0021】

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ファイル124において参照され、またはそれに関連付けられる。非ユーザインタフェースサーバコンポーネント130は、動的コンテンツファイル124で宣言されたサーバ側制御オブジェクトによって立ち上げられたイベントを処理し得る。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。

20

【0022】

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+ページ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+ページは処理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ファイル(例えば、ASP+ページ)のコンテンツに基づきサーバ側制御オブジェクト階層を生成する。

30

【0023】

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1つ以上の処理を行う。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。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。

40

50

【 0 0 2 4 】

あるいは、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 2 0 6 または 2 0 8 においてアクセスされる（ポストバック入力の手動リング、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。

【 0 0 2 5 】

処理 2 1 0 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 2 1 4 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 2 1 6 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表示する。処理 2 1 2 において、サーバ側制御オブジェクト階層を終了する。階層内サーバ側制御オブジェクトは、関連付けられたASP+ページを参照するHTTPリクエストに回答して作成され、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。あるいは、処理 2 1 2 は、処理 2 0 8 の後、処理 2 1 0 の前に行ってもよい。

10

【 0 0 2 6 】

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 3 0 0 は、HTTPパイプライン 3 0 4 にHTTPリクエスト 3 0 2 を受信する。HTTPパイプライン 3 0 4 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 3 0 0 によって受信される各入力HTTPリクエスト 3 0 2 は、最終的には、インタフェースハンドラ（Interface Handler）、例えばIHTTPハンドラクラス（ハンドラ 3 0 6 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 3 0 6 は、URLリクエストを解析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 3 0 8 ）を呼び出す。

20

【 0 0 2 7 】

図 3 において、ASP+ページ 3 1 0 に関連付けられたページファクトリ 3 0 8 が呼び出され、ASP+ページ 3 1 0 からのオブジェクトのインスタンスの作成および構成を手動リングする。ASP+ページ 3 1 0 は、一意のURLによって認識または参照され得、他の接頭辞も用い得るが、例えば、".aspx"という接頭辞によってさらに識別することができる。特定の".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 3 0 8 によって受信されると、ページファクトリモジュール 3 0 8 は、ファイルシステムを検索して適切な資源またはファイル（例えば、.aspxページ 3 1 0 ）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 3 0 8 は、ファイルを開き、そのファイルをメモリに読み出す。あるいは、要求されたファイルは存在するが、予めメモリにロードされている場合は、以下に詳細に述べるように、資源は必ずしもメモリにロードされる必要はないかもしれない。要求されたaspx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 3 0 8 は、例えばHTTP"404"メッセージをクライアントに送信することによって、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。

30

40

【 0 0 2 8 】

ASP+ページ 3 1 0 をメモリに読み出した後、ページファクトリモジュール 3 0 8 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、ページオブジェクトの構造、プロパティおよび機能を定義するコードである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コードファイルを生成する

50

のに用いられる。本発明のある実施形態において、ソースリストは、中間言語に動的にコンパイルされる。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、固有の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 3 1 0）から直接生成する。制御クラスライブラリ 3 1 2 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 3 0 8 によってアクセスされ得る。

【 0 0 2 9 】

ページファクトリモジュール 3 0 8 は、コンパイルされたクラスからページオブジェクトを作成する。ページオブジェクト 3 1 4 は、図 1 のウェブページ 1 0 4 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 3 1 4 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 3 1 8、ボタンオブジェクト 3 2 0 および別のボタンオブジェクト 3 2 2）が、制御オブジェクト階層 3 1 6 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 に示す HTML 制御に対応するオブジェクト（後述）も含まれている。ページオブジェクト 3 1 4 は、図 1 のウェブページ 1 0 4 に対応する。テキストボックスオブジェクト 3 1 8 は、図 1 のテキストボックス 1 0 6 に対応する。同様に、ボタンオブジェクト 3 2 0 は、図 1 の追加ボタン 1 0 8 に対応し、ボタンオブジェクト 3 2 2 は、図 1 の削除ボタン 1 1 0 に対応する。ページオブジェクト 3 1 4 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。

【 0 0 3 0 】

上記の実施形態において、制御オブジェクト階層 3 1 6 の制御オブジェクトは、サーバ 3 0 0 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 3 0 2 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 3 1 6 から生成（すなわち、レンダリング）され、HTTP レスポンス 3 2 4 でクライアントに送信される。例えば、結果 HTML（または他のオーサリング言語）コードは、ACTIVEX タイプの制御またはその他、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 HTML 構成を参照することができる。

【 0 0 3 1 】

ASP+リソース 3 1 0 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 3 3 0 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 3 3 0 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回答して、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 3 3 0 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。

【 0 0 3 2 】

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ファイルフォーマット（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を読み出すページファクトリモジュール308に対し命令を与え、クラスを作成し、最終的にHTMLコードまたはその他、クライアントへHTTPレスポンスで送信するオーサリング言語をレンダリングする適切なサーバ側制御オブジェクトを呼び出す。

【0033】

ファイル400の第1ラインは、以下のフォーマットにおいてデリミッタ"<%@"と"%>"との間にディレクティブを含む：

```
<%@ directive [attribute=value] %>
```

10

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、動的コンテンツファイルを処理するときページファクトリモジュール308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。

【0034】

第2ラインの<html>は、直接的な「書き込み」命令以外、結果HTMLコードをレンダリングするための情報において追加の処理が行われなようにリテラルとしてソースコードファイルへ書き込まれる標準HTML開始タグである。HTML構文において、<html>は、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。

20

【0035】

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、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>
```

30

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"属性を内コードの文法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。

40

【0036】

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呼び出される。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 Bas

50

ic文法を用いると、キーワード"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。

【0037】

「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。コードレンダリングブロックは、ページレンダリング時間で実行処理される任意の量のコードを含むことができる。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。コードの他の部分は、そのレンダリングメソッドで実行処理され得る。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。

【0038】

```
<% InlineCode %>
```

ここで"InlineCode"は、ページ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。

【0039】

インライン表現もまた、以下のような例示的な文法を用いて表現レンダリングブロックデリミッタ"<%@"と"%>"との間で用いることができる。

【0040】

```
<%= 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ルダに書き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、以下のようにファイル400に含まれ得る。

【0041】

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

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。

【0042】

```
<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。ファイル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。

【0043】

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。

【0044】

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。

【0045】

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

10

20

30

40

50

"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。

10

【0046】

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タグが構文解析され、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、タグ要素上の宣言名/値属性対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）。

20

【0047】

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文法は以下のとおりである。

【0048】

```
<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。サポートされ得るHTMLタグのリストならびに関連文法および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。

30

【0049】

【表 1】

HTML タグ名	例	COM+ クラス	
<a>	 My Link 	AnchorButton	
		Image	
	 	Label	
<div>	<div id = "MyDiv" runat = server>Some contents</div>	Panel	
<form>	<form id = "MyForm" runat = server> </form>	FormControl	10
<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	20
<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	30
<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	

【 0 0 5 0 】

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、タグ要素における属性対である宣言名/値は、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」の接頭文字列である子の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。

40

50

【 0 0 5 1 】

```
<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。

【 0 0 5 2 】

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。

【 0 0 5 3 】

```
<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。

【 0 0 5 4 】

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。

【 0 0 5 5 】

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の読み出しおよび書き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の読み出しおよび書き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に接続されている。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揮発性記憶部を提供している。

【 0 0 5 6 】

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。

【 0 0 5 7 】

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

10

20

30

40

50

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、ジョイスティック、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。

【0058】

10

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。

【0059】

20

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。

【0060】

30

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+ファイル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。

【0061】

ページファクトリモジュール308によって行われる処理600を図6のフローチャートに示す。図6では、処理600は、制御オブジェクト階層204の生成処理(図2)にほぼ対応する。処理600は、ASP+ページ(ASP+ファイルともいう)をコンパイルされたオブジェクトコードクラスに変換し、次いで、それは、制御オブジェクト314、318、320および322(図3)のインスタンスを作成するためにメモリにロードされる。

40

【0062】

サーバが処理202においてURLに対するリクエストを受信すると(図2)、ページ作成処理600が始まる。リクエストの受信後、構文解析処理602が、要求されたURLの構文解析を行い、どの資源が要求されているかを判断する。資源は、静的なファイル(.htm、.gif、.jpgなど)、ディレクトリブラウジング起動、DAV(デジタルオーディオ/ビデオ)ファイルおよび動的なコンテンツリクエスト(.aspx、.soapなど)を含み、本発明は、これらをハンドリングするように構成されている。サーバ側によって受信された、入ってくるHTTPリクエストは、ハンドラクラス、例えば、IHTTPHandlerクラスの特定のイン

50

スタンスによって最終的に処理される。ハンドラファクトリ、例えば、IHTTPHandler"ファクトリ"の使用によって、ハンドラインスタンス、例えば、IHTTPHandlerインスタンスへのURLリクエストの解釈を実際に行う差込可能なアーキテクチャが提供される。この解釈は、入ってくるURLのファイル拡張およびHTTPコマンドを、適切なハンドラインスタンス、例えばIHTTPHandlerインスタンスを最終的に作成することになっているIHTTPHandlerFactoryクラスのようなファクトリクラスへ写像することができるアプリケーション構成設定を用いて容易に行うことができる。構文解析処理において、差込可能アーキテクチャにより種々の資源を識別できるが、以下の記載は、動的ウェブページコンテンツ資源を識別するHTTPリクエストに焦点をあて、特に、.aspxまたはASP+ページもしくはファイルのような動的ウェブページコンテンツファイルに焦点を当てている。

10

【0063】

構文解析処理602で実際の資源が識別されると、チェック処理603において、メモリをチェックし、クラスがメモリに存在するかどうかを判断する。クラスがコンパイルされているかどうかを判断するために、チェック処理603において、クラスがコンパイルされるとセットされるフラグを検索することによって、または名前によってメモリ内のクラスを検索する。どちらにしても、チェック処理603において、コンパイルクラスがすでにコンパイルされ格納されていることを示す指標を検索する。クラスがメモリにある場合、フローはYES枝に分岐しインスタンス作成処理612に進む。インスタンス作成処理は、コンパイルされたクラスから制御オブジェクトのインスタンスを作成する。チェック処理603において、クラスがコンパイルされていないと判断されると、フローはNO枝に分岐し位置決定処理604に進む。

20

【0064】

チェック処理603が、クラスはメモリに存在しないと判断すると、位置決定処理604において、特定の資源を検索し探し当て、ファイルをメモリに読み出す。"System. ASP. WebForms. PageFactory"クラスのようなベースクラスが、ASPおよびASPXページのインスタンス作成および構成をハンドリングするハンドラファクトリインプリメンテーションを提供する。資源、この場合は物理的な動的なウェブページコンテンツファイルが見つからなければ、適切なエラーメッセージが戻される。

【0065】

位置決定処理604の次に、処理600において、構文解析作成処理606を行い、ここでASP+ファイルのような資源の構文解析を行う。構文解析/作成処理606は、ASP+ファイルを宣言ごとに読み出し、構文解析しながら集めた情報からデータモデルを作成する。データモデルは、ソースコードを引き出すことができるアクティブコンテンツファイルの要素に関連する要素を含むデータ構造である。データモデルは、アクティブコンテンツファイルで参照された構造的要素を含み、これらの要素は、アクティブサーバページの結果制御オブジェクトの構造を表すように連結される。本発明のある実施形態において、データモデルは、階層的なツリー構造で関連付けられたオブジェクトの組み合わせである。

30

【0066】

ウェブコンテンツファイルの各宣言は、データモデルの作成方法およびデータモデルに格納される情報を示す所定の要素を有している。例えば、宣言がリテラルテキスト宣言である場合、データモデルは、テキストが適切な位置に挿入されるような情報を含むだけでよい。しかし、宣言が入れ子の制御オブジェクトが存在することを示す場合、ASP+ファイルに宣言されたような入れ子に従ったデータモデルが作成されなければならない。本質的に、データモデルは、各宣言ごとに1つのオブジェクトを含み、各オブジェクトは、各オブジェクトに子オブジェクトが存在するかどうかに関する情報を含む。

40

【0067】

ASP+ファイルが構文解析され、データモデルが作成されると、任意である作成処理607を行ってもよい。作成処理607は、以下に記載するように、データモデルを解析し、例えば、作成すべきソースコードファイルのようなソースコードを抽象化したものである中間データ構造を作成する。本質的に、中間データ構造は、コードを記述する包括的な

50

データ構造である。中間データ構造は、クラスを記述する上位レベルのデータ構造を有する。各クラスごとに、クラス名およびメソッドのリストまたはアレイを有する別のレベルのデータ構造であってもよい。さらに、各メソッド自体、ステートメントおよび宣言のような種々の要素を有するデータ構造である。ステートメントは、表現、データ呼び出しなどのような項目を含み得る。

【 0 0 6 8 】

ASP+ファイルは構文解析され、データモデル（および任意には、中間データ構造）が作成されると、作成ソースコードモジュール 6 0 8 は、典型的には、ASP+ファイルに存在する各宣言のコードの種々のラインを書き込むことにより、各データモデルの解析を通して必要なソースコードを作成する。一般に、このステップは、データモデルにおける公知の宣言および他の情報のコードへの直接的な翻訳を伴う。このような翻訳は、翻訳を行うアプリケーションへハードコード化されるか、または、各翻訳は、検索テーブルに格納され得る。この種の検索テーブルは、ほとんどすべての言語にとって適切な翻訳を提供するように作成され得るが、オブジェクト指向ソースコード言語の使用により、既存のCOMまたはCOM+ライブラリを利用するためのCOMまたはCOM+クラスへのコンパイルが簡単になる。（例えば、ソースコード言語の一例として、C++またはVbasicなどがある。）さらに、中間データ構造が処理 6 0 7 で作成される場合は、翻訳はさらに直接的になる。このような場合、データ構造は、包括的なソースコード言語ファイルに非常に近くなって、翻訳は、単に、特定のソースコード言語のソースコードファイルを作成するのに適切な辞典および文法の追加になるにすぎない場合がある。

10

20

【 0 0 6 9 】

ASP+ファイルは、結果ソースコードファイルに特定のソースコード言語を宣言する。ASP+ファイルに言語が宣言されていない場合、Visual Basicなどのデフォルト言語を用いることができる。データモデルまたは中間データ構造から生成された結果ソースコードファイルは、本質的に、プログラマがサーバ側制御オブジェクト階層を利用する場合で、このコード生成ツールの恩恵を受けないときに書き込むことを要求されることになるタイプの高度なソースコードファイルである。

【 0 0 7 0 】

ソースコードファイルが完成すると、コンパイル処理 6 1 4 において、ソースコードファイルコンパイルし、オブジェクトコードまたは他の実行処理可能フォームでコンパイルされたクラスを作成する。あるいは、クラスは、バイトコードまたは仮想装置上で起動することができる他の言語にコンパイルすることができる。ソースコードファイルのコンパイルは、ソースコードファイルに存在するソースコード言語をコンパイルするように構成されたコンパイラの使用を伴う。そして、クラスが、インスタンス作成処理 6 1 6 によって呼び出され、ウェブページのためのページオブジェクトおよび結果制御オブジェクトを作成する。URLリクエストに明示的に備わっていないが、インスタンス作成処理は、特定の資源のURLリクエストの一部をなす非明示的処理である。

30

【 0 0 7 1 】

ASP+ファイルがページクラスにコンパイルされると、クラスは、将来のリクエストで使用可能な状態で維持される。したがって、ASP+ファイルのソースコードの生成処理は、1 度行われるだけでよく、後のリクエストはコンパイルされたクラスを利用することができる。必要なオブジェクトを必要に応じてインスタンス作成する。コンパイルされたクラスは、短期間の間キャッシュされ得、かつ/またはコンパイルされたクラスはディスクに格納し得る。実際、ASP+ファイルが一旦コンパイルされると、元々のASP+ファイルを再び触れる必要がない。もちろん、ASP+ファイルが修正される場合は、コンパイル処理を繰り返すことによってページクラスを更新する。適切なフラグまたは他の指標メカニズムを用いてASP+ファイルが更新されたかどうかを判断することができ、したがって自動クラス更新が可能である。あるいは、ウェブページファイルまでASP+を更新した後、キャッシュされたクラスを除去するという責任は開発者に残され得る。

40

【 0 0 7 2 】

50

データモデルの作成に関連した詳細を図7に示す。データモデル作成処理700は、テスト処理702でASP+ファイルにおいて宣言があるかどうかを判断することから始まる。純粋なHTMLファイルのようなコンパイルすべき宣言がない場合、フローは、NO枝に分岐し処理714の終了に進む。

【0073】

テスト処理702でASP+ファイルに少なくとも1つの宣言があると判断されると、フローはYES枝に分岐し、ASP+ファイルで第1の宣言を得る獲得処理704に進む。第1の宣言からの情報は、格納処理706によってデータ構造に格納される。データ構造に格納された実際の宣言のみならず、宣言が子オブジェクトを有するコンテナタイプの制御を宣言するかどうか、または宣言がディレクティブなどを宣言するかどうかなどの第1の宣言に関連した情報もデータ構造に格納され得る。ウェブページは、ASP+ファイルであるので、典型的に第1の宣言としてディレクティブタグを有する。ディレクティブタグは、種々のディレクティブ、例えば図4のライン1に示すようなデリミッタ"<%@"と"%>"との間の情報を含む。図4に示すデリミッタは、単に例示的であり他の選択も可能であることを理解すべきである。ディレクティブは、ソースコードファイル作成に用いられるページファクトリモジュールに情報を供給する。例えば、図4のライン1に示すディレクティブは、用いられるべきデフォルト言語が、この例では、VB(すなわち、Visual Basic)であることを示している。

10

【0074】

第1の宣言が取り出されデータ構造に格納されると、処理700は、ファイルに別の宣言があるかどうかを判断するテスト処理708に進む。この処理は、上記の処理702と同様である。もはや宣言がなければ、フローはNO枝に分岐し終了ステップ714に進む。別の宣言があると、フローはYES枝に分岐し、獲得処理710に進み、次の宣言を獲得する。

20

【0075】

次の宣言の取り出しの次に、格納処理708は、データモデルの他のデータ構造に関連付けられた異なるデータ構造に次の宣言に関連する情報を格納する。第1の宣言に関して上で述べたように、データ構造に格納された情報は、次の宣言のリテラルテキストのみならず、リテラルテキストから派生した情報をも含み得る。したがって、データ構造は、必要に応じて階層的に第1のデータ構造に関連付けられ得る。

30

【0076】

次のデータ構造への情報の格納の次に、フローはテスト処理708に分岐し、資源ファイル、例えば、ASP+ファイルにデータモデルに読み込むべき宣言がまだ存在するかどうかを判断する。存在しない場合は、フローは、NO枝に分岐し処理の終了に進む。しかし、テスト処理708がまだ宣言が存在すると判断すると、フローはYES枝に分岐し、次の宣言を獲得する獲得処理710に進む。獲得処理710の次に、格納処理708において、上記のように、次の宣言に関連した情報を格納する。これらのステップ708、710および712は、すべての宣言が順次取り出され、データ構造に格納されるまで続く。データ構造は、入れ子のオブジェクト、子オブジェクト、ノードまたは親オブジェクトのサブ要素は、それらがサブ要素として識別できるように連結される階層として関連付けられてもよい。

40

【0077】

ソースコードモジュール608(図6)の作成には、このデータモデルを用いて、特定のソースコード言語のソースコードファイルを作成する。データモデルからのソースコードファイルの作成処理は、ソースコード言語に依存することに注目することが重要である。すなわち、特定の言語で書き込まれたソースコードファイルは、典型的に、特定のフォーマット要件、例えば、すべての可変宣言がこれらの変数の使用に先立つという要件を有する。したがって、その言語のコンパイラが呼び出されると、適切なプログラム命令がソースコードファイル内の適切な位置に配置されなければならない。この作業をなし遂げるために、ASP+ファイルを評価して、ファイルの中にどんな一意の要素または特徴があるか

50

を判断しなければならないし、またファイルを処理してソースコードファイルを生成しなければならない。ソースコードファイルは、典型的に、第1の組の要素または宣言が可変宣言のようなファイルの始めにあることを要求するので、データモデル全体が、この情報のために解析されるべきである。同様に、制御オブジェクト情報は、ファイルの中央にあるべきなので、処理の第1フェーズの次の第2フェーズ時にこの情報のためにデータモデル全体を解析すべきである。

【0078】

これらのフェーズは、ソースコードファイル（または処理607での中間データ構造）に書き込まれるべきコードの別個の部分を実行的に判断する機能的な処理であり、ここでは、別個の部分は、ソースコードファイル内の結果位置に基づき論理的に組み合わせられる。したがって、これらのフェーズは、1つのデータモデルの「パス」またはトラバースであると考えられ、したがって連続的に行われる。しかし、あるいは、これらのフェーズは、別個の並行した処理として行われてもよく、別個の処理の結果は1つのソースコードにまとめられ、処理が単一データモデルまたはデータモデルの別個のコピーのいずれかを評価する。あるいは、コードの別個の部分を実行的に判断し書き込む機能的な処理またはフェーズは、単一のトラバース時に行われてもよく、そこで、ソースコードは、ソースコードファイルの別個の部分に選択的に書き込まれたり、その部分が、ソースコードの論理的な部分間の適切な連結を提供するようにまとめられるか、関連付けられた別個のファイルに書き込まれたりする。したがって、本明細書中では、別個の解析を、実質的に連続して、すなわち次々に行われるように記載しているが、解析は実質的に同時にスプリットされたり、実行されてもよい。

10

20

【0079】

図8に示す処理またはフローは、データモデルから比較的直接的にソースコードファイルを生成する実施形態、すなわち図6に示す省略処理607に関する。本実施形態は、変数が始めに宣言され、オブジェクトおよびメソッドが変数情報に続くVBコードファイルのようなソースコードファイルを生成する。図8に示すソースコード作成処理800は、データモデルの「パス」として別個のフェーズを示し、データモデルのデータ構造を通過する3つのパスになっている。第1のパスの処理において、変数および宣言情報を探している。第2のパスの処理において、オブジェクト作成情報を探し、第3のパス処理800において、コードレンダリング情報を探している。

30

【0080】

第1のパスは、変数および他の宣言情報についてデータモデルをトラバースする処理802で始まる。このステップは、処理700で作成されたデータモデルの第1のデータ構造を解析することから始まる。第1のデータ構造の解析処理において、宣言に関するソースコードファイルに書き込まれるべきソースコードの特定ラインがあるかどうか判断する。

【0081】

第1のデータ構造の解析が終了すると、処理804において、ソースコードファイルに変数および宣言情報に関するソースコードを生成し書き込む。ライタオブジェクトが呼び出され、それに与えられたテキストをソースコードファイルに単に書き込む。したがって、ライタオブジェクトへの呼び出しは、ソースコードファイル名のパラメータと変数または宣言情報に関する特定のプログラミング言語の文法のテキストストリングとの両方を含む。あるいは、テキストは、コピーされるか、または他の公知の手段を用いてソースコードファイルに書き込まれる。

40

【0082】

また、処理804において、ライタオブジェクトへ渡すべき適切な文法を決定しなければならない。本質的に、現在のデータ構造が、サーバ側制御を表す場合は、処理804において、それに対するソースコードを生成する。そうでなければ、処理804において、それをソースコードファイルにコピーする。本質的に、処理804は、3つのうちの1つを行い得る。生成処理804において、まず、ASP+ファイルにおける情報がリテラルテキ

50

ストであるかどうか、したがってソースコードファイルに直接書き込まれるべきかどうかを判断してもよい。これは、HTMLタグがASP+ファイルに挿入される場合であり、この場合、ソースコード宣言は生成されない。その代わりに、情報は単に資源ファイルにコピーされる。処理804において、第2に、テキストが単純な翻訳、特定のソースコード言語の適切な文法フォームで1対1の対応を要求するかどうかを判断してもよい。このような処理804において、1対1対応を探すか、または、ページファクトリモジュールにハードコードされた場合は1対1翻訳を行い、結果情報をライタオブジェクトに供給する。第3に、1対1翻訳ができない場合は、処理804において、より複雑な翻訳が要求され、したがってモジュールまたは次の検索テーブル処理を呼び出し、ライタオブジェクトに渡されるべき適切なソースコード文法の作成処理を行う。典型的に、宣言情報の翻訳は直接的である。

10

【0083】

処理804において、第1のデータ構造のソースコードの生成および書き込みを行うと、テスト処理806において、データモデルに解析すべきデータ構造がまだ存在するかどうかを検知する。データモデルのこの第1のパス時に解析すべきデータ構造がまだある場合は、フローはYES枝に分岐して処理800の始めに戻り、802において次のデータ構造に対してトラバースを行う。このような場合、次のデータ構造は、上記のように解析されソースコード文法を生成し、その文法コードを資源ファイルに書き込むライタオブジェクトを呼び出す。

20

【0084】

テスト処理806において、この第1のパス時にデータモデルに解析すべきデータ構造がもう存在しないと判断した場合は、フローはNO枝に分岐してトラバース処理808に進み、データ構造を通過する第2のパスを開始する。トラバース処理808は、典型的にオブジェクト制作およびメソッド作成を書き込むコードを書き込むという次のフェーズを開始する。トラバース処理808は、データモデルの上から始まり、データモデルの最初のデータ構造に戻る。

【0085】

次に、生成および書き込みソースコード処理810において第1のデータ構造を解析し、オブジェクト作成に関する情報がその第1のデータ構造にあるかどうかを判断する。このようなオブジェクト作成情報が第1のデータ構造に存在する場合、特定のプログラミング言語の文法が作成および生成されてソースコードファイルに書き込むライタオブジェクトに送信される。本質的に、処理810は、上記の処理804と同様であり、ここでは、データ構造が特定タイプの情報について解析され、その情報がデータ構造から収集されると、何らかの翻訳を行い、その情報に関するソースコード言語を作成する。情報がリテラルである場合、処理810において、その情報をソースコードファイルに直接送ることができる。同様に、その情報が簡単な翻訳を必要とする場合は、その簡単な翻訳は、プログラムにハードコードされるか、あるいは検索テーブルタイプ処理の一部であってもよい。また、さらに複雑な構造が要求される場合は、検索テーブルまたは他のモジュールを呼び出し、情報を処理し、ソースコードファイルに適切な文法ソースコードを作成する。

30

【0086】

処理810の次に、照会ステップ812は、データモデルに解析すべきデータ構造が存在するかどうかをテストする。オブジェクト作成のために解析すべきデータ構造がデータモデルに存在する場合、フローはYES枝に分岐し次のデータ構造をトラバースするトラバース処理808に進む。したがって、処理808において、次のデータ構造に戻り、処理810において、オブジェクト作成に関する情報のデータ構造を解析する。このオブジェクト作成情報は、上記のように、ライタオブジェクトに送信され、ソースコードファイルに追加することができるソースコードラインに翻訳される。ステップ812、808および810は、すべてのソースコードがオブジェクト作成に関し書き込まれるまで繰り返される。

40

【0087】

50

判断ステップ 8 1 2 が、オブジェクト作成のために解析すべきデータ構造がデータモデルにもう存在しないと判断すると、フローはNO枝に分岐し処理 8 1 4 に進み、第 2 のパスを終了する。この第 3 のパスは、データモデルのトップの処理 8 1 4 で始まる。処理 8 0 8 および 8 0 2 に関して上で述べたように、このトラバースステップは、まず、データモデル内の第 1 のデータ構造を取り出す。処理 8 1 4 でのデータモデル内の第 1 のデータ構造の取り出しの次に、処理 8 1 6 で、コードレンダリング情報のデータ構造を解析する。コードレンダリング情報は、ライターに送信されたり、ソースコードファイルに追加させるプログラミング言語での特定の文法に翻訳される。コードレンダリングメソッドは、ソースコードファイルの終わり近くにあり、したがって、第 3 のパスがオブジェクト作成パスの次に行われる。あるいは、コードレンダリングメソッドが同時スレッド処理に間に生成され、資源ファイルの終わりに追加される。

10

【 0 0 8 8 】

処理 8 1 6 は、上記の処理 8 1 0 および 8 0 4 とかなり似ており、このソースコード情報は、データ構造にあるコードレンダリング情報から収集され得る。さらに、上記のように、情報は直接挿入されてもよく、1対1の割合で翻訳されてもいいし、より複雑なモジュールを用いてソースコードファイルの文法を生成することを要求してもよい。処理 8 1 6 の次に、検知処理 8 1 8 において、データモデルにデータ構造が残っていないかどうか判断する。

【 0 0 8 9 】

処理 8 1 8 において、この第 3 のパス間に解析すべきデータ構造がまだ存在すると検知すると、フローはYES枝に分岐し、次のデータ構造のトラバースのために 8 1 4 に戻る。次のデータ構造が呼び出され、処理 8 1 8 において、次のデータ構造のコードレンダリング情報に関するソースコードの生成および書き込みを行う。

20

【 0 0 9 0 】

除り 8 1 8 において、解析すべきデータ構造がもはや存在しないと判断すると、フローはNO枝に分岐し、処理連結 8 2 0 で処理を終了する。

【 0 0 9 1 】

上記の説明により明らかなように、数回のパスがデータモデルに対して行われる。これは、データモデルの各宣言が、ソースコードファイルの種々の場所に置かれるコードの特定のラインを要求することによる。しかし、ソースコードファイルを書き込む際、ライターオブジェクトは、ファイルに連続的に追加して順次書き込むことしかできず、予め書き込まれたコードライン間にソースコードラインを挿入できない。したがって、データモデルは、ソースコードの第 1 または上部、ソースコードの第 2 または中間部、およびソースコードファイルの第 3 または最後の部分に属する適切な情報を収集するために 1 回以上トラバースしなければならない。パスの数は、種々のコンピュータ言語に対し存在するコードの部分の数によって異なることが予想される。

30

【 0 0 9 2 】

図 9 に示す処理またはフローは、データモデルから中間データ構造を生成し、次いで、中間データ構造からソースコードファイルを生成する実施形態、すなわち図 6 の処理 6 0 7 および 6 0 8 の別の実施形態に関する。本実施形態のフロー 9 0 0 は、処理 8 0 4、8 1 0 および 8 1 6 が特定の言語のソースコードを生成し、処理 9 0 4、9 1 0 および 9 1 6 が、ソースコードを生成するのではなく、中間データ構造の部分を生成することを除いて図 8 のフローと同様である。したがって、処理 9 0 4、9 1 0 および 9 1 6 が、データモデルでの情報に関する情報を生成する際に処理 8 0 4、8 1 0 および 8 1 6 と同様の処理を行う。しかし、生成された情報は包括的であり、後に異なるソースコード言語のソースコードファイルを作成するのに用いられることがある。

40

【 0 0 9 3 】

データモデルに解析すべきデータ構造がもはや存在しないと判断されると、中間データ構造が完成し、フローは分岐して作成処理 9 2 0 に進む。作成処理において、中間データ構造からソースコードファイルを作成する。データ構造はソースコードファイルの包括的

50

な記述であるので、処理 9 2 0 において中間データ構造を包括的な記述から特定の言語コードファイルに翻訳する。

【 0 0 9 4 】

図 1 0 は、ウェブページ開発者によって製作されたウェブページ資源ファイル、すなわちASP+ファイルの例である。ファイルは、ソースコードファイルを生成するための処理 6 0 0 のサブジェクトである。図 1 0 に示すASP+ページは、ライン 1 にディレクティブライン、ライン 3 から 1 3 にサーバ側スクリプトブロックおよびライン 1 6 から 2 1 にサーバ側コントロール宣言ブロックを有する。ディレクティブラインは、どのようなタイプのソースコード言語を用い、それに一般的な記述を与える（そうでなければソースコードファイルのどのコードにもならない）かを判断するために処理 6 0 0 において用いられる。コード宣言ブロック"<script runat=server> </script>"内に書かれたすべてのコードは、概念的にページメンバ宣言（変数、プロパティ、メソッド）として扱われ、図 1 1 に示すような生成ファイルの資源ファイルに直接挿入される。

10

【 0 0 9 5 】

図 1 1 は、図 1 0 に示すASP+ファイルを用いてソースコードを作成する処理によって作成されたソースコードファイルの一例である。図 1 1 の第 1 のラインは、ソースコードファイルによって作成しようとしているページクラスは、コンパイルされたとき、System.ASP.WebForms.Pageから引き継いでいることを示している。このクラスからの引継ぎは、ページクラスの多くの制御機能を提供するので重要なステップである。初期状態として、生成された資源ファイルは"System.ASP.WebForms.Page"ベースクラスのサブクラスである。任意には、開発者は、ディレクティブラインに与えられた"Inherits"属性を用いて代替クラスを特定することができる。

20

【 0 0 9 6 】

ライン 3 および 4 は、第 1 のパスでデータモデルを通過している間に作成された可変宣言を示している。

【 0 0 9 7 】

第 2 のパス時にライン 6 に示された情報が作成され、新しい制御オブジェクト"DataList"を作成し、それをMyDataと名づける。ある実施形態において、ASP+ファイル宣言ブロック"<script runat=server> </script>"内に書き込まれているすべてのコードは、ページメンバ宣言（変数、プロパティ、メソッド）として扱われ、上記のような生成ファイルの資源ファイルに直接挿入される。このように、コード 8 - 1 3 のラインは、リテラルテキストとして直接挿入された。データモデル 6 0 6 の作成の間、リテラルテキストに関連する情報は、多数のパスの間トラバースされないように、本質的にデータモデルの上部または離れて、ストリングまたはアレイで格納された（図 8 ）。データモデルの作成時に、この情報は直接挿入されると判断され、データモデルにリファレンスポインタを有するアレイとしてそれを格納することができる。

30

【 0 0 9 8 】

ライン 1 5 ~ 3 9 は、制御オブジェクト情報構築に関する第 2 のパスの間に、コードの 1 セクションが書き込まれた。ライン 1 5 ~ 2 0 は、トップレベルのオブジェクトの作成において用いられるコードを表す。トップレベルのオブジェクトは、図 3 に示すようなページオブジェクト 3 1 4 のようなページ全体のコンテナタイプのオブジェクトである。この制御オブジェクトは、作成されたASP+ページはこのタイプの制御オブジェクトを有するので、図 1 0 に示す実質的なコードから比較的独立して構築される。

40

【 0 0 9 9 】

図 1 1 のライン 2 2 ~ 2 7 は、MyListと名づけられた子制御オブジェクトの作成に用いられるコードを表す。図 1 0 のライン 1 6 に示すように、テーブルリスト制御が定義され、識別子"MyList"を与えられた。図 1 0 のライン 1 6 からの情報は、ステップ 8 1 0（図 8）において図 1 1 に示されるライン 2 2 ~ 2 7 に翻訳される。本質的に、処理 8 1 0 において、図 1 0 のライン 1 6 のコード部分の構文解析すると、"tablelist"タイプの制御を作成すべきであると認識される。認識されると、図 1 1 に示されるライン 2 2 ~ 2 7 の

50

情報は、テーブルから検索するか、またはライン 22 ~ 27 に示されたコードを生成するための "MyList" のような適切な変数を挿入して公知の計算を用いて生成される。システムは、このタイプの制御を認識するように設計されているので、図 11 に示されたコードを生成することができる。一旦、生成されると、処理 810 は、そのコードもファイルに書き込む。

【0100】

同様に、図 11 のライン 29 ~ 34 は、別の制御オブジェクト、この場合は、"template" タイプのオブジェクトを表す。テンプレートは、それらもまたコンテナ制御であり、実行時間に実際に生成されるという点で特別なオブジェクトである。いずれにしても、テンプレートの制御オブジェクトの生成に必要なコードがステップ 810 (モデルにまだデータ構造が存在するかどうかを判断するテスト 812 に引き続いて) において生成される。上記のように、テンプレート制御オブジェクトが生成されるべきであり、図 11 に示すコードが生成されファイルに書き込まれているという認識がなされる。重要なことは、図 11 のライン 32 および 33 の "control3" へのリファレンスによって明らかのように、テンプレートのコードは、子制御に関連する情報を必要とする。したがって、データモデルが 606 (図 6) で作成されると、この情報またはこの情報を決定する指標が、テンプレート制御情報とともに格納された。また、コードは、コードの生成が直接的であるようにテンプレート制御オブジェクトに対して予め決められている。

10

【0101】

子制御が、図 10 のライン 19 のコードおよび図 11 のライン 36 ~ 39 でのそれに対応する翻訳されたコードを用いてテンプレート内に作成される。図 10 のライン 19 は、識別子 "MyLabel" を有する "label" タイプの制御を呼び出す。テーブルリスト制御およびテンプレート制御と同様に、ラベル制御は、処理 810 が認識する公知のタイプの制御であり、次いで適切なコードを生成し、それを図 11 のライン 36 ~ 39 に示すようなファイルへ書き込むことができる。

20

【0102】

データモデルの最後のパス時にライン 41 ~ 51 が作成および生成され、ソースコードファイルに書き込まれた。ライン 41 ~ 51 におけるコードのラインは、クライアントへのレスポンスの一部になる HTML コードをレンダリングするために呼び出されるレンダリングメソッドを表している。レンダリングコードは、ステップ 816 でのこのようなコードを生成すべきであるという認識に基づき生成される。一旦認識されると、コードは単に検索テーブルから判断されるか、または必要に応じて生成される。

30

【0103】

図 11 に示すコードは、特定の動的ウェブページファイル、すなわち図 10 に示すファイルからのサーバ側生成コードを表している。図 11 に示すファイルが完成すると、ファイルは、図 6 に示す処理 610 に関して上に述べたようにコンパイルされてもよい。コンパイルが、動的ウェブページコンテンツファイルの制御オブジェクト制御を生成するのに用いることができるクラスとなる。クラスは、キャッシュ、または他のメモリに格納され、ページのためのオブジェクトのインスタンスを作成するために所望の回数用いることができる。結果クラスのコンパイルおよびパーシスタンスのいずれも、ディスクへのアクセスが必要でないように「インメモリ」で行われ得、これは、一般にメモリで行うより時間がかかる。

40

【0104】

サーバ側ページのセットアップ、起動および実行に必要なすべての処理は、動的にコンパイルされたページクラス内にカプセル化される。その結果、ページセットアップの間にファイルの追加的な構成 / 構文解析は必要でない。さらに、元々の ".aspx" または ASP+ ファイルは再び扱われることはなく、「実行時間ホスト環境」、例えば、ASP スクリプトエンジンは、ページ実行処理の間要求されない。

【0105】

本明細書中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

50

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。

【0106】

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。

10

【図面の簡単な説明】

【0107】

【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ファイル(例えば、ASP+ページ)の一例を示す。

20

【図5】本発明のある実施形態を実行するの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プロセスフローチャートを示す。

【図7】データモデルを作成するためのサーバ側アプリケーションの構文解析を表すプロセスフローチャートを示す。

【図8】ソースコードファイルを作成するためのデータモデルのトラバースを表すプロセスフローチャートを示す。

【図9】本発明の別の実施形態におけるソースコードファイルを作成するためのデータモデルのトラバースを表すプロセスフローチャートを示す。

【図10】本発明による構文解析され得るASP+ページの一例を示す。

30

【図11】図10に示す初期ASP+ファイルで本発明を用いて作成されたソースコードファイルの一例を示す。

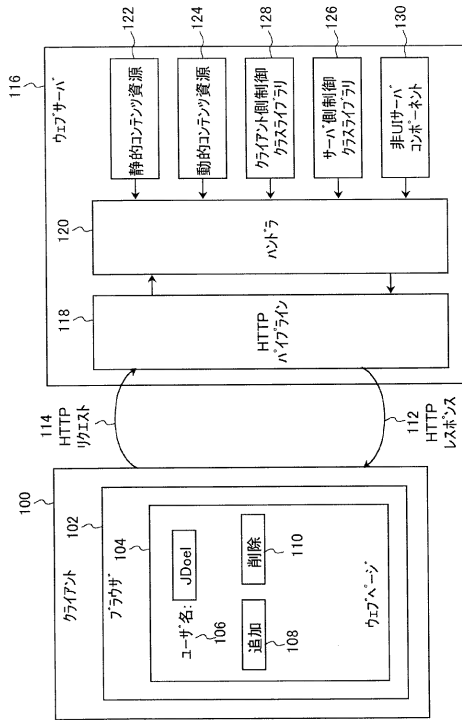
【符号の説明】

【0108】

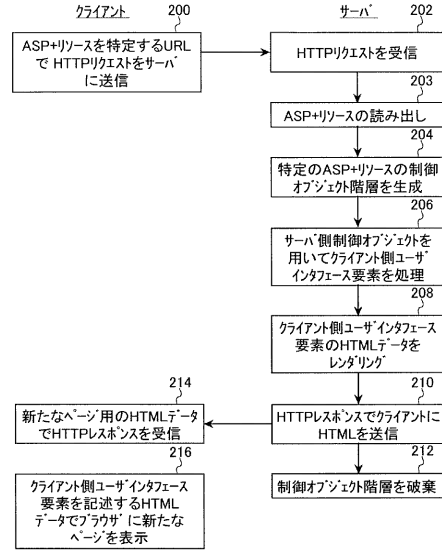
- 100 クライアント
- 102 ブラウザ
- 104 ウェブページ
- 106 テキストボックス制御
- 108、110 ボタン制御
- 112 HTTPレスポンス
- 114 HTTPリクエスト
- 116 ウェブサーバ
- 118 HTTPパイプライン
- 120 ハンドラ
- 122 静的コンテンツ資源
- 124 動的コンテンツ資源
- 126 サーバ側制御クラスライブラリ
- 128 クライアント制御側クラスライブラリ
- 130 非ユーザインタフェースサーバコンポーネント

40

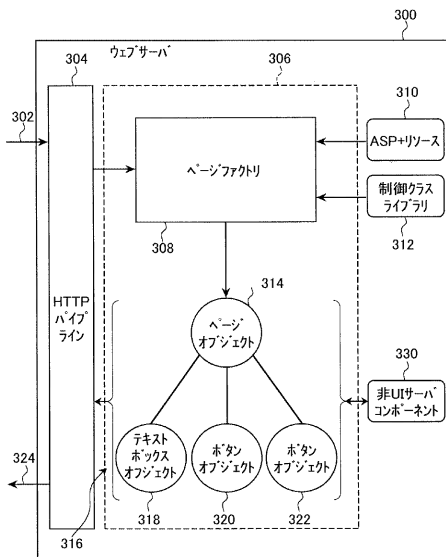
【図 1】



【図 2】



【図 3】



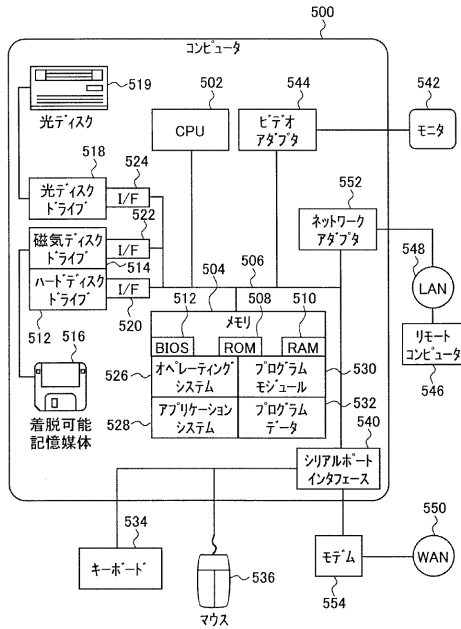
【図 4】

```

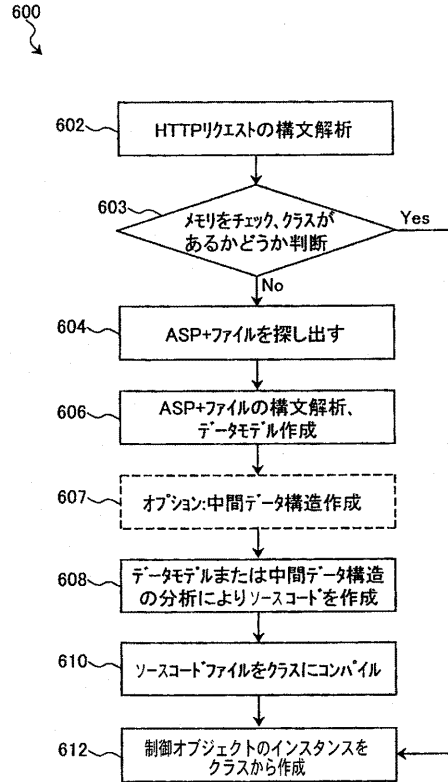
1 <%@ Page Language="VB" Description="Simple Sample Page" ErrorPage="ErrorPage.aspx" %>
2 <html>
3   <script runat="server">
4     Sub AddButton_Click(ByVal Source as Object, By Val E as EventArgs)
5       Message.Text = "Add" & UserName.Text
6     End Sub
7   Sub DeleteButton_Click(ByVal Source as Object, By Val E as EventArgs)
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>

```

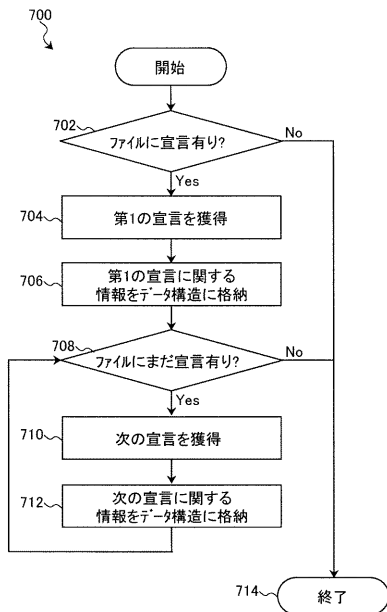
【図5】



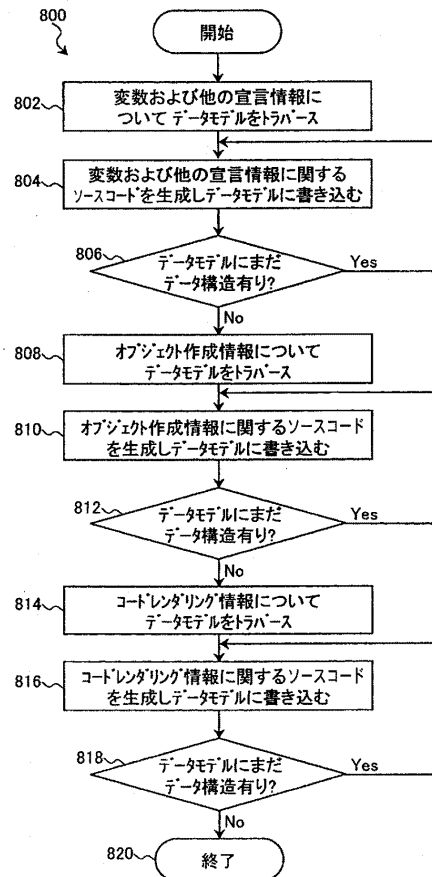
【図6】



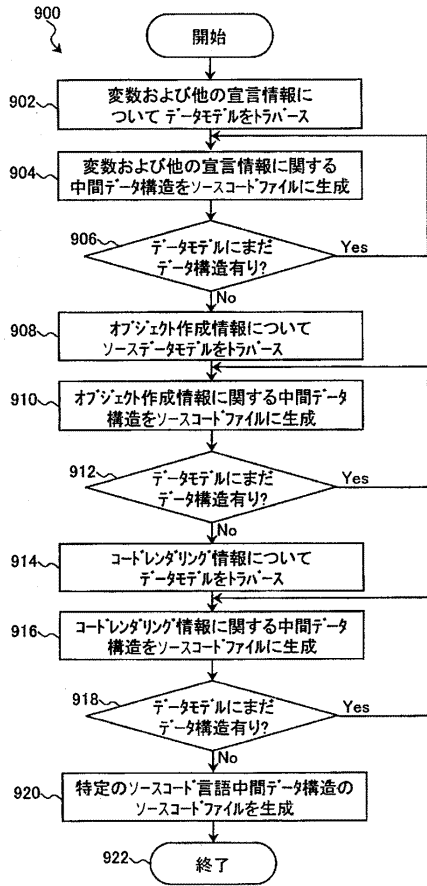
【図7】



【図8】



【 図 9 】



【 図 1 0 】

```

1 <%@ language="VB" Description="Test of the TableList control" %>
2
3 <script runat=server>
4     public MyData as New DataList
5
6     Overrides Sub Init()
7         MyData.Add "Name1"
8         MyData.Add "Name2"
9         MyData.Add "Name3"
10
11     Set MyList.DataSource = MyData
12     End Sub
13 </script>
14
15 <%>
16 <wfc:TableList id="MyList" runat=server>
17     <template:ItemTemplate runat=server>
18 <%>
19     <wfc:Label id="MyLabel" databinding="text:DataItem" runat=server/>
20 </template:ItemTemplate>
21 </wfc:TableList>
  
```

【 図 1 1 】

```

1 Inherits System.ASP.WebForms.Page
2
3 Dim MyList as TableList
4 Dim __control3 as Label
5
6 public MyData as New DataList
7
8 Overrides Sub Init()
9     MyData.Add "Name1"
10    MyData.Add "Name2"
11    MyData.Add "Name3"
12    Set MyList.DataSource = MyData
13 End Sub
14
15 Public Sub __tmp_aspx0__BuildControl__control0(ByVal __ctrl as ContainerControl)
16    __ctrl.SetRenderMethodDelegate New RenderMethod(AddressOf
17    Me.__tmp_aspx0__Render__control0)
18    __tmp_aspx0__BuildControlMyList
19    __ctrl.AddParsedSubControl MyList
20 End Sub
21
22 Public Sub __tmp_aspx0__BuildControlMyList
23    set MyList = new TableList
24    MyList.ID = "MyList"
25    set MyList.ItemTemplate = new CompiledTemplateBuilder(new
26    BuildTemplateMethod(AddressOf me.__tmp_aspx0__BuildControl__control2))
27 End Sub
28
29 Public Sub __tmp_aspx0__BuildControl__control2(ByVal __ctrl as ContainerControl)
30    __ctrl.SetRenderMethodDelegate New RenderMethod(AddressOf
31    Me.__tmp_aspx0__Render__control2)
32    __tmp_aspx0__BuildControl__control3
33    __ctrl.AddParsedSubControl __control3
34 End Sub
35
36 Public Sub __tmp_aspx0__BuildControl__control3
37    set __control3 = new Label
38    __control3.ID = "MyLabel"
39 End Sub
40
41 Public Sub __tmp_aspx0__Render__control0(ByVal output as HtmlTextWriter,
42 ByVal __container as ContainerControl)
43    Call __container.Controls(0).RenderControl(output)
44    output.Write("&vbCRLF &""&vbCRLF &""")
45 End Sub
46
47 Public Sub __tmp_aspx0__Render__control2(ByVal output as HtmlTextWriter,
48 ByVal __container as ContainerControl)
49    Call __container.Controls(0).RenderControl(output)
50    output.Write("&vbCRLF &""")
51 End Sub
  
```

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 ピーターズ、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
- Fターム(参考) 5B082 HA05 HA08

【外国語明細書】

2009070372000001.pdf