(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 601 26 016 T2** 2007.07.12

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 156 415 B1**
(21) Deutsches Aktenzeichen: **601 26 016.3**
(96) Europäisches Aktenzeichen: **01 111 681.1**
(96) Europäischer Anmeldetag: **14.05.2001**
(97) Erstveröffentlichung durch das EPA: **21.11.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **17.01.2007**
(47) Veröffentlichungstag im Patentblatt: **12.07.2007**

(51) Int Cl.[8]: ***G06F 9/44*** (2006.01)

(30) Unionspriorität:
573769 **18.05.2000** **US**

(73) Patentinhaber:
Microsoft Corp., Redmond, Wash., US

(74) Vertreter:
Meissner, Bolte & Partner GbR, 80538 München

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE, TR

(72) Erfinder:
Burd, Gary S., Kirkland, Washington 98033, US; Cooper, Kenneth B., Seattle, Washington 98199, US; Guthrie, Scott D., Bellevue, WA 98004, US; Ebbo, David S., Redmond, Washington 98052, US; Anders, Mark T., Bellevue, Washington 98007, US; Peters, Ted A., Seattle, Washington 98119, US; Millet, Stephen J., Edmonds, Washington 98026, US

(54) Bezeichnung: **Serverseitige Kontrollobjekte zur Verarbeitung von kundenseitigen Benutzerschnittstellenelementen**

Beschreibung

Gebiet der Technik

[0001] Die vorliegende Erfindung betrifft allgemein eine Webserver-Grundstruktur und speziell serverseitige Steuerobjekte, die clientseitige Benutzeroberflächenelemente einer Webseite verarbeiten.

Hintergrund der Erfindung

[0002] Ein typischer Webbrowser empfängt Daten von einem Webserver, der das Erscheinungsbild und das rudimentäre Verhalten einer Webseite zur Anzeige auf einem Clientsystem definiert. In einem typischen Szenario spezifiziert ein Benutzer eine URL-Adresse, eine globale Adresse einer Ressource im World Wide Web, um Zugang zu einer gewünschten Website zu erhalten. Im allgemeinen bezieht sich der Ausdruck "Ressource" auf Daten oder Routinen, auf die von einem Programm zugegriffen werden kann. Eine beispielhafte URL-Adresse ist "http://www.microsoft.com/ms.htm". Der zweite Teil bezeichnet den Domäne-Namen (d. h. "www.microsoft.com"), wo sich die Ressource befindet. Der dritte Teil bezeichnet die Ressource (d. h. eine Datei mit der Bezeichnung "ms.htm") innerhalb der Domäne. Somit erzeugt ein Browser eine HTTP (HyperText Transport Pxotocol)-Anfrage, die der beispielhaften URL-Adresse zugeordnet ist, um die zu der ms.htm-Datei innerhalb der www.microsoft.com-Domäne gehörigen Daten abzurufen. Ein Webserver, der für den Betrieb der www.microsoft.com-Site zuständig ist, empfängt die HTTP-Anfrage und sendet die angeforderte Webseite oder Ressource in einer HTTP-Antwort an das Clientsystem zur Anzeige im Browser zurück.

[0003] Die "ms.htm"-Datei des obigen Beispiels enthält einen statischen HTML(HypexText-MarkupLanguage)-Code. HTML ist eine Klartext-Autorensprache, die zum Erzeugen von Dokumenten (z. B. Webseiten) im World Wide Web verwendet wird. Als solche kann eine HTML-Datei von einem Webserver abgerufen und als Webseite in einem Browser angezeigt werden, um die umfangreichen grafischen Erfahrungen darzustellen, die Anwender inzwischen erwarten, wenn sie Informationen aus dem Internet betrachten. Unter Anwendung von HTML kann ein Entwickler beispielsweise formatierten Text, Listen, Formulare, Tabellen, Hypertext-Links, Inlinebilder und -töne sowie Hintergrundgrafiken zur Anzeige im Browser bezeichnen. Eine HTML-Datei ist jedoch eine statische Datei, die nicht inhärent die dynamische Erzeugung von Webseiteninhalten unterstützt.

[0004] In manchen Fällen kann es notwendig sein, daß eine Webseite dynamischen Inhalt in einem Browser anzeigt, etwa veränderliche Börsenkurs- oder Verkehrsinformationen. In solchen Situationen wird typischerweise ein serverseitiges Anwendungsprogramm entwickelt, um die dynamischen Daten zu erhalten und sie in HTML zu formatieren und an den Browser zu übermitteln, um in einer Webseite angezeigt zu werden, während die Webseite aktualisiert wird.

[0005] Zusätzlich können dieselben serverseitigen Anwendungsprogramme in Situationen verwendet werden, in denen die Information nicht strikt dynamisch ist, wo es jedoch so viele verschiedene Werte gibt, die in einer statischen Webseite angezeigt werden können, daß es impraktikabel wäre, die erforderliche Anzahl von statischen Webseiten zu erzeugen. Beispielsweise kann eine Reiseplanungsseite zwei Kalender zeigen: einen Kalender für Abreisedaten und einen Kalender für Rückreisedaten. Anstatt Hunderte von statischen Seiten mit jedem möglichen Paar von Kalenderkombinationen zu entwickeln, kann ein serverseitiges Anwendungsprogramm die entsprechende statische Seite mit Anzeige der entsprechenden Kalender dynamisch erzeugen.

[0006] Viele Webseiten erlauben dem Benutzer den Dialog mit der im Browser angezeigten Seite durch die Wahl von visuellen Elementen der Seite. In der oben erwähnten Reiseplanungsseite könnte beispielsweise der Kalender dem Benutzer den Dialog mit dem Kalender erlauben durch Anklicken eines Tags, um dieses Datum zu wählen, oder durch Anklicken eines Piktogramms, um einen Monat weiter oder zurück zu gehen. Bei vorhandenen Lösungen sendet ein Browser eine HTTP-Anfrage zurück zum serverseitigen Anwendungsprogramm. Die HTTP-Anforderung kann Parameter enthalten, die in der Anfragekette codiert sind, und zwar als Formularpostvariablen oder in einem anderen Datenformat, um die clientseitigen Ereignisse oder Daten zu beschreiben (z. B. welche Steuerung der Benutzer angeklickt hat). Beispielsweise könnten die Parameter das Datum enthalten, das der Benutzer in dem einen Kalender gewählt hat, zusammen mit dem Datum, das aktuell in dem anderen Kalender angezeigt wird.

[0007] Die Übermittlung von Ereignissen und Daten zurück zum Server wird als "Postback" bezeichnet, weil der Browser typischerweise die Anforderung unter Verwendung einer HTTP POST-Anforderung sendet. Das serverseitige Anwendungsprogramm kann die HTTP-Anforderung verarbeiten und den entsprechenden HTML-Code für eine Webseite mit einem neu berechneten Kalender, der die Benutzeraktion reflektiert, erzeu-

gen, so daß dieser in einer HTTP-Antwort zum Client übertragen wird. Danach wird das resultierende Dokument zu einem Clientsystem in einer HTTP-Antwort übermittelt, wo es im Browser als Webseite dargestellt wird, welche die aktualisierten Kalender zeigt.

[0008] Die Entwicklung eines serverseitigen Anwendungsprogramms kann eine komplexe Aufgabe sein, bei der nicht nur die Vertrautheit mit normaler HTML-Codierung, die für das Layout einer Webseite verwendet wird, sondern auch mit den Grundlagen der Programmierung erforderlich sind, was eine oder mehrere Programmiersprachen (z. B. C++, Perl, Visual Basic oder Jscript) und das HTTP-Protokoll einschließt, wie Daten zwischen Browser und Server gesendet werden. Webseiten-Designer dagegen sind häufig Grafik-Designer und -Editoren, die möglicherweise keine Programmiererfahrung haben. Außerdem kann die Vereinfachung der Entwicklung komplexer Webseiten die Entwicklung von neuen Webinhalten durch jeden Entwickler beschleunigen.

[0009] Im allgemeinen erfordert die Entwicklung eines kundenspezifischen serverseitigen Anwendungsprogramms auch eine enorme Anstrengung, und zwar derart, daß Entwickler häufig nicht geneigt sind, diese Aufgabe anzugehen. Ein Entwickler muß nicht nur den HTML-Code verstehen, der erzeugt werden muß, um eine gewünschte Webseite anzuzeigen, sondern der Entwickler muß auch verstehen, wie Benutzerdialog- und Clientdaten von der Webseite in Postback-Operationen resultieren. Es ist daher erwünscht, eine Entwicklungs-Grundstruktur bereitzustellen, die es einem Entwickler ermöglicht, eine Webseite mit minimaler Programmierung dynamisch zu erstellen und zu verarbeiten.

[0010] Eine Vorgehensweise zur Minimierung der Programmierungsanforderungen für die dynamische Webseitenerzeugung ist die Active Server Page- bzw. ASP-Grundstruktur, die von Microsoft Corporation bereitgestellt wird. Eine ASP-Ressource weist typischerweise Visual Basic oder Jscript-Code auf, um beispielsweise eine HTTP-Anforderung zu verarbeiten, welche die ASP-Ressource als die gewünschte Ressource bezeichnet, und danach den resultierenden HTML-Code in einer HTTP-Antwort zum Client zu erzeugen. Ferner kann eine ASP-Ressource Bezug nehmen auf bereits entwickelte oder von Dritten stammende clientseitige Bibliothekskomponenten (z. B. clientseitige ACTIVEX-Steuerungen), um eine gegebene Anwendungsprogrammieraufgabe zu erleichtern. bei den derzeitigen serverseitigen Anwendungs-Grundstrukturen kann jedoch die Programmierung, die notwendig ist, um clientseitige Benutzeroberflächenelemente (z. B. Textboxen, Listenboxen, Tasten, Hypertext-Links, Bilder, Töne etc.) innerhalb von serverseitigen Anwendungen dynamisch zu managen, anspruchsvolle Programmierfähigkeiten und beträchtliche Anstrengungen erfordern. Ein ungelöstes Problem ist die richtige Kapselung der Programmierung, die notwendig ist, um Benutzeroberflächenelemente einschließlich der Handhabung von Postback-Ereignissen zu verarbeiten, damit der Webseiten-Entwickler sich auf andere Aspekte der Webseite konzentrieren kann.

[0011] Weitere Informationen in bezug auf den Stand der Technik sind ersichtlich aus der US-PS 5 991 802, die ein Verfahren und ein System angibt, um durch ein Clientcomputersystem eine Funktion eines Objekts einer Objektklasse aufzurufen, die von einem Servercomputersystem bereitgestellt wird. Der Client sendet eine Anforderung an einen Server, die eine URL-Adresse aufweist, die ein Skript, eine Objektklasse und eine Funktion der aufzurufenden Objektklasse bezeichnet. Als Antwort auf den Empfang der Anforderung startet der Server das Skript und überträgt die Steuerung auf das Skript. Das Skript instanziert ein Objekt der in der URL-Adresse der empfangenen Anforderung erkannten Objektklasse und ruft die in der URL-Adresse der empfangenen Anforderung erkannte Funktion auf. Die aufgerufene Funktion führt das Verhalten der Funktion aus, erzeugt eine an den Client zu sendende Antwort und sendet die Antwort an den Client. Die Antwort enthält Zustandsinformation, die einen Zustand des Objekts beschreibt, nachdem das Verhalten der Funktion ausgeführt ist. Wenn der Client anschließend eine Anforderung sendet, eine Funktion der Objektklasse aufzurufen, wird die Zustandsinformation in die Anforderung eingefügt, so daß die Funktion ihr Verhalten auf der Basis der Zustandsinformation ausführen kann.

[0012] WO 98/21651 betrifft eine gattungsgemäße Softwarezustandsmaschine zur Implementierung einer Softwareanwendung in einer objektorientierten Umgebung und umfaßt ein Set von Entitätsobjekten, die für Softwareelemente der Softwareanwendung definiert sind, ein Set von Zustandsobjekten, die für jedes Entitätsobjekt definiert und für Zustände repräsentativ sind, in die das Softwareelement eintreten kann, und ein Set von Ereignisobjekten, die für jedes Zustandsobjekt definiert sind, das für Eingaben, welche das Softwareelement empfangen kann, oder für Aktionen repräsentativ ist, die das Softwareelement antreffen kann, während es in dem durch das Zustandsobjekt repräsentierten Zustand ist. Ein Verfahren und ein System zum Aufbau dynamischer Objekte für eine Anwendungssoftware umfaßt die folgenden Schritte: Lesen und Parsen einer Hauptkonfigurationsdatei durch ein Masterobjekt, Erhalten einer Objekt-ID für jedes in der Hauptkonfigurationsdatei bezeichnete dynamische Objekt, mittels eines Werkobjekts Erzeugen eines Beispiels jedes dynami-

schen Objekts, das in der Hauptkonfigurationsdatei bezeichnet ist, und Erhalten einer physikalischen Adresse für jedes erzeugte Objekt, Abspeichern der Objekt-IDs und der physikalischen Adressen der erzeugten Objektbeispiele in einem Objektlexikon, Aufrufen des Initialisierungsverfahrens jedes in dem Objektlexikon abgespeicherten Objekts und Initialisieren jedes erzeugten Objekts.

[0013] "Chapter 6 Session Management Service", Server-Side JavaScript Guide, [Online] 1999, S. 125-126, XP002193608, abgerufen aus dem Internet am 19. März 2002 von URL: http://developer.netscape.com/docs/manuals/ssjs/1:4/ssjs.pdf beschreibt die Session Management Service-Objekte, die im serverseitigen JavaScript verfügbar sind, um Daten gemeinsam unter einer Vielzahl von Client-Anforderungen an eine Anwendung, unter einer Vielzahl von Benutzern einer einzigen Anwendung oder auch unter einer Vielzahl von Anwendungen auf einem Server zu nutzen.

Zusammenfassung der Erfindung

[0014] Gemäß der vorliegenden Erfindung werden die vorstehenden und weitere Probleme gelöst durch Bereitstellen eines serverseitigen Steuerobjekt-Grundgerüsts zum Verwalten der Verarbeitung und Erzeugung von clientseitigen Benutzeroberflächenelementen. Ferner kann eine Hierarchie von serverseitigen Steuerobjekten zusammenwirken zum Erzeugen des resultierenden Autorensprache-Codes wie etwa Standard-HTML zum Anzeigen einer Webseite an einem Client. Der Client kann beispielsweise irgendein Browser sein, der Standard-HTML oder eine andere Autorensprache unterstützt. Die Operation des Verarbeitens des clientseitigen Benutzeroberflächenelements kann eine oder mehrere von einer Postback-Ereignisbehandlungsoperation, einer Postback-Datenbehandlungsoperation, einer Datenbindungsoperation oder einer Zustandsmanagementoperation sein, die auf den Zustand eines serverseitigen Steuerobjekts bezogen ist.

[0015] Ein erheblicher Nutzen einer Ausführungsform der vorliegenden Erfindung liegt in der verbesserten Kapselung der serverseitigen Verarbeitung von clientseitigen Benutzeroberflächenelementen (z.B. Eingängen, die von Benutzeroberflächenelementen empfangen werden, und Ausgängen, die zum Erzeugen von Benutzeroberflächenelementen dienen) und der zugehörigen Funktionalität. Ein oder mehr serverseitige Steuerobjekte können erzeugt werden, um ein oder mehr Benutzeroberflächenelementen logisch zu entsprechen. Beispielsweise kann mit einem gegebenen Benutzeroberflächenelement, das einen Monat in einem Kalenderdisplay darstellt, eine Hierarchie von serverseitigen Steuerobjekten erzeugt werden, die dem Kalenderdisplay und seinen verschiedenen Unterelementen entsprechen. Bei einer Konfiguration kann ein "Monat"-Steuerobjekt hierarchisch eine Vielzahl von "Wochen"-Steuerobjekten enthalten, wobei jedes "Wochen"-Steuerobjekt hierarchisch sieben "Tag"-Steuerobjekte enthält.

[0016] Bei einer Ausführungsform der vorliegenden Erfindung können außerdem die serverseitigen Steuerobjekte zusammenwirken zum Verarbeiten der logisch entsprechenden Benutzeroberflächenelemente. Dieser Vorteil resultiert zum Teil aus der gleichzeitigen Existenz einer Vielzahl von serverseitigen Steuerobjekten während der Verarbeitung einer Clientanforderung und der Erzeugung einer Antwort. Beispielsweise kann bei Detektierung eines ersten Postbackereignisses, das vom Client empfangen wird (z. B. einem Klick auf ein "nächster-Monat"-Tastenelement in einer Kalenderanzeige), ein Steuerobjekt ein zweites Ereignis hervorrufen (z. B. die Wahl des nächsten Monats zur Anzeige durch das "Monat"-Steuerobjekt), das anschließend von ein oder mehr gleichzeitig existierenden Steuerobjekten detektiert und verarbeitet wird. Diese Kooperation ermöglicht die Kapselung von komplexen Steuerdialogen innerhalb der serverseitigen Steuerobjekte selbst, wodurch die vom Webseiten-Entwickler verlangte kundenspezifische Ereignisbehandlung minimiert wird.

[0017] Ein Verfahren und ein Computerprogrammprodukt, die ein clientseitiges Benutzeroberflächenelement, das in eine auf einem Client gezeigte Webseite eingefügt ist, verarbeiten, werden bereitgestellt. Eine Anforderung, die auf einen serverseitigen Vereinbarungsdatenspeicher bezogen ist, wird empfangen. Eine Vereinbarung wird von dem serverseitigen Vereinbarungsdatenspeicher eingegeben. Ein serverseitiges Steuerobjekt wird erzeugt und programmiert, um auf der Basis der Vereinbarung eine Funktionalität des Benutzeroberflächenelements bereitzustellen. Das Benutzeroberflächenelement wird unter Nutzung des serverseitigen Steuerobjekts verarbeitet. Autorensprache-Daten werden von dem serverseitigen Steuerobjekt erzeugt, um das Benutzeroberflächenelement in der Webseite anzuzeigen.

[0018] Eine Hierarchie von serverseitigen Steuerobjekten, die von einem Computer ausführbar sind, um ein oder mehr clientseitige Benutzeroberflächenelemente zu verarbeiten, die in eine an einem Client gezeigte Webseite eingebaut sind, wird bereitgestellt. Ein oder mehr serverseitige Sohn-Objekte entsprechen den ein oder mehr clientseitigen Benutzeroberflächenelementen. Jedes serverseitige Sohn-Objekt behandelt Eingänge, die von dem Client empfangen werden, und erzeugt Autorensprache-Daten zur Anzeige des clientseitigen

Benutzeroberflächenelements an dem Client. Mindestens eine hierarchische Kennung wird von dem Client im Zusammenwirken mit dem Eingang empfangen und bezeichnet eines der serverseitigen Sohn-Objekte. Ein serverseitiges Seiten-Objekt enthält die serverseitigen Sohn-Objekte und empfängt den Eingang zur Verteilung an eines der serverseitigen Sohn-Objekte in Übereinstimmung mit der hierarchischen Kennung.

Kurze Beschreibung der Zeichnungen

[0019] Fig. 1 zeigt einen Webserver zum dynamischen Erzeugen von Webseiteninhalt zur Anzeige an einem Client bei einer Ausführungsform der vorliegenden Erfindung;

[0020] Fig. 2 zeigt ein Flußdiagramm von Operationen zum Verarbeiten und Bereitstellen von clientseitigen Benutzeroberflächenelementen unter Verwendung von serverseitigen Steuerobjekten bei einer Ausführungsform der vorliegenden Erfindung;

[0021] Fig. 3 zeigt beispielhafte Module in einem Webserver, die bei einer Ausführungsform der vorliegenden Erfindung verwendet werden;

[0022] Fig. 4 zeigt eine beispielhafte Ressource mit dynamischem Inhalt (z. B. eine ASP+-Ressource) bei einer Ausführungsform der vorliegenden Erfindung;

[0023] Fig. 5 zeigt ein beispielhaftes System, das zur Implementierung einer Ausführungsform der vorliegenden Erfindung brauchbar ist;

[0024] Fig. 6 ist ein Prozeßflußdiagramm, das die Verarbeitung eines Seiten- bzw. Page-Objekts bei einer Ausführungsform der vorliegenden Erfindung darstellt;

[0025] Fig. 7 zeigt eine beispielhafte serverseitige Steuer- bzw. Controlklasse bei einer Ausführungsform der vorliegenden Erfindung;

[0026] Fig. 8 zeigt eine beispielhafte serverseitige ContainerControl-Klasse bei einer Ausführungsform der vorliegenden Erfindung;

[0027] Fig. 9 zeigt eine beispielhafte serverseitige ControlCollection-Klasse bei einer Ausführungsform der vorliegenden Erfindung;

[0028] Fig. 10 zeigt eine beispielhafte serverseitige Seiten- bzw. Page-Klasse bei einer Ausführungsform der vorliegenden Erfindung.

Genaue Beschreibung der Erfindung

[0029] Eine Ausführungsform der vorliegende Erfindung umfaßt ein serverseitiges Steuerobjekt zum Verarbeiten und Erzeugen eines clientseitigen Benutzeroberflächenelements zur Anzeige in einer Webseite. Ferner kann eine Hierarchie von serverseitigen Steuerobjekten zusammenwirken zum Erzeugen des resultierenden Autorensprache-Codes wie Standard-HTML zur Anzeige einer Webseite an einem Client. Der Client kann beispielsweise irgendein Browser sein, der Standard-HTML oder eine andere Autoren-Sprache unterstützt. Bei einer Ausführungsform der vorliegenden Erfindung entsprechen serverseitige Steuerobjekte logisch clientseitigen Benutzeroberflächenelementen und erzeugen an einem Server den Autorensprache-Code, der von einem clientseitigen Browser zu verwenden ist, um eine Webseite zu zeigen und zu verarbeiten. Die Operation der Verarbeitung des Postback-Ereignisbehandlungsoperation, einer Postback-Datenbehandlungsoperation, einer Datenbindungsoperation und einer Zustandsmanagementoperation umfassen. Die Zustandsmanagementoperation betrifft den Zustand eines serverseitigen Steuerobjekts.

[0030] Fig. 1 zeigt einen Webserver zum dynamischen Erzeugen von Webseiteninhalt zur Anzeige an einem Client bei einer Ausführungsform der vorliegenden Erfindung. Ein Client **100** führt einen Browser **102** aus, der eine Webseite **104** an einer Displayeinrichtung des Client **100** anzeigt. Der Client **100** kann ein Clientcomputersystem aufweisen, das eine Displayeinrichtung wie etwa einen Videomonitor hat. Ein "INTERNET EXPLORER"-Browser, der von Microsoft Corporation vertrieben wird, ist ein Beispiel eines Browsers **102** in einer Ausführungsform der vorliegenden Erfindung. Andere beispielhafte Browser umfassen, ohne Einschränkung, "NETSCAPE NAVIGATOR" und "MOSAIC". Die beispielhafte Webseite **104** enthält eine Textboxsteuerung **106** und zwei Schaltflächensteuerungen **108** und **110**. Der Browser **102** kann HTML-Code in der HTTP-Antwort **112**

von einem Webserver **116** empfangen und zeigt die Webseite wie durch den HTML-Code beschrieben. Die HTML wird zwar unter Bezugnahme auf eine Ausführungsform beschrieben, aber im Rahmen der vorliegenden Erfindung sind auch andere Autorensprachen möglich, was – ohne Einschränkung – die folgenden einschließt: SGML (Standard Generalized Markup Language), XML (eXtensible Markup Language) und WML (Wireless Markup Language), die eine XML-basierte Markupsprache und dazu bestimmt ist, den Inhalt und Benutzeroberflächen von Schmalbandfunkgeräten wie Pagern und Mobiltelefonen zu bezeichnen. Ferner wird zwar im vorliegenden Fall hauptsächlich Standard-HTML 3.2 angegeben, aber im Rahmen der vorliegenden Erfindung kann jede Version von HTML unterstützt werden.

[0031] Die Kommunikationen zwischen dem Client **100** und dem Webserver **116** können unter Verwendung einer Folge von HTTP-Anforderungen **114** und HTTP-Antworten **112** ausgeführt werden. HTTP wird zwar unter Bezugnahme auf eine Ausführungsform beschrieben, aber andere Transportprotokolle einschließlich – ohne Einschränkung – S-HTTP sind im Rahmen der vorliegenden Erfindung denkbar. Am Webserver **116** empfängt ein HTTP-Fließbandmodul **118** eine HTTP-Anforderung **114**, bestimmt die URL-Adresse und ruft einen geeigneten Handler **120** zur Verarbeitung der Anforderung auf. Bei einer Ausführungsform der vorliegenden Erfindung sind an dem Webserver **116** eine Vielzahl von Handlern **120** vorgesehen, um verschiedene Arten von Ressourcen zu behandeln.

[0032] Wenn beispielsweise die URL-Adresse eine Ressource **122** mit statischem Inhalt wie etwa eine HTML-Datei bezeichnet, greift ein Handler **120** auf die Ressource **122** mit statischem Inhalt zu und leitet die Ressource **122** mit statischem Inhalt durch das HTTP-Fließband **118** zurück zur Übertragung zu dem Client **100** in einer HTTP-Antwort **112**. Wenn alternativ bei einer Ausführungsform der vorliegenden Erfindung die URL-Adresse eine Ressource **124** mit dynamischem Inhalt bezeichnet wie etwa eine ASP+-Ressource, greift ein Handler **120** auf die Ressource **124** mit dynamischem Inhalt zu, verarbeitet die Inhalte der Ressource **24** mit dynamischem Inhalt und erzeugt den resultierenden HTML-Code für die Webseite **104**. Bei einer Ausführungsform der vorliegenden Erfindung umfaßt der resultierende HTML-Code den Standard-HTML-Code 3.2. Im allgemeinen ist eine Ressource mit dynamischem Inhalt ein serverseitiger Vereinbarungs-Datenspeicher (z. B. eine ASP+-Ressource), der verwendet werden kann, um den Autorensprache-Code dynamisch zu erzeugen, der eine an einem Client anzuzeigende Webseite beschreibt. Der HTML-Code für die Webseite wird dann in einer HTTP-Antwort **112** durch das HTTP-Fließband **118** zur Übertragung an den Client **100** geleitet.

[0033] Während der Verarbeitung kann ein Handler **120** auch auf Bibliotheken von bereits entwickelten oder Drittcodes zugreifen, um die Entwicklungsarbeit zu vereinfachen. Eine solche Bibliothek ist eine serverseitige ClassControl-Bibliothek **126**, von welcher der Handler **120** serverseitige Steuerobjekte instanzieren kann zur Verarbeitung von Benutzeroberflächenelementen und zum Erzeugen der resultierenden HTML-Daten für die Anzeige einer Webseite. Bei einer Ausführungsform der vorliegenden Erfindung werden ein oder mehr serverseitige Steuerobjekte in ein oder mehr Benutzeroberflächenelementen entweder sichtbar oder versteckt in der Webseite abgebildet, die in der Ressource **124** mit dynamischem Inhalt beschrieben ist.

[0034] Dagegen ist eine zweite Bibliothek eine clientseitige ControlClass-Bbibliothek **128** wie etwa eine Bibliothek mit "ACTIVEX"-Komponenten von Microsoft Corporation. Eine "ACTIVEX"-Steuerung ist ein COM-Objekt ("Component Object Model"), das bestimmten Standards insofern folgt, auf welche Weise es mit seinem Client und anderen Komponenten in Dialog tritt. Eine clientseitige "ACTIVEX"-Steuerung ist beispielsweise eine COM-basierte Komponente, die automatisch zu einem Client gedownloaded und von einem Webbrowser am Client ausgeführt werden kann. Serverseitige ACTIVEX-Komponenten (nicht gezeigt) sind COM-basierte Komponenten, die an einem Server implementierbar sind, um verschiedene serverseitige Funktionen auszuführen wie etwa die Bereitstellung der serverseitigen Funktionalität einer Börsenkursnachschlage-Anwendung oder Datenbankkomponente. Eine detaillierte Erläuterung von ACTIVEX findet sich in "Understanding ACTIVEX and OLE", David Chappell, Microsoft Press, 1996.

[0035] Im Gegensatz zu "ACTIVEX"-Steuerungen entspricht ein serverseitiges Steuerobjekt in einer Ausführungsform der vorliegenden Erfindung, das in einer Ressource **124** mit dynamischem Inhalt bezeichnet ist, einem Benutzeroberflächenelement, das am Client angezeigt wird. Das serverseitige Steuerobjekt kann auch gültigen HTML-Code erzeugen, der beispielsweise eine HTML-Marke und einen Lokalisierer zur Bezugnahme auf eine gegebene clientseitige "ACTIVEX"-Steuerung aufweisen kann. Wenn der Browser den Code für die clientseitige "ACTIVEX"-Steuerung bereits in seinem Speichersystem hat, führt der Browser die "ACTIVEX"-Steuerung innerhalb der Webseite am Client aus. Andernfalls downloadet der Browser den Code für die "ACTIVEX"-Steuerung von der vom Lokalisierer bezeichneten Ressource und führt dann die "ACTIVEX"-Steuerung innerhalb der Webseite am Client aus. Ein serverseitiges Steuerobjekt bei einer Ausführungsform der vorliegenden Erfindung kann außerdem Ereignisse zu einem serverseitigen "ACTIVEX"-Objekt erheben, das

dazu genutzt wird, eine Börsennachschlage-Anwendung am Server zu implementieren.

[0036] Ein Handler **120** hat ferner Zugang zu ein oder mehr Nicht-Benutzeroberflächen-Serverkomponenten **130**, die an dem Webserver **116** oder einem anderen zugänglichen Webserver ausgeführt werden. Eine Nicht-Benutzeroberflächen-Serverkomponente **130** wie etwa eine Börsenkursnachschlage-Anwendung oder Datenbankkomponente kann in einer Ressource **124**, die von einem Handler **120** verarbeitet wird, als Referenz vorgesehen oder zugeordnet sein. Serverseitige Ereignisse, die von den in der Ressource **124** mit dynamischem Inhalt deklarierten Steuerobjekten aufgerufen werden, können mit serverseitigem Code verarbeitet werden, der geeignete Methoden in der Nicht-Benutzeroberfläche-Serverkomponente **130** aufruft. Infolgedessen vereinfacht die von den serverseitigen Steuerobjekten ermöglichte Verarbeitung die Programmierung der Nicht-Benutzeroberfläche-Serverkomponente **130** durch Kapselung der Verarbeitung und Erzeugung der Benutzeroberflächenelemente einer Webseite, was es dem Entwickler der Nicht-Benutzeroberflächen-Serverkomponente **130** erlaubt, sich auf die spezifische Funktionalität der Anwendung anstatt auf Benutzeroberflächenaspekte zu konzentrieren.

[0037] Fig. 2 ist ein Flußdiagramm von Operationen zum Verarbeiten und Erzeugen von clientseitigen Benutzeroberflächenelementen unter Verwendung von serverseitigen Steuerobjekten in einer Ausführungsform der vorliegenden Erfindung. In einer Operation **200** sendet der Client eine HTTP-Anforderung an den Server. Die HTTP-Anforderung enthält eine URL-Adresse, die eine Ressource wie etwa eine ASP+-Ressource bezeichnet. In einer Operation **202** empfängt der Server die HTTP-Anforderung und ruft den entsprechenden Handler zur Verarbeitung der angegebenen Ressource auf. Die ASP+-Ressource wird in einer Operation **203** gelesen. Eine Operation **204** erzeugt eine serverseitige Steuerobjekthierarchie auf der Basis der Inhalte der bezeichneten Ressource mit dynamischem Inhalt (z. B. der ASP+-Ressource).

[0038] In einer Operation **206** führen die serverseitigen Steuerobjekte der Steuerobjekthierarchie ein oder mehr der folgenden Operationen aus: Postback-Ereignisbehandlung, Postback-Datenbehandlung, Zustandsmanagement und Datenbindung. Postback-Ereignisse und – Daten (gemeinsam "Postback-Eingaben") von Benutzeroberflächenelementen werden von dem Client an den Server zum Zweck der Verarbeitung übermittelt. Beispielsweise kann ein Postback-Ereignis ohne Einschränkung ein "Mausklick"-Ereignis von einem clientseitigen Tastenfeldelement oder ein "Datenänderung"-Ereignis von einem clientseitigen Textboxelement umfassen, das zum Server übermittelt wird. Postback-Daten können beispielsweise ohne Einschränkung Text umfassen, der von einem Benutzer in einem Textboxelement eingegeben wird, oder einen Index eines von einer Dropdown-Box gewählten Elements. Eine Postback-Operationkann jedoch aus anderen Ereignissen und nicht nur aus einem Benutzerdialog resultieren.

[0039] In einer Operation **208** wird jedes serverseitige Steuerobjekt in der Hierarchie aufgerufen, Daten wie etwa HTML-Code zum Anzeigen von clientseitigen Benutzeroberflächenelementen in der Webseite zu generieren (oder darzustellen). Es ist zu beachten, daß der Ausdruck "darstellen" zwar verwendet werden kann, um den Vorgang des Anzeigens von grafischen Darstellungen auf einer Benutzeroberfläche zu beschreiben, er wird hier aber auch dazu verwendet, den Vorgang des Erzeugens von Autorensprache-Daten zu beschreiben, die von einer Client-Anwendung wie etwa einem Browser zum Zweck der Anzeige und der clientseitigen Funktionalität interpretiert werden können. Eine detaillierte Erörterung der Verarbeitungsoperation **206** und der Erzeugungsoperation **208** folgt in Verbindung mit Fig. 6. Bei einer Ausführungsform werden Aufrufe an Erzeugungs()-Methoden in einzelnen Steuerobjekten durchgeführt unter Verwendung einer Baumdurchquerungsfolge. Dabei resultiert ein Aufruf an die Erzeugungs()-Methode eines Seitenobjekts in einem rekursiven Durchlauf durch sämtliche geeigneten serverseitigen Steuerobjekte in der Hierarchie. Alternative Methoden zum Aufrufen der Erzeugungs()-Methoden für geeignete Steuerobjekte können ebenfalls angewandt werden einschließlich einer Ereignisanzeige oder einer Objektregistrierung. Die runden Klammern bedeuten, daß die "Erzeugungs()-Markierung eine Methode im Vergleich mit einem Datenwert bezeichnet.

[0040] Bei einer Ausführungsform der vorliegenden Erfindung kann die tatsächliche Erzeugung der einzelnen serverseitigen Steuerobjekte verzögert bzw. aufgeschoben werden, bis auf das serverseitige Steuerobjekt zugegriffen wird (etwa bei der Handhabung von Postback-Eingaben, Laden eines Zustands, Erzeugen von HTML-Code von dem Steuerobjekt etc.) in den Operationen **206** oder **208**. Wenn auf ein serverseitiges Steuerobjekt niemals für eine gegebene Anfrage zugegriffen wird, optimiert die aufgeschobene Steuerobjekterzeugung die Serververarbeitung durch Eliminieren einer unnötigen Objekterzeugungsoperation.

[0041] Eine Operation **210** überträgt den HTML-Code in einer HTTP-Antwort an den Client. In einer Operation **214** empfängt der Client den HTML-Code, der einer anzuzeigenden neuen Webseite zugeordnet ist. In einer Operation **216** fügt das Clientsystem die Benutzeroberflächenelemente der neuen Seite entsprechend dem

von der HTTP-Antwort empfangenen HTML-Code ein (d. h. zeigt ihn an). Es versteht sich jedoch, daß das Einfügen eines Benutzeroberflächenelements auch Nicht-Anzeigeoperationen umfassen kann wie etwa das Erstellen von Audio- oder Tastausgaben, Lesen und Schreiben aus einem bzw. in einen Speicher, Steuern der Operation von Skripten etc. In einer Operation **212** wird die serverseitige Steuerobjekthierarchie abgeschlossen. Bei einer Ausführungsform der vorliegenden Erfindung werden serverseitige Steuerobjekte in der Hierarchie erzeugt als Antwort auf eine HTTP-Anforderung, die auf eine zugehörige ASP+-Ressource Bezug nimmt, und werden anschließend an die Erzeugung von Autorensprache-Daten (z. B. HTML-Daten) zerstört. Bei einer alternativen Ausführungsform kann die Operation **212** nach der Operation **208** und vor der Operation **210** durchgeführt werden.

[0042] Fig. 3 zeigt beispielhafte Module in einem Webserver, die in einer Ausführungsform der vorliegenden Erfindung verwendet werden. Der Webserver **300** empfängt eine HTTP-Anforderung **302** in dem HTTP-Fließband **304**. Das HTTP-Fließband **304** kann verschiedene Module umfassen wie etwa Module zum Protokollieren von Webseiten-Statistiken, Benutzerberechtigung, Benutzerautorisierung und das Einbringen von Webseiten-Ausgängen im Cache-Speicher. Jede ankommende HTTP-Anforderung **302**, die von dem Webserver **300** empfangen wird, wird letztlich von einer spezifischen Instanz einer IHTTPHandler-Klasse (als Handler **306** gezeigt) verarbeitet. Der Handler **306** löst die URL-Anforderung auf und ruft einen geeigneten Handlerbetrieb auf (z. B. ein Seitenherstellungsmodul **308**).

[0043] In Fig. 3 wird ein der ASP+-Ressource **310** zugeordnetes Seitenherstellungsmodul **308** aufgerufen, um die Instanzierung und Konfigurierung der ASP+-Ressource **310** abzuwickeln. Bei einer Ausführungsform kann eine ASP+-Ressource durch Bezeichnen eines bestimmten Suffixes (oder einer Dateierweiterung wie ".aspx") mit der Ressource identifiziert werden. Wenn eine Anforderung für eine gegebene ASP+-Ressource erstmals von dem Seitenherstellungsmodul **308** empfangen wird, durchsucht das Seitenherstellungsmodul **308** das Dateisystem nach der entsprechenden Datei (z. B. der .aspx-Datei **310**). Die Datei kann Text (z. B. Autorensprache-Daten) oder ein anderes Datenformat (z. B. Bytecodedaten oder codierte Daten) enthalten, die später von dem Server interpretiert werden oder auf die er zugreift, um die Anforderung zu bedienen. Wenn die physikalische Datei existiert, öffnet das Seitenherstellungsmodul **308** die Datei und liest die Datei in den Speicher ein. Wenn die Datei nicht zu finden ist, sendet das Seitenherstellungsmodul **308** eine entsprechende Fehlermeldung "Datei nicht gefunden" zurück.

[0044] Nach dem Einlesen der ASP+-Ressource **310** in den Speicher verarbeitet das Seitenherstellungsmodul **308** den Dateiinhalt, um ein Datenmodell der Seite aufzubauen (z. B. Listen von Skriptblöcken, Anweisungen, statischen Textbereichen, hierarchischen serverseitigen Steuerobjekten, serverseitigen Steuereigenschaften etc.). Das Datenmodell wird genutzt, um eine Quellenauflistung einer neuen Objektklasse wie etwa einer COM+-Klasse (COM+ = "Component Object Model+") zu erzeugen, das die Seitengrundklasse erweitert. Die Seitengrundklasse umfaßt Code, der die Struktur, die Eigenschaften und die Funktionalität eines Seitengrundobjekts definiert. Bei einer Ausführungsform der vorliegenden Erfindung wird die Quellenauflistung dann dynamisch in einer Zwischensprache kompiliert und später just-in-time-kompiliert in plattformspezifische Anweisungen (z. B. X86, Alpha etc.). Eine Zwischensprache kann allgemeinen oder kundenspezifisch aufgebauten Sprachcode wie COM+ IL-Code, Java Bytecodes, Modula 3 Code, SmallTalk-Code und Visual Basic-Code umfassen. Bei einer alternativen Ausführungsform können die Zwischensprache-Operationen entfallen, so daß die programmspezifischen Anweisungen direkt aus den Quellenauflistungen oder der Quellendatei erzeugt werden können (z. B. der ASP+-Ressource **310**). Das Seitenherstellungsmodul **308** kann auf eine ControlClass-Bbliothek **312** zugreifen, um vordefinierte serverseitige Steuerklassen zu erhalten, die bei der Erzeugung der Steuerobjekthierarchie verwendet werden.

[0045] Das Seitenherstellungsmodul **308** gibt ein Seitenobjekt **314** aus, das ein serverseitiges Steuerobjekt ist, das der Webseite **104** von Fig. 1 entspricht. Das Seitenobjekt **314** und seine Söhne (d. h. ein Textboxobjekt **318**, ein Tastenfeldobjekt **320** und ein anderes Tastenfeldobjekt **322**) bilden eine beispielhafte Steuerobjekthierarchie **316**. Andere beispielhafte Steuerobjekte sind gemäß der vorliegenden Erfindung ebenfalls vorstellbar, was ohne Einschränkung Objekte, die den HTML-Steuerungen in Tabelle 1 entsprechen, sowie kundenspezifische Steuerobjekte umfaßt. Das Seitenobjekt **314** entspricht der Textbox **106** in Fig. 1. Ebenso entspricht das Tastenfeldobjekt **320** dem ADD-Tastenfeld **108** in Fig. 1, und das Tastenfeldobjekt **322** entspricht dem Entfernen-Tastenfeld **110** in Fig. 1. Das Seitenobjekt **314** ist hierarchisch mit anderen Steuerobjekten an dem Server verknüpft. Bei einer Ausführungsform ist ein Seitenobjekt ein Container-Objekt, das hierarchisch seine Sohn-Steuerobjekte enthält. Bei einer alternativen Ausführungsform können andere Formen von hierarchischen Beziehungen verwendet werden einschließlich einer Abhängigkeitsbeziehung. Bei einer komplexeren Steuerobjekthierarchie mit einer Vielzahl von Ebenen von Söhnen kann ein Sohnobjekt ein Container-Objekt für andere Sohnobjekte sein.

[0046] Bei der gezeigten Ausführungsform werden die Steuerobjekte in der Steuerobjekthierarchie **316** an dem Server **300** erzeugt und ausgeführt, und jedes serverseitige Steuerobjekt entspricht logisch einem entsprechenden Benutzeroberflächenelement an dem Client. Die serverseitigen Steuerobjekte wirken außerdem zusammen, um Postback-Eingaben von der HTTP-Anforderung **302** zu handhaben, die Zustände von serverseitigen Steuerobjekten zu managen, eine Datenbindung mit serverseitigen Datenbanken auszuführen und Autorensprache-Daten (z. B. HTML-Code) zu erzeugen, um eine resultierende Webseite am Client zur Anzeige zu bringen. Die resultierenden Autorensprache-Daten werden von der serverseitigen Steuerobjekthierarchie **316** generiert (d. h. erzeugt) und in einer HTTP-Antwort **324** zum Client übermittelt. Beispielsweise kann resultierender HTML-Code jedes gültige HTML-Konstrukt verkörpern und kann auf Steuerungen vom ACTIVEX-Typ, JAVA-Applets, Skripts und alle sonstigen Webressourcen Bezug nehmen, die clientseitige Benutzeroberflächenelemente ergeben (z. B. Steuertasten, Textboxen etc.), wenn sie von einem Browser verarbeitet werden.

[0047] Aufgrund von Vereinbarungen, die in der ASP+-Ressource **310** getroffen werden, können serverseitige Steuerobjekte auch auf ein oder mehr Nicht-Benutzeroberflächen-Serverkomponenten **330** zugreifen, um einen Dialog zwischen der Nicht-Benutzeroberflächen-Serverkomponente **330** und clientseitigen Benutzeroberflächenelementen herzustellen. Beispielsweise können als Antwort auf Postback-Eingaben serverseitige Steuerobjekte serverseitige Ereignisse zu den Nicht-Benutzeroberflächen-Serverkomponenten erheben, die für diese Ereignisse registriert sind. Auf diese Weise kann die Nicht-Benutzeroberfläche-Serverkomponente **330** mit dem Benutzer durch Benutzeroberflächenelemente in Dialog treten, ohne den zum Anzeigen und Verarbeiten dieser Elemente erforderlichen Code zu programmieren.

[0048] Fig. 4 zeigt Inhalte einer beispielhaften Ressource mit dynamischem Inhalt bei einer Ausführungsform der vorliegenden Erfindung. Bei der gezeigten Ausführungsform enthält die Datei **400** Klartextvereinbarungen in einem beispielhaften Ressourcenformat (z. B. ASP+) mit dynamischem Inhalt. Jede Vereinbarung liefert Anweisungen an einen Seitenkompilierer, der die Datei **400** liest, die entsprechenden serverseitigen Steuerobjekte erzeugt und aufruft, um das clientseitige Benutzeroberflächenelement zu verarbeiten, und schließlich den erhaltenen HTML-Code zur Übermittlung an den Client in einer HTTP-Antwort kombiniert. Insofern beschreibt oder bezieht sich eine Vereinbarung auf die Funktionalität eines clientseitigen Benutzeroberflächenelements, das von einem serverseitigen Steuerobjekt implementiert wird. Das serverseitige Steuerobjekt erzeugt dann den HTML-Code, der dazu dient, eine neue Version der Webseite am Client zu definieren.

[0049] Die erste Zeile der Datei **400** umfaßt eine Übersetzungsanweisung in dem Format:
<%@directive {attribute=value}%>
wobei directive ohne Einschränkung "Seite", "Cache" oder "importieren" umfassen kann. Übersetzungsanweisungen werden von dem Seitenkompilierer verwendet bei der Verarbeitung einer Ressource mit dynamischem Inhalt, um Charakteristiken zu bestimmen wie Zwischenspeicher-Semantik, Sitzungszustands-Anforderungen, Fehlerhandhabungspläne, Skriptsprachen, Transaktionssemantik und Importanweisungen. Anweisungen können überall innerhalb einer Seitendatei plaziert sein.

[0050] In der zweiten Zeile ist <html> eine Standard-HTML-Startmarke, die den resultierenden HTML-Code als Literalkonstante (d. h. ohne zusätzliche Verarbeitung zum Erzeugen des resultierenden HTML-Codes) durchläuft. In HTML bezeichnet <html> den Beginn der HTML-Datei und wird mit der Abschlußmarke in Zeile 21, also </html>, gepaart, die ebenfalls eine Literalkonstante ist.

[0051] Ein Codevereinbarungsblock befindet sich in Zeilen 3 bis 10 der Datei **400**. Im allgemeinen definieren serverseitige Codevereinbarungsblöcke Seitenobjekt- und Steuerobjektelement-Variablen und -methoden, die am Server ausgeführt werden. In dem Format:

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

wobei die Sprach- und die src-Parameter fakultativ sind. Bei einer Ausführungsform der vorliegenden Erfindung sind Codevereinbarungsblöcke definiert unter Verwendung von <script>-Marken, die ein "runat"-Attribut enthalten, das einen Wert hat, der auf "server" gesetzt ist. Fakultativ kann ein "language"-Attribut auch verwendet werden, um die Syntax des inneren Codes zu bezeichnen. Die Standardsprache kann die Sprachkonfiguration der Gesamtseite repräsentieren; das "language"-Attribut in dem Codevereinbarungsblock erlaubt jedoch einem Entwickler die Verwendung verschiedener Sprachen innerhalb derselben Webseitenimplementierung, was beispielsweise Jscript und PERL (Practical Extraction and Report Language) einschließt. Die

<script>-Marke kann auch fakultativ eine "src"-Dabei bezeichnet, die eine externe Datei ist, aus der Code in die Ressource mit dynamischem Inhalt zur Verarbeitung durch den Seitenkompilierer eingefügt wird. Es versteht sich, daß die angegebene Syntax bei einer Ausführungsform verwendet wird, daß jedoch alternative Ausführungsformen andere Syntaxen im Rahmen der vorliegenden Erfindung verwenden können.

[0052] In Fig. 4 sind zwei Subroutinen im Visual Basic-Format innerhalb des Codevereinbarungsblocks vereinbart: AddButton_Click und DeleteButton_Click. Beide Subroutinen benötigen zwei Eingangsparameter "Source" und "E" und werden auf dem Server als Antwort auf eine HTTP-Anforderung ausgeführt, wenn ein clientseitiges Klickereignis an der entsprechenden Taste detektiert wird. In der AddButton_Click-Subroutine wird der Text in einer UserName-Textbox mit dem Wort "Add" verkettet und in das Textdatenelement von Message geladen. In der DeleteButton_Click-Subroutine wird der Text in der UserName-Textbox mit dem Wort "Delete" verkettet und in das Textdatenelement von Message geladen. Obwohl dies in Fig. 4 nicht gezeigt ist, können Elementvariablen von serverseitigen Steuerobjekten in dem Codevereinbarungsblock der Datei **400** deklariert sein. Beispielsweise deklariert bei Anwendung einer Visual Basic-Syntax der Schlüsselwortplatz "DIM" eine Datenvariable eines serverseitigen Steuerobjekts.

[0053] Ein "code render block" (nicht gezeigt) kann ebenfalls in einer Ressource mit dynamischem Inhalt enthalten sein. Bei einer Ausführungsform der vorliegenden Erfindung wird ein Codeerzeugungsblock in einem einzigen "rendering"-Verfahren ausgeführt, das zu einem Seitenerzeugungszeitpunkt abläuft. Ein Codeerzeugungsblock genügt dem folgenden Format (obwohl bei alternativen Ausführungsformen auch an andere Formate gedacht ist):
<%InlineCode%>
wobei InlineCode eigenständige Codeblöcke oder Steuerablaufblöcke umfaßt, die auf dem Server während der Seitenerzeugungszeit ausgeführt werden.

[0054] Inline-Ausdrücke können ebenfalls innerhalb eines Codeerzeugungsblocks verwendet werden unter Verwendung der beispielhaften Syntax:
<%=InlineExpression%>
wobei der in einem Block InlineExpression enthaltene Ausdruck letztlich von einem Aufruf zu "Response.Write(InlineExpression)" in einem Seitenobjekt umfaßt ist, das den aus InlineExpression resultierenden Wert in einen geeigneten Platzhalter in der Vereinbarung schreibt. Beispielsweise können InlineExpressions in einem Codeerzeugungsblock wie folgt eingefügt sein:
<font size="<%=x%>">Hi<%=Name%>,you are%=Age%>!
wobei eine Grußformel und eine Angabe über das Alter einer Person in einem Font ausgegeben werden, der in dem Wert "x" gespeichert ist. Name und Alter der Person sind als Zeichenfolgen in einem Codevereinbarungsblock (nicht gezeigt) definiert. Der resultierende HTML-Code wird am Server zur Übermittlung an den Client in der HTTP-Antwort so erzeugt, daß er die Werte der InlineExpressions an geeigneten Stellen enthält:
<font size="12">Hi Bob, du bist 35!

[0055] In Zeile 11 der Datei **400** ist <body> eine Standard-HTML-Marke zur Definition des Beginns des Hauptteils des HTML-Dokuments. In Zeile 20 der Datei **400** ist die Abschlußmarke </body> ebenfalls gezeigt. Sowohl <body> als auch </body> sind Literalkonstanten bei einer Ausführungsform der vorliegenden Erfindung.

[0056] Innerhalb des Hauptteils der HTML-Datei **400** findet man in Zeile 12 die Startmarke <form> eines HTML-Formularblocks. Die Endmarke </form> des Formularblocks findet man in Zeile 19 der HTML-Datei **400**. Ein fakultativer Parameter "id" kann ebenfalls in der HTML-Steuermarke <form> enthalten sein, um dem Formularblock eine gegebene Kennung zuzuordnen, was es möglich macht, daß eine Mehrzahl von Formularblöcken in einer einzigen HTML-Datei enthalten sein kann.

[0057] In Zeile 18 der Datei **400** wird eine serverseitige Marke deklariert, die mit "Message" bezeichnet ist. Die "Message"-Marke wird in dem in Zeilen 5 und 8 der Datei **400** vereinbarten Code verwendet, um eine Marke auf der Webseite anzuzeigen.

[0058] Innerhalb eines Formularblocks sind drei beispielhafte HTML-Steuermarken gezeigt, die den Benutzeroberflächenelementen **106**, **108** und **110** von Fig. 1 entsprechen. Das erste Benutzeroberflächenelement ist in Zeile 13 der Datei **400** deklariert und entspricht einer Textbox. Das Textliteral "User Name:" deklariert eine Marke, die links von der Textbox positioniert ist. Die Eingabemarke mit type="Text" bezeichnet ein serverseitiges Textbox-Steuerobjekt, das ein id gleich "UserName" als serverseitiges Steuerobjekt hat, das ein clientseitiges Textbox-Benutzeroberflächenelement bildet. Zeilen 15 und 16 der Datei **400** deklarieren die clientseitigen Benutzeroberflächenelemente, die als Tastenfelder **108** und **110** von Fig. 1 gezeigt sind. Es ist zu beachten,

daß der "OnServerClock"-Parameter die geeignete Subroutine bezeichnet, die in dem Codevereinbarungsblock der Daten **400** deklariert ist. Als solche erzeugen die serverseitigen Tastenfeldsteuerobjekte, die als Antwort auf die Vereinbarungen in der Datei erzeugt wurden, den HTML-Code für die clientseitigen Tastenfelder und einen zugehörigen serverseitigen Code zur Implementierung der Tastenfeld-Klickereignisse.

[0059] Die Textbox und die Tastenfelder, die in der Datei **400** deklariert werden, sind Beispiele von HTML-Serversteuerungs-Vereinbarungen. Standardmäßig werden sämtliche HTML-Marken innerhalb einer ASP+-Ressource als literaler Textinhalt behandelt und sind für Seitenentwickler programmorientiert unzugänglich. Seitenautoren können jedoch angeben, daß eine HTML-Marke zerlegt werden und als eine zugängliche Serversteuerungsvereinbarung behandelt werden sollte, indem sie mit einem "runat"-Attribut mit einem auf "server" gesetzten Wert markiert wird. Jedes serverseitige Steuerobjekt kann fakultativ einem speziellen "id"-Attribut zugeordnet sein, um die programmorientierte Bezugnahme auf das entsprechende Steuerobjekt zu ermöglichen. Eigentumsargumente und Ereignisbindungen an serverseitigen Steuerobjekten können ebenfalls angegeben werden unter Verwendung von deklarativen Name-/Wert-Attributpaaren an dem Markenelement (z. B.: der OnServerClick ist gleich dem Paar "MyButton_Click").

[0060] Bei einer Ausführungsform der vorliegenden Erfindung ist eine allgemeine Syntax zur Deklaration von HTML-Steuerobjekten wie folgt:

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

wobei OptionalName eine besondere Kennung für das serverseitige Steuerobjekt ist. Eine Liste von aktuell unterstützten HTML-Marken und der zugehörigen Syntax und COM+-Klasse ist in der TABELLE 1 gezeigt, obwohl im Rahmen der vorliegenden Erfindung auch andere HTML-Marken in Betracht kommen.

HTML Markenname	Beispiel	COM+ Class
<a>	 My Link 	Ankertaste
<img>	<img id = "MyImage" runat = server>	Abbildung
		Etikett
	Some contents	Panel
<form>	<form id = "MyForm" runat = server> </form>	Formular-steuerung
<select>	<select id = "MyList" runat = server> <option>One</option> <option>Two</option> <option>Three</option> </select>	DropDown-Liste
<input type = file>	<input id = "MyFile" type = file runat = server>	Dateieingabe
<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>	Taste
<input type = radio>	<input id = "MyRadioButton" type = radio runat = server>	FunkTaste
<input type = checkbox>	<input id = "MyCheck" type = checkbox runat = server>	PrüfBox
<input type = hidden>	<input id ="MyHidden" type = hidden runat = server>	VerdecktesFeld
<input type = image>	<input type = image src = "foo.jpg" runat = server>	BildTaste
<input type = submit>	<input type = submit runat = server>	Taste
<input type = button>	<input type = button runat = server>	Taste
<button>	<button id = MyButton runat = server>	Taste
<textarea>	<textarea id = "MyText" runat = server> Dies ist ein Probetext </textarea>	TextBereich

TABELLE 1

[0061] Zusätzlich zu Standard-HTML-Steuermarken ermöglicht es eine Ausführungsform der vorliegenden Erfindung Entwicklern, wiederverwendbare Komponenten zu generieren, die gemeinsame programmorientierte Funktionalitäten außerhalb der gesetzten Standard-HTML-Marke kapseln. Diese kundenspezifischen serverseitigen Steuerobjekte werden bezeichnet unter Verwendung von beschreibenden Marken innerhalb einer Seitendatei. Kundenspezifische serverseitige Steuerobjektbeschreibungen umfassen ein "runat"-Attribut mit einem auf "server" gesetzten Wert. Fakultativ kann das spezielle "id"-Attribut angegeben werden, um die programmorientierte Bezugnahme des kundenspezifischen Steuerobjekts zu ermöglichen. Außerdem bezeichnen beschreibende Name-/Wert-Attributpaare an einem Markenelement Eigenschaftsargumente und Ereignisbindungen an einem serverseitigen Steuerobjekt. Inline-Schablonenparameter können ebenfalls an ein serverseitiges Steuerobjekt gebunden sein durch Vorsehen eines geeigneten Sohn-Elements mit "template"-Präfix an

dem Vater-Serversteuerobjekt. Ein Format für eine kundenspezifische serverseitige Steuerobjektbeschreibung ist:
<servercntrlclassname id="OptionalName" [propertyname="propval"] runat=server/>
wobei servercntrlclassname ein Name einer zugänglichen Serversteuerklasse ist, OptionalName eine spezielle Kennung des serverseitigen Steuerobjekts ist und propval einen fakultativen Eigenschaftswert in dem Steuerobjekt darstellt.

[0062] Unter Verwendung einer alternativen Beschreibungssyntax können XML-Marken-Präfixe verwendet werden, um eine genauere Bezeichnung zum Benennen von serverseitigen Steuerobjekten innerhalb einer Seite zu erhalten, wobei das folgende Format verwendet wird:
<tagprefix:classname id="OptionalName" runat=server/>
wobei tagprefix einer gegebenen Steuernamensplatz-Bibliothek zugeordnet ist und classname einen Namen einer Steuerung in der zugeordneten Namensplatz-Bibliothek darstellt. Ein fakultativer propertyvalue wird ebenfalls unterstützt.

[0063] Zusammenfassend umfaßt eine Ausführungsform der vorliegenden Erfindung serverseitige Steuerobjekte, die generiert und an dem Server ausgeführt werden, um HTML-Code zu erzeugen, der an einen Client gesendet wird. Der HTML-Code kann alle gültigen HTML-Konstrukte verkörpern und kann beispielsweise auf Steuerungen vom ACTIVEX-Typ, JAVA-Applets, Skripte und alle anderen Webressourcen Bezug nehmen, um Benutzeroberflächen-Tastenfelder und andere Benutzeroberflächenelemente am Client zu erzeugen. Ein Benutzer am Client kann mit diesen Benutzeroberflächenelementen in Dialog treten, die den serverseitigen Steuerobjekten logisch entsprechen, und kann an den Server eine Anforderung zurücksenden. Die serverseitigen Steuerobjekte werden an dem Server generiert zum Verarbeiten der Daten, Ereignisse und anderen Charakteristiken der Benutzeroberflächenelemente, um so die nächste Runde von HTML-Code zu erzeugen, der in einer Antwort an den Client übertragen werden soll.

[0064] Unter Bezugnahme auf **Fig. 5** weist ein beispielhaftes Rechnersystem für Ausführungsformen der Erfindung folgendes auf: eine Universalrechenvorrichtung in Form eines herkömmlichen Computersystems **500** mit einer Prozessoreinheit **502**, einem Systemspeicher **504** und einem Systembus **506**, der verschiedene Systemkomponenten einschließlich des Systemspeichers **504** mit der Prozessoreinheit **500** verbindet. Der Systembus **506** kann einer von mehreren Typen von Busstrukturen sein mit einem Speicherbus oder einer Speichersteuerung, einem peripheren Bus und einem lokalen Bus, wobei jede von verschiedenen Busarchitekturen verwendet werden kann. Der Systemspeicher umfaßt einen Festwertspeicher bzw. ein ROM **508** und einen Speicher mit wahlfreiem Zugriff bzw. ein RAM **510**. Ein Ein-/Ausgabe-Grundsystem **512** (BIOS), das Grundroutinen enthält, die dazu beitragen, Informationen zwischen Elementen innerhalb des Computersystems **500** zu übertragen, ist in dem ROM **508** gespeichert.

[0065] Das Computersystem **500** hat ferner ein Festplattenlaufwerk **512** zum Lesen von einer und Schreiben auf eine Festplatte, ein Magnetplattenlaufwerk **514** zum Lesen von einer oder Schreiben auf eine entfernbare Magnetplatte **516** und ein Bildplattenlaufwerk **518** zum Lesen von einer oder Schreiben auf eine entfernbare Bildplatte **519** wie etwa ein CD-ROM, eine DVD oder einen anderen Bilddatenträger. Das Festplattenlaufwerk **512**, das Magnetplattenlaufwerk **514** und das Bildplattenlaufwerk **518** sind mit dem Systembus **506** über eine Festplattenlaufwerk-Schnittstelle **520**, eine Magnetplattenlaufwerk-Schnittstelle **522** und eine Bildplattenlaufwerk-Schnittstelle **524** verbunden. Die Laufwerke und ihre zugehörigen computerlesbaren Datenträger bilden den nichtflüchtigen Speicher von computerlesbaren Anweisungen, Datenstrukturen, Programmen und anderen Daten für das Computersystem **500**.

[0066] Die hier beschriebene beispielhafte Umgebung verwendet zwar eine Festplatte, eine entfernbare Magnetplatte **516** und eine entfernbare Bildplatte **519**, aber andere Arten von computerlesbaren Medien, die Daten speichern können, können in dem beispielhaften System Anwendung finden. Beispiele dieser anderen Arten von computerlesbaren Datenträgern, die in der beispielhaften Betriebsumgebung verwendet werden können, umfassen Magnetbandkassetten, Flash-Speicherkarten, digitale Bildplatten, Bernoulli-Kartuschen, RAMs und ROMs.

[0067] Eine Reihe von Programm-Modulen kann auf der Festplatte, der Magnetplatte **516**, der Bildplatte **510**, dem ROM **508** oder RAM **510** gespeichert sein einschließlich eines Betriebssystems **526**, eines oder mehrerer Anwendungsprogramme **528**, anderer Programm-Module **530** und Programmdaten **532**. Ein Benutzer kann Befehle und Informationen in das Computersystem **500** durch Eingabeeinrichtungen wie eine Tastatur **534** und eine Maus **536** oder eine andere Zeigereinrichtung eingeben. Beispiele von anderen Eingabeeinrichtungen umfassen etwa ein Mikrofon, einen Joystick, ein Gamepad, eine Satellitenschüssel und einen Scanner. Diese

und weitere Eingabeeinrichtungen werden häufig mit der Verarbeitungseinheit **502** über eine Serienport-Schnittstelle **540** verbunden, die mit dem Systembus **506** gekoppelt ist. Dennoch können diese Eingabeeinrichtung auch über andere Schnittstellen wie etwa einen Parallelport, einen Gameport oder einen universellen seriellen Bus bzw. USB angeschlossen werden. Ein Monitor **542** oder eine andere Art von Displayeinrichtung ist ebenfalls über eine Schnittstelle wie etwa einen Videoadapter **544** mit dem Systembus **506** verbunden. Zusätzlich zu dem Monitor **542** weisen Computersysteme typischerweise andere periphere Ausgabeeinrichtungen (nicht gezeigt) auf, etwa Lautsprecher und Drucker.

[0068] Das Computersystem **500** kann in einer vernetzten Umgebung betrieben werden unter Anwendung logischer Verbindungen mit einem oder mehreren abgesetzten Computern wie etwa einem abgesetzten Computer **546**. Der abgesetzte Computer **546** kann ein Computersystem, ein Server, ein Router, ein Netz-PC, ein gleichrangiger Computer oder normaler Netzknoten sein und weist charakteristisch viele oder alle von den oben relativ zu dem Computersystem **500** beschriebenen Elementen auf. Die Netzverbindungen umfassen ein lokales Netz bzw. LAN **548** und ein Fernnetz bzw. WAN **550** auf. Diese Netzwerkumgebungen sind in Büros, unternehmensweiten Computernetzen, Intranetzen und dem Internet allgemein üblich.

[0069] Bei Verwendung in einer LAN-Netzumgebung ist das Computersystem **500** mit dem lokalen Netz **548** über eine Netzschnittstelle oder einen Netzadapter **552** verbunden. Bei Verwendung in einer WAN-Netzumgebung weist das Computersystem **500** charakteristisch ein Modem **554** oder eine andere Einrichtung zum Herstellen von Verbindungen über das Fernnetz **550**, wie etwa das Internet auf. Das Modem **554**, das intern oder extern sein kann, ist mit dem Systembus **506** über die serielle Portschnittstelle **540** verbunden. In einer vernetzten Umgebung können Programm-Module, die relativ zu dem Computersystem **500** oder Bereichen davon gezeigt sind, in der abgesetzten Speichereinrichtung gespeichert sein. Es versteht sich, daß die gezeigten Netzwerkverbindungen beispielhaft sind und daß andere Einrichtungen zum Herstellen einer Kommunikationsstrecke zwischen den Computern verwendet werden können.

[0070] Bei einer Ausführungsform der vorliegenden Erfindung stellt der Computer **500** einen Webserver dar, wobei der Prozessor **502** ein Seitenherstellungsmodul an einer ASP+-Ressource ausführt, die auf wenigstens einem der Datenträger **516**, **512**, **514**, **518**, **519** oder dem Speicher **532** abgespeichert ist. HTTP-Antworten und -Anforderungen werden über das LAN **548** übertragen, das mit einem clientseitigen Computer **546** gekoppelt ist.

[0071] Fig. 6 zeigt ein Prozeßablaufdiagramm, das die serverseitige Verarbeitung eines Seitenobjekts und anderer Steuerobjekte in einer Ausführungsform der vorliegenden Erfindung darstellt. In einer Operation **600** wird ein Seitenobjektkonstruktor von dem Seitenherstellungsmodul **308** aufgerufen (siehe Fig. 3). Infolgedessen wird ein Seitenobjekt (siehe z. B. das Seitenobjekt **314** in Fig. 3) generiert, um dem Webseiten-Benutzeroberflächenelement am Client logisch zu entsprechen. In einer Operation **602** ruft das Seitenherstellungsmodul die ProcessRequest()-Elementfunktion des Seitenobjekts auf, welche die stufenweisen Operationen zum Verarbeiten der von einem Client empfangenen HTTP-Anforderung initiiert. In einer ersten Stufe einer Ausführungsform der vorliegenden Erfindung generiert eine serverseitige Create-Operation (nicht gezeigt) die serverseitigen Sohn-Steuerobjekte, die in der Steuerobjekthierarchie des Seitenobjekts enthalten sind, d. h. Konstruktoren für Sohn-Steuerobjekte werden rekursiv aufgerufen, um die Steuerobjekte während der Lebensdauer der Verarbeitung der HTTP-Anforderung zu generieren.

[0072] Bei einer alternativen Ausführungsform wird jedoch die Erschaffung von Sohn-Steuerobjekten aufgeschoben, bis das Steuerobjekt für einen gegebenen Verarbeitungsschritt erforderlich ist (z. B. die Handhabung eines Postback-Ereignisses, die Handhabung von Postback-Daten, das Laden oder Sichern eines Betrachtungszustands, das Umwandeln einer Datenbindung oder die Erzeugung von HTML-Code für das entsprechende Benutzeroberflächenelement). Die letztere Ausführungsform, von der es heißt, daß sie "aufgeschobene Steuerobjekterzeugung" implementiert, ist eine Optimierung, die eine unnötige Nutzung von CPU und Speicher reduzieren kann. Beispielsweise kann ein vom Client empfangenes, vom Benutzer eingegebenes Ereignis in der Erschaffung einer vollständig verschiedenen Webseite resultieren. In diesem Fall ist es nicht notwendig, eine vollständige Steuerobjekthierarchie der vorhergehenden Seite zu instanzieren, nur um ein Ereignis zu verarbeiten, das unmittelbar in der Beendigung der Steuerobjekthierarchie und der Instanzierung einer neuen und davon verschiedenen Steuerobjekthierarchie für eine neue Seite resultiert.

[0073] Als Antwort auf den Serveraufruf für die ProcessRequest-Methode des Seitenobjekts können die Operationen **604** bis **620** von dem Seitenobjekt und von einzelnen Sohn-Steuerobjekten ausgeführt werden, was teilweise von den Daten einer gegebenen HTTP-Anforderung abhängig ist. Bei einer Ausführungsform der vorliegenden Erfindung werden die Operationen **604** bis **620** für jedes einzelne Objekt in der in Fig. 6 gezeigten

Reihenfolge ausgeführt; eine gegebene Operation für ein Objekt kann jedoch außerhalb der Reihenfolge oder überhaupt nicht in bezug auf eine gegebene Operation eines anderen Objekts auftreten, was von der HTTP-Anforderung abhängig ist. Beispielsweise kann ein erstes Objekt seine Init-Operation **604** und seine Lade-Operation **606** ausführen und mit der Postback-Datenverarbeitungsoperation **608** beginnen, bevor ein Sohn-Steuerobjekt seine eigene Init-Operation **604** und Lade-Operation **606** aufgrund der aufgeschobenen Steuerobjekterschaffung ausführt. Die Reihenfolge der Operationsverarbeitung durch das Seitenobjekt und Sohn-Steuerobjekte ist von verschiedenen Faktoren abhängig, einschließlich und ohne Einschränkung von der Beschaffenheit der Daten in der HTTP-Anforderung, der Konfiguration der Steuerobjekthierarchie, dem aktuellen Zustand der Steuerobjekte und davon, ob eine aufgeschobene Steuerobjekterstellung implementiert ist.

[0074] Die Init-Operation **604** initialisiert ein Steuerobjekt, nachdem es durch Ausführen eines serverseitigen Codes generiert wurde, welcher der Initialisierung in der Ressource mit dynamischem Inhalt zugeordnet ist. Auf diese Weise kann jedes serverseitige Steuerobjekt mit spezieller serverseitiger Funktionalität, die in der Ressource mit dynamischem Inhalt beschrieben ist, kundenangepaßt werden. Bei einer Ausführungsform der vorliegenden Erfindung ist der dynamische Inhaltscode, der die Grundseitensteuerungsklassen kundenanpassen soll, von dem Seitenentwickler in der ASP+-Ressource am Server deklariert. Wenn die ASP+-Ressource kompiliert wird, wird der deklarierte Code in den geeigneten Initialisierungscode (z. B. die Init()-Methoden des Seitenobjekts und der Sohn-Steuerobjekte) eingefügt. Die Init-Operation **604** führt diesen Code aus, um die Seitengrundklasse und die Grundklassen für Sohn-Steuerobjekte zu personalisieren oder zu erweitern.

[0075] Bei einer Ausführungsform der vorliegenden Erfindung wird das Zustandsmanagement der serverseitigen Steuerobjekte in einer Lade-Operation **606** und einer Sicherungs-Operation **616** unterstützt, die eine transportfähige Zustandsstruktur verwenden zur Anpassung an das zustandslose Modell für Client-Serversysteme durch Rückstellen von serverseitigen Steuerobjekten in ihre vorherigen Zustände. Bei einer Ausführungsform wird der Zustand in einem oder mehreren verdeckten HTML-Feldern eines HTTP-Anforderungs/-Antwortpaars zum und vom Server übermittelt, obwohl andere transportfähige Zustandsstrukturen im Rahmen der vorliegenden Erfindung liegen.

[0076] In einer gegebenen Folge von auf die aktuelle Seite bezogenen Anforderungen und Antworten zwischen einem Client und einem Server werden die Zustände von ein oder mehr Steuerobjekten mittels der Sicherungs-Operation **616** nach Verarbeitung einer vorhergehenden Anforderung in einer transportfähigen Zustandsstruktur gespeichert. Bei einer Ausführungsform der vorliegenden Erfindung ist in die transportfähige Zustandsstruktur auch zusätzliche Zustandsinformation eingefügt, was hierarchische Information oder Steuerobjektkennungen einschließt, um dem Server zu ermöglichen, einen gegebenen Zustand dem entsprechenden Steuerobjekt zuzuordnen. In einer anschließenden HTTP-Anforderung wird die Zustandsinformation in der transportfähigen Zustandsstruktur zum Server zurückgebracht. Der Server extrahiert die Zustandsinformation aus der empfangenen transportfähigen Zustandsstruktur und lädt die Zustandsdaten in die entsprechenden Steuerobjekte innerhalb der Steuerobjekthierarchie, um jedes Steuerobjekt in seinen Zustand zurückzubringen, wie er vor einer vorhergehenden HTTP-Antwort existiert hat. Nachdem die aktuelle Anforderung verarbeitet ist, werden die Zustände von ein oder mehr serverseitigen Steuerobjekten erneut in der transportfähigen Zustandsstruktur durch die Sicherungs-Operation **616** abgespeichert, und die transportfähige Zustandsstruktur wird in der nächsten HTTP-Antwort zum Client zurückgebracht.

[0077] Als Ergebnis der Lade-Operation **606** wird jedes serverseitige Steuerobjekt in einen Zustand gebracht, der mit seinem Zustand vor einer vorhergehenden HTTP-Antwort übereinstimmt. Wenn beispielsweise ein Textbox-Steuerobjekt einen Merkmalswert aufweist, der gleich "JDOE" vor einer vorhergehenden HTTP-Antwort ist, bringt die Lade-Operation **606** dieses Steuerobjekt in seinen vorhergehenden Zustand zurück, und zwar zum Teil durch Laden der Textfolge "JDoe" in den Merkmalswert. Ob der Zustand eines gegebenen Objekts gespeichert und wieder hergestellt wird, ist außerdem konfigurierbar.

[0078] Um eine Ausführungsform der vorliegenden Erfindung zusammenzufassen: Der Zustand von ein oder mehr serverseitigen Steuerobjekten wird nach der Verarbeitung "gesichert". Die gesicherte Zustandsinformation wird in einer Antwort zum Client übermittelt. Der Client sendet in einer anschließenden Antwort die gesicherte Zustandsinformation zum Server zurück. Der Server lädt die Zustandsinformation einer frisch instanzierten serverseitigen Steuerobjekthierarchie, so daß der Zustand der Hierarchie in ihren vorhergehenden Zustand zurückgebracht wird.

[0079] Eine alternative Ausführungsform kann die Zustandsinformation am Server oder an einer anderen Weblokation aufrechterhalten, die für den Server während der Rundreise vom Server zum Client und dann zurück zum Server zugänglich ist. Nach Empfang der Anforderung vom Client durch den Server kann diese Zu-

standsinformation vom Server wieder abgerufen und in der Steuerobjekthierarchie in das entsprechende serverseitige Steuerobjekt bzw. die Steuerobjekte geladen werden.

[0080] In der Operation **608** werden von der HTTP-Anforderung empfangene Postback-Daten verabeitet. Postback-Daten können in die Nutzinformation der HTTP-Anforderung eingefügt sein in Schlüsselwert-Paaren, in einer hierarchischen Darstellung (z. B. CML) oder in anderen Datendarstellungen wie RDF ("Resource Description Framework"). Die Operation **608** zerlegt die Nutzinformation, um eine spezielle Kennung eines serverseitigen Steuerobjekts zu identifizieren. Wenn die Kennung (z. B. "page1:text1") gefunden ist und das erkannte serverseitige Steuerobjekt in der Steuerobjekthierarchie existiert, werden die entsprechenden Postback-Daten zum Steuerobjekt weitergeleitet. Unter Bezugnahme auf **Fig. 1** werden beispielsweise eine spezielle Kennung, die der Textbox **106** zugeordnet ist, und der Text "JDoe" in die Nutzinformation der HTTP-Anforderung **114** an den Webserver **116** übermittelt. Die Operation **608** zerlegt die Nutzinformation der HTTP-Anforderung **114** und gewinnt die spezielle Kennung der Textbox **106** und ihren zugehörigen Wert (d. h. "JDoe"). Die Operation **608** wandelt dann die spezielle Kennung der Textbox **106** um, um das entsprechende serverseitige Steuerobjekt zu identifizieren, und leitet den "JDoe"-Wert zum Zielobjekt zum Zweck der Verarbeitung.

[0081] Wie in bezug auf die Lade-Operation **606** erörtert wird, können die Merkmalswerte der serverseitigen Steuerobjekte in ihre früheren Zustände zurückgebracht werden. Als Antwort auf den Empfang von Postback-Daten bestimmt das serverseitige Steuerobjekt, ob der eingeleitete Postback-Wert eine Änderung gegenüber dem vorhergehenden Wert des entsprechenden Merkmals bewirkt. Wenn ja, wird die Änderung in einer Änderungsliste protokolliert, um für das zugeordnete Steuerobjekt eine Datenänderung zu bezeichnen. Nachdem sämtliche Postback-Daten innerhalb der Steuerobjekthierarchie verarbeitet worden sind, kann eine Steuerobjektmethode aufgerufen werden, um ein oder mehr durch Postback-Daten geänderte Ereignisse zu ein oder mehr Nicht-Benutzeroberflächen-Serverkomponenten zu erhöhen, etwa eine Börsenkurs-Nachschlageanwendung, die auf dem Server läuft. Ein Beispiel für ein durch Postback-Daten geändertes Ereignis ist ein Ereignis, das anzeigt, daß Postback-Daten eine Änderung eines Merkmals eines serverseitigen Steuerobjekts bewirkt haben. Bei einer beispielhaften Ausführungsform kann ein solches Ereignis zu einer vom System bereitgestellten Ereignisschlange gesendet werden, so daß serverseitiger Code, der zum Verarbeiten des Ereignisses gespeichert sein kann, aufgerufen werden kann. Der serverseitige Code kann dann eine Methode der Nicht-Benutzeroberfläche-Serverkomponente aufrufen. Auf diese Weise kann eine serverseitige Nicht-Benutzeroberfläche-Serverkomponente auf Ereignisse ansprechen, die von einer Änderung in Daten eines serverseitigen Steuerobjekts ausgelöst wurden. Alternative Methoden zur Implementierung von Ereignissen sind im Umfang der vorliegenden Erfindung ebenfalls möglich, etwa die Verwendung von durch eine Anwendung bereitgestellten Ereignisschlangen, Abfragen und Verarbeitungsunterbrechungen.

[0082] In der Operation **610** werden Postback-Ereignisse gehandelt. Postback-Ereignisse können in der Nutzinformation der HTTP-Anforderung übermittelt werden. Die Operation **610** zerlegt ein bezeichnetes Ereignisziel (z. B. mit "_EVENTTARGET" bei einer Ausführungsform der vorliegenden Erfindung markiert) zur Identifizierung des serverseitigen Steuerobjekts, auf das sich das Ereignis richtet. Ferner zerlegt die Operation **610** die eventuell vorhandenen ermittelten Ereignisargumente und liefert das Ereignisargument (z. B. markiert mit "_EVENTARGUMENT" in einer Ausführungsform der vorliegenden Erfindung) an das angegebene serverseitige Steuerobjekt. Das Steuerobjekt hebt seine Ereignisse zur Verarbeitung durch serverseitigen Code an, der eine Methode einer Nicht-Benutzeroberfläche-Serverkomponente (z. B. einer serverseitigen Börsenkurs-Nachschlaganwendung) aufruft, die der Ressource mit dynamischem Inhalt zugeordnet ist.

[0083] Die Operation **612** löst Datenbindungsbeziehungen zwischen den serverseitigen Steuerobjekten und ein oder mehr Datenbanken, auf die der Server Zugriff hat, auf, wodurch in Steuerobjektmerkmalen Datenbankwerte aktualisiert werden und/oder Datenbankfelder mit Werten von Steuerobjektmerkmalen aktualisiert werden. Bei einer Ausführungsform der vorliegenden Erfindung können Merkmale von serverseitigen Steuerobjekten Merkmalen eines Vater-Datenbindungscontainers wie etwa einer Tabelle in einer serverseitigen Anwendungsdatenbank zugeordnet (oder datengebunden) werden. Während der Datenbindungsoperation **612** kann das Seitengerüst ein datengebundenes Steuerobjektmerkmal mit dem Wert des entsprechenden Merkmals des Vater-Datenbindungscontainers aktualisieren. Auf diese Weise reflektieren Benutzeroberflächenelemente auf der Webseite der nächsten Antwort exakt aktualisierte Merkmalswerte, weil die Steuerobjektmerkmale, denen die Benutzeroberflächenelemente entsprechen, während der Datenbindungsoperation **612** automatisch aktualisiert worden sind. Ebenso können Steuerobjektmerkmale zu den Vater-Datenbindungs-Containerfeldern aktualisiert werden, wodurch eine serverseitige Anwendungsdatenbank mit Postback-Eingaben von einem serverseitigen Steuerobjekt aktualisiert wird.

[0084] Die Operation **614** führt verschiedene Aktualisierungsoperationen aus, die ausgeführt werden können,

bevor der Steuerobjektzustand gesichert und die Ausgabe erzeugt wird. Die Operation **616** fordert Zustandsinformationen (d. h. den Betrachtungszustand) von ein oder mehr Steuerobjekten in der Steuerobjekthierarchie an und speichert die Zustandsinformationen zur Einfügung in eine transportfähige Zustandsstruktur, die in der HTTP-Nutzantwort zum Client übermittelt wird. Beispielsweise kann ein "grid"-Steuerobjekt eine aktuelle Indexseite einer Liste von Werten sichern, so daß das "grid"-Steuerobjekt nach einer darauf folgenden HTTP-Anforderung (d. h. in der Operation **606**) in diesen Zustand zurückgebracht werden kann. Wie oben beschrieben wird, repräsentiert die Betrachtungszustandsinformation den Zustand der Steuerobjekthierarchie vor allen folgenden Aktionen durch den Client. Wenn die Betrachtungszustandsinformation zurückgegeben wird, wird sie genutzt, um die Steuerobjekthierarchie in diesen vorhergehenden Zustand zu bringen, bevor irgendwelche clientseitigen Postback-Eingaben oder Datenbindungen verarbeitet werden.

[0085] Die Erzeugen-Operation **618** erzeugt die geeignete Autorensprache-Ausgabe (z. B. HTML-Daten) für die Übermittlung zum Client in einer HTTP-Antwort. Die Erzeugung erfolgt durch einen von oben nach unten verlaufenden hierarchischen Baumdurchlauf von allen serverseitigen Steuerobjekten und von eingebettetem Erzeugungscode. Die Operation **620** führt alle letzten Aufräumarbeiten durch (z. B. Schließen von Dateien oder Datenbankverbindungen), bevor die Steuerobjekthierarchie abgeschlossen wird. Die Verarbeitung springt dann zu Operation **602** zurück und läuft bis zu Operation **622**, wo das Seitenobjekt durch Aufrufen seines Destruktors abgeschlossen wird.

[0086] Fig. 7 zeigt eine Darstellung einer beispielhaften serverseitigen Steuerklasse bei einer Ausführungsform der vorliegenden Erfindung. Die serverseitige Steuerklasse definiert die Methoden, Merkmale und Ereignisse, die allen serverseitigen Steuerobjekten einer Ausführungsform der vorliegenden Erfindung gemeinsam sind. Speziellere Steuerklassen (z. B. ein serverseitiges Tastensteuerobjekt, das einer clientseitigen Taste in einer Webseite entspricht) werden von der Steuerklasse abgeleitet. Eine dargestellte Steuerklasse **700** weist einen Speicher auf, in dem Merkmale **702** und Methoden **704** abgespeichert sind. Andere Ausführungsformen von Steuerklassen mit einer anderen Kombination von Datenelementen und Methoden liegen ebenfalls im Rahmen der vorliegenden Erfindung.

[0087] Bei der gezeigten Ausführungsform sind Merkmale **702** öffentlich. Das Merkmal "ID" ist ein lesbarer und schreibbarer String- bzw. Kettenwert, der eine Steuerobjektkennung bezeichnet. Das Merkmal "Visible" ist ein lesbarer und schreibbarer boolescher Wert, der angibt, ob die Autorensprache-Daten für ein entsprechendes clientseitiges Benutzeroberflächenelement erzeugt werden sollen. Das Merkmal "MaintainState" ist ein lesbarer und schreibbarer boolescher Wert, der angibt, ob das Steuerobjekt seinen Betrachtungszustand (und den Betrachtungszustand seiner Söhne) am Ende der aktuellen Seitenanforderung (d. h. als Antwort auf eine Sicherungsoperation **616** in Fig. 6) sichern soll. Das Merkmal "Parent" ist eine lesbare Referenz zu einem ControlContainer (siehe Fig. 8), der dem aktuellen Steuerobjekt in der Steuerobjekthierarchie zugeordnet ist. Das Merkmal "Page" ist eine lesbare Referenz zu dem Stammseitenobjekt, in dem das aktuelle Steuerobjekt beherbergt ist. Das Merkmal "Trace" ist eine lesbare Referenz, die es einem Entwickler erlaubt, ein Ablaufverfolgungsprotokoll zu schreiben. PropertyBindingContainer ist eine lesbare Referenz auf den unmittelbaren Datenbindungsbehälter des Steuerobjekts. Das Merkmal "Bindings" ist eine lesbare Referenz auf eine Sammlung der Datenbindungszuordnungen des Steuerobjekts.

[0088] Die Methoden **704** umfassen Methoden zum Verarbeiten von Anforderungen und zum Zugriff auf Datenelemente des Steuerobjekts. Bei einer Ausführungsform erfolgt Bezugnahme auf die Methoden durch Zeiger, die in dem Speicherplatz des Steuerobjekts abgespeichert sind. Diese Referenzeinrichtung wird auch bei Ausführungsformen von anderen serverseitigen Objekten angewandt, etwa einem Container-Steuerobjekt, einem Steuersammelobjekt und einem Seitenobjekt. Die Methode "Init()" dient dazu, Sohn-Steuerobjekte zu initialisieren, nachdem sie erzeugt worden sind (siehe Operation **604** von Fig. 6). Die Methode "Load()" dient dazu, die Betrachtungszustandsinformation aus einer vorhergehenden HTTP-Anforderung wieder herzustellen (siehe Operation **606** von Fig. 6). Die Methode "Save()" dient dazu, Betrachtungszustandsinformation zur Verwendung mit einer späteren HTTP-Anforderung zu sichern (siehe Operation **616** von Fig. 6). Die Methode "PreRender()" dient dazu, alle notwendigen Vor-Erzeugungsschritte vor der Sicherung des Betrachtungszustands und des Erzeugungsinhalts auszuführen (siehe Operation **614** von Fig. 6). Die Methode "Render (TextWriter output)" dient dazu, Autorensprache-Code für das Benutzeroberflächenelement auszugeben, das dem aktuellen Steuerobjekt entspricht (siehe Operation **618** von Fig. 6). Der Code wird durch den Ausgabestrom (der ihm im "output"-Parameter zugeführt wird) übermittelt, um den Code in der Antwort an den Client zu speichern. Die Methode "Dispose()" dient dazu, letzte Aufräumarbeiten auszuführen, bevor das Steuerobjekt beendet wird (siehe Operation **620** von Fig. 6).

[0089] Die Methode "GetUniqueID()" beschafft eine spezielle, hierarchisch qualifizierte Kettenkennung für das

aktuelle Steuerobjekt. Die Methode "GetControlWithID(String id)" gibt eine Referenz zu einem unmittelbaren Sohn-Steuerobjekt mit der bereitgestellten Kennung ("id") zurück. Die Methode "GetControlWithUniqueID(String id)" gibt eine Referenz zu einem Sohn-Steuerobjekt zurück, das eine spezielle hierarchische Kennung ("id") hat.

[0090] Die Methode "PushControlPxopertyTwoBindingContainer (String prop Name)" dient dazu, einen Bindungsbehälter für die Zweiwege-Datenbindung von Postback-Daten zu aktualisieren, wenn sich der Postback-Datenwert innerhalb des serverseitigen Steuerobjekts ändert. Die Methode "PushBindingContainerPropertyTwoControl(String prop Name)" dient dazu, ein serverseitiges Steuerobjektmerkmal mit einem aktuellen Bindungsbehälterwert zu aktualisieren. Die Methode "HasBindings()" gibt einen booleschen Wert zurück, der bezeichnet, ob ein Steuerobjekt irgendwelche Bindungszuordnungen hat. Diese drei Funktionen werden verwendet, um Bindungsbeziehungen zwischen Merkmalen von serverseitigen Steuerobjekten und Attributen in serverseitigen Datenspeichern aufzulösen (siehe die Datenbindungsoperation **612** von Fig. 6).

[0091] Fig. 8 zeigt eine beispielhafte serverseitige ContainerControl-Klasse in einer Ausführungsform der vorliegenden Erfindung. Die ContainerControl-Klasse ermöglicht eine Implementierung, die verschachtelte Sohn-Steuerobjekte unterstützt und automatisch Betrachtungszustand-Information in eine transportfähige Zustandsstruktur serialisiert und deserialisiert. Eine ContainerControl-Klasse **800** weist einen Speicher auf, der Merkmale **802** und Methoden **804** speichert. Das Merkmal "Controls" ist eine lesbare Referenz zu einer "ControlCollection" der Sohn-Steuerobjekte in der Steuerobjekthierarchie (siehe Fig. 9). Das Merkmal "StateCollection" ist eine lesbare Referenz auf ein Lexikon von Betrachtungszustandsinformation, die dazu dient, den Zustand eines Steuerobjekts über eine Vielzahl von Seitenanforderungen aufrechtzuerhalten. Die Methode "HasControls()" gibt einen booleschen Wert zurück, der anzeigt, ob das Steuerobjekt irgendwelche Sohn-Steuerobjekte hat.

[0092] Bei einer alternativen Ausführungsform können die Elementmerkmale und Methoden der ContainerControl-Klasse **800** in die Control-Klasse **700** von Fig. 7 kombinatorisch eingebracht werden, so daß jedes Steuerobjekt imstande ist, in sich und von sich aus Söhne zu unterstützen. Wenn ein Steuerobjekt einer solchen Ausführungsform jedoch keine Sohn-Objekte aufweist, ist das Controls-Element (vom ControlCollection-Typ) leer (d. h. es hat keine Sohn-Objekte).

[0093] Fig. 9 zeigt eine beispielhafte serverseitige ControlCollection-Klasse in einer Ausführungsform der vorliegenden Erfindung. Eine ControlCollection-Klasse **900** umfaßt einen Speicher zum Speichern von Merkmalen **902** und Methoden **904**. Das Merkmal "All" ist eine lesbare und schreibbare Schnappschußanordnung aller Sohn-Steuerobjekte des aktuellen Steuerobjekts, geordnet nach einem Index. Merkmal "this[index]" ist eine lesbare Referenz zu einem nach Ordnungszahlen indexierten Steuerobjekt innerhalb der Steuerkollektion. Das Merkmal "Count" ist ein lesbarer Wert, der die Anzahl von Sohn-Steuerobjekten in der Kollektion bezeichnet.

[0094] Die Methode "Add(Control child)" dient dazu, ein bestimmtes Steuerobjekt zu der aktuellen Kollektion zu addieren. Die Methode "IndexOf(Contxol child)" gibt den Ordnungszahlenindex des bestimmten Sohn-Steuerobjekts in die Kollektion zurück. Die Methode "GetEnumerator(boolAllowRemove)" gibt den Zähler aller Sohn-Steuerobjekte innerhalb der Kollektion zurück. Die Methode "Remove(Control value)" entfernt das bestimmte Steuerobjekt aus der aktuellen Kollektion. Die Methode "Remove(int index)" entfernt ein bestimmtes Steuerobjekt aus der aktuellen Sammlung auf der Basis des angegebenen Index. Die Methode "Clear()" entfernt sämtliche Steuerobjekte aus der aktuellen Kollektion.

[0095] Fig. 10 zeigt ein beispielhaftes serverseitiges Seiten- bzw. Page-Objekt bei einer Ausführungsform der vorliegenden Erfindung. Eine Seitengrundklasse definiert die Methoden, Merkmale und Ereignisse, die allen server-manipulierten Seiten innerhalb einer Ausführungsform der vorliegenden Erfindung gemeinsam sind. Ein Seitenobjekt **1000** weist einen Merkmale **1002** speichernden Speicher und Methoden **1004** auf. Das Merkmal "ErrorPage" ist eine lesbare und schreibbare Kette, die im Fall einer nichtgehandelten Seitenausnahme erzeugt wird. Auf diese Weise gibt ein Seitenobjekt eine lesbare Fehlerseite an den Client in einer HTTP-Antwort zurück. Das Merkmal "Requests" ist eine lesbare Referenz zu einem HTTPRequest, das von der Webserver-Grundstruktur geliefert wird und die Funktionalität für den Zugriff auf ankommende HTTP-Anforderungsdaten liefert. Das Merkmal "Response" ist eine lesbare Referenz zu einer HTTP-Antwort, die von der Webserver-Grundstruktur bereitgestellt wird und die Funktionalität zum Übertragen von HTTP-Antwortdaten zu einem Client bereitstellt. Das Merkmal "Application" ist eine lesbare Referenz zu einer HTTPApplication, die von der Webserver-Grundstruktur bereitgestellt wird. Das Merkmal "Session" ist eine lesbare Referenz zu einer HTTPSession, die von der Webserver-Grundstruktur bereitgestellt wird. Das Merkmal "Server" ist eine lesbare Re-

ferenz zu einem ServerObjekt, das ein seitenkompatibles Anwendungsserver-Nutzobjekt ist. Das Merkmal "Context" ist eine lesbare Referenz zu sämtlichen Webserver-Objekten, die von der Webserver-Grundstruktur bereitgestellt werden, was es einem Entwickler ermöglicht, Zugriff auf zusätzliche exponierte Fließband-Modul-Objekte zu erhalten. Das Merkmal "IsFirstLoad" ist ein lesbarer boolescher Wert, der angibt, ob das Seitenobjekt gerade erstmals oder als Antwort auf eine clientseitige Postback-Anforderung geladen wird und ob darauf zugegriffen wird.

[0096] Die Methode "Load()" dient dazu, das Seitenobjekt zu initialisieren und Betrachtungszustandsinformation von der vorhergehenden Seitenanforderung wiederherzustellen. Die Methode "Save()" dient dazu, Betrachtungsinformation zum Gebrauch mit einer späteren Seitenanforderung zu sichern. Die Methode "HandleErxor(Exception errorInfo)" dient dazu, einen nicht-gehandelten Fehler, der während einer Seitenausführung auftritt, zu handeln. In diesem Fall leitet die Grundklassen-Implementierung den Client zu einer URL-Adresse, die eine Standardfehler-Webseite hat, um. Die Methode "GetPostbackScript(Control target, String name, String arg)" bringt eine einem gegebenen Steuerobjekt zugeordnete clientseitige Scriptmethode zurück. Die Methode "RegisterClientScript-Block(String key, String script)" dient dazu, doppelte Blöcke von clientseitigem Scriptcode, der an den Client gesendet wird, zu eliminieren. Die Methode "IHTTPHandler.ProcessRequest(HTTPContextContext)" dient dazu, Webanforderungen zu verarbeiten. IHTTPHandler.ProcessRequest wird aufgerufen, um die Verarbeitung einer HTTP-Anforderung zu initiieren, die von einem Client empfangen wurde (siehe Operation **602** von Fig. 6). Die Methode "IHTTPHandler.IsReusable()" gibt an, ob ein Seitenobjekt erneut verwendet werden kann, um eine Vielzahl von Webanforderungen zu bedienen.

[0097] Die hier beschriebenen Ausführungsformen der Erfindung sind als logische Schritte in ein oder mehr Computersystemen implementiert. Die logischen Operationen der vorliegenden Erfindung sind implementiert (1) als eine Folge von prozessor-implementierten Schritten, die in ein oder mehr Computersystemen ausgeführt werden, und (2) als miteinander verbundene Maschinenmodule innerhalb von ein oder mehr Computersystemen. Die Implementierung ist eine Frage der Wahl und abhängig von den Leistungsanforderungen des die Erfindung implementierenden Computersystems. Daher werden die logischen Operationen, welche die Ausführungsformen der Erfindung bilden, hier unterschiedlich als Operationen, Schritte, Objekte oder Module bezeichnet.

[0098] Die vorstehende Beschreibung, die Beispiele und die Daten ergeben eine vollständige Beschreibung der Herstellung und des Gebrauchs des Aufbaus der Erfindung. Viele Ausführungsformen der Erfindung können realisiert werden, ohne vom Umfang der Erfindung abzuweichen, wie er in den beigefügten Patentansprüchen definiert ist.

Patentansprüche

1. Verfahren zum serverseitigen Verarbeiten mindestens eines clientseitigen Benutzeroberflächenelements (**106**, **108**, **110**), das in eine Web-Seite (**104**) eingebaut ist, die auf einem Client (**100**) angezeigt wird, wobei das Verfahren die folgenden Schritte aufweist:
Lesen (**203**) einer Vereinbarung von einer Resource (**310**; **400**);
Erzeugen (**204**) einer Vielzahl von gleichzeitig existierenden serverseitigen Steuerobjekten (**318**, **320**, **322**), die dem clientseitigen Benutzeroberflächenelement logisch entsprechen, auf der Basis der Vereinbarung;
Verarbeiten (**206**) der clientseitigen Benutzeroberflächenelement-Ereignisse unter Verwendung der gleichzeitig existierenden serverseitigen Steuerobjekte; und
Erzeugen (**208**) von Autorensprache-Daten aus den gleichzeitig existierenden serverseitigen Steuerobjekten, um das clientseitige Benutzeroberflächenelement im Anschluß an den Verarbeitungsvorgang in die Web-Seite einzubauen.

2. Verfahren nach Anspruch 1, wobei der Vorgang des Erzeugens einer Vielzahl von gleichzeitig existierenden serverseitigen Steuerobjekten die folgenden Schritte aufweist:
Erzeugen eines ersten serverseitigen Steuerobjekts, das einem ersten clientseitigen Benutzeroberflächenelement entspricht; und
Erzeugen eines zweiten serverseitigen Steuerobjekts, das einem ersten clientseitigen Benutzeroberflächenelement entspricht, so dass das erste und das zweite serverseitige Steuerobjekt gleichzeitig existieren.

3. Verfahren nach Anspruch 1 oder 2, wobei der Verarbeitungsvorgang den folgenden Schritt aufweist:
Aufrufen eines serverseitigen Ereignisses, das einem ersten serverseitigen Steuerobjekt zugeordnet ist.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei der Verarbeitungsvorgang ferner den folgenden

Schritt aufweist:
Bearbeiten des serverseitigen Ereignisses unter Verwendung eines zweiten serverseitigen Steuerobjekts.

5. Verfahren nach einem der Ansprüche 1 bis 4, wobei der Verarbeitungsvorgang ferner den folgenden Schritt aufweist:
Bearbeiten des serverseitigen Ereignisses unter Verwendung einer Nicht-Benutzeroberflächen-Serverkomponente.

6. Verfahren nach einem der Ansprüche 1 bis 5, wobei der Vorgang des Erzeugens einer Vielzahl von gleichzeitig existierenden serverseitigen Steuerobjekten die folgenden Schritte aufweist:
Erzeugen eines ersten serverseitigen Steuerobjekts, das einem clientseitigen Benutzeroberflächenelement entspricht; und
Erzeugen eines zweiten serverseitigen Steuerobjekts, das dem clientseitigen Benutzeroberflächenelement entspricht, so dass das erste und das zweite serverseitige Steuerobjekt gleichzeitig existieren.

7. Computerdatensignal, das in einer Trägerwelle verkörpert ist, die mit einem Computersystem (**500**) lesbar ist und ein Computerprogramm codiert, das, wenn es auf dem Computersystem abläuft, sämtliche Schritte des Verfahrens von Anspruch 1 ausführt.

8. Computerprogramm-Datenträger (**516**), der mit einem Computersystem (**500**) lesbar ist und ein Computerprogramm codiert, das, wenn es auf dem Computersystem abläuft, sämtliche Schritte des Verfahrens von Anspruch 1 ausführt.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

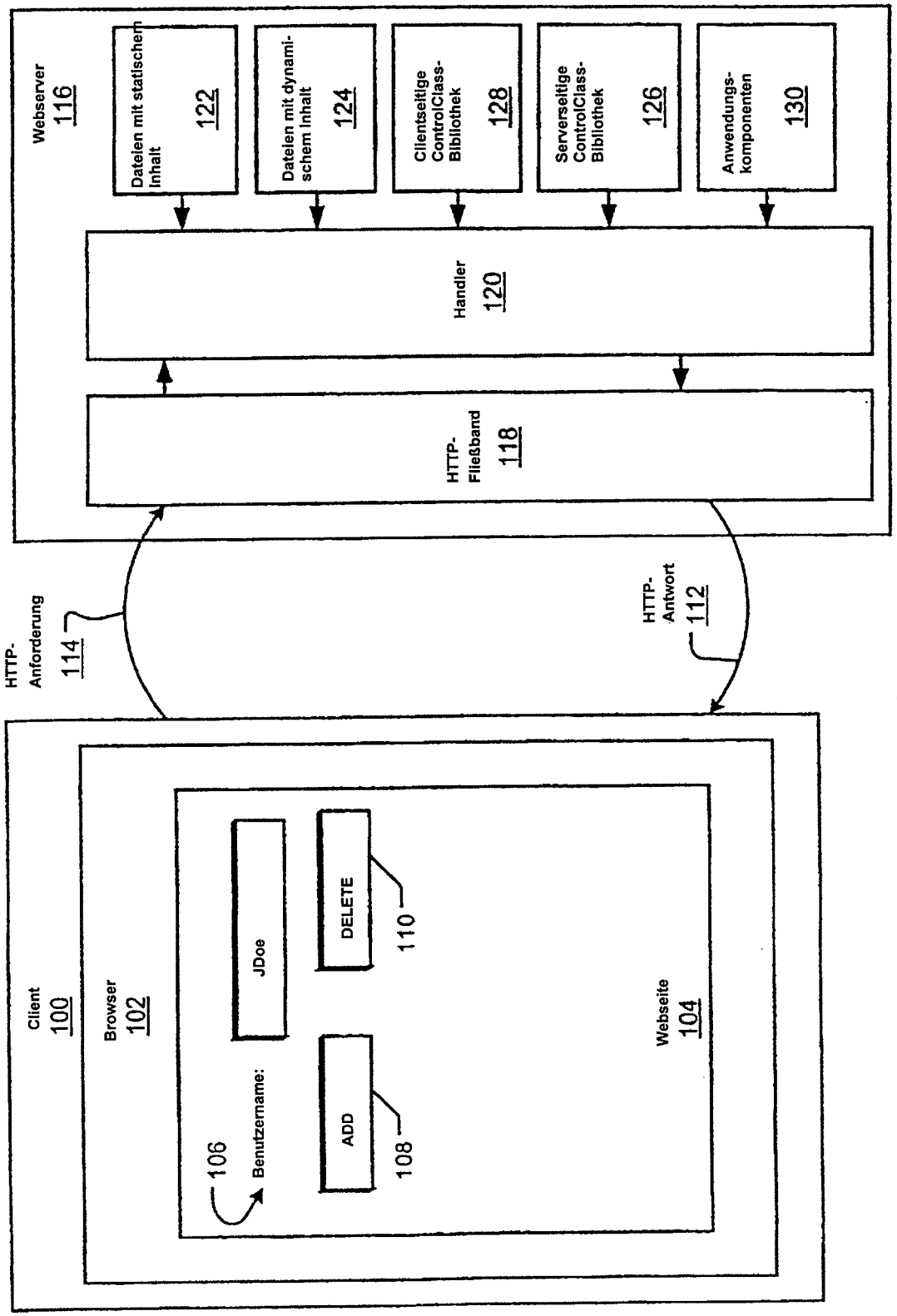


FIG. 1

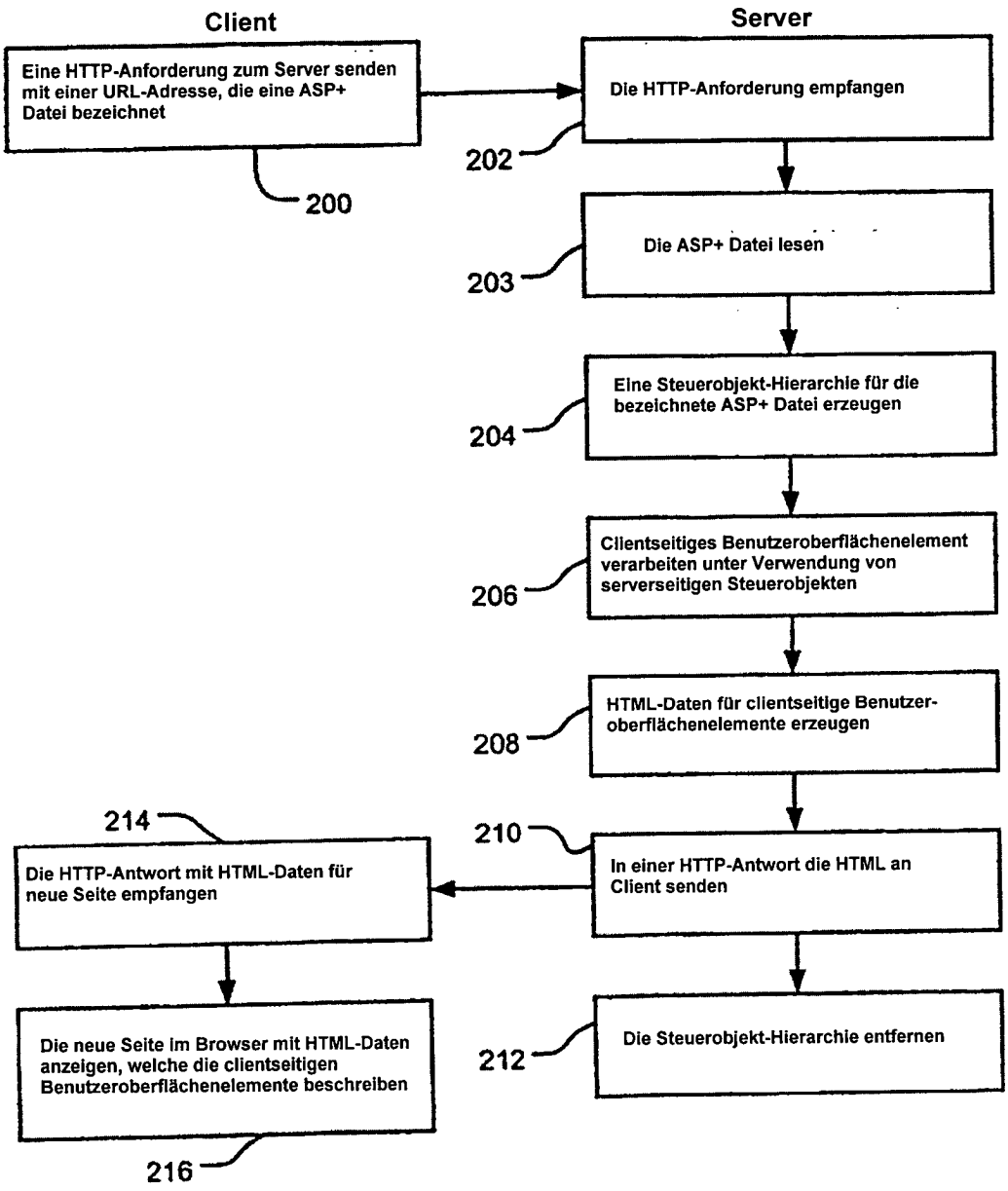

Client
Server
Eine HTTP-Anforderung zum Server senden mit einer URL-Adresse, die eine ASP+ Datei bezeichnet
200
Die HTTP-Anforderung empfangen
202
Die ASP+ Datei lesen
203
Eine Steuerobjekt-Hierarchie für die bezeichnete ASP+ Datei erzeugen
204
Clientseitiges Benutzeroberflächenelement verarbeiten unter Verwendung von serverseitigen Steuerobjekten
206
HTML-Daten für clientseitige Benutzer-oberflächenelemente erzeugen
208
210
In einer HTTP-Antwort die HTML an Client senden
212
Die Steuerobjekt-Hierarchie entfernen
214
Die HTTP-Antwort mit HTML-Daten für neue Seite empfangen
Die neue Seite im Browser mit HTML-Daten anzeigen, welche die clientseitigen Benutzeroberflächenelemente beschreiben
216

FIG. 2

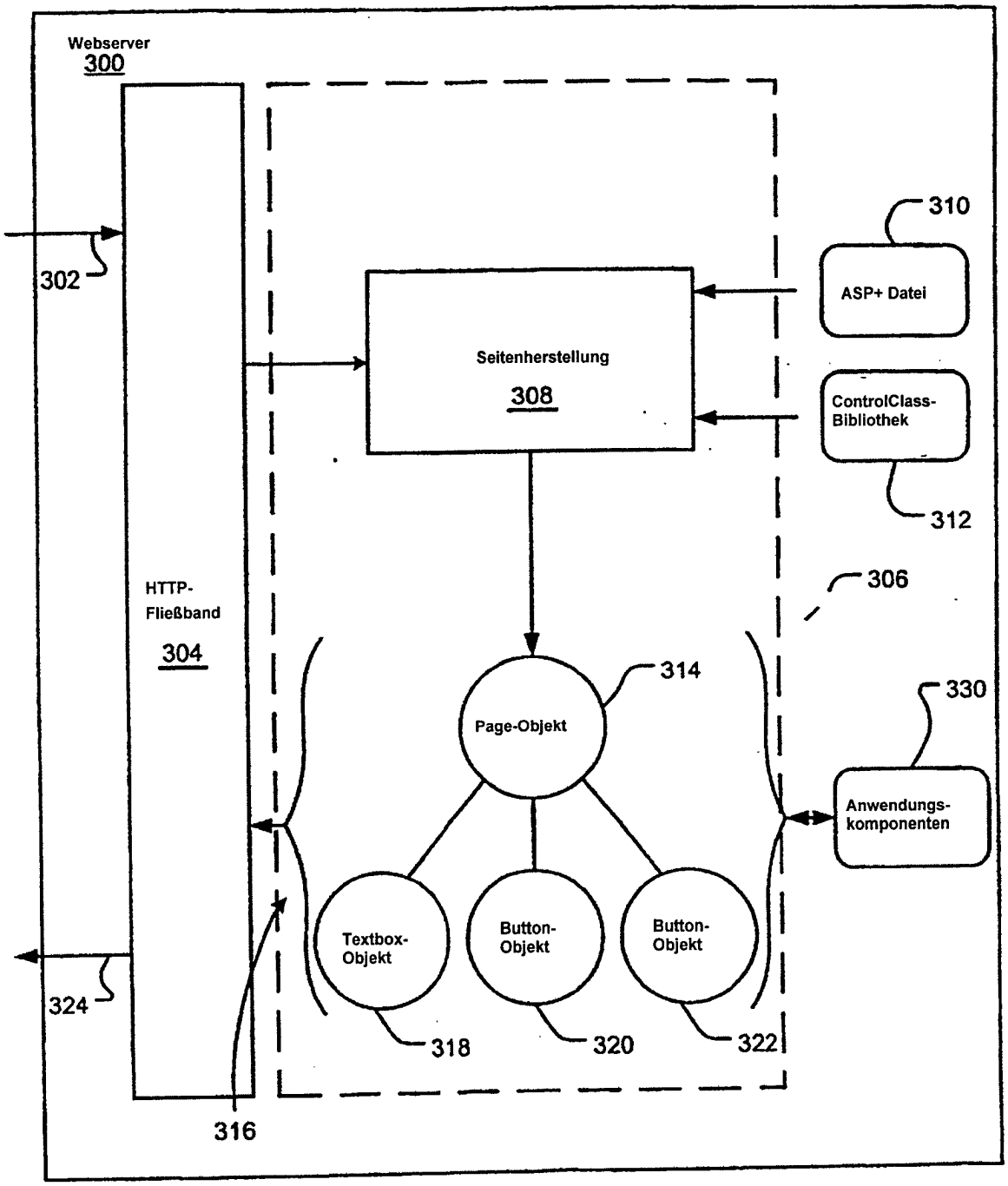
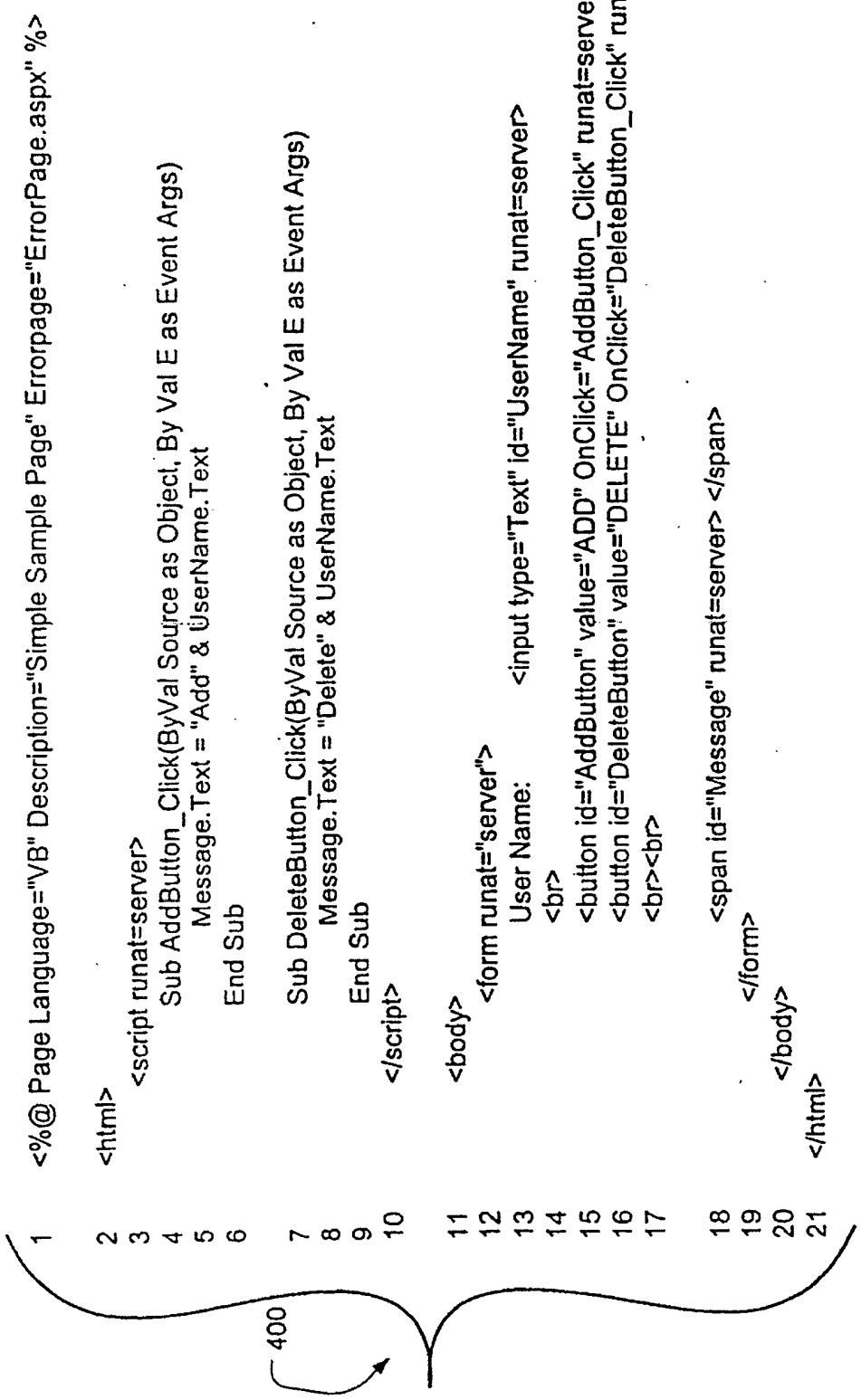

Webserver
300
302
HTTP-
Fließband
304
Seitenherstellung
308
310
ASP+ Datei
ControlClass-
Bibliothek
312
306
314
Page-Objekt
330
Anwendungs-
komponenten
Textbox-
Objekt
Button-
Objekt
Button-
Objekt
318
320
322
324
316

FIG. 3


400
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
15 <button id="AddButton" value="ADD" OnClick="AddButton_Click" runat=server>
16 <button id="DeleteButton" value="DELETE" OnClick="DeleteButton_Click" runat=server>
17

18
19 </form>
20 </body>
21 </html>

FIG. 4

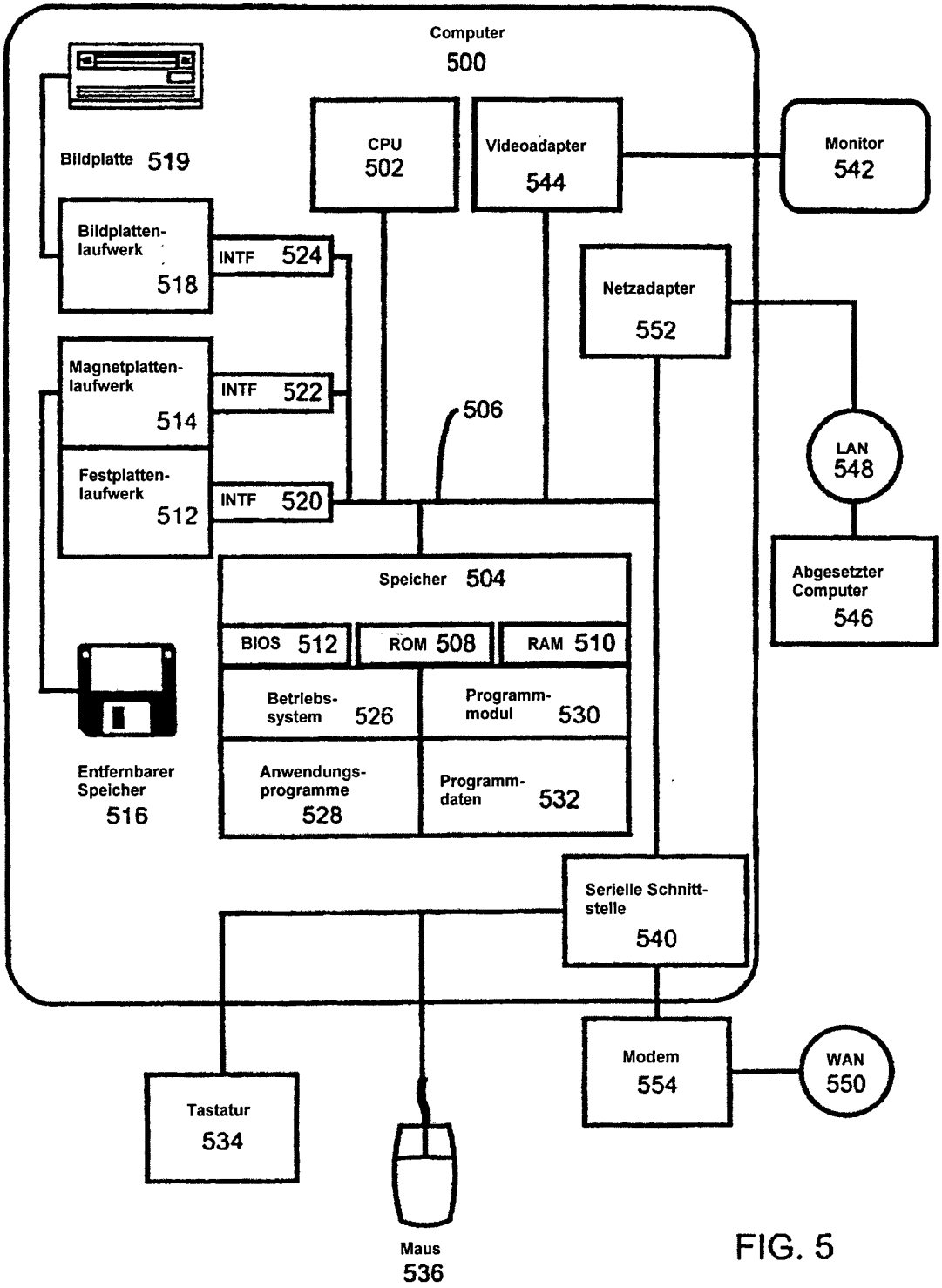

Computer
500
Bildplatte 519
CPU
502
Videoadapter
544
Monitor
542
Bildplatten-
laufwerk
518
INTF 524
Netzadapter
552
Magnetplatten-
laufwerk
514
INTF 522
506
LAN
548
Festplatten-
laufwerk
512
INTF 520
Abgesetzter
Computer
546
Speicher 504
BIOS 512
ROM 508
RAM 510
Betriebs-
system 526
Programm-
modul 530
Entfernbarer
Speicher
516
Anwendungs-
programme
528
Programm-
daten 532
Serielle Schnitt-
stelle
540
Modem
554
WAN
550
Tastatur
534
Maus
536

FIG. 5

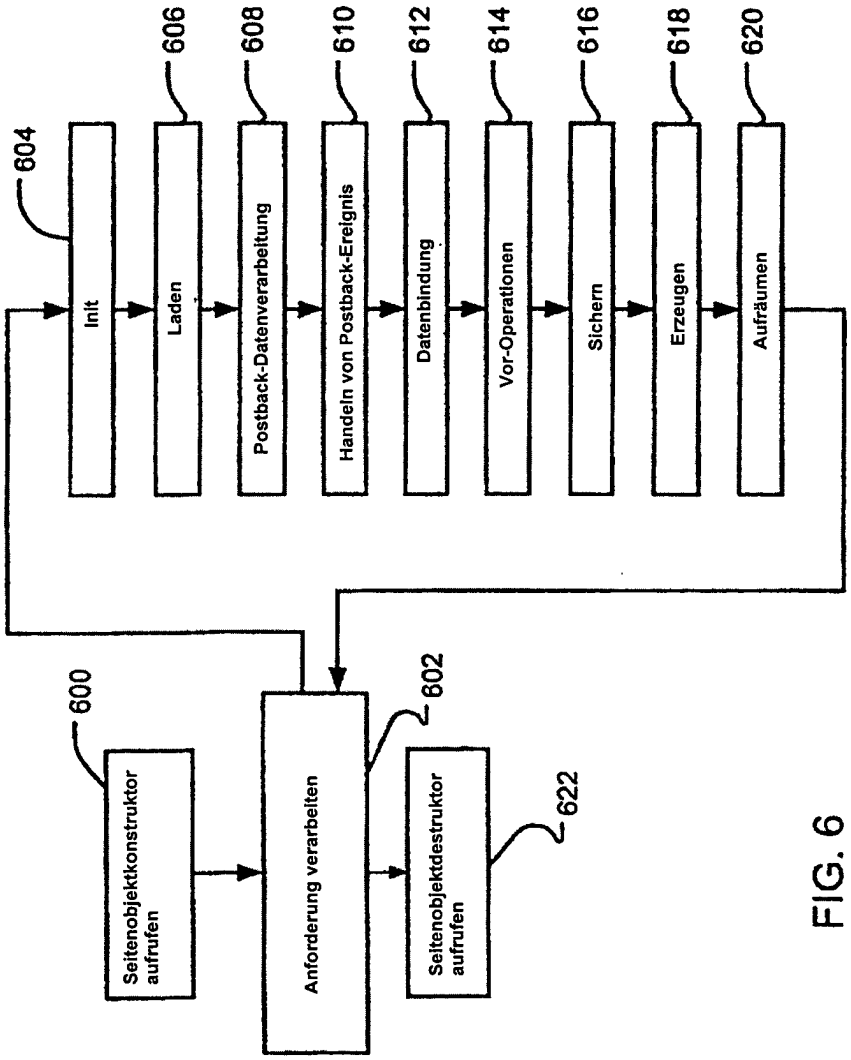

604
606
608
610
612
614
616
618
620
Init
Laden
Postback-Datenverarbeitung
Handeln von Postback-Ereignis
Datenbindung
Vor-Operationen
Sichern
Erzeugen
Aufräumen
600
602
622
Seitenobjektkonstruktor aufrufen
Anforderung verarbeiten
Seitenobjektdestruktor aufrufen

FIG. 6

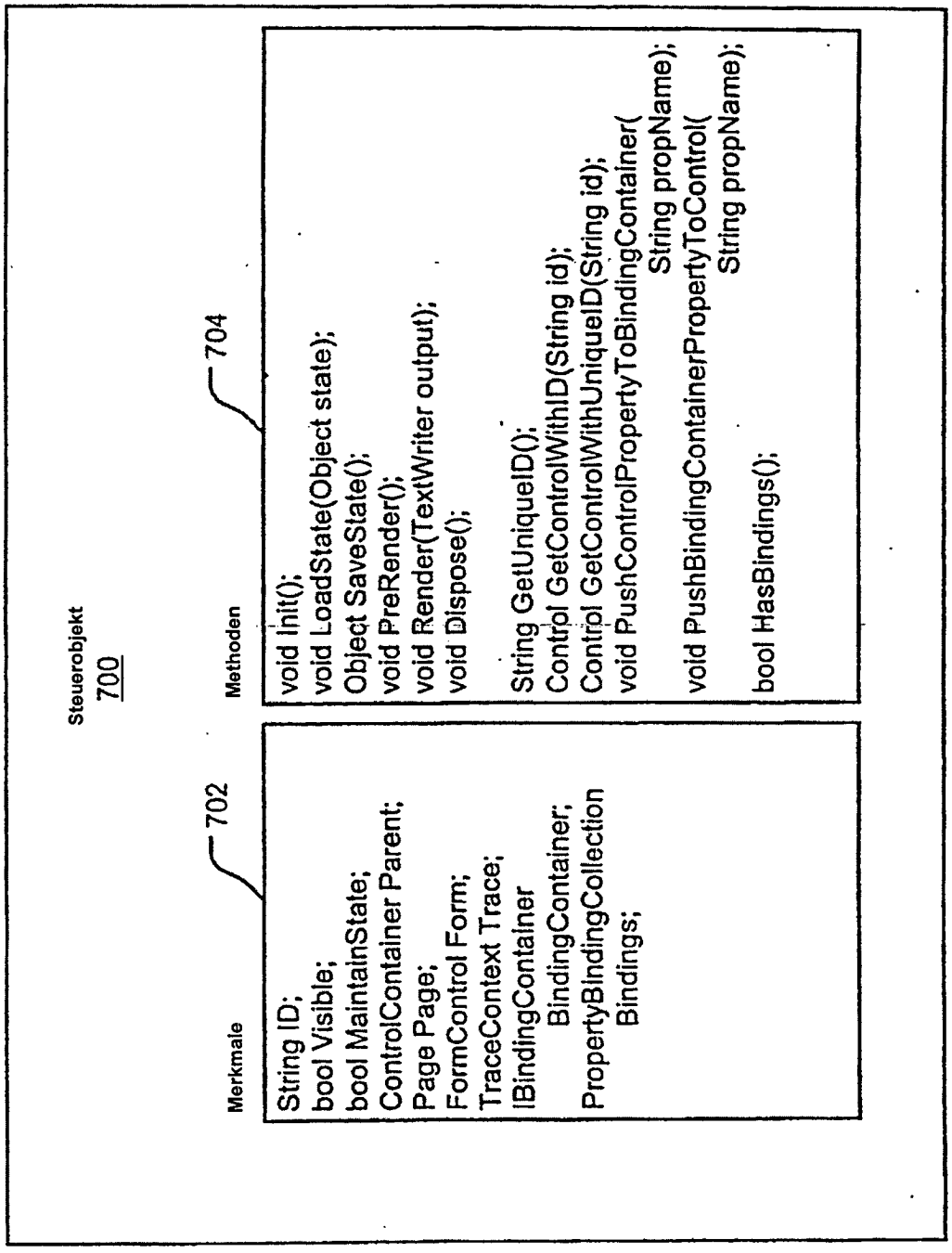

Steuerobjekt
700
Merkmale
702
String ID;
bool Visible;
bool MaintainState;
ControlContainer Parent;
Page Page;
FormControl Form;
TraceContext Trace;
IBindingContainer
BindingContainer;
PropertyBindingCollection
Bindings;
Methoden
704
void Init();
void LoadState(Object state);
Object SaveState();
void 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();

FIG. 7

FIG. 8

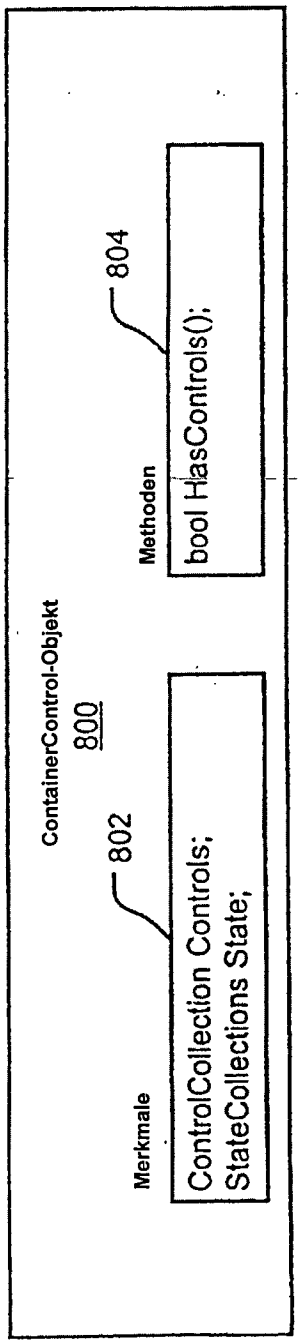
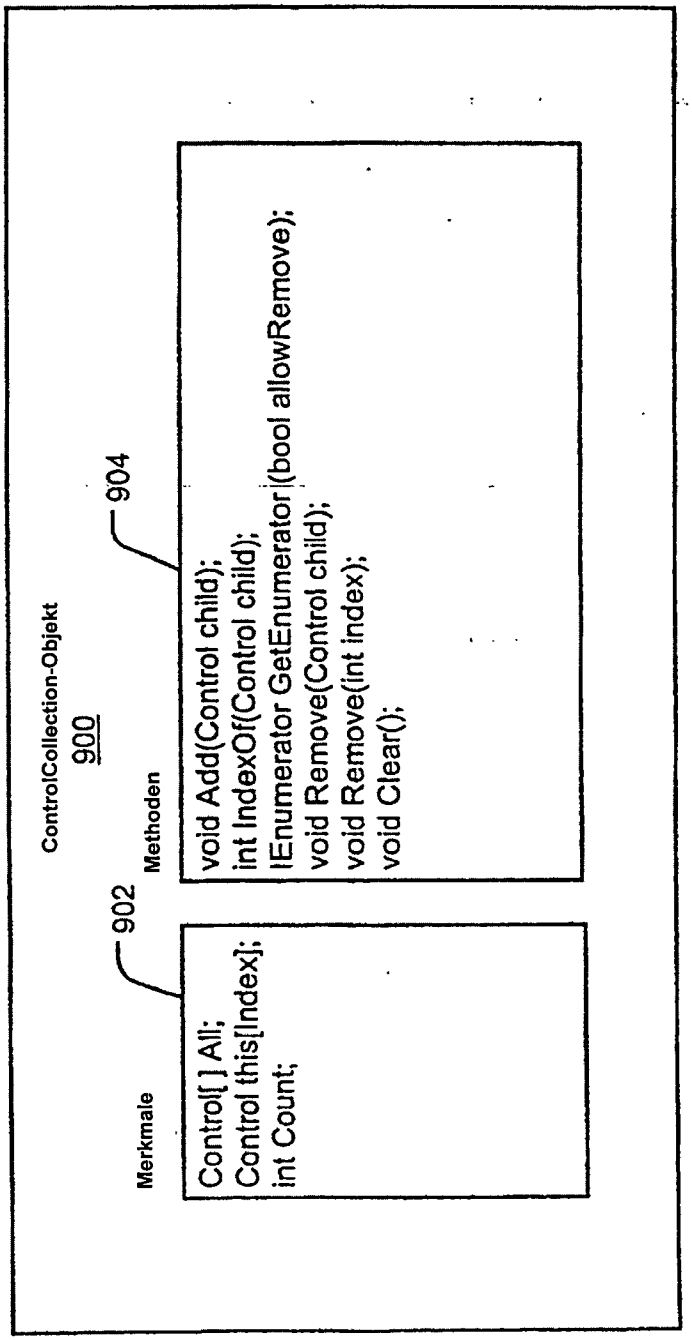

ContainerControl-Objekt
800
802
Merkmale
ControlCollection Controls;
StateCollections State;
804
Methoden
bool HasControls();

FIG. 9


ControlCollection-Objekt
900
902
Merkmale
Control[ ] All;
Control this[Index];
int Count;
904
Methoden
void Add(Control child);
int IndexOf(Control child);
IEnumerator GetEnumerator(bool allowRemove);
void Remove(Control child);
void Remove(int index);
void Clear();

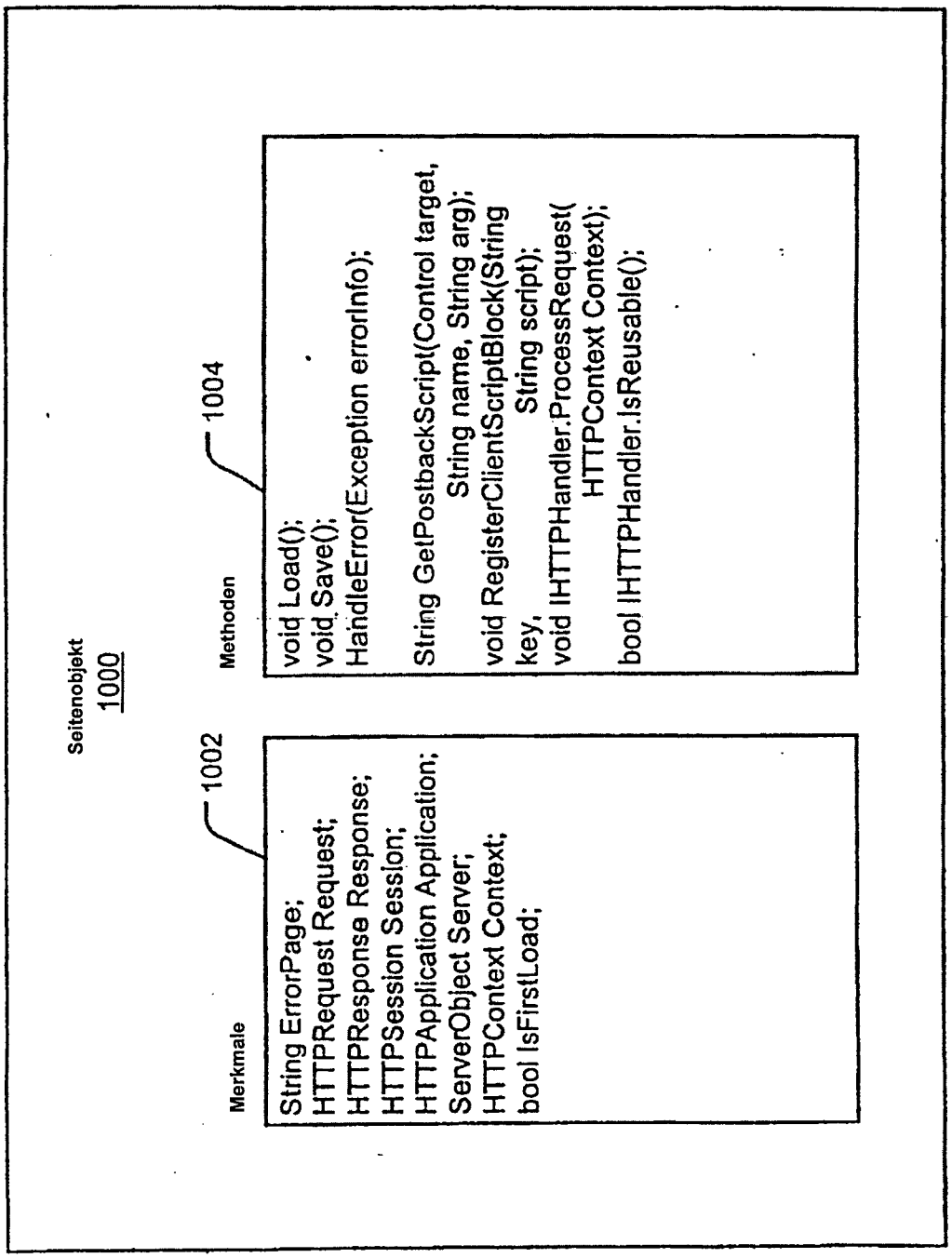

Seitenobjekt
1000
Merkmale
1002
String ErrorPage;
HTTPRequest Request;
HTTPResponse Response;
HTTPSession Session;
HTTPApplication Application;
ServerObject Server;
HTTPContext Context;
bool IsFirstLoad;
Methoden
1004
void Load();
void Save();
HandleError(Exception errorInfo);
String GetPostbackScript(Control target,
String name, String arg);
void RegisterClientScriptBlock(String
key, String script);
void IHTTPHandler.ProcessRequest(
HTTPContext Context);
bool IHTTPHandler.IsReusable();

FIG. 10