(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-293452
(P2007-293452A)
(43)公開日　平成19年11月8日(2007.11.8)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/21	(2006.01)	G06F	17/21	538M	5B009
G06F 3/048	(2006.01)	G06F	3/048	651B	5E501
		G06F	17/21	501T	

審査請求 未請求　請求項の数 8　ＯＬ　（全 18 頁）

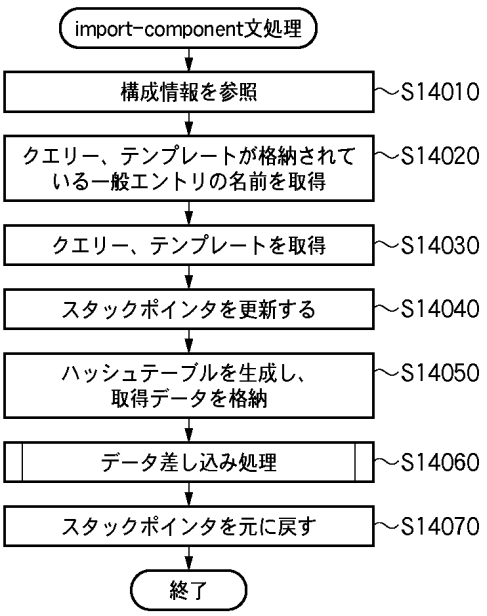
(21)出願番号　特願2006-118444 (P2006-118444)
(22)出願日　平成18年4月21日 (2006.4.21)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(72)発明者　牛久　豊彦
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

Ｆターム(参考)　5B009 ND02　ND03　QA06　VA02
5E501 AA02　AC34　CC16　DA06

(54)【発明の名称】文書処理装置及びその制御方法、プログラム、記憶媒体

(57)【要約】
【課題】　データ参照を伴う文書部品の差し込みを可能にすることで、文書部品の再利用性を高める技術を提供する。
【解決手段】　文書生成装置は、第１及び第２の文書のテンプレート情報と、第２の文書におけるデータ参照に係る情報を示すクエリー情報と、を記憶する記憶手段と、外部装置からクエリー情報に基づいてデータを取得する取得手段と、取得手段により取得されたデータと、第１及び第２の文書のテンプレート情報と、に基づいて第１の文書に係る文書データを生成する生成手段と、を備え、第１の文書のテンプレート情報には第２の文書の差し込み位置を示す差し込み情報が含まれ、生成手段は、クエリー情報に基づいて取得手段により取得されたデータを含む、第２の文書が差し込み位置に差し込まれた、第１の文書に係る文書データを生成することを特徴とする文書生成装置。
【選択図】　図１４



【特許請求の範囲】
【請求項１】
第２の文書が差し込まれた第１の文書に係る文書データを生成する文書生成装置であって、
前記第１の文書のテンプレート情報と、前記第２の文書のテンプレート情報と、前記第２の文書におけるデータ参照に係る情報を示すクエリー情報と、を記憶する記憶手段と、
外部装置から前記クエリー情報に基づいてデータを取得する取得手段と、
前記取得手段により取得された前記データと、前記第１及び第２の文書のテンプレート情報と、に基づいて前記第１の文書に係る文書データを生成する生成手段と、を備え、
前記第１の文書のテンプレート情報には前記第２の文書の差し込み位置を示す差し込み情報が含まれ、
前記生成手段は、前記クエリー情報に基づいて前記取得手段により取得された前記データを含む、前記第２の文書が前記差し込み位置に差し込まれた、前記第１の文書に係る文書データを生成する
ことを特徴とする文書生成装置。
【請求項２】
前記生成手段は、
前記第１の文書のテンプレート情報の解析において前記差し込み情報を発見したことに応じて、前記クエリー情報に基づいて前記取得手段により前記データを取得し、取得された該データと前記第２の文書のテンプレート情報とに基づいて前記第２の文書に係る文書データを生成し、該文書データを用いて前記第１の文書に係る文書データを生成する
ことを特徴とする請求項１に記載の文書生成装置。
【請求項３】
前記生成手段は、
前記第１の文書のテンプレート情報の解析において前記差し込み情報を発見したことに応じて、当該差し込み情報を前記第２の文書のテンプレート情報で置き換えて前記生成を行う
ことを特徴とする請求項１に記載の文書生成装置。
【請求項４】
第２の文書が差し込まれた第１の文書に係る文書データを生成する文書生成装置の制御方法であって、
前記文書生成装置は、
前記第１の文書のテンプレート情報と、前記第２の文書のテンプレート情報と、前記第２の文書におけるデータ参照に係る情報を示すクエリー情報と、を記憶する記憶手段を備え、
外部装置から前記クエリー情報に基づいてデータを取得する取得工程と、
前記取得工程において取得された前記データと、前記第１及び第２の文書のテンプレート情報と、に基づいて前記第１の文書に係る文書データを生成する生成工程と、を備え、
前記第１の文書のテンプレート情報には前記第２の文書の差し込み位置を示す差し込み情報が含まれ、
前記生成工程においては、前記クエリー情報に基づいて前記取得工程により取得された前記データを含む、前記第２の文書が前記差し込み位置に差し込まれた、前記第１の文書に係る文書データが生成される
ことを特徴とする文書生成装置の制御方法。
【請求項５】
前記生成工程においては、
前記第１の文書のテンプレート情報の解析において前記差し込み情報が発見されたことに応じて、前記クエリー情報に基づいて前記取得工程により前記データが取得され、取得された該データと前記第２の文書のテンプレート情報とに基づいて前記第２の文書に係る文書データが生成され、該文書データを用いて前記第１の文書に係る文書データが生成さ

れる
ことを特徴とする請求項４に記載の文書生成装置の制御方法。
【請求項６】
前記生成工程においては、
前記第１の文書のテンプレート情報の解析において前記差し込み情報が発見されたことに応じて、当該差し込み情報を前記第２の文書のテンプレート情報で置き換えて前記生成が行われる
ことを特徴とする請求項４に記載の文書生成装置の制御方法。
【請求項７】
コンピュータを請求項１乃至３のいずれかに記載の文書生成装置として機能させるためのプログラム。
【請求項８】
請求項７に記載のプログラムを格納したコンピュータで読み取り可能な記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、文書処理装置及びその制御方法、プログラム、記憶媒体に関する。
【背景技術】
【０００２】
従来より、データベースなどから取得した表データを、メイン文書と呼ばれるデータを差し込むべき場所を指定した文書に挿入して新たな文書を作成・印刷するいわゆる差し込み印刷機能を有する構成が知られている。例えば、Ｍｉｃｒｏｓｏｆｔ（登録商標）社のワードプロセッサ・アプリケーションであるＷｏｒｄ等はこれに相当する。差し込む表データは表計算ソフトのファイルや、データベースのテーブルやデータ取得のためのクエリーによって指定される。メイン文書のデータを差し込む場所は、差し込み印刷フィールドとして指示される。この差し込み印刷フィールドは差し込みデータの表のカラムを特定できるデータを持っている。
【０００３】
差し込み印刷処理が開始されると、文書データに基づいて印刷処理を実行するためにプリンターに送られる印刷データが生成される。その場合、文書データを読み込んで差し込み印刷フィールドを検出したときに、差し込み印刷フィールドに設定されているカラム特定情報から表データのカラムを特定しデータを取得する。そして、取得したデータを差し込み印刷フィールドと置き換えて印刷データを生成する。このような差し込み印刷技術は、特許文献１に開示されている。
【０００４】
また、Apache Software Foundationがオープンソースとして提供している汎用テンプレートエンジンVelocityを用いて、データとテンプレートをマージして文書を作成する手法も知られている。VelocityにはVelocity Template Language（ＶＴＬ）という言語が存在し、テンプレートをこのＶＴＬを使用して記述すると、テンプレート外で定義されたデータをテンプレートにマージして文書を作成することができる。このVelocityには別のテキストファイルを読み込んで文書の一部とする機能や、ＶＴＬを含むテンプレートを読み込んでマージして読み込む機能がある（非特許文献１参照）。
【０００５】
また、ＸＭＬデータを取得したときに、スタイルシートを指定して文書に変換するＸＳＬＴおよびＸＳＬ－ＦＯ技術が標準化されている（非特許文献２、非特許文献３参照）。
【特許文献１】特公平３－６１５９６号公報
【非特許文献１】Velocity User Guide (http://jakarta.apache.org/velocity/docs/user-guide.html)
【非特許文献２】Extensible Stylesheet Language (XSL) Version 1.0, W3C Recommendation 15 October 2001 (http://www.w3.org/TR/2001/REC-xsl-20011015/)

【非特許文献３】XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999 (http://www.w3.org/TR/1999/REC-xslt-19991116)
【発明の開示】
【発明が解決しようとする課題】
【０００６】
データを取得し、テンプレートやメイン文書（以降テンプレートと総称）に埋め込んで文書を作成する文書処理装置において、データを取得する記述（クエリー）とテンプレートの組み合わせを指定して文書作成を行うものがある。例えば、Ｍｉｃｒｏｓｏｆｔ（登録商標）社のＷｏｒｄはこの装置と同等の機能を備えたソフトウェアに相当する。このような装置において、差し込みを行なう文書（以後元文書と呼ぶ）の一部を他の文書（以後新文書と呼ぶ）で流用する場合、次のような手順が必要になる。
（１）元文書のデータを取得するためのクエリーを新文書でデータを取得するクエリーに設定する。
（２）元文書のテンプレートで流用する箇所以外を変更して、新文書のテンプレートとする。
【０００７】
この変換作業を行う場合、次のような課題が存在する。
（ａ）新文書で新たに必要となるデータ取得の追加定義が煩雑。
従来の構成においては、文書にデータ取得の定義が一つしか指定できないものが多かった。このため新文書で新たにデータ取得が必要となる場合は、元文書のデータ取得の必要な部分を残し、新文書のデータ取得を新たに追加したデータ取得を定義する必要があった。従って、元文書のデータ取得のうち、流用する元文書の一部が使用しているデータを取得するための部分がどこであるかを調べて、それ以外を削除するという作業が必要となっていた。
（ｂ）元文書のテンプレートから必要な部分を抽出することが難しい。
文書の一部を流用しようとする場合は、元文書の差し込み結果を見て、どの部分を流用するかを決定することが多いものと考えられる。しかし、その際に、最終結果の欲しい部分がテンプレートのどの部分にあたるかを、正確に見極めることはテンプレートの構造を熟知していなければ難しい。
【０００８】
このように、従来の構成においては、文書の部品として、差し込みによって生成されるものを流通させようとすると、文書部品を組み込む際の工数がかかっていた。従って、文書の部品を再利用することが現実的に困難であった。
【０００９】
本発明は上記問題に鑑みなされたものであり、データ参照を伴う文書部品の差し込みを可能にすることで、文書部品の再利用性を高める技術を提供することを目的とする。
【課題を解決するための手段】
【００１０】
上記目的を達成するため、本発明による文書生成装置は以下の構成を備える。即ち、
第２の文書が差し込まれた第１の文書に係る文書データを生成する文書生成装置であって、
前記第１の文書のテンプレート情報と、前記第２の文書のテンプレート情報と、前記第２の文書におけるデータ参照に係る情報を示すクエリー情報と、を記憶する記憶手段と、
外部装置から前記クエリー情報に基づいてデータを取得する取得手段と、
前記取得手段により取得された前記データと、前記第１及び第２の文書のテンプレート情報と、に基づいて前記第１の文書に係る文書データを生成する生成手段と、を備え、
前記第１の文書のテンプレート情報には前記第２の文書の差し込み位置を示す差し込み情報が含まれ、
前記生成手段は、前記クエリー情報に基づいて前記取得手段により取得された前記データを含む、前記第２の文書が前記差し込み位置に差し込まれた、前記第１の文書に係る文

書データを生成する。
【００１１】
　また、本発明による文書生成装置の制御方法は以下の構成を備える。即ち、
　第２の文書が差し込まれた第１の文書に係る文書データを生成する文書生成装置の制御方法であって、
　前記文書生成装置は、
　前記第１の文書のテンプレート情報と、前記第２の文書のテンプレート情報と、前記第２の文書におけるデータ参照に係る情報を示すクエリー情報と、を記憶する記憶手段を備え、
　外部装置から前記クエリー情報に基づいてデータを取得する取得工程と、
　前記取得工程において取得された前記データと、前記第１及び第２の文書のテンプレート情報と、に基づいて前記第１の文書に係る文書データを生成する生成工程と、を備え、
　前記第１の文書のテンプレート情報には前記第２の文書の差し込み位置を示す差し込み情報が含まれ、
　前記生成工程においては、前記クエリー情報に基づいて前記取得工程により取得された前記データを含む、前記第２の文書が前記差し込み位置に差し込まれた、前記第１の文書に係る文書データが生成される。
【発明の効果】
【００１２】
　本発明によれば、データ参照を伴う文書部品の差し込みを可能にすることで、文書部品の再利用性を高める技術を提供することができる。
【発明を実施するための最良の形態】
【００１３】
　以下、添付図面を参照して本発明に係る実施の形態を詳細に説明する。ただし、この実施の形態に記載されている構成要素はあくまでも例示であり、本発明の範囲をそれらのみに限定する趣旨のものではない。
【００１４】
　＜＜実施形態１＞＞
　（ハードウェア構成）
　図１は本実施形態に係る文書生成装置のハードウェア構成を示した図である。図１において、１０１０は処理部であり、プログラムを実行する。処理部１０１０は、例えば、中央処理演算装置（ＣＰＵ：Central Processing Unit）により構成される。
【００１５】
　１０２０は一次記憶部であり、プログラムやデータを一時的に記憶する。一次記憶部１０２０は、例えば、ＲＡＭ（Random Access Memory）等により構成される。１０３０は二次記憶部であり、プログラムやデータを長期的に記憶する。二次記憶部１０３０は、例えば、ハードディスク（ＨＤ：Hard Disk）等により構成される。
【００１６】
　１０４０は入力部であり、オペレータが処理を指示するための入力を受け付ける。入力部１０４０は、例えば、キーボード、マウス、タッチパネルなどの装置により構成される。１０５０は出力部であり、入力部１０４０から入力されたコマンドや、それに対する装置の応答出力等を表示する。出力部１０５０は、例えば、ＣＲＴや液晶パネル等のディスプレイや、或いは、スピーカー等の音声出力装置により構成することができる。
【００１７】
　１０６０は印刷部であり、記録用紙（記録媒体）に所定の画像を形成して印刷を行う。印刷部１０６０は、例えば、インクジェット・プリンタやレーザ・プリンタ等により構成することができる。
【００１８】
　１０７０はネットワーク部であり、ネットワークによって他の装置と通信を行うための処理を実行する。ネットワーク部１０７０は、例えばＬＡＮや無線ＬＡＮ、ＩＳＤＮ等の

、ネットワークへのインタフェースとして外部装置とデータのやりとりを行う。１０８０はバスであり、本実施形態に係る文書生成装置における各構成要素を接続し、データの流れを司る。

【００１９】

上記のような構成は、例えば、複合機やプリンタ、パーソナルコンピュータ、ＰＤＡ等の情報機器により実現することができる。尚、以上の各装置と同等の機能を実現するソフトウェアにより、ハードウェア装置の代替として構成することもできる。

【００２０】

本実施形態では、プログラムを動作させる度に、本実施形態に係るプログラム及び関連データを、プログラム等がインストールされている二次記憶部１０３０から一次記憶部１０２０にロードして実行させる場合を想定して説明するが、これに限られない。例えば、本実施形態に係るプログラムをＲＯＭ（不図示）に記録しておき、これをメモリマップの一部をなすように構成し、処理部１０１０で直接実行するように構成することができる。或いは、不図示のメディアから本実施形態に係るプログラム及び関連データを二次記憶部１０３０にロードして実行させるように構成することもできる。

【００２１】

また、本実施形態では、説明の便宜のため、本実施形態に係る文書生成装置を１つの装置で実現した構成について述べるが、複数の装置にリソースを分散した構成によって実現してもよい。例えば、記憶や演算のリソースを複数の装置に分散した形に構成してもよい。或いは、文書生成装置上で仮想的に実現される構成要素毎にリソースを分散し、並列処理を行うようにしてもよい。

【００２２】

（ソフトウェア構成）

図２は本実施形態に係る文書生成装置上で実行されるソフトウェアの構成を示した図である。図２で示されるソフトウェアはプログラムとして二次記憶部１０３０に記憶される。本実施形態に係る文書生成装置が起動すると、処理部１０１０は、二次記憶部１０３０から一次記憶部１０２０にこのプログラムをロードし、処理部１０１０において処理実行するように制御する。

【００２３】

図２において、２０１０はデータ取得部であり、データベースやＡＳＰ（Application Service Provider）にアクセスしてデータを取得する。２０２０は文書管理部であり、二次記憶部１０３０に記憶されている処理すべき文書データを管理する。２０３０はデータマージ部であり、取得したデータをテンプレートに埋め込む処理を行なう。２０４０は表示制御部であり、生成された文書を出力部１０５０に表示するように制御する。

【００２４】

２０５０はレイアウト部であり、データマージ部２０３０によってマージされた文書を表示／印刷するためにレイアウト処理を行なう。２０６０は印刷制御部であり、生成された文書を印刷部１０６０によって印刷する。２０７０はレンダリング部であり、レイアウト部２０５０によってレイアウトされた文書を表示制御部２０４０または印刷部２０６０によって表示／印刷するのに適した形式に変換する。

【００２５】

図２に示されるソフトウェアの各機能は、図１を参照して上述した文書生成装置の処理部１０１０が一次記憶部１０２０にロードされたプログラムを実行し、図２に示される各ハードウェアと協働することによって実現される。もちろんソフトウェアの各機能の一部或いは全てが専用のハードウェアで実現されてもよい。

【００２６】

（文書ファイル）

図３は、本システムで扱う文書ファイルの形式を模式的に表した図である。図３において、３０１０は文書ファイルの内部構成を表した構成情報エントリである。後述するように、本実施形態では構成情報が所定のＸＭＬ Ｓｃｈｅｍａに適合するＸＭＬにより記述

される場合について例示的に説明するが、構成情報の形式はこれに限られない。３０２０は、クエリーやテンプレートを格納する一般エントリである。

【００２７】

なお、本実施形態においては、文書ファイルはＺＩＰ形式のアーカイブファイルとして構成される場合を例示的に想定する。ただし、構成情報エントリに対応する内容と一般エントリに対応する内容とを区別して処理を行うことが可能ならば、文書ファイルの形式はこれに限られない。

【００２８】

図４は、構成情報エントリ３０１０内に格納される、構成情報を記述するためのＸＭＬの構文を示すＸＭＬ Ｓｃｈｅｍａの一例を示した図である。図４において、４０１０には、構成情報全体を示す＜contents＞タグに関するスキーマが記述されている。４０２０には、クエリーを示す＜query＞タグに関するスキーマが記述されている。４０３０には、テンプレートを示す＜template＞タグに関するスキーマが記述されている。４０４０には、コンポーネント（部品）を示す＜component＞タグに関するスキーマが記述されている。

【００２９】

例えば、４０１１は、構成情報には０個以上のクエリーが存在することを示している。４０１２は、構成情報には１個のテンプレートが存在することを示している。４０１３は、構成情報には０個以上のコンポーネントが存在することを示している。また、例えば、４０２１、４０２２は、データ型がそれぞれ"string"（文字列）である属性"name"と"entry"が＜query＞タグに存在することを示している。ただし、属性"name"はクエリーの名前を示し、属性"entry"は一般エントリ３０２０（図３）の中のエントリを識別する情報を示す。本実施形態に係る構成においては、４０４０のように、コンポーネント（部品）毎に、０個以上のクエリーと１個のテンプレートが関連づけられている点が特徴的である。

【００３０】

図５は、図４に例示したＸＭＬ Ｓｃｈｅｍａに合致する、図３の構成情報エントリ３０１０に格納される構成情報の例を示した図である。構成情報には、０個以上のコンポーネント（０個以上のクエリーと１個のテンプレートから構成される）と、０個以上のクエリー、１個のテンプレートを格納するエントリが名前定義と共に記述されている。また、図５において、５０１０は"comp1"という名前のコンポーネント（部品）に関する情報を記述している。５０１１は一般エントリの集合３０２０のうち"entry-1"に格納された"query1"という名前のクエリーを用いることを示しており、５０１２は、"entry-2"に格納されたテンプレートを用いることを示している。

【００３１】

本実施形態に係るシステムにおいては、ＨＴＭＬにおいて差し込み内容がVelocityのＶＴＬを拡張した言語で指定されたものをテンプレートとして利用する。ただし、ＨＴＭＬは、HyperText Markup Language（HTML 4.01 Specification W3C Recommendation 24 December 1999）の略称である。

【００３２】

また、データアクセスは図６に示すようなクエリーによって定義される。図６は、データアクセスを記述するクエリーを例示した図である。本実施形態に係る文書生成装置は、クエリーに指定されたサーバにアクセスし、指定されたＳＱＬによる問い合わせを行なう。そして、クエリーに基づいて取得された値を、テーブルの各列がdataItem要素で指定された名前をもつ配列として扱う。例えば、図６の例では、６０１０は、サーバ上のテーブル"person"からフィールド"a","b","c"のデータを取得することを示している。取得したデータは６００５の＜name＞タグによって示されている"testdata"という名前の配列に格納される。この配列の列は、６０２１～６０２３に指定されているように、フィールド"a"のデータの列を"data1"、フィールド"b"のデータの列を"data2"、フィールド"c"のデータの列を"data3"という名前で扱うことを示している。なお、上述のスキーマや構成情報、クエリー等の記述形式は、説明の便宜のために例示したものであり、これに限られるわ

けではない。

【００３３】

図７は、図５の構成情報例に示した"comp1"という名前の部品における、クエリーとテンプレートを例示した図である。図７において、（ａ）はクエリーの例、（ｂ）はテンプレートの例を示している。

【００３４】

図７（ａ）のクエリーは、データベースにアクセスして商談情報を取得し、商品名、単価、数量を"name","price","quantity"というラベルのついた列を持つ"Activities"という配列にデータを格納する処理を記述している。即ち、７０１０では、"Activities"から"name","price","quantity"について参照することを示している。一方、７０２０では、それらの参照値の列をそれぞれ"name","price","quantity"という名前の配列に格納することを示している。

【００３５】

本実施形態に係る文書生成装置はこのようなクエリーを実行することで、商談情報をテンプレートで扱うことができる。図７（ｂ）のテンプレートは、図７（ａ）のクエリーにより取得された配列データを元に表を作成する。図７（ｂ）において、７０３０、７０４０、７０５０は、"Name", "Price", "Quantity"という見出しを有するテーブルを表示する記述を示している。７０６０は、配列"Activities"中のそれぞれの行を"product"に代入し、その行のそれぞれの列である"name"（７０６１）、"price"（７０６２）、"quantity"（７０６３）を取り出して表示する記述を示している。

【００３６】

このように、本実施形態に係る構成においては、コンポーネント（部品）毎に０個以上のクエリーと１個のテンプレートが関連づけられている。

【００３７】

（処理概要）

図８は、文書部品を取り込むためにＶＴＬに拡張を施した文を説明するための図である。本実施形態に係る構成においては、２つの文を用いて部品の取り込みを指定する。一つはimport_component文であり、もう一つはinclude_componet文である。図８において、図８（ａ）はimport_component文により部品を取り込む様子を模式的に示しており、図８（ｂ）はinclude_component文により部品を取り込む様子を模式的に示している。

【００３８】

import_component文は、指定した部品用に定義されたクエリーを問い合わせ、その問い合わせによって得られたデータを差し込んで部品を完成させてから取り込むものである。図８（ａ）において、８０１０はテンプレートの例、８０２０はテンプレート８０１０に記述されたimport_component文８０１１に基づいて生成されたＨＴＭＬを例示している。図８（ａ）の例では、図７に例示した"comp1"という名前のコンポーネントをimport_component文により取り込んでいる。

【００３９】

一方、include_component文は、指定した部品用に定義されたクエリーによって得られたデータだけではなく、文書用に取得されたデータも合わせて差し込んで部品を完成させて取り込むものである。図８（ｂ）において、８０３０はテンプレートの例、８０４０はテンプレート８０３０に記述されたinclude_component文８０３４に基づいて生成されたＨＴＭＬを例示している。図８（ｂ）の例においても、図７に例示した"comp1"という名前のコンポーネントをinclude_component文により取り込んでいる。ただし、８０３０では、８０３１～８０３３において、ラベル名に相当する変数"$name","$price","$quantity"に文字列に係る値"名前","単価","数量"を設定している。この結果、当該テンプレートに基づいて文書を生成すると、８０４１～８０４３のように、テンプレートにおいて設定した文字列が表示されたＨＴＭＬ８０４０が生成される。

【００４０】

（データ構造）

次に、上記のimport_component文、include_component文を実行するためのデータ管理について、図９を参照して説明する。図９は、差し込みデータを管理するためのデータ構造を示した図である。

【００４１】

図９において、９０１０は文書と部品のデータをスタックとして管理する差し込みデータスタックである。当該スタックは、例えば、一次記憶部１０２０等の記憶装置に設けることができる。９０２０は差し込みデータスタック９０１０上に格納されるスタックエントリである。スタックエントリ９０２０には、後述のハッシュテーブル９０４０を指すためのアドレス情報が格納されている。９０３０はスタックポインタであり、現在処理を行なっている部品用のデータを格納しているスタックエントリ９０２０を指すものである。

【００４２】

９０４０はスタックエントリ９０２０から指されるデータハッシュテーブルである。９０５０はデータハッシュテーブル９０４０に格納されるデータ群である。データ群９０５０には、例えば、配列や整数クラス、実数クラス等の様々な形式のデータが格納される。また、データハッシュテーブル９０４０は、データ群９０５０とそれにつけられた名前との対応関係を管理している。これにより、名前からデータ群９０５０の実体を検索することができるようになっている。なお、スタックの一番下には親文書について取得されたデータが格納される。

【００４３】

（データ差込処理）

まず、一般的なデータ差し込みの処理を説明する。ＶＴＬのデータ参照は"${name}"、または"${name.method()}"のように識別子もしくは識別子に対するメソッド呼び出しを{}で囲んだものに$をつけて表現する。ただし、"$name"のように{}を省略して簡略的に表記することもできる。ここでは簡略形を使用して説明する。差し込み処理を図１０の処理フローを用いて説明する。図１０は、差し込み処理の流れを示したフローチャートである。差し込み処理とは、テンプレートを入力とし、ＶＴＬで記述された指示に基づき差し込んだ結果を出力するものである。ここでは、差込結果をＨＴＭＬ文書として出力する場合について例示的に説明する。なお、以下の処理は図２にその構成を例示したプログラムを読み込んで、ＣＰＵ１０１０の制御に基づいて実行される。

【００４４】

まず、ステップＳ１００１０において、入力されたテンプレートからトークンを切り出し、ＶＴＬの構文であるかどうかを判断する。ただし、トークンとは処理において意味を有する文字列の単位であり、本実施形態に係る文書生成装置は、入力されたテンプレートを先頭からトークン単位で順に読み出して処理を進める。判断の結果、ＶＴＬ構文であれば（ステップＳ１００１０でＹＥＳ）ステップＳ１００２０に進む。そうでなければ（ステップＳ１００１０でＮＯ）、ステップＳ１００５０に進む。

【００４５】

ステップＳ１００２０では、ステップＳ１００１０で切り出されたトークンが、データ参照のための構文であるかを判断する。具体的には、例えば、"${}"等の記号の有無によりデータ参照のための構文であるかを否かを判断することができる。判断の結果、データ参照のための構文であれば（ステップＳ１００２０でＹＥＳ）ステップＳ１００３０へ、そうでなければ（ステップＳ１００２０でＮＯ）ステップＳ１００４０へ進む。

【００４６】

ステップＳ１００３０では、まず、データ参照を示す"${}"等の記号を切り出されたトークンから取り外す。取り外されたものからデータ識別子を取り出す。取り出したデータ識別子を用いて、スタックポインタ９０３０の指しているスタックエントリ９０２０からデータハッシュテーブル９０４０を得る。得られたデータハッシュテーブル９０４０に対して、取り出したデータ識別子をキーにしてデータ識別子に対応するインスタンスを取得する。データ参照がデータ識別子を直接参照している場合は、取得されたインスタンスの表現しているデータを文字列として出力する。データ参照がデータ識別子に対するメソッ

ド呼び出しを行っている場合は、取得されたインスタンスに対してメソッド呼び出しを行なった結果のデータを文字列として表現して出力する。なお、データ識別子をキーにしてハッシュテーブル９０４０を検索をしてもインスタンスが見つからない場合は、ヌル文字列（長さ０の文字列）を出力する。そして、ステップ１００６０に進む。

【００４７】

ステップＳ１００４０では、データ参照以外のＶＴＬ（foreach文やif-else文、set文など）の処理を行なう。import_componet文とinclude_componet文以外の処理は通常のＶＴＬの処理を実行する場合と同様であるため、説明を割愛する。これら２つの文についての処理の詳細は後述する。ステップＳ１００４０の処理が終了すると、ステップＳ１００６０に進む。

【００４８】

ステップＳ１００５０では、ステップＳ１００１０で切り出されたトークンをそのまま表示データとして出力する。ステップＳ１００５０の処理が終了すると、ステップＳ１００６０に進む。

【００４９】

ステップＳ１００６０では、テンプレートの最後まで処理したかを判断し、最後まで処理が終わっていれば（ステップＳ１００６０でＹＥＳ）差し込み処理を終了する。そうでなければ（ステップＳ１００６０でＮＯ）ステップＳ１００１０に戻り、次のトークンを読み込んで処理を継続する。

【００５０】

（import_component文処理）

ステップＳ１００４０における、import_component文の処理について、図１４を参照して説明する。図１４は、import_component文に係る処理の流れを示したフローチャートである。

【００５１】

まず、ステップＳ１４０１０において、構成情報を参照し、import_component文の引数として指定された名前を有する部品に関する情報を、構成情報から取得する。例えば、import_component文の引数が"comp1"であり、構成情報が図５のように示される場合、図５の５０１０の部分を参照する。

【００５２】

次に、ステップＳ１４０２０において、参照した構成情報から、その部品のクエリーの格納されている一般エントリの名前と、テンプレートの格納されている一般エントリの名前を取得する。上記の例では、クエリーが格納されている一般エントリの名前として"entry-1"を、テンプレートの格納されている一般エントリの名前として"entry-2"を取得することになる。

【００５３】

次に、ステップＳ１４０３０において、得られたエントリ名の一般エントリから文書管理部２０２０を使用してクエリーとテンプレートを取り出す。上記の例では、クエリーをentry-1から、テンプレートをentry-2から取り出す。

【００５４】

次に、ステップＳ１４０４０において、クエリーを用いてデータを取得し、スタックポインタ９０３０を進め、次のスタックエントリ９０２０を指すようにする。例えば、スタックポインタ９０３０が図９の「ドキュメント用データ」に対応するスタックエントリ９０２０を指している場合は、「部品Ａ用データ」に対応するスタックエントリ９０２０を指すように、スタックポインタ９０３０を更新する。

【００５５】

次に、ステップＳ１４０５０において、スタックポインタ９０３０の指しているスタックエントリ９０２０に対してデータハッシュテーブル９０４０を生成し設定する。例えば、更新後のスタックポインタ９０３０が「部品Ａ用データ」に対応するスタックエントリ９０２０を指している場合は、当該スタックエントリ９０２０に対してデータハッシュテ

ーブル９０４０を生成し設定する。そして、クエリーで取得したデータを、生成・設定したデータハッシュテーブル９０４０に、クエリーに指定された名前をキーにして格納する。

【００５６】

次に、ステップＳ１４０６０において、この状態で、部品用のテンプレートを入力として、図１０を参照して説明した差し込み処理と同様の処理を行う。そして、当該差し込み処理の文書出力を、このimport_component文処理を呼び出した差し込み処理の文書出力先に出力する。このため部品の差し込み処理が終わった出力は、そのまま親文書の差し込み結果に反映されることになる。

【００５７】

ステップＳ１４０６０の処理が終了したら、スタックポインタ９０３０を元に戻す。そして、import_component文処理を終了する。

【００５８】

（include_component文処理）

include_component文の処理は、import_component文の処理とほとんど同じであるが、以下の点がimport_component文の処理及び通常の処理と異なる。

【００５９】

（１）ステップＳ１００３０でデータ参照によって参照しているデータ識別子が取得されていない場合（データハッシュテーブル９０４０に格納されていない場合）はヌル文字列を出力していた。include_component文の差し込み処理においては、ヌル文字列に代わりデータ参照構文（例えば"$name"）そのものを出力する。このような処理を行なうために、差し込み処理を行なっていることを示すフラグを設け、include_component文の処理を実行している場合はフラグをＴＲＵＥに設定する。そして、フラグがＴＲＵＥに設定されている場合は、ヌル文字列に代えてデータ参照構文そのものを出力する処理を行うようにステップＳ１００３０の処理を制御する。

【００６０】

（２）import_component文の出力箇所は文書と同じであったが、include_component文の出力箇所は、文書とは別に新たに用意される。この別途用意された出力箇所には部品に対する差し込み結果のみが出力される。即ち、データ参照を行わずに部品のテンプレートを差し込む処理のみを実行する。この出力結果を処理されているテンプレートのinclude_component文に置き換えて、部品の出力結果の先頭からテンプレートの差し込み処理を再開する。

【００６１】

このような処理を実行することによって、include_component文の差し込み処理中に解決できなかったデータ参照構文はそのまま残る。また、テンプレートの差し込み処理をさらに続けることによって、部品ではなくそれを包含する文書（部品のテンプレート中にinclude_componentがある場合はその部品）のデータを使用して差し込み処理を行なうことができる。

【００６２】

なお、ステップＳ１００３０において、データを検索するのはスタックポインタ９０３０の指しているスタックエントリ９０３０のみを使用するように処理していた。これに対して、include_component文を処理する場合は、スタックポインタ９０３０の指しているスタックエントリ９０２０以下の全てのスタックエントリ９０２０を対象にデータを検索するようにする。これにより、親文書においても対応するデータを適切に検索して出力することができる。

【００６３】

このinclude_component文の処理の詳細について、図１５を参照して説明する。図１５は、include_component文の処理の流れを示したフローチャートである。

【００６４】

図１５において、まず、ステップＳ１５０１０において、include_component文の処理

を実行しているか否かを示すフラグをＴＲＵＥに設定する。このフラグの初期値はＦＡＬＳＥであり、ＴＲＵＥ値はinclude_component文の処理を実行していることを示し、ＦＡＬＳＥ値は実行していないことを示す。

【００６５】

次に、ステップＳ１４０１０～ステップＳ１４０５０の処理を実行する。これらの処理はimport_component文の処理の説明において参照した、図１４のフローチャートにおける処理と同様であるため、説明を省略する。ステップＳ１４０５０の処理を終了すると、ステップＳ１５０２０へ進む。

【００６６】

ステップＳ１５０２０では、ステップＳ１４０６０のようにデータ差し込み処理を実行するのではなく、取得された文書部品のテンプレートを、データ参照等は行わずに、include_component文が呼ばれた箇所にそのまま差し込む。

【００６７】

次に、ステップＳ１４０７０の処理を実行する。ここで、ステップ１４０７０の処理はimport_component文の処理の説明において参照した、図１４のフローチャートにおける処理と同様であるため、説明を省略する。

【００６８】

次に、ステップＳ１５０３０において、当該include_component文の処理を呼び出したテンプレートに係る、データ差し込み処理におけるトークンの参照位置を、ステップＳ１５０２０で差し込んだテンプレートの先頭位置に設定する。つまり、当該include_component文の処理を呼び出したテンプレートに係るデータ差し込み処理において、当該include_component文処理において差し込まれたテンプレートについて先頭から参照するように設定する。

【００６９】

次に、ステップＳ１５０４０において、当該include_component文の処理を呼び出したテンプレートに係る、データ差し込み処理を継続する。この処理は図１４のステップＳ１４０６０の処理のように部品用のテンプレートを入力として、データ差し込み処理を実行するのではなく、ステップＳ１５０２０によりテンプレートが差し込まれた親文書についてデータ差し込み処理を継続するものである。この処理においては、上述のように、ステップＳ１００３０でデータ参照によって参照しているデータ識別子が取得されていない場合、ヌル文字列に代えて"$name"等のデータ参照構文そのものを出力する。また、データ検索は、スタックポインタ９０３０の指しているスタックエントリ９０２０以下の全てのスタックエントリ９０２０を対象に実行する。

【００７０】

次に、ステップＳ１５０５０において、上述のフラグをＦＡＬＳＥに設定する。そして、include_component文の処理を終了する。

【００７１】

上記のように、本実施形態に係る構成においては、文書部品についてもテンプレート情報及びクエリー情報が定義されており、クエリー情報に基づいて取得されたデータを含む文書部品が差し込み位置に差し込まれた、文書データを生成する。このため、本実施形態に係る構成によれば、親文書側で文書部品のデータ参照構造等を知ることなく、データ参照を伴う文書部品を用いることができる。従って、データ参照を伴う文書部品の差し込みを可能であり、文書部品の再利用性が高められている。

【００７２】

また、import_component文の処理においては、文書部品に係る文書データを生成してから、親文書へ差し込むため、親文書側で文書部品のデータ参照構造等を知ることなく、データ参照を伴う文書部品を用いることができる。また、include_component文の処理においては、文書部品に係るテンプレートを親文書のテンプレートに差し込んでから、データ参照等の処理を実行する。このため、親文書側で文書部品中の変数にアクセスして、文書の出力結果に反映させることが可能である。

【００７３】
なお、本実施形態に係る装置では、部品のクエリーとテンプレートは文書ファイル内に存在するように構成されていたが、別ファイルに存在していてそれらのファイルを読み込んで処理を行なうよう構成してもよい。
【００７４】
また、本実施形態では、親文書のみが文書部品を差し込む例について説明したが、図１４のように、本実施形態に係る文書生成装置は、差し込み処理を再帰的に実行するため、文書部品が更にデータ参照を伴う文書部品を差し込むようにすることができる。従って、文書部品の設計者は、用途や目的に応じて柔軟な設計を行えることが可能であるとともに、文書部品の再利用性が高められている。
【００７５】
＜＜実施形態２＞＞
実施形態１では、テンプレート及びクエリーに基づいてＨＴＭＬドキュメントを生成する構成について説明した。実施形態２では、ＸＭＬデータをＸＳＬＴによって変換しＨＴＭＬドキュメントを生成する文書生成装置の実施形態を説明する。本実施形態に係る文書生成装置のハードウェア構成およびソフトウェアの構成は実施形態１と同様である。
【００７６】
ＸＳＬＴではＸＭＬデータを、指定されたスタイルシートに基づいて変換し、所望のＨＴＭＬ文書を取得する。本実施形態では生成したＨＴＭＬを表示するＨＴＭＬブラウザのために、以下のような処理命令（processing-instruction）を追加する。
【００７７】
＜?import-component data="http://localhost/xmldata/data１.xml" type="text/xsl" href="http://localhost/stylesheets/ss１.xsl" ?＞。
【００７８】
この処理命令は、data属性で指定されたＸＭＬデータに対して、type属性とhref属性をそのままスタイルシート指定のための処理命令であるxml-stylesheetのtype属性とhref属性として使用してＸＭＬデータの中に埋め込む。例えば、data属性で指定されたＸＭＬデータが次のようなものである場合を考える。
【００７９】
＜?xml version="１.０" encoding="shift_jis"?＞
＜document＞
＜title＞ＸＳＬＴについて＜/title＞
＜para＞本文＜/para＞
＜/document＞
このＸＭＬデータに対して、以下のようにxml-stylesheet処理命令を追加する。
【００８０】
＜?xml version="１.０" encoding="shift_jis"?＞
＜?xml-stylesheet type="text/xsl" href="http://localhost/stylesheet/ss１.xsl"?＞
＜document＞
＜title＞ＸＳＬＴについて＜/title＞
＜para＞本文＜/para＞
＜/document＞
このＸＭＬデータをＸＳＬＴで処理してその結果をＨＴＭＬ内の処理命令が記述されていた部分と置き換えて出力する。例えば、図１１に示すようなＸＭＬデータに対して図１２に示すＸＳＬＴのスタイルシートを適用する場合を考えると図１３に示すようなＨＴＭＬデータが取得できる。このデータを実施形態１に係る文書生成装置のＨＴＭＬブラウザで表示するとimport-componentの部分がcomp1.xmlで示されるＸＭＬデータをcomp1.xslによって変換されたＨＴＭＬに置き換えて表示される。
【００８１】

＜＜その他の実施形態＞＞

以上、本発明の実施形態例について詳述したが、本発明は、例えば、システム、装置、方法、プログラムもしくは記憶媒体等としての実施態様を取ることが可能である。具体的には、複数の機器から構成されるシステムに適用してもよいし、また、一つの機器からなる装置に適用してもよい。

【００８２】

尚、本発明は、前述した実施形態の機能を実現するプログラムを、システムあるいは装置に直接あるいは遠隔から供給し、そのシステムあるいは装置のコンピュータが該供給されたプログラムコードを読み出して実行することによっても達成される場合を含む。

【００８３】

従って、本発明の機能処理をコンピュータで実現するために、該コンピュータにインストールされるプログラムコード自体も本発明の技術的範囲に含まれる。つまり、本発明は、本発明の機能処理を実現するためのコンピュータプログラム自体も含む。

【００８４】

その場合、プログラムの機能を有していれば、オブジェクトコード、インタプリタにより実行されるプログラム、ＯＳに供給するスクリプトデータ等の形態であってもよい。

【００８５】

プログラムを供給するための記録媒体としては、例えば、次のものが含まれる。即ち、フロッピー（登録商標）ディスク、ハードディスク、光ディスク、光磁気ディスク、ＭＯ、ＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＣＤ－ＲＷ、磁気テープ、不揮発性のメモリカード、ＲＯＭ、ＤＶＤ（ＤＶＤ－ＲＯＭ，ＤＶＤ－Ｒ）等が含まれる。

【００８６】

その他、プログラムの供給形態としては、次のようなものも考えられる。即ち、クライアント装置のブラウザを用いてインターネットのホームページに接続し、該ホームページから本発明に係るコンピュータプログラム、或いは、圧縮され自動インストール機能を含むファイルをＨＤ等の記録媒体にダウンロードする形態も考えられる。また、本発明に係るプログラムを構成するプログラムコードを複数のファイルに分割し、それぞれのファイルを異なるホームページからダウンロードすることによっても実現可能である。つまり、本発明の機能処理をコンピュータで実現するためのプログラムファイルを複数のユーザに対してダウンロードさせるＷＷＷサーバも、本発明に含まれるものである。

【００８７】

また、次のような供給形態も考えられる。即ち、まず、本発明に係るプログラムを暗号化してＣＤ－ＲＯＭ等の記憶媒体に格納してユーザに配布する。そして、所定の条件をクリアしたユーザに対し、インターネットを介してホームページから暗号化を解く鍵情報をダウンロードさせ、その鍵情報の使用により暗号化されたプログラムを実行してコンピュータにインストールさせて本発明に係る構成を実現する。このような供給形態も可能である。

【００８８】

また、コンピュータが、読み出したプログラムを実行することによって、前述した実施形態の機能が実現される他、次のような実現形態も想定される。即ち、そのプログラムの指示に基づき、コンピュータ上で稼動しているＯＳなどが、実際の処理の一部または全部を行ない、その処理によっても前述した実施形態の機能が実現され得る。

【００８９】

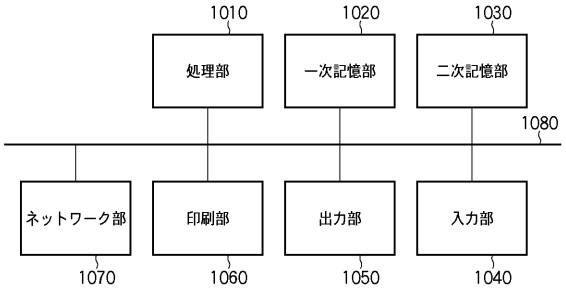
さらに、記録媒体から読み出されたプログラムが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書き込まれた後、そのプログラムの指示に基づいても前述した実施形態の機能が実現される。即ち、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部または全部を行い、その処理によっても前述した実施形態の機能が実現される。

【図面の簡単な説明】

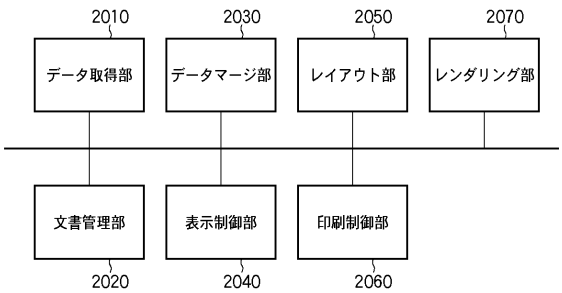
【００９０】

【図１】文書生成装置のハードウェア構成を示した図である。
【図２】文書生成装置上で実行されるソフトウェアの構成を示した図である。
【図３】文書ファイルの形式を模式的に表した図である。
【図４】構成情報を記述するためのＸＭＬの構文を示すＸＭＬ　Ｓｃｈｅｍａの一例を示した図である。
【図５】構成情報の例を示した図である。
【図６】データアクセスを記述するクエリーを例示した図である。
【図７】クエリーとテンプレートを例示した図である。
【図８】文書部品を取り込むためにＶＴＬに拡張を施した文を説明するための図である。
【図９】差し込みデータを管理するためのデータ構造を示した図である。
【図１０】差し込み処理の流れを示したフローチャートである。
【図１１】ＸＭＬデータの一例を示す図である。
【図１２】ＸＳＬＴスタイルシートの一例を示す図である。
【図１３】スタイルシートに基づく変換結果を例示する図である。
【図１４】import_component文に係る処理の流れを示したフローチャートである。
【図１５】include_component文の処理の流れを示したフローチャートである。

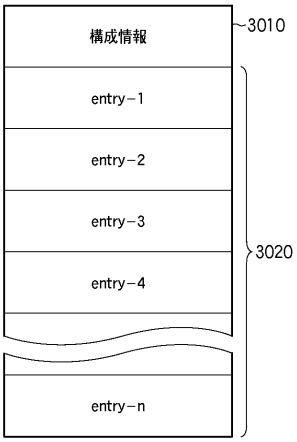
【図１】



【図２】



【図３】



【図４】

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="contents">
    <xsd:complexType>
      <xsd:sequence>                                              ～4011
        <xsd:element ref="query" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="template"/> ～4012                      }4010
        <xsd:element ref="component" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>                                             ～4013
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="query">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string"/> ～4021       }4020
      <xsd:attribute name="entry" type="xsd:string"/> ～4022
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="template">
    <xsd:complexType>
      <xsd:attribute name="entry" type="xsd:string"/>             }4030
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="component">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="query" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="template"/>                             }4040
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>

</xsd-schema>
```

【図５】

```
<?xml version="1.0" encoding="UTF-8"?>
<contents>
  <component name="comp1">
    <query name="query1" entry="entry-1"/> ～5011                 }5010
    <template entry="entry-2"/> ～5012
  </component>
  .....
  <query name="q1" entry="entry-3"/>
  <template entry="entry-n"/>
</contents>
```

【図６】

```
<query>
  <name>testdata</name>  ～6005
  <server url="" "userid=" "passwd="" "/>
  <sql>
  select a, b, c from person;  ～6010
  </sql>
  <datamap>
    <dataItem>data1</dataItem>  ～6021
    <dataItem>data2</dataItem>  ～6022
    <dataItem>data3</dataItem>  ～6023
  </datamap>
</query>
```

【図７】

```
<query>
  <name>Activities</name>
  <server url="" "userid=" "passwd="" "/>
  <sql> select name, price, quantity from Activities; </sql>  ～7010
  <dataItem>name</dataItem>          }7020
  <dataItem>price</dataItem>
  <dataItem>quantity</dataItem>
</query>
```

(a) クエリー

```
<table>
  <tr>
#if ($name)
    <td>$name</td>                   }7030
#else
    <td>Name</td>
#end
#if ($price)
    <td>$price</td>                  }7040
#else
    <td>Price</td>
#end
#if ($quantity)
    <td>$quantity</td>               }7050
#else
    <td>Quantity</td>
#end
  </tr>
#forearch ($product in $Activity)
  <tr>
    <td>$product.name</td> ～7061   }7060
    <td>$product.price</td> ～7062
    <td>$product.quantity</td> ～7063
  </tr>
#end
</table>
```

(b) テンプレート

【図８】

8010

```
<html>
<head>
</head>
<body>
        ........
#import_component("comp1") ～8011
        ........
  </body>
</html>
```

8020

```
<table>
  <tr>
    <td>Name</td>
    <td>Price</td>
    <td>Quantity</td>
  </tr>
  <tr>
    <td>LCD TV A001</td>
    <td>100</td>
    <td>12</td>
  </tr>
  <tr>
    <td>CRT TV D022</td>
    <td>35</td>
    <td>3</td>
  </tr>
.........
</table>
```

(a) import_componentの例

8030

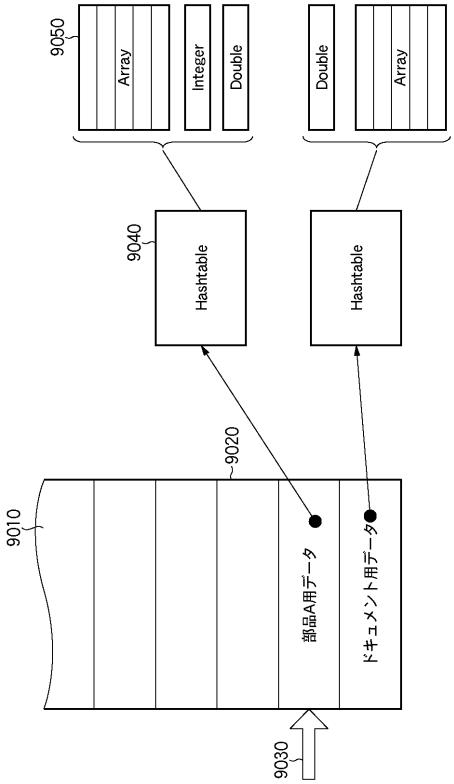
```
<html>
<head>
</head>
<body>
        ........
#set($name="名前") ～8031
#set($price="単価") ～8032
#set($quantity="数量") ～8033
#include_component("comp1") ～8034
        ........
  </body>
</html>
```

8040

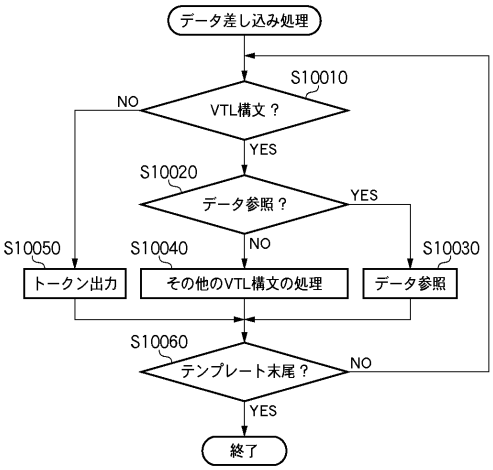
```
<table>
  <tr>
    <td>名前</td> ～8041
    <td>単価</td> ～8042
    <td>数量</td> ～8043
  </tr>
  <tr>
    <td>LCD TV A001</td>
    <td>100</td>
    <td>12</td>
  </tr>
  <tr>
    <td>CRT TV D022</td>
    <td>35</td>
    <td>3</td>
  </tr>
.........
</table>
```

(b) include_componentの例

【図９】



【図１０】



【図１１】

```
<?xml version="1.0" encoding="Shift_JIS"?>
<page>
  <employees>
    <employee>
      <employeeno>1</employeeno>
      <employeename>奥田富士夫</employeename>
      <job>President</job>
    </employee>
    <employee>
      <employeeno>50</employeeno>
      <employeename>山田太郎</employeename>
      <job>Director</job>
    </employee>
    <employee>
      <employeeno>1401</employeeno>
      <employeename>丹羽花子</employeename>
    </employee>
  </employees>
  <component>
    <data>comp1.xml</data>
    <stylesheet>comp1.xsl</stylesheet>
  </component>
</page>
```

【図１２】

```
<?xml version="1.0" encoding="Shift_JIS"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="html" encoding="Shift_JIS"/>

  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="page">
    <html>
    <body>
    <xsl:apply-templates/>
    </body>
    </html>
  </xsl:template>

  <xsl:template match="employees">
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="employees">
    <xsl:value-of select="employeename"/><br/>
  </xsl:template>

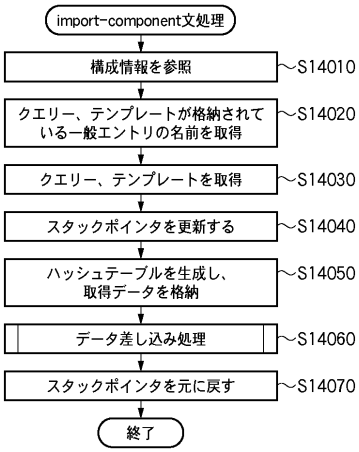
  <xsl:template match="component">
    <xsl:processing-instruction name="import-component">
    data=&quot;<xsl:value-of select="data"/>&quot;
    type=&quot;text/xsl&quot;
    href=&quot;<xsl:value-of select="stylesheet"/>&quot;
    <xsl:processing-instruction>
  </xsl:template>

</xsl:stylesheet>
```

【図１３】

```
<html>
<body>
奥田富士夫<br/>
山田太郎<br/>
丹羽花子<br/>
<?import-component data="comp1.xml" type="text/xsl" href="comp1.xsl"?>
</body>
</html>
```

【図１４】



【図１５】

