(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-523013
(P2004-523013A)
(43)公表日　平成16年7月29日(2004.7.29)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 15/00	G06F 15/00	310S	5B085
G06F 3/00	G06F 3/00	653A	5E501

審査請求 有　予備審査請求 有　（全 241 頁）

(21)出願番号	特願2001-545284(P2001-545284)
(86)(22)出願日	平成12年12月15日(2000.12.15)
(85)翻訳文提出日	平成14年6月14日(2002.6.14)
(86)国際出願番号	PCT/US2000/034013
(87)国際公開番号	WO2001/045069
(87)国際公開日	平成13年6月21日(2001.6.21)
(31)優先権主張番号	60/172,134
(32)優先日	平成11年12月17日(1999.12.17)
(33)優先権主張国	米国(US)
(31)優先権主張番号	09/728,073
(32)優先日	平成12年12月4日(2000.12.4)
(33)優先権主張国	米国(US)
(81)指定国	EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),AU,BR,CA,JP,MX

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】Ｗｅｂベースの説明

(57)【要約】
【課題】コンピュータ・システムのＷｅｂベース環境で実行される、アプリケーションの実施の仕方をユーザに教示する方法を提供すること。
【解決手段】この方法は、所定のアプリケーションを提供すること、および所定のアプリケーションを説明する１つまたは複数の注釈を含む注釈ページを提示することを含む。各注釈は、アプリケーションのキーワード・リンク、注釈リンク、および実施の詳細を含む。この方法は、ユーザに注釈内のリンクを選択することを許可する。ユーザがキーワード・リンクを選択した場合、そのキーワードに関連付けられた参考文書が提示される。ユーザが注釈リンクを選択した場合、所定アプリケーションの別のソース・ファイルを説明する別の注釈が提示される。

【特許請求の範囲】
【請求項１】
コンピュータ・システムにおいてＷｅｂベース環境で実行される、ユーザがアプリケーションを実施することを学習するために役立つ方法であって、
複数の所定のアプリケーションを提供するステップと、
所定のアプリケーションのソース・ファイルを説明し、キーワード・リンク、注釈リンク、および前記アプリケーションの実施の詳細を有する１つまたは複数の注釈を含む注釈ページを提示するステップと、
前記ユーザが注釈内のリンクを選択することを許可するステップと、
前記ユーザがキーワード・リンクを選択した場合、キーワードに関連付けられた参考文書を提示するステップと、
前記ユーザが注釈リンクを選択した場合、所定のアプリケーションの別のソース・ファイルを説明する別の注釈を提示するステップと
を含む、方法。
【請求項２】
所定のアプリケーションを実行するステップと、
前記所定のアプリケーションの実行と連携して実行された該アプリケーションを説明する１つまたは複数の注釈を提示するステップと
をさらに含む、請求項１に記載の方法。
【請求項３】
前記所定のアプリケーションが前記ユーザからの入力を受け取るステップを含む、請求項２に記載の方法。
【請求項４】
前記ユーザからの入力に基づいて、前記所定のアプリケーションの実行と連携して別の注釈ページを提示するステップをさらに含む、請求項３に記載の方法。
【請求項５】
前記別の注釈ページを提示するステップが、
詳細が表示されるべきプログラム・ユニットのアプリケーション、ファイル、クラスおよび関数の名称を含む注釈要求モジュールを自動的かつ同時に呼び出すステップと、
要求を注釈にマップするステップと、
別の注釈ページを表示するようにＷｅｂベース環境のブラウザ・ウィンドウに通知するステップと
を含む、請求項４に記載の方法。
【請求項６】
前記所定のアプリケーションの実行と連携して別の注釈ページを提示する、請求項３に記載の方法。
【請求項７】
注釈ページを含むＷｅｂページ内の構造化されたリンクを構文解析することによって、注釈へのリンクを含むグローバル目次（ａ ｇｌｏｂａｌ ｔａｂｌｅ ｏｆ ｃｏｎｔｅｎｔｓ）を自動的に生成するステップをさらに含む、請求項６に記載の方法。
【請求項８】
現行の注釈ページに対応するリンクを強調表示することによって、前記グローバル目次内のリンクが提示された注釈と同期する、請求項７に記載の方法。
【請求項９】
前記グローバル目次が第１のブラウザ・ウィンドウの第１のフレームに提示され、前記注釈ページが前記第１のブラウザ・ウィンドウの第２のフレームに提示され、前記所定のアプリケーションが第２のブラウザ・ウィンドウで実行される、請求項８に記載の方法。
【請求項１０】
前記所定のアプリケーションが、Ｊａｖａ（Ｒ）アプレットまたはアプリケーションを起動するステップを含む、請求項２に記載の方法。

【請求項１１】
前記Ｊａｖａ（Ｒ）アプレットまたはアプリケーションを起動するステップが、前記注釈ページを表示するようＷｅｂブラウザに要求するために、Ｊａｖａ（Ｒ）アプリケーション・プログラミング・インターフェースを呼び出すステップを含む、請求項１０に記載の方法。
【請求項１２】
前記所定のアプリケーションを実行するステップが、Ｊａｖａ（Ｒ）アプレットを含むハイパーテキスト・マークアップ言語のページをダウンロードするステップを含む、請求項２に記載の方法。
【請求項１３】
前記所定のアプリケーションを実行するステップが、共通ゲートウェイ・インターフェース要求を、Ｗｅｂベース環境のウィンドウ内でアプリケーションを起動するＷｅｂサーバに送信するステップを含む、請求項２に記載の方法。
【請求項１４】
前記アプリケーションが、１つまたは複数の注釈を表示するようＷｅｂブラウザに要求するために、ＪａｖａＳｃｒｉｐｔを含むハイパーテキスト・マークアップ言語のページを戻す、請求項１３に記載の方法。
【請求項１５】
前記注釈ページが第１のブラウザ・ウィンドウに提示され、前記所定のアプリケーションが第２のブラウザ・ウィンドウで実行される、請求項２に記載の方法。
【請求項１６】
アプリケーションの実施の詳細が、該アプリケーションを説明するテキスト、該アプリケーションからのソース・コードのフラグメント、またはこれらの両方を含む、請求項１に記載の方法。
【請求項１７】
ソース・コードのフラグメントが、提示された前記アプリケーションのソース・コード・ファイルから直接的にインポートされる、請求項１６に記載の方法。
【請求項１８】
自動的に、所定のアプリケーションのソース・コード・ファイルを説明する注釈ページを生成するステップをさらに含む、請求項１に記載の方法。
【請求項１９】
前記注釈ページを生成するステップが、
説明でマークアップされたテキストを既に埋め込んであるソース・コード・ファイルを受け取るステップと、
前記所定のアプリケーションの構造を決定するためにソース・コードを構文解析するステップと、
前記所定のアプリケーションの構造および説明に基づいて、１つまたは複数の注釈を生成するステップと
を含む、請求項１８に記載の方法。
【請求項２０】
前記注釈ページを生成するステップが、
前記所定のアプリケーションの注釈をナビゲートする、１つまたは複数の注釈リンクを生成するステップと、
埋め込まれた情報に基づいて、アプリケーションの実施の詳細を生成するステップと、
参考文書に対する１つまたは複数のキーワード・リンクを生成するステップと
を含む、請求項１９に記載の方法。
【請求項２１】
前記注釈ページを生成するステップが、該注釈ページ内のキーワード・リンクおよび注釈リンクを強調表示するステップを含む、請求項２０に記載の方法。
【請求項２２】

更新されたソース・コード・ファイルを受け取ったときに、前記所定のアプリケーションのソース・コード・ファイルを説明する前記注釈ページを自動的に更新するステップをさらに含む、請求項１９に記載の方法。
【請求項２３】
注釈リンクのための前記複数の注釈を構文解析することによって、グローバル目次を自動的に生成するステップをさらに含む、請求項１に記載の方法。
【請求項２４】
注釈へのリンクを含む前記グローバル目次を提供するステップをさらに含む、請求項２３に記載の方法。
【請求項２５】
アプリケーションに関する注釈ページを有するＷｅｂページへのリンクを含むローカル目次を生成するステップをさらに含む、請求項２３に記載の方法。
【請求項２６】
前記グローバル目次のローカル・リンクが選択されるとき、前記ローカル目次を提供するステップをさらに含む、請求項２５に記載の方法。
【請求項２７】
提示された前記注釈ページが実行された前記アプリケーションを説明し、前記注釈ページは前記所定のアプリケーションの実行と連携して提示される、請求項１に記載の方法。
【請求項２８】
前記アプリケーションのソース・コードを含むが前記注釈からのテキストは含まない、注釈マークアップを除去したソース・コード・ファイルを生成するステップと、
注釈マークアップを除去した前記ソース・コード・ファイルを提示するステップと、
注釈マークアップを除去した前記ソース・コード・ファイルを前記ユーザが編集することを許可するステップと
をさらに含む、請求項１に記載の方法。
【請求項２９】
コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションの実施を教示する方法であって、
複数の所定のアプリケーションを提供するステップと、
所定のアプリケーションを実行するステップと、
実行されたアプリケーションを説明し、アプリケーションの実施の詳細と、注釈および参考文書へのリンクとを含む注釈ページを前記所定のアプリケーションの実行と連携して提示するステップと
を含む、方法。
【請求項３０】
コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションの実施を教示する方法であって、
前記環境内の内容に基づいて、前記環境内の内容への複数のリンクを含むグローバル目次を自動的にアセンブルするステップと、
前記グローバル目次を提供するステップと、
ローカル・トピック内に前記ユーザを方向付ける内容へのリンクを含むローカル目次を生成するステップと、
ローカル・トピックにアクセスするために前記ユーザが前記ローカル目次からリンクを選択することを許可するステップと
を含む、方法。
【請求項３１】
コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションの実施を教示する方法であって、
複数の事前定義された対話型例示を提供するステップと、
ユーザの選択に応答して、１つまたは複数の前記事前定義された対話型例示を実行するス

テップと、
実行された前記対話型例示を説明する１つまたは複数の注釈を前記事前定義された対話型例示の実行と連携して提示するステップと、
実行された前記対話型例示、前記注釈、またはその両方の異なる態様を
前記ユーザが選択的に探索することを許可するステップと
を含む、方法。
【請求項３２】
ユーザにアプリケーションの実施を教示するＷｅｂベースのコンピュータ・システムであって、
前記Ｗｅｂベースのコンピュータ・システムのユーザによって、選択的に実行可能な１つまたは複数の事前定義された対話型アプリケーションと、
事前定義された対話型アプリケーションを説明する１つまたは複数の注釈を含む注釈ページとを含み、前記注釈ページが、
１つまたは複数のリンクと、
前記アプリケーションの実施の詳細とを含み、
事前定義された対話型アプリケーションの選択的実行に応答して、前記注釈ページに異なる注釈が自動的に提供される、
システム。
【請求項３３】
前記ユーザが、事前定義された対話型アプリケーションに関連付けられたソース・コードにアクセスすることができるユーティリティをさらに含む、請求項３２に記載のシステム。
【請求項３４】
前記ユーティリティが、事前定義された対話型アプリケーションのソース・コードを前記ユーザに表示またはコピーすることを可能にする、請求項３３に記載のシステム。
【請求項３５】
前記アプリケーションの実施の詳細が、前記アプリケーションを説明するテキスト、前記アプリケーションに関連付けられたソース・コードのフラグメント、またはこれらの両方を含む、請求項３２に記載のシステム。
【請求項３６】
リンクが、前記ユーザに参考文書の本文へのアクセスを提供するキーワード・リンク、または前記ユーザに別の注釈ページへのアクセスを提供する注釈リンクを含む、請求項３２に記載のシステム。
【請求項３７】
前記注釈を表示する内容フレームと、
ナビゲーション・バーを表示するフレームワーク・アプレットと、
リンクの目次階層を表示する目次フレームと
を有するフレームワークを含むＷｅｂブラウザ・ウィンドウをさらに含む、請求項３２に記載のシステム。
【請求項３８】
前記フレームワーク・アプレットがＪａｖａ（Ｒ）アプレットを含む、請求項３７に記載のシステム。
【請求項３９】
ＪａｖａＳｃｒｉｐｔが、
前記Ｗｅｂブラウザ・ウィンドウにフレームワークが提示されているかどうかを自動的に判断し、前記フレームワークが提示されている場合、前記フレームワーク・アプレットに前記フレームワーク内の内容に関して通知する、請求項３７に記載のシステム。
【請求項４０】
前記目次が、前記フレームワーク内の内容に基づいて前記階層内のリンクを自動的に強調表示する、請求項３９に記載のシステム。

【請求項４１】
前記ユーザが、前記目次階層のリンクを選択することによって、注釈ページにアクセスする、請求項４０に記載のシステム。
【請求項４２】
前記ユーザが、前記ナビゲーション・バーと対話することによって、注釈ページにアクセスする、請求項４０に記載のシステム。
【請求項４３】
前記目次が、内容フレームに表示された注釈ページに基づいて前記階層を強調表示する、請求項４０に記載のシステム。
【請求項４４】
前記目次が除去可能またはサイズ変更可能である、請求項３７に記載のシステム。
【請求項４５】
ユーザにアプリケーションの実施を教示するためのＷｅｂベースのコンピュータ・システムであって、
内容フレーム、フレームワーク・アプレット、および前記内容フレーム内の内容に関するリンクのグローバル目次階層を表示する目次フレームを含むＷｅｂブラウザ・ウィンドウと、
事前定義された対話型アプリケーションを説明し、別の内容へのリンクを含み、前記内容フレームに表示される１つまたは複数の注釈と、
前記表示された注釈のローカル内容に関するリンクのローカル目次階層を表示する目次のウィンドウと
を含む、システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本願は、Ｗｅｂベースの文書化および説明に関する。
【０００２】
【従来の技術】
本願は、米国仮特許第６０／１７２，１３４号明細書および米国特許第０８／８８８，９２５号明細書の特典を請求するものであり、これらの開示の全体を参照により本明細書に組み込む。
【０００３】
図１に示すような典型的なコンピュータ・システムは、中央演算処理装置１０５と、入出力装置１１０と、例えばオペレーティング・システム１２０および１つまたは複数のアプリケーション・プログラム１２５などの、コンピュータ１００によって使用される様々なプログラムを含んでいるメモリ１１５とを有するコンピュータ１００を含む。コンピュータ・システムのエンド・ユーザは、情報を入出力装置１１０を介してコンピュータ１００に転送する様々な入力装置（キーボード１３０、マウス１３５）によってコンピュータ１００と対話する。コンピュータ１００は、この入力されたデータに対して、特に、例えば表示モニター１４０の画面上に適切なテキストおよび画像を表示することによるなどのエンド・ユーザへの応答出力を提供する方法によって応答する。
【０００４】
オペレーティング・システム１２０は、グラフィカル・ユーザ・インターフェース（ＧＵＩ）を含むことができる。このＧＵＩによって、オペレーティング・システムおよび実行中の可能性を有するいかなるアプリケーション（例えば、ワードプロセッシング・プログラム）でも、そのコンピュータ・システムのユーザと対話することができる。一般に使用されるＧＵＩ実施態様は、モニター画面を仮想デスクトップに見立てたデスクトップ・メタフォーを使用する。デスクトップは、基本的に、１つまたは複数の表示領域を含む、様々なグラフィック・オブジェクトをサポートする２次元作業テンプレート領域である。図２に示すように、アプリケーション・プログラムまたはオペレーティング・システムによ

って生成された情報は、表示領域２０５（例えば、ウィンドウ、ダイアログ・ボックス、ポップアップ・メニュー、プルダウン・メニュー、ドロップダウン・リスト、アイコン）の範囲内でデスクトップ２００上に表示することができる。ユーザは、表示領域内で適切にカーソル２１０を操作することによって、また、キーボードまたは他の入力装置を用いて情報を入力することによって、オペレーティング・システム、および実行中の可能性のあるいかなるアプリケーションとでも対話することができる。

【０００５】

コンピュータ１００は、通信リンク１５５（例えば、電話回線）を介してネットワーク１５０とデータを交換するためのある種の通信カードまたは装置１４５（例えば、モデムまたはネットワーク・アダプタ）も含む。ネットワーク１５０は、例えば、ローカル・エリア・ネットワーク（ＬＡＮ）、イントラネット、またはインターネットであってよい。サービス・プロバイダはネットワークへのアクセスを提供するが、さらに、ネットワークに関連付けられた様々なユーティリティまたはサービス（電子メールなど）を提供することもできる。サービス・プロバイダの例には、ＡＴ＆Ｔ ＷｏｒｌｄＮｅｔなどのインターネット・サービス・プロバイダ（ＩＳＰ）またはＡｍｅｒｉｃａ ＯｎｌｉｎｅおよびＣｏｍｐｕＳｅｒｖｅなどのオンライン・サービス・プロバイダ（ＯＳＰ）が含まれる。

【０００６】

開発者は、アプリケーション・プログラムを実施するために、プログラミングの概念を知る必要がある。したがって、アプリケーションの実施態様の説明（アプリケーションの操作の説明だけでなく）が役立つだろう。

【０００７】

多くのコンピュータ・アプリケーションは、アプリケーションの使用を支援するオンライン・ヘルプ／文書化機能を提供する。図３に示すような典型的なオンライン・ヘルプ・システムはＧＵＩによってアクセスされるが、ここでは、ヘルプ・ウィンドウ３００内でユーザに対してテキスト情報およびグラフィック情報の画面が表示される。次いでユーザは、アプリケーションとその様々な機能のより良い理解を得るためにヘルプ・テキストの画面を読むことができる。

【０００８】

ユーザは、キー・シーケンスによって（例えば、キーボードのＦ１キーを押し下げること）、または適切なグラフィック・アイコンまたはメニュー項目上でマウスをクリックすることによって、ヘルプ・システムを呼び出す。それに応答して、ヘルプ・システムは、ユーザが要望通りにアクセスし表示することができる使用可能なヘルプ・トピックスおよびサブトピックスを列挙する目次３０５を表示することができる。ユーザは、目次３０５全体を参照し、関心のあるヘルプ・トピックをクリックして、対応する情報の本文をヘルプ・ウィンドウの中に表示させることができる。図３に示すヘルプ・ウィンドウ３００で、ユーザは、対応するヘルプ画面３１５を図４に示すウィンドウ３００に表示させるように既に「Ｐｒｏｇｒａｍｍｉｎｇ ｗｉｔｈ Ｍｉｃｒｏｓｏｆｔ Ｗｏｒｄ」を選択済みである。

【０００９】

図４に示す「Ｐｒｏｇｒａｍｍｉｎｇ ｗｉｔｈ Ｍｉｃｒｏｓｏｆｔ Ｗｏｒｄ」トピック３１０は、別個の「リンク」によって示される複数のサブトピック３２０を含む。ユーザがそれらのリンクの１つ、例えば「Ｅｒｒｏｒ Ｍｅｓｓａｇｅｓ」リンク３２５の上でマウスをクリックすると、図５に示すように、対応するヘルプ・トピックのテキストがヘルプ・ウィンドウ３００に自動的に表示される。この実施例では、「Ｅｒｒｏｒ Ｍｅｓｓａｇｅｓ」トピック３３０は、特定種類のエラー・メッセージに関するさらなるサブトピックへの複数のリンクを含む。図６に示すように、ユーザがそれらリンクのうちの１つ、例えば「Ｏｕｔ ｏｆ ｍｅｍｏｒｙ（ｓｔａｃｋ ｓｐａｃｅ）」リンク３３５をクリックすると、その選択されたトピックに関する対応するヘルプ情報（「Ｆｒｅｅｉｎｇ ｕｐ ｍｅｍｏｒｙ」）を表示するために新しいヘルプ・ウィンドウ３４０が作成される。ウィンドウ３４０に表示されたヘルプ情報は、別のサブトピック「ａｃｔｉｖｅ ｗｉ

ｎｄｏｗ」へのさらに別のリンク３４５を含むが、これは、ユーザがそれをクリックすると、対応するヘルプ・テキストをポップアップ・ダイアログ・ボックス３５０内に表示させる。ほぼすべてのレベルのそのようなネストされたヘルプ表示が可能である。ヘルプ情報を表示するために使用される表示領域（ウィンドウ、ダイアログ・ボックスなど）の量および種類は、主としてヘルプ・システム開発者の好みに基づく設計上の選択の問題である。

【００１０】

ヘルプ・システムは「文脈に依存した」ヘルプ情報を提供することができる。すなわち、そのヘルプ・システムが、すべての入手可能なヘルプ・トピックスを単純に表示して、適切なヘルプ・トピックをユーザが手動で識別して読み出すように強いるのではなく、アプリケーションの現行タスクに特に関係したヘルプ情報を自動的に表示するということである。文脈に依存したヘルプ・システムは、アプリケーションの現行状況（例えば、ユーザによって呼び出された特定の関数）および現在のカーソル位置などの要因に基づいて、どのヘルプ情報を表示するかを決定する。

【００１１】

大部分のオンライン・ヘルプ・システムによって提供される情報は、アプリケーションの使用機能の機構に関する。図７では、例えば、選択されたヘルプ・トピック４０５「Ｃａｎｃｅｌ ｐｒｉｎｔｉｎｇ」に対応するテキスト４００は、アプリケーション４１０（Ｍｉｃｒｏｓｏｆｔ（Ｒ） Ｗｏｒｄ）によって提供される印刷機能をどのように制御するかを説明している。

【００１２】

ヘルプ・システムは、所望の目的を達成するためにアプリケーションをどのように活用するかに関する価値ある情報を提供することもできる。図８では、例えば、オンライン・ヘルプ・システムは、ＷｏｒｄＢａｓｉｃプログラミング言語用の参考資料５００と、ＷｏｒｄＢａｓｉｃを使用して有用なプログラムを作成するための方法に関する実践的な説明５０５の二種類の価値ある情報を提供する。参考資料５００は、ＡｄｄＡｄｄｌｎステートメントなどの様々なＷｏｒｄＢａｓｉｃステートメントの構文規則および意味を説明するテキスト注釈を含むが、それに関するヘルプ・テキストを図９に示す。実践的な説明５０５は、ＷｏｒｄＢａｓｉｃプログラミング言語をより良く理解するためにユーザが学ぶことのできるプログラム・コードの静的な例を含むことができる。図１０に、ＧｅｔＣｕｒＶａｌｕｅｓ ＷｏｒｄＢａｓｉｃステートメントを使用するプログラム・コードの一例を示す。

【００１３】

【発明が解決しようとする課題】

オンライン・ヘルプ・システムは、通常、テクニカル・ライターによって既に作成済みのヘルプ情報を含んでいる複数の異なるヘルプ・ソース・ファイルをコンパイルすることによって「構築」される（すなわち、ランタイム操作を容易にする形式で処理される）。一般に、これらのヘルプ・ソース・ファイルは、ヘルプ・システムが対応するアプリケーションとは別の情報の別個の本文として保存される。したがって、アプリケーション開発者がアプリケーションの機能性を変更または更新するとき、テクニカル・ライターは、オンライン・ヘルプ・システムがそのアプリケーションの動作を正確に説明することを保障するために、ヘルプ・ソース・ファイルに対して対応する変更または更新を行う必要がある。しかし、一般に、オンライン・ヘルプ・システムは、アプリケーションの実施を説明できていない。

【００１４】

ヘルプ・システムは「ブラウザ」を使用してネットワーク環境で実施することができ、このブラウザは、ユーザがネットワーク環境で保管されている電子内容にアクセスし、これを表示させることを可能にする。ブラウザは、通常、ハイパーテキスト・マークアップ言語（ＨＴＭＬ）で記述された文書を表示するために使用され、インターネットなどのネットワークに接続されたサーバ上に保管される。図１１は、典型的なＨＴＭＬ文書またはＷ

ｅｂページ６０５を表示するブラウザ・アプリケーション６００（この場合はインターネット・エクスプローラ）の画面の一場面である。ユーザは、所望の文書が常駐するネットワーク・アドレス６１０、すなわちユニフォーム・リソース・ロケータ（ＵＲＬ）を指定することによってＷｅｂページ６０５にアクセスするようブラウザ６００に指示する。それに応答して、ブラウザ６００は、要求されたＷｅｂページをホストする対応するサーバに接触し、そのＷｅｂページを構成する１つまたは複数のファイルを取り出し、次いでコンピュータのディスプレイ１４０にそのＷｅｂページを表示する。

【００１５】

１つのＷｅｂページは、潜在的にデータ・タイプの異なる複数の異なるファイル（例えば、テキスト６１５、イメージ６２０、仮想世界、音声、またはムービー）から構成されることができる。さらに、Ｗｅｂページは、ネットワーク上で使用可能な別の資源（例えば、Ｗｅｂページ、個別ファイル、またはダウンロード可能なファイル）へのリンク６２５またはポインタを含むことができる。各リンクは、ネットワーク上の位置を指す、関連付けられたＵＲＬを有している。表示されたリンクをユーザがクリックするか、または別の手段で選択すると、ブラウザは、そのリンクの関連付けられたＵＲＬに対応するＷｅｂページまたは他の資源を取り出し、それをユーザに対して表示または実行する。

【００１６】

【課題を解決するための手段】

図１２を参照すると、Ｗｅｂページ６０５は、内容７００の他に、そのＷｅｂページに関連付けられたすべてのリンクを通してユーザがナビゲートするのに役だつサイト・ガイド７０５を提供することができる。サイト・ガイド７０５は目次に類似したもので、通常はツリー構造に似ている。同様に、Ｗｅｂページ６０５は、そのＷｅｂページに関連付けられたリンク内にある特定のキーワードをユーザが検索することを可能にする検索機能７１０を含むことができる。このＷｅｂページは、すべての内容およびリンクにそこからアクセスすることができる主Ｗｅｂページにユーザを引き戻す「ホーム」リンク７１５を提供することができる。このＷｅｂページは、アクセスされたときに別のＷｅｂページまたはコンピュータからユーザのコンピュータにファイルを送信するダウンロード・リンク７２０を提供することができる。

【００１７】

【発明の実施の形態】

本発明の一態様によれば、コンピュータ・システムのＷｅｂベース環境で実行される方法は、ユーザにアプリケーションの実施の仕方を教示する。この方法は、所定のアプリケーションを提供することと、所定のアプリケーションのソース・ファイルを説明する１つまたは複数の注釈を含む注釈ページを提示することを含む。各注釈は、そのアプリケーションのキーワード・リンク、注釈リンク、および実施の詳細を含む。方法は、ユーザが注釈内のリンクを選択できるようにする。ユーザがキーワード・リンクを選択した場合、そのキーワードに関連付けられた参考文書が提示される。ユーザが注釈リンクを選択した場合、所定アプリケーションの別のソース・ファイルを説明する別の注釈が提示される。

【００１８】

実施形態は、１つまたは複数の以下の機能を含むことができる。例えば、所定アプリケーションを実行することができ、実行されたアプリケーションを説明する１つまたは複数の注釈を所定アプリケーションの実行と連携して提示することができる。所定アプリケーションを実行することは、ユーザからの入力を受け取ることを含むことができる。ユーザからの入力に基づいて、所定アプリケーションの実行と連携して別の注釈ページを提示することができる。

【００１９】

別の注釈ページを提示することは、詳細が表示されるべきプログラム・ユニットのアプリケーション、ファイル、クラスおよび関数の名称を含む注釈要求モジュールを自動的かつ同時に呼び出すことを含むことができる。別の注釈ページを提示することは、その要求を注釈にマッピングすること、Ｗｅｂベース環境のブラウザ・ウィンドウに別の注釈ページ

を表示するよう伝達することも含む。

【００２０】

所定アプリケーションの実行と連携して別の注釈ページを提示することができる。注釈ページを含むＷｅｂページの構造化リンクを構文解析することによって、注釈へのリンクを含むグローバル目次が自動的に生成される。グローバル目次内のリンクの生成は、現行の注釈ページに対応するリンクを強調表示することによって注釈の提示と同期することができる。グローバル目次は第１のブラウザ・ウィンドウの第１のフレーム内に提示することができ、注釈ページは第１のブラウザ・ウィンドウの第２のフレーム内に提示することができ、所定アプリケーションは第２のブラウザ・ウィンドウ内で実行することができる。

【００２１】

所定アプリケーションを実行することは、Ｊａｖａ（Ｒ）アプレットまたはアプリケーションを起動することを含むことができ、これは、Ｗｅｂブラウザに注釈ページを表示するよう要求するためにＪａｖａ（Ｒ）アプリケーション・プログラム・インターフェースを呼び出すことを含むことができる。所定アプリケーションを実行することは、Ｊａｖａ（Ｒ）アプレットを含んでいるハイパーテキスト・マークアップ言語のページをダウンロードすることを含むことができる。

【００２２】

所定アプリケーションを実行することは、共通ゲートウェイ・インターフェース要求をＷｅｂサーバに送信することを含むことができる。このＷｅｂサーバは、Ｗｅｂベース環境のウィンドウでアプリケーションを起動する。アプリケーションは、Ｗｅｂブラウザに１つまたは複数の注釈を表示するよう要求するＪａｖａＳｃｒｉｐｔを含むハイパーテキスト・マークアップ言語のページを戻すことができる。

【００２３】

注釈ページを第１のブラウザ・ウィンドウに提示することができ、所定アプリケーションを第２のブラウザ・ウィンドウで実行することができる。アプリケーションの実施の詳細は、そのアプリケーションを記述しているテキスト、そのアプリケーションからのソース・コードのフラグメント、またはこれらの両方を含むことができる。ソース・コードのフラグメントは、提示されたアプリケーションのソース・コード・ファイルから直接的にインポートされることができる。

【００２４】

注釈ページを自動的に生成することができる。この生成には、説明がマークアップされたテキストを既に埋め込んであるソース・コード・ファイルを受け取ることが含まれる。さらに、所定アプリケーションの構造を決定するためにソース・コードを構文解析することができ、所定アプリケーションの構造および説明に基づいて１つまたは複数の注釈を生成することができる。注釈ページの生成は、所定アプリケーションの注釈をナビゲートするための１つまたは複数の注釈リンクを生成することを含むことができる。さらに、埋め込まれた情報に基づいてアプリケーションの実施の詳細を生成することができ、参考文書に対して１つまたは複数のキーワード・リンクを生成することができる。注釈ページを生成することは、注釈ページのキーワード・リンクおよび注釈リンクを強調表示することを含むことができる。更新されたソース・コード・ファイルを受け取ったときに、注釈ページを自動的に更新することができる。

【００２５】

注釈リンクに対する１つまたは複数の注釈を構文解析することによって、グローバル目次を自動的に生成することができる。このグローバル目次は、提供されることができ、注釈へのリンクを含むことができる。別法として、グローバル目次は、生成されることができ、アプリケーションに対する注釈ページを含めてＷｅｂページへのリンクを含むことができる。グローバル目次内のローカル・リンクが選択されたときは、ローカル目次を提供することができる。

【００２６】

提示された注釈ページは実行されるアプリケーションを記述することができ、その注釈ペ

ージは、所定アプリケーションの実行と連携して提示されることができる。
【００２７】
注釈マークアップを除去し、アプリケーションのソース・コードは含むが注釈からのテキストは含まないソース・コード・ファイルを生成することができる。除去済みのソース・コード・ファイルを提示することができ、その除去済みのソース・コード・ファイルをユーザが編集することを可能にすることができる。
【００２８】
本発明の別の態様によれば、コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションを実施することを教示する方法は、所定の複数アプリケーションを提供することを含む。所定アプリケーションが実行され、その実行されたアプリケーションを記述する注釈ページを、所定アプリケーションの実行と連携して提示することができる。注釈ページは、アプリケーションの実施の詳細と、注釈および参考文書へのリンクを含む。
【００２９】
本発明の別の態様によれば、コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションを実施することを教示する方法は、その環境内における内容に基づいてグローバル目次を自動的にアセンブルし、提供することを含む。グローバル目次は、環境内の内容への複数のリンクを含む。ローカル・トピック内でユーザを方向付ける内容へのリンクを含むローカル目次が生成される。ユーザは、ローカル・トピックスにアクセスするために、ローカル目次からリンクを選択することが許可される。
【００３０】
本発明のさらなる態様によれば、コンピュータ・システムのＷｅｂベース環境で実行される、ユーザにアプリケーションを実施することを教示する方法は、複数の事前定義された対話型の例示を提供することを含む。１つまたは複数の事前定義された対話型の例示は、ユーザ選択に応答して実行され、その実行されている対話型の例示を記述する１つまたは複数の注釈は、事前定義された対話型の例示の実施と連携して提示される。ユーザは、実行されている対話型の例示、注釈、またはその両方の異なる態様を選択的に探索することができる。
【００３１】
本発明の別の態様によれば、ユーザにアプリケーションを実施することを教示するＷｅｂベースのコンピュータ・システムは、１つまたは複数の事前定義された対話型アプリケーションと、１つまたは複数の注釈を含む注釈ページとを含む。事前定義された対話型アプリケーションは、Ｗｅｂベースのコンピュータ・システムのユーザによって選択的に実行可能である。注釈ページは、事前定義された対話型アプリケーションを説明する。注釈ページは、１つまたは複数のリンクと、アプリケーションの実施の詳細とも含む。事前定義された対話型アプリケーションの選択的な実行に応答して、異なる注釈が注釈ページに自動的に提供される。
【００３２】
本発明のさらなる態様によれば、ユーザにアプリケーションを実施することを教示するＷｅｂベースのコンピュータ・システムは、内容フレーム、フレームワーク・アプレット、および内容フレームの内容に関するリンクのグローバル目次の階層を表示する目次フレームを含むＷｅｂブラウザ・ウィンドウを含む。システムは、内容フレーム内に表示された１つまたは複数の注釈も含み、そこで、各注釈は、事前定義された対話型アプリケーションを説明し、別の内容へのリンクを含む。このシステムは、表示された注釈においてローカル内容に関するリンクのローカル目次の階層を表示する目次ウィンドウを含む。
【００３３】
１つまたは複数の実施形態の詳細を、添付の図面と以下の説明で述べる。この他の機能、目的、および利点は、説明、図面、および特許請求の範囲から明らかになろう。
【００３４】
従来のオンライン・ヘルプ・システムによって提供されるヘルプ情報は、ユーザがアプリ

ケーション・プログラムを効果的に活用することを支援する上で有用であることは既に証明済みである。しかし、これらの従来のオンライン・ヘルプ・システムは、基本的に静的テキストまたはグラフィック情報を提供することに限定されているので、それらの有効性は大幅に低下する。従来のオンライン・ヘルプ・システムのユーザは、ヘルプ・システムによって提供されるテキスト情報およびグラフィック情報を読み、注意深くこれを検討し、次いでその教示を解決すべき問題点に適用することによって解決を得る。その結果、少しでも複雑なアプリケーションを使用することを学ぶことは、しばしば面倒で時間のかかるプロセスである。

【００３５】

ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅとして知られているＩｎｆｏｒｍｉｘ（Ｒ）Ｓｏｆｔｗａｒｅ社によって開発されたオンライン・ヘルプおよび指示システムは、ユーザが情報を吸収し、新しいアプリケーションの使用法を学習することのできる容易さを劇的に強化する。ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅは、Ｗｉｎｄｏｗｓ（Ｒ）９５／ＮＴオペレーティング・システム下で実行するＮｅｗＥｒａ（Ｒ）に関するオンライン文書化機能であり、Ｉｎｆｏｒｍｉｘ（Ｒ）Ｓｏｆｔｗａｒｅ社製のオブジェクト指向アプリケーション開発環境である。ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅのそれ自体の使用法および操作に関するオンライン説明を付録Ａとして添付する。

【００３６】

ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅ（またはより一般的には、ＮｅｗＥｒａ（Ｒ）、ＶｉｓｕａｌＢａｓｉｃ（Ｒ）およびＪａｖａ（Ｒ）開発環境に提供されている例示ベースの説明システムをカバーするＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ）は、所要のタスクをどのように達成するかを実例で示す、動的で対話型の例示がユーザに提供される「例示によるドキュメンテーション」と称されるより一般的な概念の特定の実施である。例示の様々な態様を説明する注釈は、実行中の例示に付随する。例示によるドキュメンテーションは、一部には、ユーザが、単にトピックに関して読むことによるだけでなく、何かを実行する（例えば、ある活動に参加して、その結果を監視したり、その結果に影響を与えたりする）ことによって最もよく学ぶという前提に基づいている。

【００３７】

図１３の流れ図に示すように、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのユーザは、様々な異なるトピックから選択すること（ステップ８００）、それに対するヘルプが探索されるアプリケーションがアクティブに維持されている間に例示を実行すること（ステップ８０５）、例示を実行中に同時に、またはその例示とは関係なく、例示に関して読むこと（ステップ８１０）、異なるエディタ・ユーティリティで例示のソース・コードを検査すること（ステップ８１５）、およびユーザが例示を理解するのに役立つオンライン・バックグラウンド参考資料にアクセスすること（ステップ８２０）を含む情報を獲得するためのいくつかの異なるオプションを有し、これらはすべてヘルプ環境を離れずに行われる。ステップ８１５では、例示のためのソース・コードを、新しいアプリケーションを作成する際にＮｅｗＥｒａ（Ｒ）開発環境でテンプレートとして使用するために、カット・アンド・ペーストすることができるサンプル・プログラムとして使用することができる。さらに、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、試行、例えば、例示またはそれらのパラメータを選択的に変更し、かつその変更が例示の結果にどのように影響を与えるかを監視することによってユーザが学ぶことを可能にする。

【００３８】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅの特定の機能を、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションからとられた例示的画面の一場面である図１４～２９を参照して詳述する。

【００３９】

ユーザが最初にＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅを起動するとき、図１４に示すデフォルト画面構成が表示される。この初期画面には、現行トピック９０２下でユーザに対して現在使用可能なサブトピックを示すリスト（または「目次」）ウィンドウ９００

と、リスト・ウィンドウ９００から選択されたトピックまたはサブトピックに対応するヘルプ情報を表示するテキスト・ウィンドウ９０４との２つの別個の表示ウィンドウが含まれる。リスト・ウィンドウ９００に表示された８個のサブトピック９０６の様々なサブトピックをユーザがクリックするとき、テキスト・ウィンドウ９０４内の情報は、選択されたサブトピック９０６に対応するように自動的に更新される。ユーザは、必要に応じて、前進（「＞＞」）ボタン９０８または後退（「＜＜」）ボタン９１０をクリックすることによって現行トピック内の異なるページに移動することができる。

【００４０】

図１４の例示では、リスト・ウィンドウ９０４に示されるサブトピックは、トピック「ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅ Ｉｎｔｒｏｄｕｃｔｏｒｙ Ｔｏｐｉｃｓ」に関係する。別のヘルプ・トピックに切替え、それによって、オンライン・ヘルプ資料の異なるサブセットを使用可能にするために、ユーザは、図１５に示すように、９つの異なるヘルプ・トピックのリスト９１６を含むＨｅｌｐ Ｔｏｐｉｃメニュー９１４を含んでいるウィンドウを提示するＨｅｌｐ Ｔｏｐｉｃｓボタン９１２をクリックする。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションのどのポイントにおいても、ユーザは、Ｈｅｌｐ Ｔｏｐｉｃｓメニュー９１４を提示させ、所望のトピックをクリックすることによって、オンライン・ヘルプ・システムの別の任意の部分にジャンプすることができる。ユーザは、適切な回数だけＢａｃｋボタン９１６を押し下げることによって前のトピックに戻ることができる。

【００４１】

図１５のリスト９１６内のヘルプ・トピックのそれぞれは、複数レベルのサブトピックの階層を開示するために拡張することができる。ユーザが例えばトピック９１８（「ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅ」）をクリックすると、そのトピックは、図１６に示すサブトピックの２つの付加的なレベルを開示するために拡張する。２つの付加的なレベルとはすなわち、サブトピック「Ｉｎｔｒｏｄｕｃｔｉｏｎ」「Ｃｏｍｍｏｎ ＮｅｗＥｒａ（Ｒ）Ｐｒｏｇｒａｍｍｉｎｇ Ｔｅｃｈｎｉｑｕｅｓ」および「Ｔｈｅ Ｅｘａｍｐｌｅｓ」を含む第１のレベル９２０と、４３個の対話型の例示を含む「Ｔｈｅ Ｅｘａｍｐｌｅｓ」サブトピック下の第２のレベル９２２である。

【００４２】

ユーザが例示、例えば「Ｅｎａｂｌｉｎｇ ａｎｄ Ｄｉｓａｂｌｉｎｇ Ｂｕｔｔｏｎｓ」の例示９２４をクリックすると、選択された例示に関連付けられた注釈セグメント９２６（「Ｏｖｅｒｖｉｅｗ ｏｆ Ｂｕｔｔｏｎｓ２ Ｅｘａｍｐｌｅ」「Ｇｒａｐｈｉｃａｌ Ｏｂｊｅｃｔ Ｓｕｍｍａｒｙ」「Ｅｖｅｎｔ Ｈａｎｄｌｅｒ Ｓｕｍｍａｒｙ」「Ｉｍｐｏｒｔａｎｔ Ｅｖｅｎｔ Ｈａｎｄｌｅｒｓ」および「Ｅｎｈａｎｃｅｍｅｎｔ ａｎｄ Ｖａｒｉａｔｉｏｎｓ」）を表示するためにリスト・ウィンドウ９００が図１７に示すように更新される。注釈セグメント９２６は、対応する例示を一括して説明し、例示のウィンドウ、そのグラフィック・オブジェクト、およびそのイベント・ハンドラの説明を含む。例示の散文説明に加え、注釈セグメントは、通常、検討中の例示のソース・コードから直接的にインポートされた特定の対象のソース・コード・フラグメントも含む。

【００４３】

図１７に示すように、注釈セグメントの１つ（「Ｉｍｐｏｒｔａｎｔ Ｅｖｅｎｅｔ Ｈａｎｄｌｅｒｓ」）は１３個のトピック９２８を含むが、これはＢＵＴＴＯＮＳ２の例示で使用される主イベント・ハンドラのリストである。各イベント・ハンドラのトピック９２８は、ソース・コード・フラグメントと、トピックが対応するイベント・ハンドラを説明する散文説明を含む。例えば、ユーザがイベント・ハンドラ・トピック９３０をクリックすると、テキスト・ウィンドウ９０４は、図１８に示すように、対応するイベント・ハンドラ（ｎｅｘｔＢＴ：：ａｃｔｉｖａｔｅ（））に関するソース・コード・フラグメント９３２をコードの操作を説明する注釈９３４と共に表示する。

【００４４】

テキスト・ウィンドウは、関係する情報への１つまたは複数のリンク、例えばバックグラ

ウンド参考資料を含むことができ、このバックグラウンド参考資料は、付加的なバックグラウンド情報へのさらなるリンクを含むことができ、階層方式によってこれはさらに累々と続く。階層における各連続的なレベルは、例示に関する情報をさらに詳細に、かつさらに低レベルの抽象性で提供する。この方法でさらに詳細な文書に対するリンクの階層を提供することによって、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、ユーザの経験と知識のレベルが様々であることに関係なく、すべてのユーザに対して文脈に合った情報を有用かつ有効な方法で提供する。ユーザは、ユーザのニーズとバックグラウンドに相応の理解のレベルに到達するまで、説明的情報の階層リンクを選択的に横断して降下することができる。この構成は、新人ユーザに対しては、詳細な説明情報への容易なアクセスを提供し、同時に、特定の点に関するヘルプを探している経験豊富なユーザを多量の不要な情報の中をナビゲートしなければならないことから保護する。

【００４５】

図１７に階層リンクの一例を示す。ここでは、テキスト・ウィンドウ９０４は、１つの関数はＢＵＴＯＴＯＮ２の例示の関数のうちの１つであるＭＡＩＮ（）関数へのリンク９３６（緑色の下線を引いたテキストによって表示される）を含む。ユーザがＭＡＩＮ（）関数リンク９３６をクリックすると、図１９に示すようにテキスト・ウィンドウ９０４はその関数に対するソース・コード９４０を表示する。ソース・コード９４０は、キーワードおよびオブジェクトのクラスの説明を含むオンライン言語解説書などの関連情報へのさらなるリンクを含む。ユーザがそれらのリンクの１つ、例えばＬＥＴステートメントへのキーワード・リンク９４２をクリックすると、テキスト・ウィンドウ９０４は、図２０に示すように対応するオンライン言語解説項目を表示するように変わる。同様に、ユーザがオブジェクト・クラス・リンク９４４をクリックした場合、テキスト・ウィンドウ９０４はｉｘＳＱＬＣｏｎｎｅｃｔクラスに関する情報を既に表示しているはずである。図２０では、ユーザは、例えば、テキスト・ウィンドウ９０４を図２１のように表示させるようにＯｂｊｅｃｔ Ｅｘｐｒｅｓｓｉｏｎボックス９４６をクリックすることによって、さらなるバックグラウンド情報へのリンクに従うことができる。その後、または別法として、ユーザは、図２２に示すように、用語集目次ウィンドウ９５０および用語集テキスト・ウィンドウ９５２の一対のウィンドウ中にオンライン用語集を提示するためにＧｌｏｓｓａｒｙボタン９４８をクリックすることができる。用語集目次ウィンドウ９５０の用語をクリックすることによって、その用語の定義が用語集テキスト・ウィンドウ９５２に表示される。

【００４６】

例示の注釈、そのソース・コード・フラグメント、対応する言語解説項目、用語集、またはそれらの組み合わせを検討した後、ユーザは、図１５に示すようにＨｅｌｐ Ｔｏｐｉｃメニュー９１４を提示させるＣｏｎｔｅｎｔｓボタン９５４（または検索可能な索引形式の使用可能なヘルプ・トピックを提示するＩｎｄｅｘボタン９５６）をクリックし、次いで図１５および１６に関連して説明した方法で所望のトピックをユーザが選択することによって、ヘルプ・システム内の任意の別の位置に選択的にジャンプすることができる。

【００４７】

図１９のＬＥＴステートメント・リンク９４２およびｉｘＳＱＬＣｏｎｎｅｃｔクラス・リンク９４４などのキーワード・リンクおよびクラス名リンクは、それぞれに、可視的な固有の方法（例えば、キーワードには青色の上付きテキスト、クラス名には青色の下付きテキスト）で示され、その結果、容易に相互から区別することができ、また、図１７に示すようにＭＡＩＮ（）関数リンク９３６（緑色の下付きテキスト）などの他の種類のリンクからも容易に区別することができる。異なる種類のリンクに対して異なるスタイルを用いることによって、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、使用可能なオンライン情報の性質および例示の異なる構成要素（注釈、ソース・コード・フラグメント、言語解説など）間の相互関係に関して直感的で有用な情報をユーザに提供する。ほぼ任意の数の異なるリンクの種類は、システム設計者の好みに応じて異なるスタイルで表示される場合がある。

【００４８】
例示の注釈に含まれるソース・コード・フラグメントのそれぞれに関して、ユーザは、例示のソース・コードを検査し、編集し、またはコピーするために、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅから適切な編集ユーティリティを呼び出すことができる。これによって、ユーザは、ソース・コード・フラグメントが取り出されるさらに大きなプログラムに照らしてソース・コード・フラグメントを表示させることができる。
【００４９】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、２つの異なる種類のソース・コードに由来するソース・コード・フラグメントを含む。それらは、ＮｅｗＥｒａ（Ｒ）プログラミング言語のテキスト・プログラム・コード（４ＧＬまたは４ＧＨファイル接尾辞によって示される）と、開発中のアプリケーションのエンド・ユーザに対してＧＵＩをどのように表示し、そのエンド・ユーザとどのようにインターフェースするかを定義するウィンドウ・インターフェース定義ファイル（接尾辞ＷＩＦを有するファイル）である。どちらか一方の種類のソース・コード・フラグメントを表示するために、ユーザは、コード・フラグメントの隣にあるショートカットの矢印、例えば図１７～１９に示すショートカット矢印９５８および９６０のうちの１つをクリックする。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅはこれに応答して、検討中のソース・コードの種類に対応するエディタを起動する。ユーザが、図１７および１９のショートカット矢印９５８などの４ＧＨまたは４ＧＬファイルの隣のショートカット矢印をクリックすると、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、図２３に示すように、コード・フラグメントがそこから取り出されたソース・コード・ファイルを表示させるために、適切なエディタであるＮｅｗＥｒａ（Ｒ）Ｃｏｄｅｗｒｉｇｈｔを自動的に起動する。同様に、ユーザが、図１７および１８のショートカット矢印９６０などのＷＩＦファイルの隣のショートカット矢印をクリックすると、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、図２４に示すように、コード・フラグメントがそこから取り出されたＷＩＦファイルを表示させるために、適切なエディタであるＮｅｗＥｒａ（Ｒ）Ｗｉｎｄｏｗ Ｐａｉｎｔｅｒ３．０を自動的に起動する。
【００５０】
この方法で複数のエディタのうちから適切なエディタを選択的に起動することは、ＮｅｗＥｒａ（Ｒ）開発環境の標準的な編集動作を反映する。ＮｅｗＥｒａ（Ｒ）開発環境と、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ資料システムの両方とも、同じ方法で同じエディタを利用する。その結果、ユーザは、オンライン・ヘルプ・システム（すなわち、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ）との通常の対話によって、ヘルプを捜すアプリケーション（すなわち、ＮｅｗＥｒａ（Ｒ）開発環境）に精通する。
【００５１】
ユーザが例示に対するソース・コードの開発を完了すると、ＷＩＦファイルのビジュアル・オブジェクトまたは４ＧＨまたは４ＧＬファイル内のプログラム・ステートメントがユーザ自身のソース・ファイルに入っているかどうかに関わらず、ユーザは、簡単にそのコードを検討したり、そのコードの一部をカット・アンド・ペーストしたりすることができる。別法として、ユーザは、「Ｓａｖｅ Ａｓ．．．」操作を実行することができ、それによって、新しいファイル名でその例示に対するソース・コードを保存することができる。次いでユーザは、必要に応じて新しいファイルを編集し、または他の方法で操作することができる。この方法で、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅによって提供された例示は、ＮｅｗＥｒａ（Ｒ）開発環境で新しいアプリケーションを開発する際に使用するためのテンプレートとしての機能を果たすことができる。
【００５２】
ユーザは、それらの動作方法を直接的に監視するためにＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅに提供された４３個の対話型の例示のどれかまたはすべてを実行することができる。例示は事前に構築されており、それらの対応するＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ注釈から直接的に起動されることができる。これを実行するためには、ユーザは、まず図１６に示すＨｅｌｐ Ｔｏｐｉｃｓウィンドウ９１４から関心のある例示を選

択し、対応する注釈がテキスト・ウィンドウに表示されると、そのテキスト・ウィンドウ上部付近に表示されているＲｕｎボタンをクリックする。それに応答して、例示が実行され、また、ユーザから受け取った入力に基づいて、その例示がスタンドアロン・アプリケーションの場合は様々な画面をユーザに対して表示する。同時に、テキスト・ウィンドウは、ユーザによって現時点で実行中の例示の一部に関連する説明情報を表示させるために自動的に更新する。ユーザが実行中の例示で実行している各連続的な操作によって、ユーザに対して現在例示に何が起こっているかを説明する注釈の対応するセクションを表示することによって、対話型の例示の状況との同期を維持するために、テキスト・ウィンドウが同時に（またはほぼ同時に）更新される。ヘルプ表示を例示の現在の状態と調整することによって、ユーザの関心のある現行トピックに直接的に関連するタイムリーかつ有用な情報（例えば、その例示によって実行中の特定のソース・コード）が一貫してユーザに提供される。その結果、情報を理解し吸収するユーザの能力は劇的に高められる。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅの例示の自動的に調整されたヘルプ表示を図２５～２９に示す。

【００５３】

図２５は、ユーザがＨｅｌｐ　Ｔｏｐｉｃｓメニューから「Ｄｉｓｐｌａｙｉｎｇ　ａｎ　Ｅｄｉｔ　Ｍｅｎｕ」の例示を選択したときに、表示される初期リスト・ウィンドウ９００およびテキスト・ウィンドウ９０４を示す。この例示を実行するために、ユーザはＲｕｎボタン９６２をクリックする。このＲｕｎボタン９６２は、図２６に示すように、編集ウィンドウの基礎を示す例示ウィンドウ９６４を生成する。同時に、テキスト・ウィンドウ９０４は、「Ｄｉｓｐｌａｙｉｎｇ　ａｎ　Ｅｄｉｔ　Ｗｉｏｎｄｏｗ」の例示に対するＭＡＩＮ（）関数に関する情報を表示するために更新される。

【００５４】

ユーザが例示ウィンドウ９６４内でＧＵＩ制御を選択的に操作すると、対応する方法でテキスト・ウィンドウ９０４に表示される情報が自動的に更新される。図２７では、ユーザは、テキスト・ウィンドウ９０４にｅｄｉｔ１ＴＢ：：ｆｏｃｕｓｌｎ（）に関する情報を表示させるテキスト・ボックス９６６内を既にクリックしている。同様に、ユーザがテキスト・ボックス９６８をクリックすると、テキスト・ウィンドウ９０４は、図２８に示すようなｅｄｉｔ２ＴＢ：：ｆｏｃｕｓｌｎ（）に関する情報を表示する。ユーザがＣｈｅｃｋＢｏｘ９７０をクリックすると、テキスト・ウィンドウ９０４は、図２９に示すようにｎｏｎｅｄｉｔＣＢ：：ｆｏｃｕｓｌｎ（）に関する情報を表示する。

【００５５】

ユーザは、例示のソース・コードを変更するか、またはそのパラメータを修正し、それらの変更が例示にどのような影響を与えるかを監視することによって、例示を試行することができる。これを実行するために、ユーザは、所望のソース・コード・ファイルを編集し、事前定義された例示を乱さないように、編集したそのソース・コード・ファイルを別個の作業ディレクトリとして保存し、次いでＮｅｗＥｒａ（Ｒ）開発環境に提供される機構を使用してその例示を再構築する。作成され、実行されることのできるそのような試行の数と種類は、ユーザの想像力によってのみ限定されるものである。

【００５６】

例示を実行する際の別のオプションも可能である。例えば、ユーザは、同時に注釈を表示させずに例示を実行することができる。さらに、ＮｅｗＥｒａ（Ｒ）に提供されたＤｅｂｕｇｇｅｒは、例示を実行する前に例示のソース・コードに区切り点を設定するために使用することができ、それによって、ユーザには例示がどのように機能するかに関するさらなる洞察が与えられる。

【００５７】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアーキテクチャの説明と、ＮｅｗＥｒａ（Ｒ）開発環境およびＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションが構築される方法を、図３０～３２を参照して提供する。

【００５８】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅは、Ｗｉｎｄｏｗｓ（Ｒ）９５／ＮＴオペレーティング・システムに提供されたオンライン・ヘルプ（ＯＬＨ）機能を基礎としている。図３０に示すように、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００は、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅに対して特に作成された資源、ならびにＮｅｗＥｒａ（Ｒ）開発環境１００５に固有の資源を示す。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００に特有の構成要素は、対話型例示１００７、その例示に対するソース・コード１０１０、およびその例示を説明する注釈１０１５を含む。注釈１０１５は、例示のソース・コードの代表フラグメント１０２０、例示のソース・コードを表示させるための適切なエディタ（例えば、ＮｅｗＥｒａ（Ｒ）ＣｏｄｅｗｒｉｇｈｔまたはＮｅｗＥｒａ（Ｒ）Ｗｉｎｄｏｗ Ｐａｉｎｔｅｒ）を起動するショートカット１０２５、対話型例示へのジャンプ１０３０、ＮｅｗＥｒａ（Ｒ）オンライン解説１０４０内に含まれる指定されたキーワードおよびクラス名の説明へのリンク１０３５を含む複数の異なる副構成要素を有する。

【００５９】

図３０に示すように、オンライン解説１０４０、コードライト・エディタ１０５０およびウィンドウ・ペインタ・エディタ１０５５は、アプリケーション・ビルダ１０６０、デバッガ１０６５およびＩＰＣ（Ｉｎｔｅｒｐｒｏｃｅｓｓ Ｃｏｍｍｕｎｉｃａｔｉｏｎ）ライブラリ１０７０と共に、開発環境１００５の一部として存在し、したがってＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００とは論理的に分離される。その結果、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００のユーザが、キーワードまたはクラス名へのリンク１０３５をクリックすることによって、またはソース・コードを表示させるためのショートカット１０２５をクリックすることによって、ＮｅｗＥｒａ（Ｒ）開発環境内に常駐する資源を要求すると、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ１０００は、要求された資源にアクセスするために、まずインターフェース・ダイナミック・リンク・ライブラリ（ＤＬＬ）１０８０を介してＮｅｗＥｒａ（Ｒ）開発環境１００５と対話する必要がある。インターフェースＤＬＬ１０８０は、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００に、開発環境の構成要素などの他のアプリケーションと対話することを可能にするルーチンのコンパイルされたライブラリである。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ１０００は、要求されたオンライン解説情報を表示するため、または適切なソース・コード・エディタを起動するために、ユーザによって行われた要求の性質に応じて適切なＤＬＬルーチンを呼び出す。

【００６０】

より具体的には、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのユーザが、例示のソース・コード１０１０の位置へのショートカット１０２５をクリックすると、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００はＤＬＬ内の関数を呼び出すが、そのＤＬＬ内の関数は、ＩＰＣライブラリ１０７０内の関数を呼び出し、そのＩＰＣライブラリ１０７０内の関数は適切なエディタを起動する。この関数呼び出し（後述する、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅの構築中にソース・コード・フラグメントを処理することによって自動的に生成される）の一部として、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００は、起動されるべきエディタ（コードライト１０５０またはウィンドウ・ペインタ１０５５）を指定し、かつ例示のソース・コード１０１０が指定されたエディタによって開かれるべき行数を識別するパラメータを渡す。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのユーザがキーワードまたはクラス名へのリンク１０２５をクリックすると、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００はＤＬＬ内の関数を呼び出し、その関数は、Ｗｉｎｄｏｗｓ（Ｒ） ＯＬＨ機能を使用してオンライン解説１０４０内の対応する定義を表示させる。

【００６１】

インターフェースＤＬＬ１０８０によって提供される他の関数は、対話型例示１００７の実行を制御し、リスト・ウィンドウおよびテキスト・ウィンドウの表示が確実に対応を維

持するように調整する。インターフェースＤＬＬ１０８０と、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーション１０００のランタイム動作に関するさらなる詳細を付録Ｂに記載する。

【００６２】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ１０００とその構成要素（例えば、例示１００７、その例示のソース・コード１０１０および注釈１０１５）が生成される方法は、高度のコードの「保全性」を実現する。この保全性とは、アプリケーションを修正することができる効率および容易性の尺度である。高度のコード保全性は、対話型例示と、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅの対応する注釈構成要素を生成するために使用されるすべての情報を、統合された論理エンティティ、すなわち対話型例示自体に対するソース・コードに埋め込むことによって達成される。その結果、情報の唯一の中央ソースだけを維持すればよい。その中央情報に対してなされたいかなる変更または更新も、例示と、その例示に関する資料／説明／ヘルプ機能（Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ）の両方に自動的に埋め込まれることになる。この自動化された構築手順により、例示と、対応するＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅの注釈は、基礎をなすソース・コードに行われた修正の数および回数に関わらず同期して維持されることが保証される。

【００６３】

図３１に示すように、ＮｅｗＥｒａ（Ｒ）ｂｙＥｘａｍｐｌｅのソース・コード１１００は、物理的には相互依存ファイルの集合で形成されているが、単一の論理エンティティとみなすことができる。ソース・コード１１００は、ソース・コード全体で混合されているプログラム説明１１０５、プログラム・コメント１１１０、および注釈１１１５の３つの基本的な種類のテキストを含んでいる。異なるテキストの種類は、プログラミングの慣例により、また、ソース・コード全体を通して様々な異なるマークアップ・シンボル１１２０を戦略的に配置することによって、相互に区別される。

【００６４】

ソース・コード１１００内のいくつかのテキストは、複数の目的に役立つことができる。例えば、ソース・コード１１００内のプログラム説明１１０５は、例示の２進数の実行可能ファイル１１２５内にコンパイルされる。これらのプログラム説明は、例示の実行中の適切なポイントで対応する注釈を表示するためのＯＬＨ機能を呼び出すことを含む。エンド・ユーザが例示を実行するとき、これらのＯＬＨ呼び出しは、その例示に現時点で何が起こっているかを自動的に説明するためにテキスト・ウィンドウに適切な注釈を表示させる。

【００６５】

それらの同じプログラム説明１１０５の部分は、また、その例示のソース・コードのクリーン・コピーとしての機能を果たすために抽出され、それは編集環境でユーザに対して表示することができる。同様に、プログラム・コメント１１１０としての機能を果たす説明テキスト（Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅプロジェクト開発者に方向付けた未処理のプログラミング説明）は、また、注釈１１１５（実行時にＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのエンド・ユーザに表示されるプログラミング説明）としての機能を果たすことができる。

【００６６】

マークアップ・シンボル１１２０は、ソース・コード内の様々な種類のテキストを図で示し、対話型例示およびＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ注釈が構築されるときにそれらがどのように扱われるべきかを指定する。図３２に、「通常の」シンボル１２００および１２０５の２つのインスタンス、「［ｅｄｉｔ」シンボル１２１０および「］ｆｉｌｅ」シンボル１２１５を含むいくつかのマークアップ・シンボルを含むＮｅｗＥｒａ（Ｒ）ソース・コードのサンプルを示す。これらのマークアップ・シンボルのそれぞれは、それぞれの引数と共に、それらがコメント・フィールド内に常駐しており、ＮｅｗＥｒａ（Ｒ）プログラム説明として扱われないことを示す一対の括弧（「｛．．．｝」）によってバインドされる。ＮｅｗＥｒａ（Ｒ）以外のプログラミング言語は、コメント・フィ

ールドを図で示すために異なる規則を使用することができる。例えば、Ｊａｖａ（Ｒ）プログラミング言語では、コメント・フィールドの開始は「／＊」シンボルによって示され、「＊／」シンボルによって終了する。いずれにせよ、対応するプログラミング言語のコンパイラは、コメント・フィールドに常駐するよう示されたすべてのテキストを無視することになる。

【００６７】

「．ｎｏｒｍａｌ」マークアップ・シンボルは、そのシンボル（例えば、シンボル１２００に続く「Ｓｉｎｃｅ ｏｂｊｅｃｔｓ、．．．，」）に続くテキストは説明コメントとして扱われるべきであり、したがって、対応する対話型例示を実行中の適切なポイントで注釈テキストの一部としてテキスト・ウィンドウ内でエンド・ユーザに対して表示されるべきであることを示す。他のマークアップ・シンボルは、出力ファイル名、例示のソース・コードの代表フラグメントとしての機能を果たすべきソース・コードの一部、ジャンプおよびリンクのホットスポットおよび宛先、または表示特性およびオブジェクト（ウィンドウ、ポップアップ、ボタンなど）に関するＧＵＩ関連情報を指定する。マークアップ言語の詳細な説明を付録Ｃに記載する。

【００６８】

一旦ソース・コード１１００が所望通りに修正されると、このコードは、いくつかの異なるステップを通してＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションの対話型例示および説明的な内容を構築するために使用される。第１に、ソース・コード１１００は、ＰＥＲＬスクリプト（テキストを構文解析するためにしばしば使用される汎用インタープリタ型言語であるＰｒａｃｔｉｃａｌ Ｅｘｔｒａｃｔｉｏｎ ａｎｄ Ｒｅｐｏｒｔ Ｌａｎｇｕａｇｅ）およびＷｏｒｄＢａｓｉｃスクリプトの２つの異なるスクリプトによって処理される。スクリプト１１３０は、２つの基本的なタイプの出力を生成する。２つの基本的なタイプの出力とは、すなわち対話型例示に対するソース・コード・ファイル１１３５と、Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションの説明的かつ視覚的な内容（例えば、注釈、ソース・コード・フラグメント、ソース・コード・エディタへのショートカット、オンライン解説へのリンク、実行可能な例示へのジャンプ）を表現するＲＴＦファイル１１４０（ＯＬＨコンパイラが要求するリッチ・テキスト・フォーマット）である。

【００６９】

ＰＥＲＬスクリプトは、マークアップ・シンボルを検索中のソース・コード１１００を構文解析し、遭遇した特定のマークアップ・シンボルに基づいていくつかのＲＴＦファイル・フラグメントといくつかのソース・コード・ファイル１１３５とを作成するが、これらは、ソース・コード１１００全体の様々なサブセットを表現する。次いでＷｏｒｄＢａｓｉｃ Ｓｃｒｉｐｔは、ＲＴＦファイル・フラグメントを完全なＲＴＦファイル１１４０にマージし、これはＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションに対する説明的かつ視覚的な内容を含んでいるＯＬＨファイル１１５０を作成するためにＷｉｎｄｏｗｓ（Ｒ） ＯＬＨコンパイラ１１４５によって処理される。同時に、例示のソース・コード１１３５は、対話型例示１１２５に対応する実行可能な２進数を生成するために、ＮｅｗＥｒａ（Ｒ）コンパイラ１１５５によってコンパイルされる。

【００７０】

ＰＥＲＬスクリプトによって生成されたＲＴＦファイル・フラグメントは、ソース・コード１１００内に表示される注釈１１１５の他にいくつかの異なる構成要素を含む。ＰＥＲＬスクリプトは、例示のために抽出されたソース・コードに表示されるキーワードまたはクラス名の各インスタンスを識別する。検出された各キーワードおよびクラス名に関して、ＰＥＲＬスクリプトは、オンライン参考資料内の対応する項目へのリンクをＲＴＦファイル内に作成する。

【００７１】

ＰＥＲＬスクリプトは、また、説明的なコメントと共に表示されるテキストとしてＲＴＦファイルに含むために、ソース・コードを表現するフラグメントも抽出する。ソース・コ

ード・フラグメントは、先導し、また後に続く空白行によって図で示されるモノスペースのラップされていないテキストとしてフォーマットされ、説明的なコメントは比例的にスペースの与えられたラップ・テキスト（ｗｒａｐｐｅｄ　ｔｅｘｔ）としてフォーマットされる。ＲＴＦファイルに含まれるソース・コード・フラグメントごとに、ＰＥＲＬスクリプトは、また、ＲＴＦファイル内に、対応するショートカット・ボタンを挿入する。このショートカット・ボタンによって、エンド・ユーザは、ソース・コード・エディタを起動し、フラグメントが開始する行でソース・コードを表示させることができる。ＰＥＲＬスクリプトは、また、その例示のために抽出されたソース・コードからすべてのマークアップ・シンボル１１２０を除去する。これにより、関連付けられたエディタで表示させるためのソース・コードのクリーン・バージョンがエンド・ユーザに提供される。

【００７２】

ＰＥＲＬスクリプトによって実行される他の関数は、注釈トピックに対する識別子が、対話型例示においてもＷｉｎｄｏｗｓ（Ｒ） ＯＬＨ機能に含まれるのと同じであることを自動的に保証することを含む。すなわち、ＰＥＲＬスクリプトは、Ｗｉｎｄｏｗｓ（Ｒ） ＯＬＨ機能のためのヘルプ・トピック識別子を読み込み、対応するＮｅｗＥｒａ（Ｒ）定数を生成する。ＰＥＲＬスクリプトは、また、例示を構築するために使用されるＮｅｗＥｒａ（Ｒ）メークファイル（アプリケーションを構築するためのコンピュータ可読命令を含むファイル）の修正されたバージョンも生成する。ＰＥＲＬスクリプトとその操作のさらなる詳細を付録Ｂに記載する。

【００７３】

上述のＰＥＲＬおよびＷｏｒｄＢａｓｉｃスクリプトはＮｅｗＥｒａ（Ｒ）プログラミング言語で書かれたソース・コードで動作するが、他の種類のソース・コード、例えばＪａｖａ（Ｒ）またはＶｉｓｕａｌＢａｓｉｃ（Ｒ）などを構文解析するために異なるスクリプトを使用することも可能である。一般に、提供されるほぼいかなる種類のプログラミング言語でも処理するために、適切なＰＥＲＬおよびＷｏｒｄＢａｓｉｃスクリプトを書くことでき、そのプログラミング言語は、ＡＳＣＩＩソース・コード（ＰＥＲＬによって要求される）を使用し、ある種のソース・コードのコメント機構を提供する。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ技術の使用を容易にする他のプログラミング言語の属性は、トピック識別子を有するＷｉｎｄｏｗｓ（Ｒ） ＯＬＨ機能を呼び出すための機構（例示がその注釈を表示することができるための）、開発環境の編集関数を呼び出すための機構（検討中のプログラミング言語が開発環境を提供または要求すると仮定して、注釈がソース・コード・ファイルを開くことができるための）、Ｗｉｎｄｏｗｓ（Ｒ） ＯＬＨフォーマットでのオンライン解説（ソース・コードのキーワードがオンライン解説へのジャンプを有することができるための）を含む。基礎をなすプログラミング言語に１つまたは複数のそれらの他の属性が欠如しても、上述のＩｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅ機能の多くは実施可能である。

【００７４】

ＰＥＲＬスクリプトは、ＲＴＦ以外のフォーマットでファイルを出力するために修正されることができる。例えば、修正されたＰＥＲＬスクリプトは、いかなる使用可能なＷｅｂブラウザ（例えば、Ｎｅｔｓｃａｐｅ Ｎａｖｉｇａｔｏｒ）を使用して表示させることができるハイパーテキスト・マークアップ言語（ＨＴＭＬ）を出力することができる。

【００７５】

例示によるドキュメンテーションの他の変形形態も可能である。例えば、対話型例示を説明する注釈は、テキスト以外の方法で提示することもできる。必要に応じて、音声、グラフィック・シンボル、ピクチャー、ムービー、または他のいかなる伝達手段をも使用することができる。さらに、実行すべき対話型例示の選択は、ユーザによる指定以外の要因に基づくか、またはユーザによる指定に付加的な要因に基づくことができる。例えば、対話型例示は、基礎となるアプリケーショの実行中のあるポイントで、またはヘルプ・システムのある実行ポイントで自動的に起動することができる。ユーザがキーワード、クラス名、または他のリンクをクリックすると、リンクによって指し示されたテキスト解説情報を

表示することに加え、またはこれの代わりに例示を自動的に起動することができる。
【００７６】
上述の例示によるドキュメンテーションの方法および技術は、ソフトウェア開発システムのユーザを支援することにのみ限定されるものではなく、アプリケーションを任意のコンピュータ・ベースのアプリケーションまたはユーティリティのための一般的なトレーニングおよび教育ツールとみなすことができる。さらに、本明細書に記載の技術は、ハードウェアまたはソフトウェアでも、またはこれらの組み合わせによって実施することができる。この技術は、それぞれがプロセッサ、（揮発性および不揮発性メモリ、または記憶素子、あるいはその両方を含めて）そのプロセッサによって可読の記憶媒体、および適切な入出力装置を含むプログラミング可能なコンピュータ上で実行中のコンピュータ・プログラムによって実施されることが好ましい。プログラム・コードは、記述された関数を実行し、出力情報を生成するために、入力装置を使用して入力されたデータに適用される。この出力情報は、１つまたは複数の出力装置に適用される。
【００７７】
各プログラムは、好適には、コンピュータ・システムと対話するために、高レベル手続き型のまたはオブジェクト指向プログラミング言語によって実施される。しかし、プログラムは所望ならばアセンブリまたは機械言語でも実施することができる。いずれにせよ、言語はコンパイル型言語またはインタープリタ型言語のどちらかであってよい。
【００７８】
そのような各コンピュータ・プログラムは、記憶媒体または記憶装置（例えば、ＣＤ－ＲＯＭ、ハードディスクまたは磁気ディスケット）に記憶することが好ましい。尚、これら記憶媒体または記憶装置は、これらが上述の手続きを実行するためにコンピュータによって読み込まれたときに、コンピュータを構成し、また、操作するために、汎用または専用のプログラム可能コンピュータによって可読である。このシステムは、コンピュータ・プログラムによって構成されたコンピュータ可読記憶媒体としても実施することができるが、この場合、そのように構成された記憶媒体はコンピュータを特定の事前定義した方法で動作させる。
【００７９】
他の実施形態は頭記の特許請求の範囲に含まれる。
【００８０】
例えば、例示によるドキュメンテーションの方法および技術は、ローカル・エリア・ネットワーク（ＬＡＮ）、イントラネット、またはインターネットなどのネットワークまたはＷｅｂベース環境に適用することができる。Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅとして知られているＩｎｆｏｒｍｉｘ（Ｒ）Ｓｏｆｔｗａｒｅ社製のＷｅｂベースの説明システムは、開発者が有効かつ効率的にデータベース・アプリケーションを実施することができる容易性を劇的に高める。このＷｅｂベースの説明システムは、ユーザにアプリケーションの実施方法を教示するＷｅｂページ形式の例示および説明を提供する。アプリケーション自体は、Ｗｅｂベースのアプリケーションであっても、そうでなくてもよい。このＷｅｂベースの説明システムによって、ユーザは、アプリケーション・プログラムの実施の詳細にアクセスするために、フォーマットされたソース・コードを読み、プログラム構造（例えば、ファイル、クラス、関数の階層）をナビゲートすることができる。このＷｅｂベース・システムは、ユーザが言語キーワード（コード・フラグメント形式として示すことができる）から、そのキーワードに関係する完全な参考文書にジャンプすることも可能にする。ユーザは、エディタでソース・コードを開き、ツールバーの実行ボタンのクリックでＷｅｂブラウザから直接的にプログラムを実行することができる。例示が実行されると、現行方法またはプログラム内の関数に関する注釈ページが表示される。したがって、ユーザは、そのようなアプリケーション・プログラムを実施する方法をより迅速に理解するために方法または関数が実行されるときに、注釈を表示させることができる。
【００８１】
これは、単純にユーザにアプリケーションの使用方法を示す従来のオンライン・ヘルプ・

システムとは対照的である。一部の実施はアプリケーションの動作を変更しないが、すべての実施態様の変更は、定義により、実施を左右するので、これは実施資料の維持をより重要なものにする。Ｗｅｂベースの説明とヘルプ・システムは、従来のオンライン・ヘルプ・システムのこの問題点を、ソース・コードに埋め込まれたコメントとして注釈を維持し、ソース・ファイルが変更するときはいつでも注釈ページを自動的に生成することによって解決する。

【００８２】

Ｗｅｂベースのシステムは、また、ユーザに対して特定タスクを達成するためのハウツー資料を提供する。このハウツー資料は、明瞭なステップ・バイ・ステップの説明を提供し、エキスパートおよび「初心者」に対して柔軟なインターフェースを提供する。このハウツー資料は有用なグラフィックを含み、必須のソフトウェアおよび関連するデモおよび他の技術情報へのリンクを提供する。さらに、このハウツー資料は、目的を達成するためにどの製品を使用すべきかをユーザが決定するために役立つ。

【００８３】

図３３に示すように、ユーザが最初に、Ｗｅｂベースの説明システムを提供するＷｅｂブラウザにアクセスしたとき、ディフォルト・ブラウザ構成１３００が表示される。初期ブラウザは、３つの別個の表示フレームのフレームワークを含む。そのフレームワークとは、すなわち、典型的にはＷｅｂページを表示するが、注釈ページ（特定タイプのＷｅｂページ）などの他の関連ページも表示する内容フレーム１３０５と、他のサービスを提供することに加えてツールバーまたはナビゲーション・バーを表示するフレームワーク・アプレット（例えばＪａｖａ（Ｒ）アプレット）を含むトップ・フレーム１３１０と、例えばＪａｖａ（Ｒ）アプレットを使用して実施することができる目次（ＴＯＣ）フレーム１３１５である。

【００８４】

ブラウザがｂｙＥｘａｍｐｌｅ（Ｒ）ＨＴＭＬページを表示するとき、そのページはＪａｖａＳｃｒｉｐｔを使用して自動的にフレームワークの存在をチェックする。フレームワークが存在する場合、ＪａｖａＳｃｒｉｐｔは、トップ・フレーム１３１０のフレームワーク・アプレットに対して、新しいページの名称およびそれが提供する内容の種類に関して通知する。ページにフレームワークが無い場合、ＪａｖａＳｃｒｉｐｔは、そのページに、初期ページ（すなわち、通知が行われるページ）としてフレームワークを開く。

【００８５】

図３４および３５を参照すると、ページ１４００の名称に基づいて、目次１３１５は対応するタイトル１４０５を自動的に選択する。目次１３１５は、クライアントで使用可能なＪａｖａ（Ｒ）ライブラリをテストし、使用可能ならばツリー制御を有する目次階層を自動的に表示する。目次フレーム１３１５は、トップ・フレーム１３１０の内容ボタン１４１０を使用してユーザによってサイズ変更可能かつ除去可能である。例えば、図３４では、目次フレーム１３１５は、ブラウザの約４０％を占有しているが、図３５では、目次フレーム１３１５は除去されており、したがって不可視である。内容ボタン１４１０は、除去されたアイコン（例えば、図３５）または開かれたアイコン（例えば、図３４）を表示することによってＴＯＣの状況を自動的に示す。

【００８６】

ユーザは、ＴＯＣツリーの対応するサブトピック、例えばサブトピック「Ｅｘｔｅｎｄｉｎｇ Ｄａｔａｂａｓｅ Ｓｅｒｖｅｒｓ」１４１５をクリックすることによって、ＴＯＣフレーム１３１５から内容フレーム１３０５内の他のリンクにアクセスすることができる。それに応答して、図３６に示すように、内容フレーム１３０５は、そのサブトピックに対応する内容を表示し、ＴＯＣフレーム１３１５はその選択されたサブトピック１４１５を強調表示する。トップ・フレーム１３１０が、ブラウザがホームページを表示しているかどうかによってその外観（内容ボタンのように）を自動的に変更するホーム・ボタン１４２０を含むということは、この特定の実施例でも明らかである。

【００８７】

別法として、ユーザは、その内容フレーム内の対応するリンク上、例えば、図３５に示すサブトピック・リンク「Ｊａｖａ Ｄａｔａｂａｓｅ Ｃｏｎｎｅｃｔｉｖｉｔｙ（ＪＤＢＣ）」１４２５をクリックすることによって、内容フレーム１３０５内のリンクにアクセスすることができる。これに応答して、図３７に示すように、内容フレーム１３０５は、そのサブトピックを表現するタイトル１４３０を含めてそのサブトピックに対応する内容を表示する。さらに、ＴＯＣフレーム１３１５は、ユーザに対して内容フレーム１３０５内にどの内容が表示されるかを示すために、ＴＯＣツリー内の選択されたサブトピック１４３５を自動的に強調表示する。ＴＯＣと内容の間のこのような同期は、ユーザがアプリケーションの多くのサブトピックおよびトピックを効果的にナビゲートするために役立つ。

【００８８】

再び図３６を参照して、ユーザが内容フレーム１３０５に例示アイコンまたはリンク１４４０、またはＴＯＣフレーム１３１５内のサブトピック・アイコンまたはリンク１４４５のどちらかを使用して例示を選択するとき、ブラウザは、内容フレーム１３０５内に、図３８に示す対応する例示を説明する注釈ページ１５００を表示する。さらに図３９を参照すると、注釈ページ１５００は、例示（散文とも呼ばれる）を説明する注釈１５０５と、別の注釈ページ１５１０への少なくとも１つのリンクとを含む。

【００８９】

例えば、ユーザがソース・ファイル「ｑｕｅｒｙｄｂ．ｈｔｍｌ」へのリンクをクリックするとき、内容フレーム１３０５は、図４０のブラウザに示すようにソース・コード・ファイル注釈ページ「ｑｕｅｒｙｄｂ．ｈｔｍｌ」１５２５を表示する。内容フレームは、図４１に示すように、対応するソース・コード・ファイルｑｕｅｒｙｄｂ．ｈｔｍｌに関するソース・コード・フラグメント１５３０を表示する。ソース・コード・フラグメント１５３０は、検討中の例示のソース・コード・ファイルから直接的にインポートされたものである。内容フレーム内のソース・コード・ファイル注釈ページは、トピックが対応するソース・コード・ファイルを説明する注釈１５３５も表示するが、このような注釈１５３５は散文と称される場合がある。

【００９０】

ソース・コードは、標準ＨＴＭＬでマークアップされている。注釈コメントがソース・コード・ファイルのソース・コード・コメント内に埋め込まれているので、注釈１５３５は維持することが容易である。例えば、図４２を参照すると、ソース・コード・ファイル１５６０は、ｑｕｅｗｒｙｄｂ．ｈｔｍｌソース・コード１５２５に関して示される。ソース・コード・フラグメントを含んでいる注釈ページは、マークアップされたソース・コード・ファイルから自動的に生成される。

【００９１】

図４３の流れ図１６００を参照すると、例示の作者は、ソース・コード・コメントに注釈を埋め込むことによってソース・コード・ファイルを注釈する（ステップ１６０５）。この作者は、また、ＨＴＭＬタグおよびＷｅｂベース・システムの１つまたは複数の特別な説明タグ付きの埋め込まれた注釈もマークアップする（ステップ１６１０）。図４４の流れ図１６２０に示すように、一旦、ソース・コード・ファイルがマークアップされると、ＰＥＲＬなどのスクリプト・プログラミング言語は、プログラミング言語からプログラム構造を判定するためにソース・コード・ファイルを構文解析し、注釈と特別な説明タグとを読み込む（ステップ１６２５）。ＰＥＲＬスクリプトは、ソース・コード・ファイルのプログラム構造を反映する注釈ページを生成する（ステップ１６３０）。さらに、ＰＥＲＬスクリプトは、プログラム構造をナビゲートするための注釈ページ間にリンクを提供し（ステップ１６３５）、フレームワークを通知するためにＪａｖａＳｃｒｉｐｔを使用する（ステップ１６４５）一方で、注釈コメントを組み込み（ステップ１６４０）、関連情報へのリンクとしてソース・コード内の言語キーワードを強調表示する（ステップ１６５０）。したがって、注釈は容易に維持される。図４３を再度参照すると、例示の作者は、ソース・コードと注釈を同時に同じ場所で調整または編集することができる（ステップ１

６１５）。したがって、注釈は自動的に再生成される。

【００９２】

内容フレーム内の注釈ページは、また、関連情報へのリンク、またはある注釈ページから別の注釈ページへの１つまたは複数のリンクを含むことができる。このような柔軟性は、ＰＥＲＬスクリプトがプログラム構造を決定し、プログラムの別の部分への参照をリンクとしてフォーマットするので、上記の注釈ページの自動生成によるものである。例えば、ユーザがｑｕｅｒｙｄｂ．ｈｔｍｌファイル１５２５内の「ｓｐｏｒｔＨｅａｄｅｒ．ｔａｇ」リンクを選択するとき、内容フレーム１３０５は、図４６に示すように、ｓｐｏｒｔｓＨｅａｄｅｒ．ｔａｇソース・コード注釈ページ１５４０を表示し、ＴＯＣフレーム１３１５は対応するサブトピック・リンク１７００を自動的に強調表示する。

【００９３】

内容フレーム注釈ページの選択可能なリンクは、バックグラウンド参考資料に対応する。例えば、「ＭＩＶＡＲ」リンクが選択されるとき、図４７に示すように、ＭＩＶＡＲタグに関する情報を表示する別個のブラウザ１７０５が作成される。この場合、ＰＥＲＬスクリプトは、ソース・コード内のキーワードを発見し、それらを参考文書へのリンクとしてフォーマットする。ユーザが内容ウィンドウ内の「ＷＥＢ ＨＯＭＥ」リンクを選択すると、図４８に示すように、用語集を含むブラウザ１７１０が作成される。

【００９４】

再度図４０を参照すると、ユーザは、図４９に示すように、内容フレーム１３０５内の注釈ページに対応するソース・ファイルをブラウザ・ウィンドウ１７１５内の平文テキストとして開くために、トップ・フレーム１３１０内の編集ボタン１５６２を選択することができる。この場合、ＰＥＲＬスクリプトはソース・ファイルから注釈コメントを自動的に除去し、その結果、ユーザは長文説明が挿入されていないソース・ファイルを編集することができる。次いでユーザは、このウィンドウと、ユーザ固有のプログラムを表示する別のテキスト・ウィンドウとの間でテキストをカット・アンド・ペーストすることができる。

【００９５】

再度図３８を参照すると、Ｊａｖａ（Ｒ）例示アプレットまたはアプリケーションを起動したり、Ｊａｖａ（Ｒ）アプレットまたは他の埋め込まれたプログラム・オブジェクトを含んでいるＨＴＭＬページをダウンロードしたり、出力をブラウザ・ウィンドウに宛先変更して非対話型プログラムを実行するように、Ｗｅｂサーバに共通ゲートウェイ・インターフェース（ＣＧＩ）要求を送信するために、ユーザはフレームワーク・アプレットを使用して例示を実行することができる。Ｊａｖａ（Ｒ）例示の場合、Ｗｅｂブラウザに注釈ページを表示するよう要求するために、実行中の例示はＪａｖａ（Ｒ）アプリケーション・プログラミング・インターフェース（ＡＰＩ）を呼び出す。ＣＧＩプログラムの場合、Ｗｅｂブラウザに注釈ページを表示するよう要求するために、例示プログラムは、ＪａｖａＳｃｒｉｐｔを含んでいるＨＴＭＬページを戻す。したがって、例示は、サーバ上、またはＷｅｂブラウザの制限付きサンドボックス内で実行される。ユーザが例示をローカルに実行する許可を要求するためにフレームワークがブラウザのセキュリティ機能を使用する場合、例示は、ローカル・クライアント上で実行することができる。

【００９６】

図３８において、ユーザはＲｕｎボタン１５７０を選択することによって例示を実行することができる。図４５の流れ図１６５５を参照すると、Ｒｕｎボタン１５７０を選択すると、図５０に示すように、上記の任意の方法を使用して、ブラウザ・ウィンドウ１８００内で例示が起動する（ステップ１６６０）。ＴＯＣフレーム１３１５は、例示を実施するソース・コードに対する現行注釈ページを強調表示することによって、同時かつ自動的に実行中の例示と同期する。ＰＥＲＬスクリプトは、注釈ページの自動生成によく似たように、階層構造を示すリンクに関してＨＴＭＬページを構文解析することによってＴＯＣを自動的に生成する。

【００９７】

さらに、例示は、現行注釈ページ、この例では上記のｑｕｅｒｙｄｂ．ｈｔｍｌファイル１５２５を表示することによって、内容フレーム１３０５を実行中の例示と同時かつ自動的に同期する。次いでユーザは、例示が実行されている間に、注釈付きのソース・コードを表示させるか、それともそのソース・ファイルを編集することができ、上記のようにキーワードからそのキーワードに対する参考文書にジャンプすることができる。

【００９８】

ユーザがブラウザ・ウィンドウ１８００内の例示と対話するとき（ステップ１６６５）、（例えば、顧客番号を入力し、ブラウザ・ウィンドウ１８００内の「Ｓｕｂｍｉｔ」ボタンを選択することによって）、ユーザ・インターフェースは、図５１に示すようにブラウザ・ウィンドウ１８００内で変化する（例えば、入力された顧客番号に対応するカスタマー・レポート）（ステップ１６７０）。ユーザ・インターフェースは、実行中の例示で使用される次の注釈ページｃｕｓｔ＿ｄｂ．ｈｔｍｌファイル１５８６に対応して変化する。実行中の例示は、注釈を表示すべきプログラム・ユニットの例示、ファイル、クラスおよび関数の名称を有する注釈要求モジュールを自動的かつ同時に呼び出す（ステップ１６７５）。注釈要求モジュールは、その要求を注釈ページにマップし、内容フレーム内に注釈ページを表示するようブラウザに伝える（ステップ１６８０）。例えば、図５１では、次の注釈ページはｃｕｓｔ＿ｄｂ．ｈｔｍｌ１５８６である。Ｃｕｓｔ＿ｄｂ．ｈｔｍｌサブトピック１５８５は、ＴＯＣフレーム１３１５で強調表示され、図５１に示すように、内容フレーム１３０５はｃｕｓｔ＿ｄｂ．ｈｔｍｌ注釈ページ１５８６を表示する。埋め込まれた注釈マークアップ１８１０は、図５２に表示されるｃｕｔ＿ｄｂ．ｈｔｍｌソース・コード・ファイル内に示される。埋め込まれたマークアップ１８１０は、ソース・コード・ファイルから注釈ページを生成するときに使用される。

【００９９】

再度図３６を参照すると、例示を選択することに加え、ユーザは、アイコンまたはリンク１４５０を内容フレーム１３０５内に文書化する方法、またはサブトピック・リンク１４５５をＴＯＣフレーム１３１５内に文書化する方法を選択することもできる。この場合、図５３に示すように、文書化する方法１９００が内容フレーム１３０５に表示される。図を見やすくするために、完全な文書１９００は図５４に示す。同期において、ＴＯＣフレーム１３１５は、文書化する方法１９００に対応する表示されたサブトピック１９０２を強調表示する。

【０１００】

さらに同期において、選択された文書化する方法１９００を反映するために、トップ・フレーム１３１０は動的に変化する。例えば、トップ・フレームには、現時点でアップ・ボタン１９０５、第１のページ・ボタン１９１０、前ページ・ボタン１９１５、次ページ・ボタン１９２０、最終ページ・ボタン１９２５、およびリスト・ボタン１９３０が表示されている。アップ・ボタン１９０５は、選択されると、文書化する方法１９００を列挙するカテゴリまたはサブトピックにジャンプする。例えば、サブトピック「Ｅｘｔｅｎｄｉｎｇ ａ Ｄａｔａｂａｓｅ Ｓｅｒｖｅｒ」は、「Ｃｒｅａｔｅ Ｗｅｂ ａｐｐｌｉｃａｔｉｏｎ」の文書化する方法１９００を列挙し、アップ・ボタン１９０５を選択すると、図３６に示すように、ブラウザは「Ｅｘｔｅｎｄｉｎｇ ａ Ｄａｔａｂａｓｅ Ｓｅｒｖｅｒ」サブトピックに関するすべての情報を表示する。一般に、トップ・フレームは、新しい内容のページが表示された旨をフレームワークに通知するＪａｖａＳｃｒｉｐｔ呼び出しで指定される内容の種類と一致するように変化する。

【０１０１】

文書１９００は他の文書またはページへのポインタを含んでおり、このポインタは複数の方法の１つによってアクセスされる。ポインタは、例えば異なる色、フォント、またはスタイルによって示される内容フレーム内のリンク上をクリックすることによって、内容フレーム１３０５から直接的に選択することができる。例えば、ユーザは図５４に示すように内容フレーム１３０５内のＡｐｐｌｉｃａｔｉｏｎ Ｐａｇｅリンク１９３０またはＰｒｅｐａｒｅ Ｄａｔａｂａｓｅリンク１９３５上をクリックすることができる。Ｃｒｅ

ａｔｅ Ｗｅｂ ａｐｐｌｉｃａｔｉｏｎ文書１９０２のサブトピックを選択することによって、ＴＯＣフレーム１３１５からポインタを選択することができる。例えば、サブトピックＰｒｅｐａｒｅ Ｄａｔａｂａｓｅ１９４０はリンクＰｒｅｐａｒｅ Ｄａｔａｂａｓｅ１９３５に対応し、それらのポインタのどちらを選択してもユーザは同じ文書に移動される。ポインタを選択する際、そのポインタに対応する文書またはページが次いでユーザに表示される。

【０１０２】

さらに図５５を参照すると、ユーザがトップ・フレーム内のリスト・ボタン１９３０を選択すると、ローカル目次ウィンドウ１３２０が表示される。例えば、ユーザが図５３のリスト・ボタン１９３０を選択すると、図５５に示すようにウィンドウ１３２０が作成される。文書をナビゲートする従来型モデルは階層型目次付きである。ＴＯＣモデルは、常時ユーザを方向付けるが、内容を制限する。これは階層には適合しない場合がある。文書をナビゲートするより新しいモデルがハイパーテキストＷｅｂである。このＷｅｂモデルは文書内のいかなるトピックでも、同じトピック上で拡張する、より詳細な文書にリンクする。このＷｅｂモデルは人為的な制限を除去するが、あっという間にユーザを混乱させる。

【０１０３】

これとは対照的に、Ｅｘａｍｐｌｅ（Ｒ）のＷｅｂバージョンは、リンクの非構造化Ｗｅｂページのローカル・コーナーで構造化された階層図を提供するローカル目次モデルを使用する。このモデルでは、文書内のいくつかのページは、ローカルＴＯＣへのルート・ページである。例えば、Ｃｒｅａｔｅ Ｗｅｂ ａｐｐｌｉｃａｔｉｏｎ文書１９００に対応するページは、ローカルＴＯＣへのルート・ページである。ユーザが、ルート・ページ（固有である）などのローカルＴＯＣ固有の任意のページにナビゲートするとき、フレームワーク・アプレットはそのルート・ページに対するローカルＴＯＣを読み込む。図５６に示すように、ユーザがローカルＴＯＣ１３２０のＩｎｔｒｏｄｕｃｔｉｏｎページにナビゲートするとき、フレームワーク・アプレットはそのルート・ページに対するローカルＴＯＣを読み込み、対応する情報をＴＯＣフレーム１３１５および内容フレーム１３０５に表示する。ローカルＴＯＣの各ページはローカルＴＯＣ内では固有である。ユーザはローカルＴＯＣ内のページを連続的に一歩ずつ進むことができ、ローカルＴＯＣの順番で複数のページを参照したり、選択した現行ページでローカルＴＯＣ階層を表示させたりすることができる。したがって、ユーザは、ローカル・トピック内で方向付けられる。

【０１０４】

ローカルＴＯＣは、ローカルＴＯＣ内のページのリンクを制約しない。すなわち、ローカルＴＯＣ内のページは、ローカルＴＯＣ１３２０には存在しないページへのリンクを有することができる。例えば、ユーザがＩｎｔｒｏｄｕｃｔｉｏｎ文書内のＩｎｆｏｒｍｉｘ Ｗｅｂ Ｉｎｔｅｇｒａｔｉｏｎ Ｏｐｔｉｏｎリンク１９４２を選択するとき、ブラウザは、図５７に示すように、ローカルＴＯＣ１３２０には存在しないＩｎｆｏｒｍｉｘ Ｗｅｂ Ｉｎｔｅｇｒａｔｉｏｎ Ｏｐｔｉｏｎページ１９４５を開く。

【０１０５】

さらに、ページは複数のローカルＴＯＣ内に表示させることができる。例えば、図５６のローカルＴＯＣ内にある「Ｃｒｅａｔｅ ｓｕｂｓｐａｃｅ」という題名のページ１９５０は、別の文書に関する別のローカルＴＯＣ内に表示することができる。ルート・ページ（この例では、「Ｃｒｅａｔｅ ＷＥＢ ａｐｐｌｉｃａｔｉｏｎ」ページ１９００）は単一ローカルＴＯＣに制約される。

【０１０６】

ユーザは、ローカルＴＯＣ内の関心のあるトピックを選択することによって、ローカルＴＯＣウィンドウから直接的にローカルＴＯＣ全体をナビゲートすることができる。さらに、ユーザは、第１のページ・ボタン１９１０、前ページ・ボタン１９１５、次ページ・ボタン１９２０、または最終ページ・ボタン１９２５を使用して、トップ・フレーム１３１０を介してローカルＴＯＣ全体をナビゲートすることができる。これらのボタンは、基本

的に、ローカルＴＯＣ中を移動することによって、ユーザのために文書内のページをめくる。例えば、Ｃｒｅａｔｅ Ｗｅｂ ａｐｐｌｉｃａｔｉｏｎ文書１９００内にあるときにユーザが次ページ・ボタン１９２０を選択するとき、ブラウザは、Ｃｒｅａｔｅ Ｗｅｂ ａｐｐｌｉｃａｔｉｏｎ文書内の次の文書を表示する。これは、図５８に示すように「Ｔａｂｌｅ ｏｆ Ｃｏｎｔｅｎｔｓ」文書１９５５である。

【０１０７】

図５９を参照すると、ローカルＴＯＣ２００５が構築される方法のモデル２０００が示される。ローカルＴＯＣの構築は、ページ２０１０の非構築（またはランダムな）Ｗｅｂ上に順序付き表示を強制し、このようにして、Ｗｅｂのページ２０１０を表示することを容易にする。

【０１０８】

他の実施形態は、頭記の特許請求の範囲に含まれる。

【０１０９】

Ｗｅｂベースの説明システムは、Ｊａｖａ（Ｒ），ＶｉｓｕａｌＢａｓｉｃ（Ｒ）、Ｃ、Ｃ＋＋、ＨＴＭＬ、Ｐｅｒｌ、ＪａｖａＳｃｒｉｐｔ、ＳＱＬ、Ｉｎｆｏｒｍｉｘ Ｓｔｏｒｅｄ Ｐｒｏｃｅｄｕｒｅ Ｌａｎｇｕａｇｅ（ＳＰＬ）、ＥＳＱＬ／Ｃ（Ｅｍｂｅｄｄｅｄ ＳＱＬ ｆｏｒ Ｃ）、ＳＱＬＪ、ＪＳＰ、ＡＳＰ、およびＩｎｆｏｒｍｉｘ Ｗｅｂ ＤａｔａＢｌａｄｅ Ｍｏｄｕｌｅ言語をサポートすることができる。

【図面の簡単な説明】

【図１】

従来技術のコンピュータ・システムのブロック図である。

【図２】

図１のコンピュータ・システムで使用されるグラフィカル・ユーザ・インターフェース内の表示領域を示す。

【図３】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図４】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図５】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図６】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図７】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図８】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図９】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図１０】

従来技術のオンライン・ヘルプおよび文書システムの画面の一場面である。

【図１１】

ブラウザ・アプリケーションの画面の一場面である。

【図１２】

ブラウザ・アプリケーションの表示領域を示す。

【図１３】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションのユーザに使用可能なオプションを示す流れ図である。

【図１４】

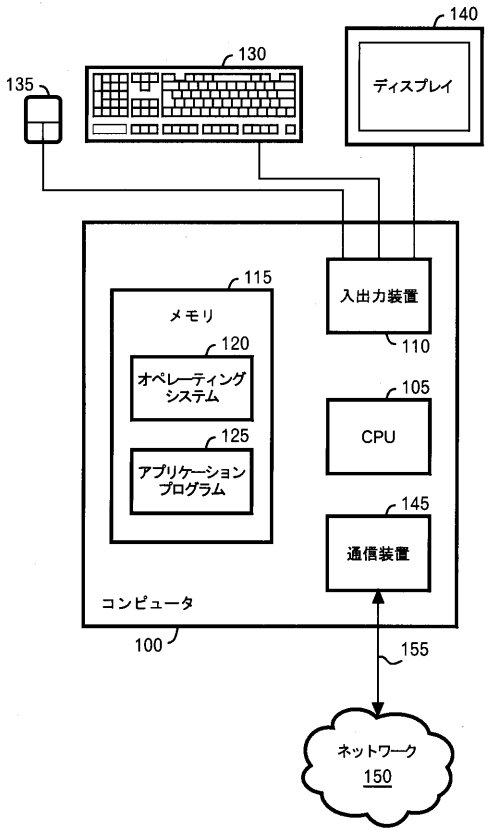
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。

【図１５】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図１６】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図１７】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図１８】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図１９】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２０】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２１】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２２】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２３】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２４】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２５】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２６】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２７】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２８】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図２９】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションおよびＮｅｗＥｒａ（Ｒ）開発環境からの画面の一場面である。
【図３０】
ＮｅｗＥｒａ（Ｒ）アーキテクチャのブロック図である。
【図３１】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅアプリケーションがどのように構築されるかを示すブロック図である。

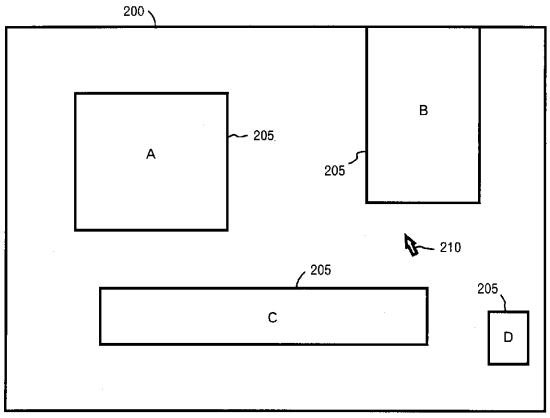
【図３２】
ＮｅｗＥｒａ（Ｒ）ソース・コードのサンプルである。
【図３３】
Ｗｅｂベースの指示システムにおける表示領域である。
【図３４】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図３５】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図３６】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図３７】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図３８】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図３９】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションの表示領域内の内容を示す図である。
【図４０】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図４１】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションの表示領域内の内容を示す図である。
【図４２】
注釈を埋め込んだソース・コード・ファイルである。
【図４３】
注釈ページの作成者がとるステップを示す流れ図である。
【図４４】
注釈ページでリンクを自動的に生成するためのスクリプト・プログラムがとるステップを示す流れ図である。
【図４５】
例示的プログラムがとるステップを示す流れ図である。
【図４６】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図４７】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図４８】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図４９】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５０】

Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５１】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５２】
注釈を埋め込んだソース・コード・ファイルである。
【図５３】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５４】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションの表示領域内の内容を示す図である。
【図５５】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５６】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５７】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５８】
Ｉｎｆｏｒｍｉｘ（Ｒ）ｂｙＥｘａｍｐｌｅのＷｅｂベースのアプリケーションからの画面の一場面である。
【図５９】
Ｗｅｂベースのアプリケーションで使用されるローカル目次モデルを示すブロック図である。

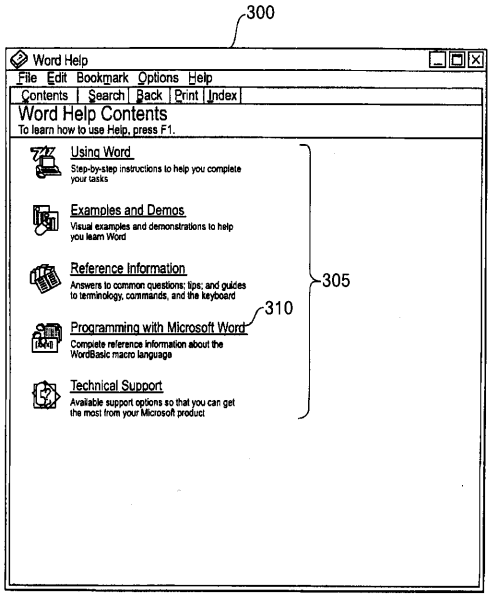
【図１】


140
ディスプレイ
130
135
115
メモリ
120
オペレーティングシステム
125
アプリケーションプログラム
入出力装置
110
105
CPU
145
通信装置
コンピュータ
100
155
ネットワーク
150

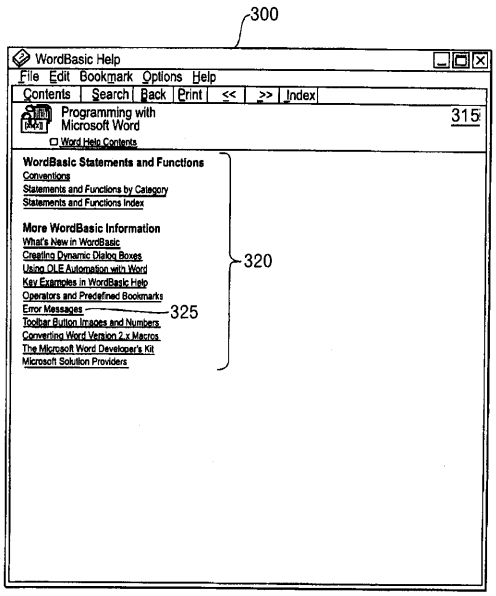
【図２】


200
A
205
B
205
210
205
C
205
D

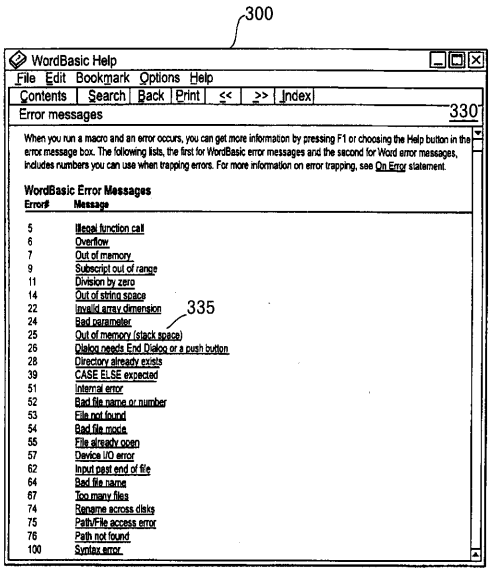
【図３】


300
Word Help
File Edit Bookmark Options Help
Contents Search Back Print Index
Word Help Contents
To learn how to use Help, press F1.
Using Word
Step-by-step instructions to help you complete your tasks
Examples and Demos
Visual examples and demonstrations to help you learn Word
Reference Information
Answers to common questions; tips; and guides to terminology, commands, and the keyboard
Programming with Microsoft Word
Complete reference information about the WordBasic macro language
Technical Support
Available support options so that you can get the most from your Microsoft product
305
310

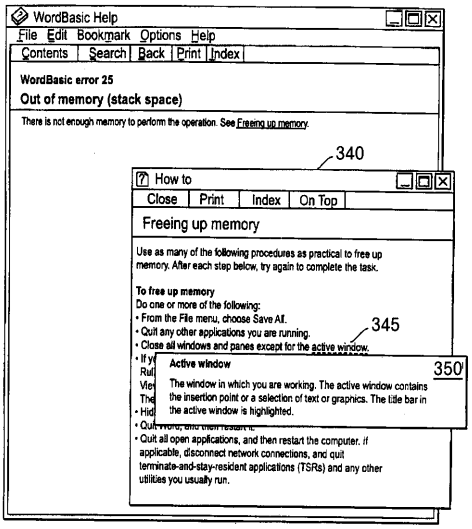
【図４】


300
WordBasic Help
File Edit Bookmark Options Help
Contents Search Back Print << >> Index
Programming with Microsoft Word
315
Word Help Contents
WordBasic Statements and Functions
Conventions
Statements and Functions by Category
Statements and Functions Index
More WordBasic Information
What's New in WordBasic
Creating Dynamic Dialog Boxes
Using OLE Automation with Word
Key Examples in WordBasic Help
Operators and Predefined Bookmarks
Error Messages
Toolbar Button Images and Numbers
Converting Word Version 2.x Macros
The Microsoft Word Developer's Kit
Microsoft Solution Providers
320
325

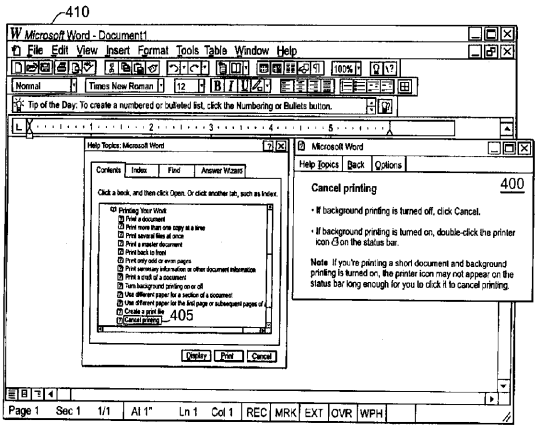
【図５】


300
WordBasic Help
File Edit Bookmark Options Help
Contents Search Back Print << >> Index
Error messages
330
When you run a macro and an error occurs, you can get more information by pressing F1 or choosing the Help button in the error message box. The following lists, the first for WordBasic error messages and the second for Word error messages, includes numbers you can use when trapping errors. For more information on error trapping, see On Error statement.
WordBasic Error Messages
Error# Message
5 Illegal function call
6 Overflow
7 Out of memory
9 Subscript out of range
11 Division by zero
14 Out of string space
22 Invalid array dimension
335
24 Bad parameter
25 Out of memory (stack space)
26 Dialog needs End Dialog or a push button
28 Directory already exists
39 CASE ELSE expected
51 Internal error
52 Bad file name or number
53 File not found
54 Bad file mode
55 File already open
57 Device I/O error
62 Input past end of file
64 Bad file name
67 Too many files
74 Rename across disks
75 Path/File access error
76 Path not found
100 Syntax error

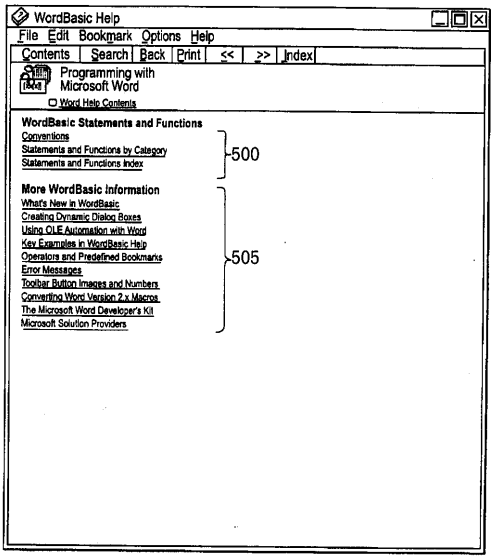
【図６】


WordBasic Help
File Edit Bookmark Options Help
Contents Search Back Print Index
WordBasic error 25
Out of memory (stack space)
There is not enough memory to perform the operation. See Freeing up memory.
340
How to
Close Print Index On Top
Freeing up memory
Use as many of the following procedures as practical to free up memory. After each step below, try again to complete the task.
To free up memory
Do one or more of the following:
• From the File menu, choose Save All.
• Quit any other applications you are running.
345
• Close all windows and panes except for the active window.
Active window
350
The window in which you are working. The active window contains the insertion point or a selection of text or graphics. The title bar in the active window is highlighted.
• Quit all open applications, and then restart the computer. If applicable, disconnect network connections, and quit terminate-and-stay-resident applications (TSRs) and any other utilities you usually run.

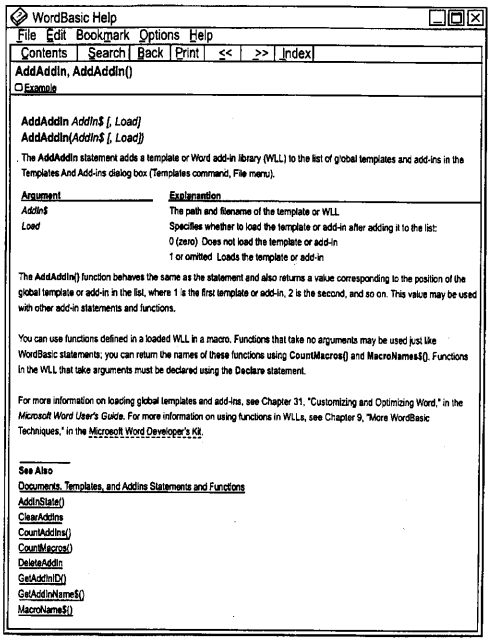
【図７】


410
Microsoft Word - Document1
File Edit View Insert Format Tools Table Window Help
Tip of the Day: To create a numbered or bulleted list, click the Numbering or Bulleted button.
Help Topics: Microsoft Word
Contents Index Find Answer Wizard
405
Microsoft Word
Help Topics Back Options
Cancel printing
400
• If background printing is turned off, click Cancel.
• If background printing is turned on, double-click the printer icon on the status bar.
Note If you're printing a short document and background printing is turned on, the printer icon may not appear on the status bar long enough for you to click it to cancel printing.
Display Print Cancel
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC MRK EXT OVR WPH

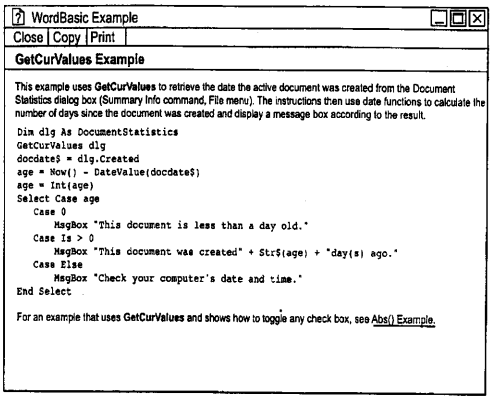
【図８】


WordBasic Help
File Edit Bookmark Options Help
Contents Search Back Print << >> Index
Programming with Microsoft Word
Word Help Contents
WordBasic Statements and Functions
Conventions
Statements and Functions by Category
Statements and Functions Index
500
More WordBasic Information
What's New in WordBasic
Creating Dynamic Dialog Boxes
Using OLE Automation with Word
Key Examples in WordBasic Help
Operators and Predefined Bookmarks
Error Messages
Toolbar Button Images and Numbers
Converting Word Version 2.x Macros
The Microsoft Word Developer's Kit
Microsoft Solution Providers
505

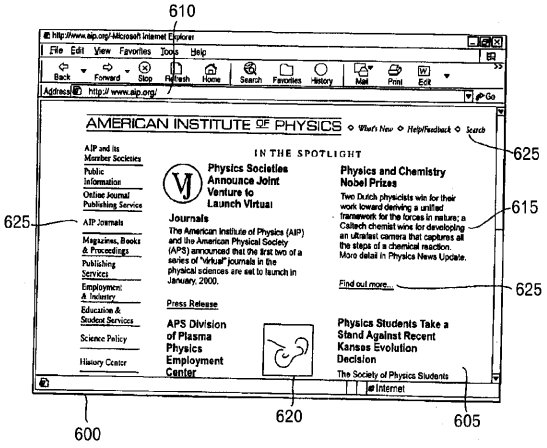
【図９】



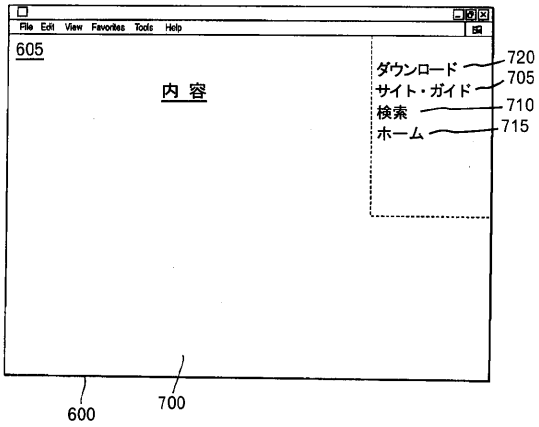
【図１０】



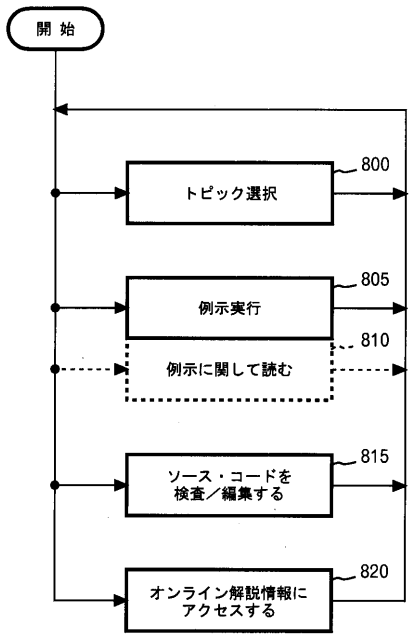
【図１１】



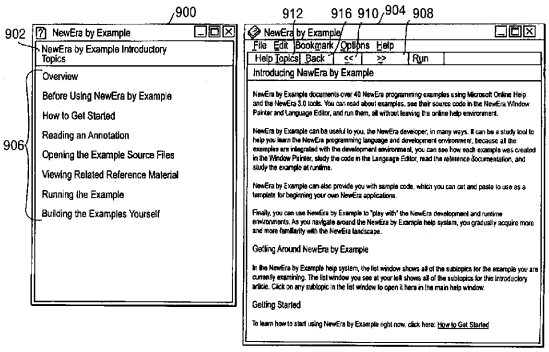
【図１２】



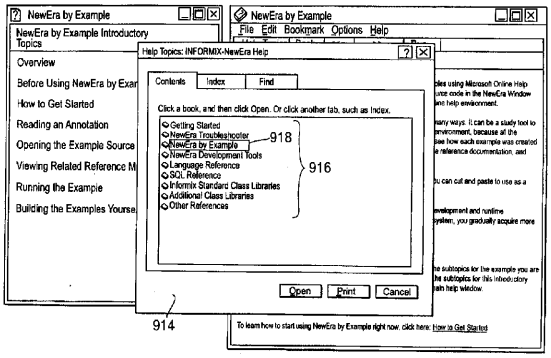
【図１３】



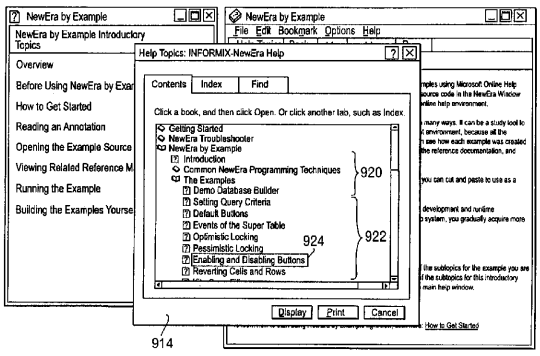
【図１４】



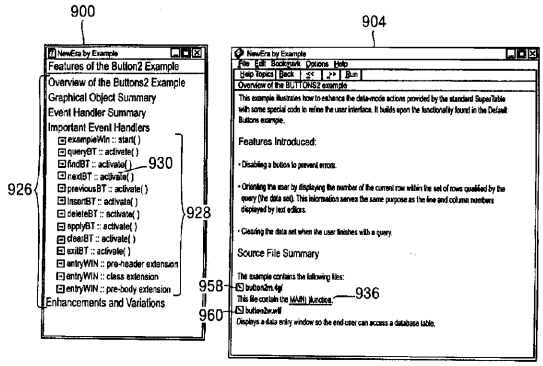
【図１５】



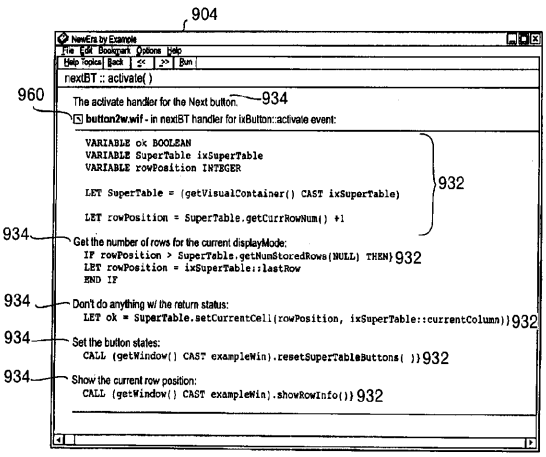
【図１６】



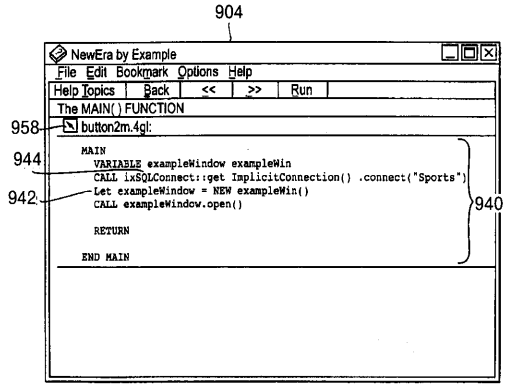
【図１７】



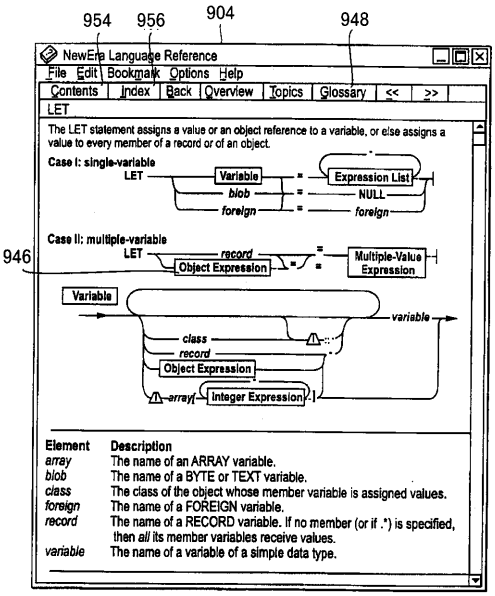
【図１８】



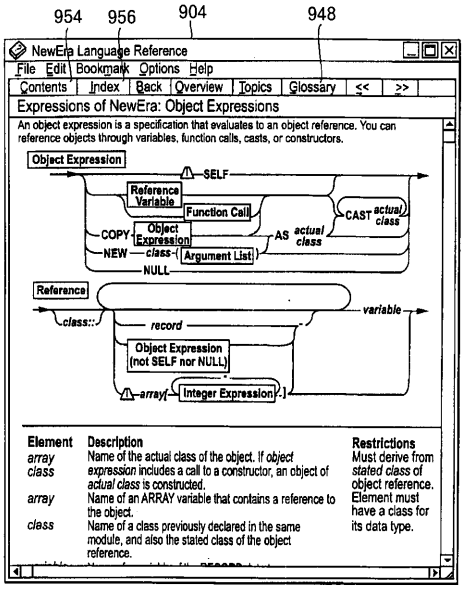
【図１９】



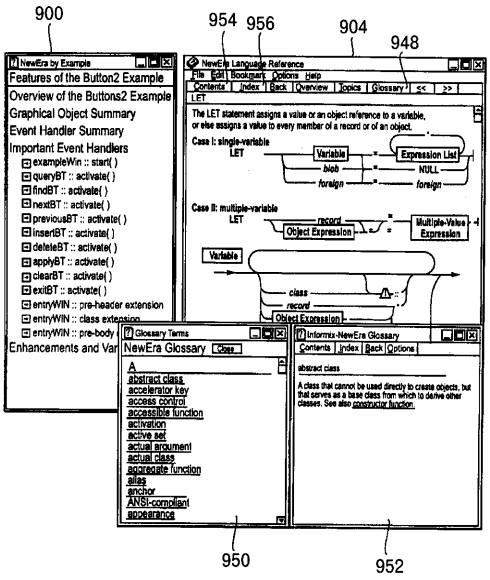
【図２０】



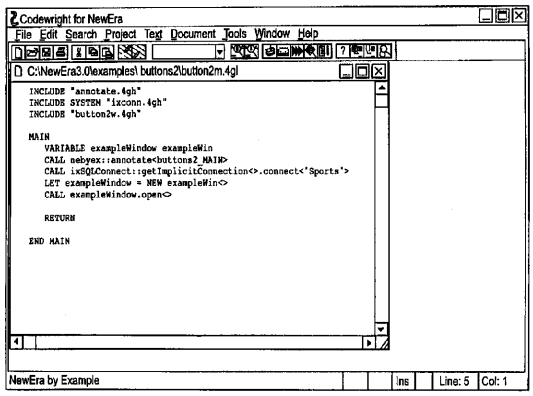
【図２１】



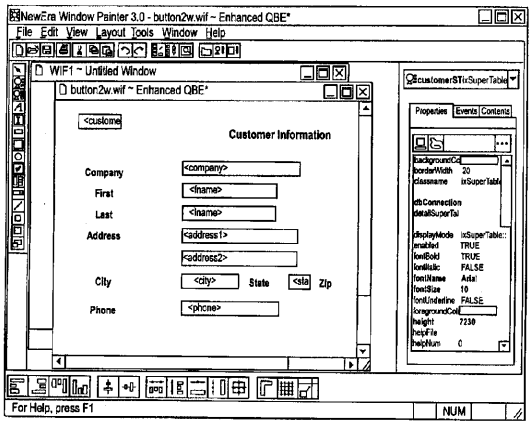
【図２２】



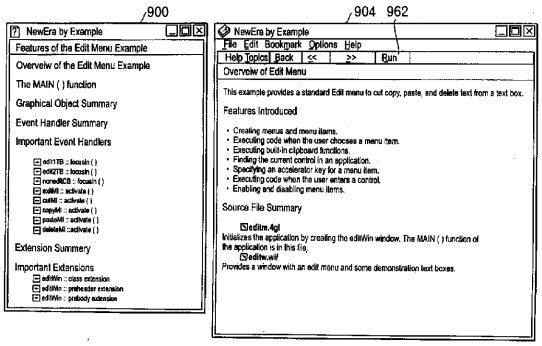
【図２３】



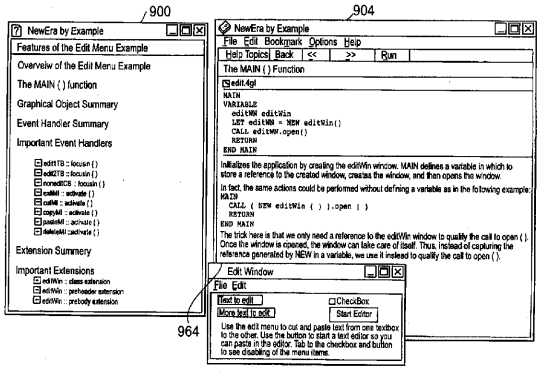
【図２４】



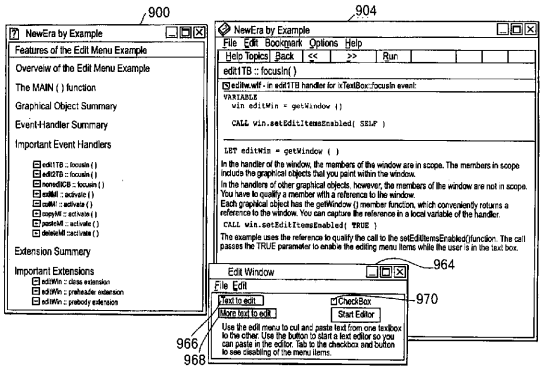
【図２５】



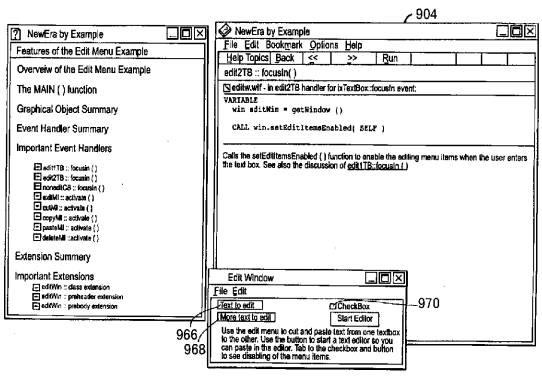
【図２６】



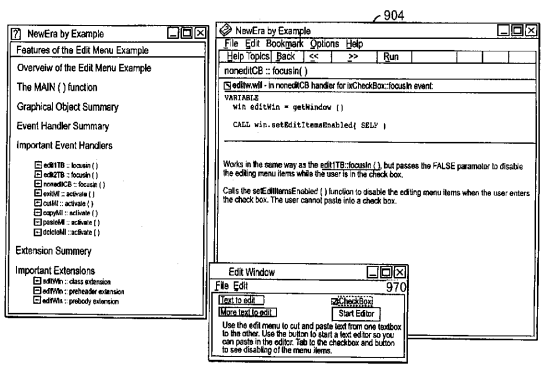
【図２７】



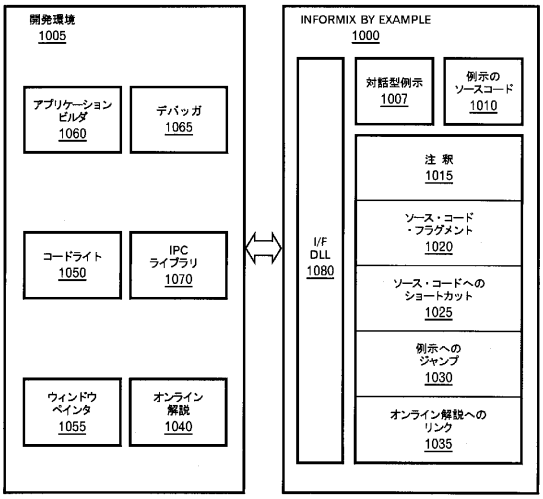
【図２８】



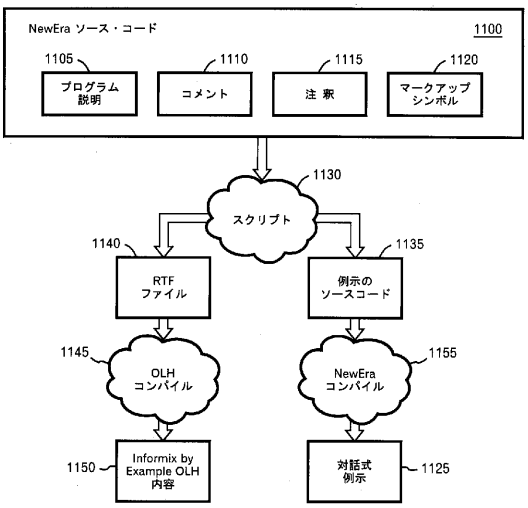
【図２９】



【図３０】



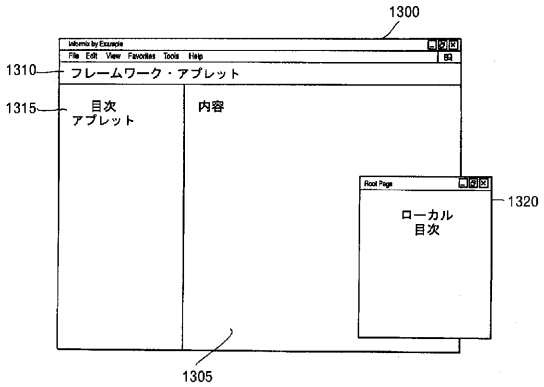
【図３１】



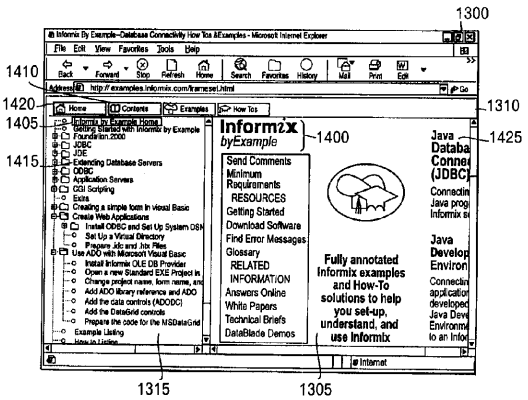
【図３２】

```
      FUNCTION driveStockRpt( destType SMALLINT, destName CHAR(*) )
1200  RETURNING VOID
      {.normal
      Since objects, in particular ixRow objects, cannot be passed
      as arguments to the report formatter, rows of fetched data will
      be unpacked into a record that matches the data types and lengths
      of elements in the fetched rows.
      }
      VARIABLE
           stockRec RECORD
                mn CHAR(15),     -- manufact.manu_name
                sn SMALLINT,     -- stock.stock_num
                sd CHAR(15),     -- stock.description
                sp MONEY(6,2),   -- stock.unit_price
                su CHAR(4)       -- stock.unit
                END RECORD

           stockStmt ixSQLStmt,
           stmtString CHAR(*),
           stockRow ixRow,

           errorCode INTEGER,
           logFile ixErrorLog
1205
      {.normal
      Use the implicit connection object to create an SQL statement
      object. The connection object must already be connected to a
      database.
      Checking the status of the prepare( ) call will confirm this.
1210  }
      {.[edit stmt}
      LET stockStmt =
      ixSQLConnect::getImplicitConnection().createStmtObject()
      {.}file stmt}
1215
```

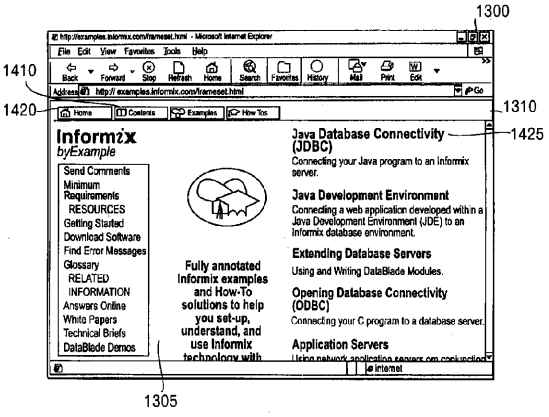
【図３３】



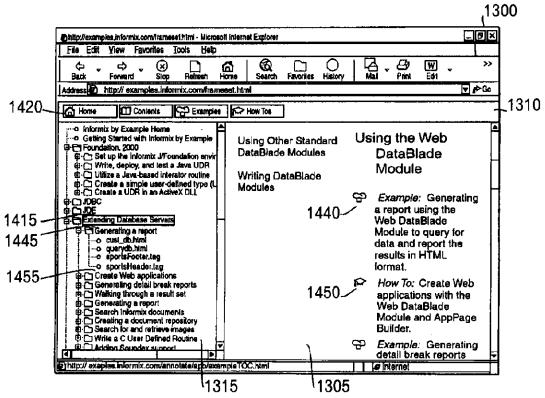
【図３４】



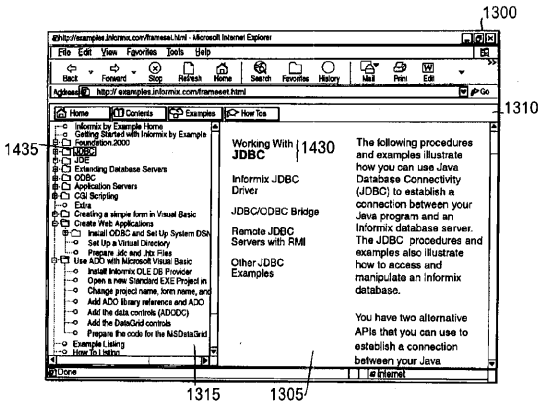
【図３５】



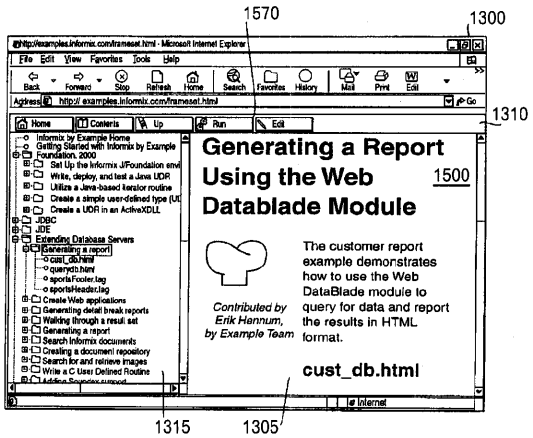
【図３６】



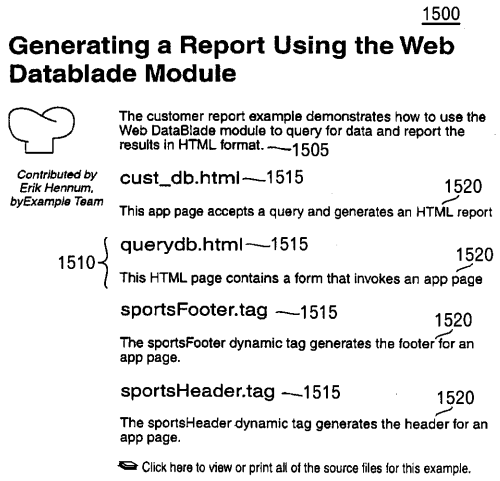
【図３７】



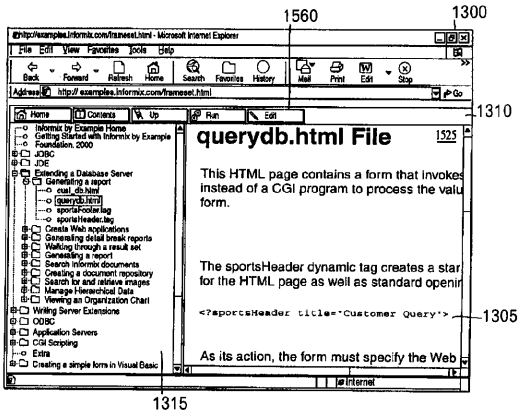
【図３８】



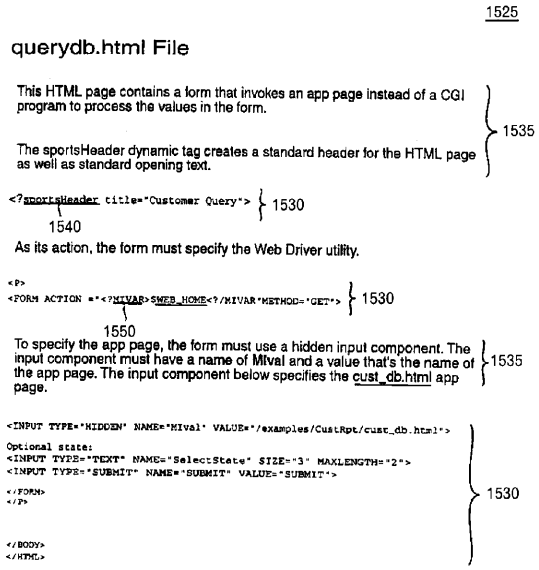
【図３９】



【図４０】



【図４１】



【図４２】

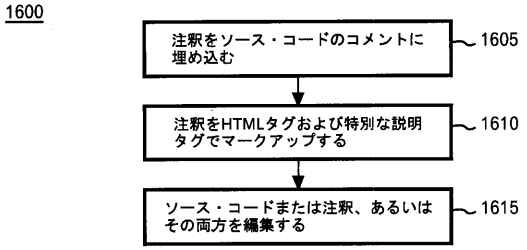
1560

```
<!--<ibyx>
<intro>
<p><abstract>This HTML page contains a form that invokes
an app page</abstract> instead of a CGI program to process
the values in the form.
</p>
</intro>
</ibyx> -- >

<!-- <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> -->
<?sportsHeader title="Customer Query">

<!-- <ibyx>
<p> As its action, the form must specify the Web Driver utility.
</p>
</ibyx>-->
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR>" METHOD="GET">

<!-- <ibyx>
<p> To specify the app page, the form must use a hidden input component.
The input component must have a name of <strong>MIval</strong> and
a value that's the name of the app page. The input component below
specifies the <a href="cust_db.html">cust_db.html</a> app page.
</p>
</ibyx> -->
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPE="TEXT" NAME="selectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">

</FORM>
</P>

<?annotate>

</BODY>
</HTML>
```

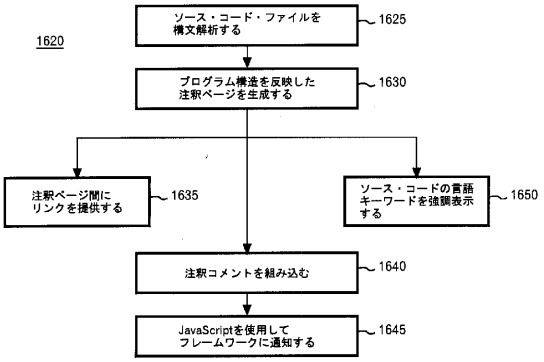
1535

1535

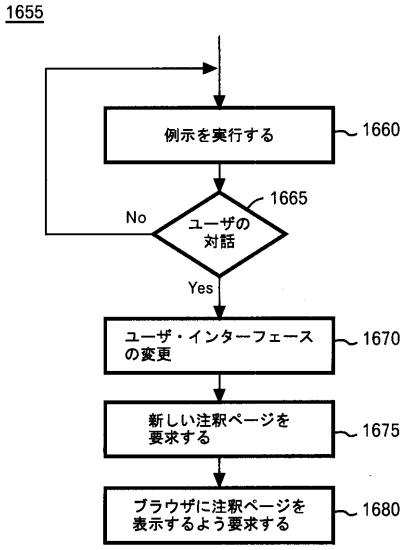
【図４３】



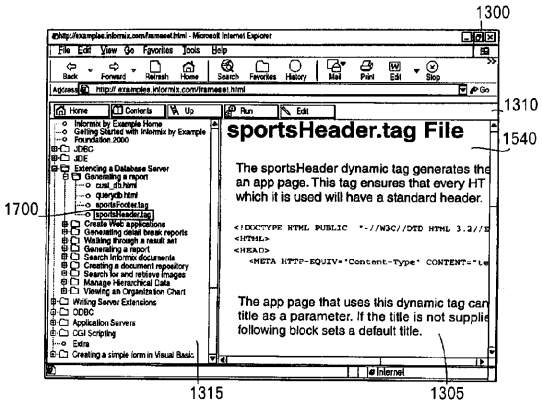
【図４４】



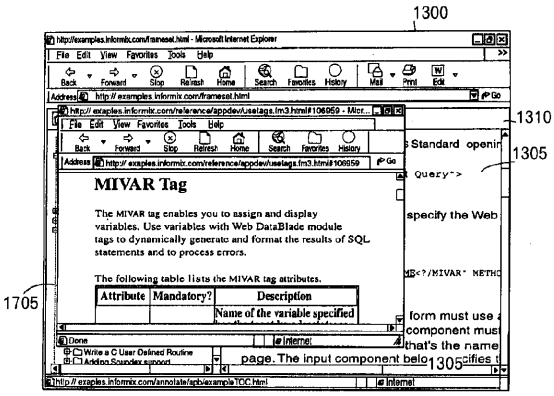
【図４５】



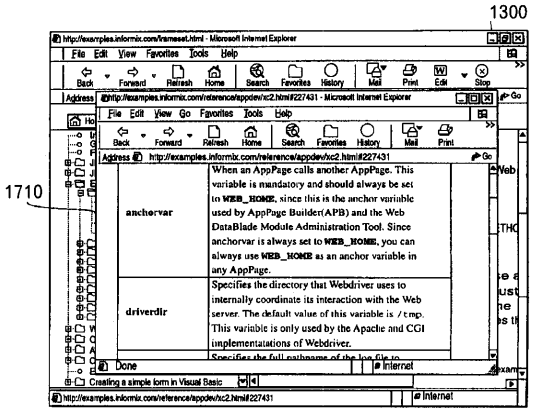
【図４６】



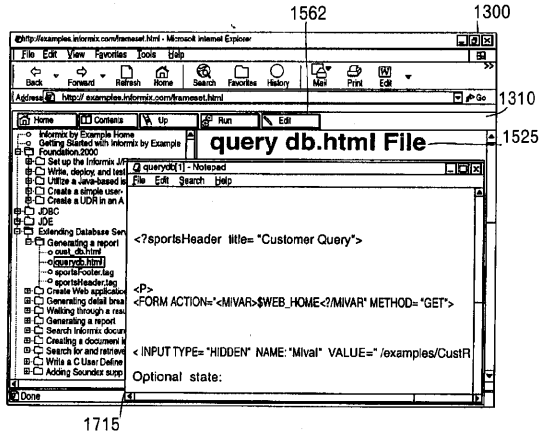
【図４７】



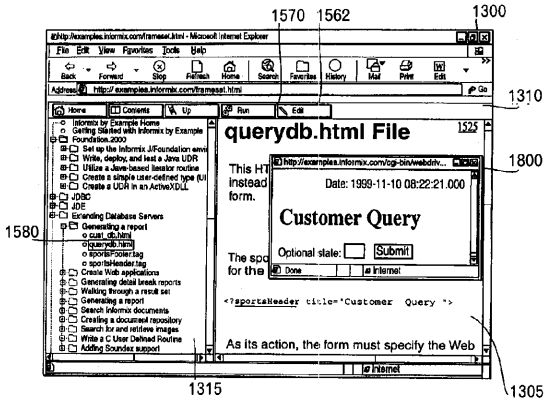
【図４８】



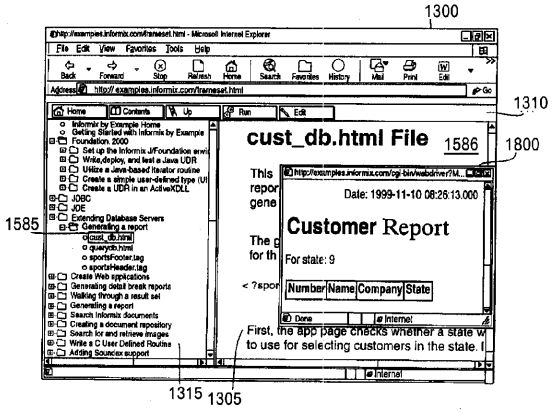
【図４９】



【図５０】



【図５１】

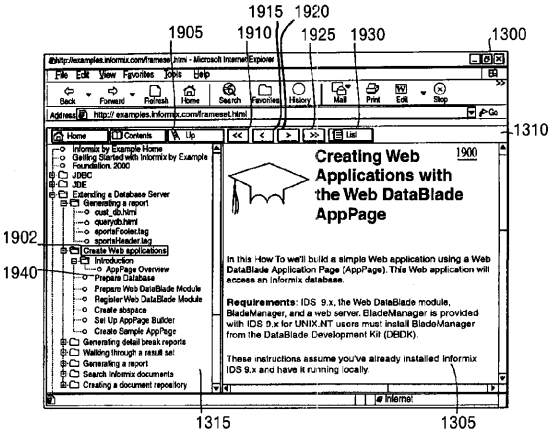


【図５２】

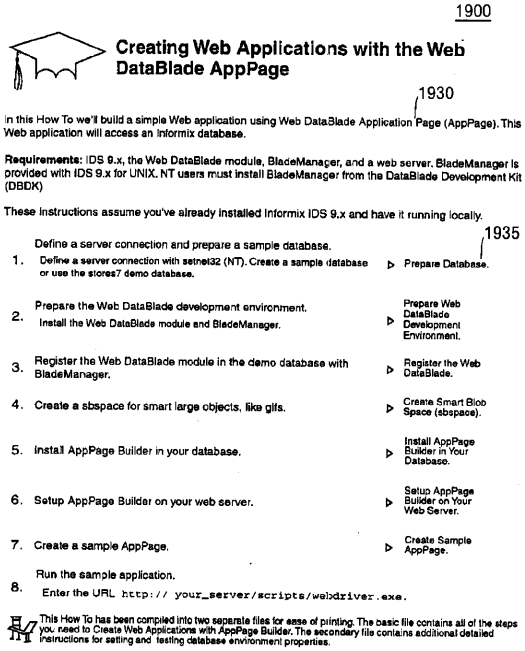
1810

```
<! - - <ibyx>
<intro>
<p> <abstract>This app page accepts a query and generates an HTML
report</abstract > in response.
The app page uses dynamic tags to generate the header and footer for the
HTML report.
</p>
</intro>
</ibyx> - - >
<! - - <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> - ->
<?sportsHeader title='Customer Report'>

<! - - <ibyx>
<p> First, the app page checks whether a state was specified to use for
selecting customers in the state. If so, the block generates a
paragraph to identify the state.
</p>
</ibyx> - - >
<?MIVAR NAME=$WHERE_STR><?/MIVAR>
<?MIBLOCK COND='$(AND,$(NST.$selectState) ,  $ (<,0,$ (STRLEN, $selectState))]'>
        <?MIVAR NAME= $WHERE_STR>WHERE state= ' $selectState'<?/MIVAR>
        <?MIVAR><P>For state: $selectState</P><?/MIVAR>
<?/MIBLOCK>

<! - - <ibyx>
<p>Next,  the app page starts the table that will contain the data.
</p>
</ibyx> - ->
        <P><TABLE BORDER='1'>
                <TR>
                <TH>Number</TH><TH>Name</TH><TH>Company</TH><TH>State</TH>
                </TR>

<! - - <ibyx>
<p> The MISQL block queries for customers, optionally selecting only customers
from the specified state. Because the contents of the block are generated
for every row of data, a new table row describes each customer.

The &amp ;nbsp; HTML entity is a non-breaking space. By putting a non-breaking
space in each column, we force the Web Browser to display the column even
if the value is null.
</p>
</ibyx> - ->
<?MISQL SQL= "SELECT customer_num, fname, lname, company, state FROM customer $WHERE_STR; ">
                <TR>
                <TD> $1  </TD><TD>$2  $3</TD><TD>$4 </TD><TD>$5 </TD>
                </TR>
<?/MISQL>

        </TABLE></P>

< ! - -  <ibyx>
<p> The sportsFooter dynamic tag creates a standard footer
for the HTML page.
</p>
</ibyx> - ->
<?sportsFooter>
```

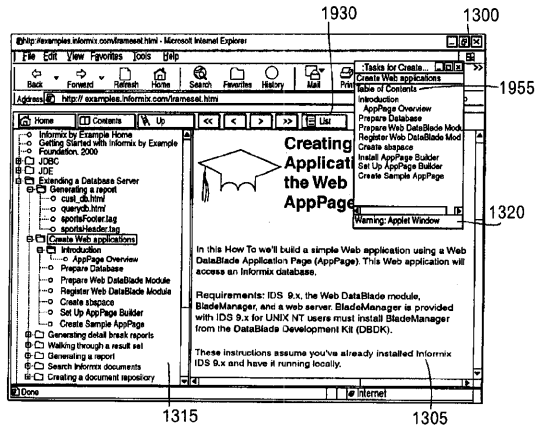
【図５３】



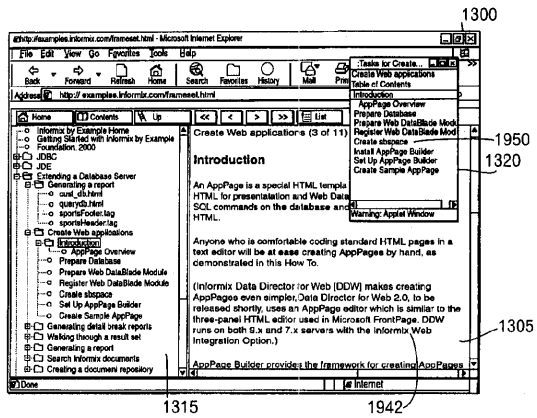
【図５４】



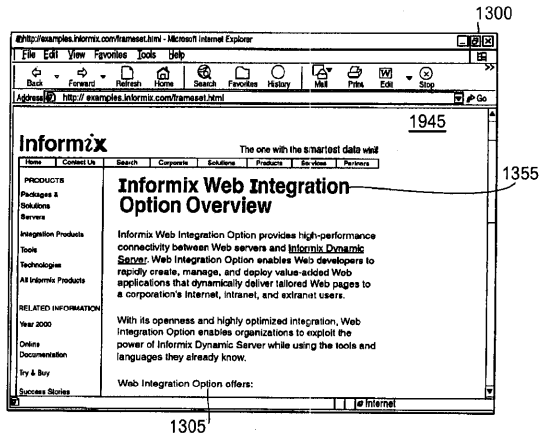
【図５５】



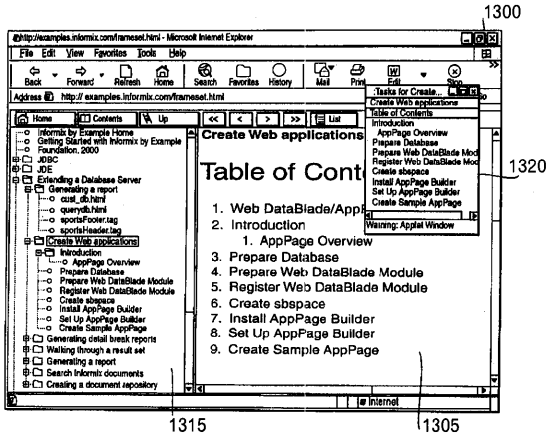
【図５６】



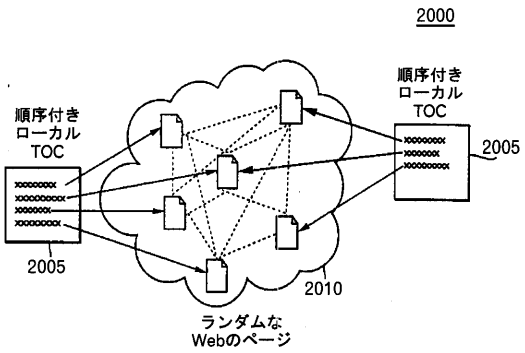
【図５７】


1300
1945
1355
Informix
The one with the smartest data wins!
Informix Web Integration Option Overview
Informix Web Integration Option provides high-performance connectivity between Web servers and Informix Dynamic Server. Web Integration Option enables Web developers to rapidly create, manage, and deploy value-added Web applications that dynamically deliver tailored Web pages to a corporation's Internet, intranet, and extranet users.
With its openness and highly optimized integration, Web Integration Option enables organizations to exploit the power of Informix Dynamic Server while using the tools and languages they already know.
Web Integration Option offers:
1305

【図５８】


1300
Create Web applications
Table of Cont
1. Web DataBlade/App
2. Introduction
1. AppPage Overview
3. Prepare Database
4. Prepare Web DataBlade Module
5. Register Web DataBlade Module
6. Create sbspace
7. Install AppPage Builder
8. Set Up AppPage Builder
9. Create Sample AppPage
1320
1315
1305

【図５９】


2000
順序付き
ローカル
TOC
順序付き
ローカル
TOC
2005
2005
2010
ランダムな
Webのページ

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/45069 A2

(51) International Patent Classification⁷: G09B 7/00

(21) International Application Number: PCT/US00/34013

(22) International Filing Date:
15 December 2000 (15.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/172,134 17 December 1999 (17.12.1999) US
Not furnished 4 December 2000 (04.12.2000) US

(71) Applicant: **INFORMIX SOFTWARE, INC.** [US/US]; 4100 Bohannon Drive, Menlo Park, CA 94025 (US).

(72) Inventor: **HENNUM, Erik**; 78 St. Mary's Avenue, San Francisco, CA 94112 (US).

(74) Agents: **DIBERARDINO, Diana**; Fish & Richardson P.C., 601 Thirteenth Street N.W., Washington, DC 20005 et al. (US).

(81) Designated States *(national)*: AU, BR, CA, JP, MX.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/45069 A2

(54) Title: WEB-BASED INSTRUCTION

(57) Abstract: A method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

WEB-BASED INSTRUCTION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Application No. 60/172,134, filed December 17, 1999 and U.S. Serial No. 08/888,925, filed July 7, 1997, the entire disclosures of which are incorporated herein by reference.

TECHNICAL FIELD

This application relates to web-based documentation and instruction.

BACKGROUND

A typical computer system as shown in Fig. 1 includes a computer 100 having a central processing unit 105, an input/output unit 110 and a memory 115 containing various programs used by the computer 100 such as an operating system 120 and one or more application programs 125. An end-user of the computer system communicates with the computer 100 by means of various input devices (keyboard 130, mouse 135) which transfer information to the computer 100 via input/output unit 110. The computer 100 replies to this input data, among other ways, by providing responsive output to the end-user, for example, by displaying appropriate text and images on the screen of a display monitor 140.

The operating system 120 may include a graphical user interface (GUI) by which the operating system and any applications it may be running (for example, a word-processing program) can communicate with a user of the computer system. A commonly used GUI implementation employs a desktop metaphor in which the screen of the monitor is regarded as a virtual desktop. The desktop is an essentially two-dimensional working template area supporting various graphical objects, including one or more display regions. As shown in Fig. 2, information generated by application programs or the operating system can be displayed on the desktop 200 within display regions 205 (for example, windows, dialog boxes, pop-up menus, pull-down menus, drop-down lists, icons). The user can interact with the operating system, and any applications it may be running, by manipulating the cursor 210

appropriately within the display regions and by entering information with the keyboard or other input device.

The computer 100 also includes some sort of communications card or device 145 (for example, a modem or network adapter) for exchanging data with a network 150 via a communication link 155 (for example, a telephone line). The network 150 may be, for example, a local area network (LAN), an intranet, or the Internet. A service provider provides access to the network and may additionally provide various utilities or services (such as electronic mail) associated with the network. Examples of service providers include Internet service providers (ISPs) such as AT&T WorldNet or online service providers (OSPs) such as America Online and CompuServe.

Developers need to know programming concepts to implement the application program. Therefore, a description of the implementation of the application (and not only the operation of the application) would be helpful.

Most computer applications provide an online help / documentation facility which aids in the use of the application. A typical online help system such as shown in Fig. 3A is accessed through a GUI in which screens of textual and graphical information are displayed to the user in a help window 300. The user can then read the screens of help text to get a better understanding of the application and its various features.

The user invokes the help system with a key sequence (for example, pressing the F1 key on the keyboard) or by clicking the mouse on an appropriate graphical icon or menu item. In response, the help system may display a table of contents 305 listing the available help topics and subtopics which can be accessed and viewed by the user as desired. The user can browse through the table of contents 305 and click a help topic of interest to cause its corresponding body of information to be displayed in a help window. In the help window 300 shown in Fig. 3A, the user has clicked the "Programming with Microsoft Word" topic 310 to cause the corresponding help screen 315 to be displayed in window 300 as shown in Fig. 3B.

The "Programming with Microsoft Word" topic 310 shown in Fig. 3B includes several subtopics 320, each represented by a separate "link." When the user clicks the mouse on one of these links--for example, the "Error Messages" link 325--the text for the corresponding help topic is displayed automatically in the help window 300, as shown in Fig. 3C. In this example, the "Error Messages" topic 330 includes several links to further subtopics relating to

specific types of error messages. As shown in Fig. 3D, when the user clicks one of these links, for example, the "Out of memory (stack space)" link 335, a new help window 340 is spawned to display the corresponding help information ("Freeing up memory") for the selected topic. The help information displayed in window 340 includes yet another link 345 for another subtopic, "active window," which when clicked by the user causes corresponding help text to appear in a pop-up dialog box 350. Virtually any level of such nested help displays is possible. The quantity and types of display regions (windows, dialog boxes, etc.) used to display help information is largely a matter of design choice based on the preferences of the help system developer.

A help system may provide "context-sensitive" help information, meaning that the help system automatically displays help information specifically relevant to the application's current task, rather than simply displaying all available help topics and forcing the user to identify and call-up the appropriate help topic manually. A context-sensitive help system decides which help information to display based on factors such as the current state of the application (for example, the particular function being invoked by the user) and the current cursor position.

The information provided by most online help systems relates to the mechanics of using features of an application. In Fig. 4, for example, the text 400 corresponding to the chosen help topic 405, "Cancel printing," describes how to control the print feature provided by the application 410 (Microsoft Word).

A help system also may provide substantive information on how to make use of the application to achieve a desired goal. In Fig. 5A, for example, the online help system provides two types of substantive information: reference material 500 for the WordBasic programming language and practical explanations 505 of how to use WordBasic to write useful programs. The reference material 500 includes textual annotations describing the syntax and meaning of various WordBasic statements, such as the AddAddIn statement, the help text for which is shown in Fig. 5B. The practical explanations 505 can include static examples of program code which the user can study to gain a better understanding of the WordBasic programming language. Fig. 5C shows an example of a program code that makes use of the GetCurValues WordBasic statement.

Online help systems typically are "built" (that is, processed into a form that facilitates run-time operation) by compiling several different help source files containing help information that has been composed by technical writers. In general, these help source files are maintained as a separate body of information apart from the application to which the help system corresponds. Consequently, when the application developers change or update the functionality of the application, the technical writers must make corresponding changes to the help source files to ensure that the online help system accurately describes the operation of the application. In general, however, online help systems fail to describe the implementation of the application.

A help system may be implemented in a network environment using a "browser", which enables users to access and view electronic content stored in the network environment. A browser typically is used for displaying documents described in Hyper-Text Markup Language (HTML) and stored on servers connected to a network such as the Internet. Fig. 6 is a screen shot of a browser application 600 (in this case, Internet Explorer) displaying a typical HTML document, or web page 605. A user instructs the browser 600 to access the web page 605 by specifying a network address 610 -- or Uniform Resource Locator (URL) -- at which a desired document resides. In response, the browser 600 contacts the corresponding server hosting the requested web page, retrieves the one or more files that make up the web page, and then displays the web page in the computer display 140.

A single web page may be composed of several different files potentially of different data types (for example, text 615, images 620, virtual worlds, sounds, or movies). In addition, a web page can include links 625, or pointers, to other resources (for example, web pages, individual files, or downloadable files) available on the network. Each link has an associated URL pointing to a location on the network. When a user clicks on, or otherwise selects a displayed link, the browser will retrieve the web page or other resource corresponding to the link's associated URL and display it to, or execute it for, the user.

Referring to Fig. 7, a web page 605 may provide, in addition to content 700, a site guide 705 that helps the user navigate through all the links associated with that web page. The site guide 705 is similar to a table of contents and typically resembles a tree structure. Likewise, the web page 605 could include a search facility 710 that enables the user to search for particular key words that appear within the links associated with that web page. The web

page may provide a "Home" link 715 that sends the user back to a main web page from which all content and links can be accessed. The web page may provide a download link 720 that, when accessed, transmits a file from another web page or computer to the user's computer.

According to one aspect of the invention, a method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a source file of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

Embodiments may include one or more of the following features. For example, a predetermined application may be performed and one or more annotations descriptive of the performed application may be presented in coordination with performance of the predetermined application. Performing the predetermined application may include receiving input from the user. Another annotation page may be presented in coordination with performance of the predetermined application based on input from the user.

Presenting the other annotation page may include automatically and simultaneously calling an annotation request module including application, file, class and function names of a program unit for which detail should be displayed. Presenting the other annotation page may also include mapping the request to an annotation, and informing a browser window in the web-based environment to display the other annotation page.

Another annotation page may be presented in coordination with performance of the predetermined application. A global table of contents that includes links to annotations may be automatically generated by parsing structured links in web pages including annotation pages. Generation of links in the global table of contents may be synchronized with presentation of annotations by highlighting links corresponding to a current annotation page. The global table of contents may be presented in a first frame of a first browser window, the annotation page may be presented in a second frame of the first browser window, and the predetermined application may be performed in a second browser window.

Performing the predetermined application may include launching a Java applet or application, which may include calling a Java application program interface to ask a web browser to show the annotation page. Performing the predetermined application may include downloading a hyper-text markup language page containing a Java applet.

Performing the predetermined application may include sending a common gateway interface request to a web server that launches the application in a window in the web-based environment. The application may return a hyper-text markup language page that includes JavaScript to ask a web browser to display the one or more annotations.

The annotation page may be presented in a first browser window and the predetermined application may be performed in a second browser window. The application implementation detail may include text descriptive of the application, fragments of source code from the application, or both. The source code fragments may be imported directly from the source code file of the presented application.

The annotation page may be automatically generated. This generation may include receiving a source code file that has embedded text marked up with instructions. Additionally, the source code may be parsed to determine a structure of the predetermined application, and one or more annotations may be generated based on the predetermined application structure and instructions. Generation of the annotation page may include generating one or more annotation links for navigating the annotations of the predetermined application. Additionally, application implementation detail may be generated based on the embedded information, and one or more keyword links may be generated for reference documentation. Generating the annotation page may also include highlighting the keyword links and the annotation links in the annotation page. The annotation page may be automatically updated when an updated source code file is received.

A global table of contents may be automatically generated by parsing the one or more annotations for annotation links. The global table of contents may be provided, and may include links to annotations. Alternatively, the global table of contents may be generated, and may include links to web page including annotation pages relating to an application. The local table of contents may be provided when a local link in the global table of contents is selected.

The presented annotation page may be descriptive of the performed application, and the annotation page may be presented in coordination with performance of the predetermined application.

A source code file, which is stripped of annotation mark up and includes source code of the application but does not include text from the annotations, may be generated. The stripped source code file may be presented and the user may be permitted to edit the stripped source code file.

According to another aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes providing a predetermined plurality of applications. A predetermined application is performed, and an annotation page descriptive of the performed application is presented in coordination with performance of the predetermined application. The annotation page includes detail of application implementation and links to annotations and reference documentation.

According to another aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes automatically assembling and providing a global table of contents based on content in the environment. The global table of contents includes a plurality of links to content within the environment. A local table of contents that includes links to content that orient the user within a local topic, is generated. The user is permitted to select links from the local table of contents to access local topics.

According to a further aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes providing a plurality of predefined interactive examples. One or more of the predefined interactive examples is performed in response to user selection, and one or more annotations descriptive of the performed interactive example are presented in coordination with performance of the predefined interactive example. The user is permitted to selectively explore different aspects of the performed interactive example, the annotations, or both.

According to another aspect of the invention, a web-based computer system for teaching a user to implement an application includes one or more predefined interactive applications, and an annotation page including one or more annotations. A predefined

interactive application is selectively executable by the user of the web-based computer system. The annotation page describes a predefined interactive application. The annotation page also includes one or more links, and detail of implementation of the application. Different annotations are automatically provided in the annotation page in response to selective execution of a predefined interactive application.

According to a further aspect of the invention, a web-based computer system for teaching a user to implement an application includes a web-browser window that includes a content frame, a framework applet, and a table of contents frame that displays a global table of contents hierarchy of links related to content in the content frame. The system also includes one or more annotations displayed in the content frame, where each annotation describes a predefined interactive application and includes links to other content. The system includes a table of contents window that displays a local table of contents hierarchy of links related to local content in the displayed annotation.

The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects and advantages will be apparent from the description, the drawings, and the claims.

DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a prior art computer system.

Fig. 2 shows display regions in a graphical user interface as used in the computer system of Fig. 1.

Figs. 3A – 5C are screen shots from a prior art online help and documentation system.

Fig. 6 is a screen shot of a browser application.

Fig. 7 shows a display region in a browser application.

Fig. 8 is a flow diagram showing the options available to a user of the Informix® byExample application.

Figs. 9A – 9P are screen shots from the Informix® byExample application and from the NewEra™ development environment.

Fig. 10 is a block diagram of the NewEra™ architecture.

Fig. 11 is a block diagram showing how the Informix® byExample application is built.

Fig. 12 is a sample of NewEra™ source code.

Fig. 13 shows display regions in a web-based instruction system.

Figs. 14A – 15A, 15C, 17A-17D, 18A, 18B, 19A and 19C-19F are screen shots from the Informix® byExample web-based application.

Figs. 15B, 15D and 19B show content in a display region of the Informix® byExample web-based application.

Fig. 15E and 18C are source code files with embedded annotations.

Fig. 16A is a flow diagram showing steps taken by an author of annotation pages.

Fig. 16B is a flow diagram showing steps taken by a scripting program to automatically generate links in the annotation pages.

Fig. 16C is a flow diagram showing steps taken by an example program.

Fig. 20 is a block diagram of a local table of contents model used in the web-based application.

DETAILED DESCRIPTION

The help information provided by conventional online help systems has proven useful in aiding users to make effective use of application programs. However, because these conventional online help systems essentially are limited to providing static textual or graphical information, their effectiveness is diminished considerably. Users of conventional online help systems gain instruction by reading and carefully studying the textual and graphical information provided by the help system and then applying its teachings to the problem to be solved. As a result, learning to use applications of any complexity often is a painstaking and time consuming process.

An online help and instruction system developed by Informix® Software, Inc., known as NewEra™ byExample, dramatically enhances the ease with which users can absorb information and learn to use new applications. NewEra™ byExample is an online documentation facility for NewEra™, an object-oriented application development environment from Informix® Software, Inc. which runs under the Windows 95/NT operating systems. A copy of NewEra™ byExample's online description of its use and operation is attached as Appendix A.

NewEra™ byExample (or more generally, Informix® byExample, which covers the example-based instruction systems provided for the NewEra™, Visual Basic and Java development environments) is a specific implementation of a more general concept referred to as "documentation by example" in which users are provided with dynamic, interactive examples demonstrating how to accomplish a given task. Annotations describing various aspects of the examples accompany the examples as they are being executed. Documentation by example is based in part on the premise that users learn best by doing something (for example, participating in an activity and observing or influencing its outcome) rather than by merely reading about the topic.

As illustrated in the flow diagram of Fig. 8, an Informix® byExample user has several different options for obtaining information including selecting among various different topics (step 800); running examples while the application for which help is sought remains active (step 805); reading about the examples, either concurrently while running the example or independent of the example (step 810); inspecting the examples' source code in different editor utilities (step 815); and accessing online background reference materials that help the user to understand the examples (step 820)--all without leaving the help environment. While in step 815, the source code for the examples can be used as sample program code which can be cut-and-pasted for use as a template in the NewEra™ development environment in creating new applications. Moreover, Informix® byExample enables users to learn through experimentation, for example, by selectively changing the examples or their parameters and observing how the changes affect the examples' outcomes.

Specific features of Informix® byExample are described in detail with reference to Figs. 9A-9P, which are exemplary screen shots taken from the Informix® byExample application.

When a user first launches Informix® byExample, the default screen configuration shown in Fig. 9A is displayed. This initial screen includes two separate display windows, a list (or "table-of-contents") window 900 showing the subtopics presently available to the user under the current topic 902, and a text window 904 which displays the help information corresponding to the topic or subtopic selected from the list window 900. As the user clicks different ones of the eight subtopics 906 displayed in the list window 900, the information in the text window 904 is updated automatically to correspond to the chosen subtopic 906. The

user can move to different pages within the current topic by clicking the forward (">>") button 908 or the backward ("<<") button 910 as desired.

In the example of Fig.9A, the subtopics shown in the list window 904 relate to the topic "NewEra™ byExample Introductory Topics." To switch to another help topic, and thereby make available a different subset of the online help documentation, the user clicks the Help Topics button 912 which brings up a window containing the Help Topics menu 914 with a list 916 of nine different help topics, as shown in Fig. 9B. At any point in the Informix® byExample application, the user can jump to any other portion of the online help system by bringing up the Help Topics menu 914 and clicking the desired topic. The user can return to a previous topic by pressing the Back button 916 an appropriate number of times.

Each of the help topics in the list 916 in Fig. 9B can be expanded to reveal a hierarchy of multiple levels of subtopics. When the user clicks, for example, on topic 918 ("NewEra™ byExample"), it expands to reveal two additional levels of subtopics as shown in Fig. 9C--a first level 920 including the subtopics "Introduction," "Common NewEra™ Programming Techniques," and "The Examples," and a second level 922 under "The Examples" subtopic which includes the 43 interactive examples.

When the user clicks one of the examples, for example, the "Enabling and Disabling Buttons" example 924, the list window 900 is updated as shown in Fig. 9D to display the annotation segments 926 ("Overview of Buttons2 Example," Graphical Object Summary," "Event Handler Summary," "Important Event Handlers," and "Enhancements and Variations") associated with the selected example. The annotation segments 926 collectively describe the corresponding example and include descriptions of the example's window, its graphical objects, and its event handlers. In addition to the prose descriptions of the example, an annotation segment usually also includes a source code fragment of particular interest which has been imported directly from the source code of the example under consideration.

As shown in Fig. 9D, one of the annotation segments ("Important Event Handlers") includes 13 topics 928-- a list of the primary event handlers used in the BUTTONS2 example. Each event handler topic 928 includes source code fragments and prose explanations describing the event handler to which the topic corresponds. For example, when the user clicks event handler topic 930, the text window 904 displays source code fragments 932

relating to the corresponding event handler (*nextBT :: activate()*) along with annotations 934 describing the code's operation, as shown in Fig. 9E.

The text window also may contain one or more links to related information, for example, background reference material, which in turn may include still further links to additional background information and so on in hierarchical fashion, each successive level in the hierarchy providing information about the example in greater detail and at a lower level of abstraction. By providing a hierarchy of links to increasingly detailed documentation in this manner, Informix® byExample supplies context-appropriate information in a helpful and efficient manner to all users, regardless of their varying levels of experience and sophistication. A user can traverse down the hierarchical links of descriptive information selectively until a level of understanding is reached that is commensurate with the user's needs and background. This arrangement provides novice users with easy access to detailed descriptive information while, at the same time, experienced users seeking help on a specific point are protected from having to navigate through large volumes of unneeded information.

An example of hierarchical linking is shown in Fig. 9D in which text window 904 includes a link 936 (displayed as green, underlined text) to the *MAIN()* function, one of functions in the BUTTONS2 example. When the user clicks the *MAIN()* function link 936, the text window 904 displays the source code 940 for that function, as shown in Fig. 9F. The source code 940 includes further links to related information such as an online language reference manual containing descriptions of keywords and object classes. When the user clicks one of these links--for example, the keyword link 942 for the LET statement--the text window 904 changes to display the corresponding online language reference entry as shown in Fig. 9G. Similarly, if the user had clicked the object class link 944, the text window 904 would have displayed information about the *ixSQLConnect* class. In Fig. 9G, the user can follow links to still further background information, for example, by clicking the Object Expression box 946 to cause the text window 904 to appear as in Fig. 9H. Subsequently, or alternatively, the user can click the Glossary button 948 to bring up an online glossary in a pair of windows--a glossary table of contents window 950 and a glossary text window 952--as shown in Fig. 9I. Clicking a term in the glossary table of contents window 950 causes its definition to appear in the glossary text window 952.

After studying an example's annotation, its source code fragments, corresponding language reference entries, the glossary, or a combination thereof, the user can jump selectively to any other location in the help system by clicking the Contents button 954, which brings up the Help Topics menu 914 shown in Fig. 9B (or the Index button 956, which presents the available help topics in a searchable-indexed form), and then selecting the desired topic in the manner described in connection with Figs. 9B and 9C.

Keyword links and class name links, such as the LET statement link 942 and the *ixSQLConnect* class link 944, respectively, in Fig. 9F are represented in visually unique manners (for example, blue uppercase text for keywords, blue upper and lowercase text for class names) so that they may be distinguished easily from each other and from other types of links such as the *MAIN()* function link 936 in Fig. 9D (green, underlined text). By using different styles for different types of links, Informix® byExample provides the user with intuitive and useful information concerning the nature of the online information available and the interrelationships between the different components (annotations, source code fragments, language references, etc.) of the examples. Virtually any number of different link types may be represented by different styles according to the preferences of the system designer.

For each of the source code fragments included in an example's annotation, a user can invoke an appropriate editing utility from within Informix® byExample to inspect, edit or copy the example's source code. This allows users to view a source code fragment in the context of the larger program from which it was taken.

Informix® byExample includes source code fragments from two different types of source code--textual program code in the NewEra™ programming language (as indicated by a 4GL or 4GH file suffix), and windows interface definition files (files having the suffix WIF) which define how the GUI will appear to, and interact with, the end-user of the application undergoing development. To view either type of source code fragment, the user clicks a short-cut arrow next to a code fragment, for example, one of the short-cut arrows 958 and 960 shown in Figs. 9D-9F, and Informix® byExample responds by launching an editor that corresponds to the type of source code under consideration. When the user clicks a short-cut arrow next to a 4GH or 4GL file, such as short-cut arrow 958 in Figs. 9D and 9F, Informix® byExample automatically launches the appropriate editor--NewEra™ Codewright--to view the source code file from which the code fragment was taken, as shown in Fig. 9J. Similarly,

when the user clicks a short-cut arrow next to a WIF file, such as short-cut arrow 960 in Figs. 9D and 9E, Informix® byExample automatically launches the appropriate editor--NewEra™ Window Painter 3.0--to view the WIF file from which the code fragment was taken, as shown in Fig. 9K.

Selectively launching an appropriate one of multiple different editors in this manner reflects the standard editing behavior of the NewEra™ development environment. Both the NewEra™ development environment and the Informix® byExample documentation system make use of the same editors in the same manner. As a result, users gain familiarity with the application for which help is sought (that is, the NewEra™ development environment) through normal interaction with the online help system (that is, Informix® byExample).

Once the user has opened up the source code for an example, the user simply can study the code or can cut-and-paste portions of the code, whether visual objects from a WIF file or program statements in a 4GH or 4GL file, into the user's own source files. Alternatively, the user can perform a "Save As..." operation and thereby save the source code for the example under a new file name. The user then can edit or otherwise manipulate the new file as desired. In this manner, the examples provided by Informix® byExample can serve as templates for use in developing new applications in the NewEra™ development environment.

Users also may execute any or all of the 43 interactive examples provided with Informix® byExample to observe first hand how they operate. The examples are prebuilt and can be launched directly from their corresponding Informix® byExample annotations. To do so, a user first selects an example of interest from the Help Topics window 914 shown in Fig. 9C and, when the corresponding annotation appears in the text window, clicks the Run button appearing near the top of the text window. In response, the example executes and, based on the input received from the user, displays various screens to the user as if the example were a standalone application. At the same time, the text window automatically updates to display descriptive information that is pertinent to the portion of the example that was just executed by the user. With each successive operation that the user performs on the running example, the text window is updated simultaneously (or nearly so) to maintain synchronization with the state of the interactive example by displaying corresponding sections of the annotations which explain to the user what just happened in the example. By coordinating the help display with

the current state of the examples, users consistently are provided with timely and useful information (for example, the particular source code being executed by the example) that is directly relevant to the user's current topic of interest. As a result, the user's ability to comprehend and absorb information is enhanced dramatically. An example of Informix® byExample's automatically coordinated help display is illustrated in Figs. 9L-9P.

Fig. 9L shows the initial list window 900 and text window 904 that are displayed when the user selects the "Displaying an Edit Menu" example from the Help Topics menu. To run this example, the user clicks the Run button 962 which, as shown in Fig. 9M, spawns an example window 964 illustrating the basics of an edit window. At the same time, the text window 904 is updated to display information on the *MAIN()* function for the "Displaying an Edit Window" example.

As the user selectively manipulates the GUI controls in the example window 964, the information displayed in the text window 904 is updated automatically in a corresponding manner. In Fig. 9N, the user has clicked in text box 966 which causes the text window 904 to display information relating to *edit1TB :: focusIn()*. Similarly, when the user clicks text box 968, text window 904 displays information relating to *edit2TB :: focusIn()* as shown in Fig. 9O. When the user clicks the CheckBox 970, text window 904 displays information relating to *noneditCB :: focusIn()* as shown in Fig. 9P.

Users can experiment with an example by changing its source code or modifying its parameters and observing how these changes affect the example. To do so, the user edits the desired source code file, saves it a separate working directory so as not to disturb the predefined examples, and then rebuilds the example using mechanisms provided with the NewEra™ development environment. The number and types of such experiments that can be created and performed are limited only by the imagination of the user.

Other options in running the examples are possible. For example, users can run an example without concurrently viewing annotations. Additionally, the Debugger provided with NewEra™ can be used to set breakpoints in the example source code before running the example, thereby giving the user even further insight into how an example works.

A description of the Informix® byExample architecture, and the manner in which the NewEra™ development environment and the Informix® byExample application are built, is provided with reference to Figs. 10-12.

Informix® byExample builds upon the Online Help (OLH) facility provided with the Windows 95/NT operating systems. As shown in Fig. 10, the Informix® byExample application 1000 draws both upon resources created specifically for Informix® byExample as well as resources that are native to the NewEra™ development environment 1005. The components specific to the Informix® byExample application 1000 include the interactive examples 1007, source code 1010 for the examples, and annotations 1015 describing the examples. The annotations 1015 include several different subcomponents including representative fragments 1020 of the examples' source code, short-cuts 1025 that launch an appropriate editor (for example, NewEra™ Codewright or NewEra™ Window Painter) for viewing the examples' source code, jumps 1030 to the interactive examples, and links 1035 to descriptions of specified keywords and class names contained in the NewEra™ online reference 1040.

As indicated in Fig. 10, the online reference 1040, the Codewright editor 1050 and the Window Painter editor 1055--along with other components such as Application Builder 1060, Debugger 1065 and Interprocess Communications (IPC) library 1070--exist as part of the development environment 1005 and thus are logically separated from the Informix® byExample application 1000. Consequently, when a user of the Informix® byExample application 1000 requests a resource residing in the NewEra™ development environment--either by clicking a link 1035 for a keyword or class name or by clicking a shortcut 1025 to view source code--Informix® byExample 1000 first must communicate with the NewEra™ development environment 1005 via an interface dynamic linked library (DLL) 1080 to access the requested resources. The interface DLL 1080 is a compiled library of routines that enable the Informix® byExample application 1000 to communicate with other applications such as the components of the development environment. Informix® byExample 1000 calls the appropriate DLL routines to display the requested online reference information or to launch the appropriate source code editor, depending on the nature of the request made by the user.

More specifically, when an Informix® byExample user clicks on a shortcut 1025 to a location in an example's source code 1010, the Informix® byExample application 1000 calls a function in the DLL, which in turn calls a function in the IPC library 1070 which launches the appropriate editor. As part of this function call (which is generated automatically by processing source code fragments during the build of Informix® byExample, discussed

below), the Informix® byExample application 1000 passes parameters that designate the editor to be launched (Codewright 1050 or Window Painter 1055), and that identify the line number at which the examples' source code 1010 is to be opened by the designated editor. When an Informix® byExample user clicks on a link 1025 for a keyword or class name, the Informix® byExample application 1000 calls a function in the DLL, which in turn uses the Windows OLH facility to display the corresponding definition in the online reference 1040.

Other functions provided by the interface DLL 1080 control execution of the interactive examples 1007 and coordinate the list window and the text window displays to ensure that they maintain correspondence. Further details on the interface DLL 1080 and the runtime operation of the Informix® byExample application 1000 are set forth in Appendix B.

The manner in which the Informix® byExample application 1000 and its components (for example, examples 1007, examples' source code 1010 and annotations 1015) are generated realizes a high degree of code "maintainability"--a measure of the efficiency and ease with which an application can be modified. The high degree of code maintainability is achieved by incorporating all of the information used to generate both the interactive examples and the corresponding annotative components of Informix® byExample into a unified logical entity--namely, the source code for the interactive examples themselves. As a result, only one central source of information need be maintained. Any changes or updates made to that central information source will be incorporated automatically both into the examples and into the documentation / instruction / help facility (Informix® byExample) for the examples. This automated build procedure ensures that the examples and the corresponding Informix® byExample annotations are kept in synchronization regardless of the number and frequency of modifications made to the underlying source code.

As shown in Fig. 11, the NewEra™ byExample source code 1100 can be thought of as a single logical entity, although physically it is formed of a collection of interdependent files. The source code 1100 contains three basic types of text--program instructions 1105, program comments 1110 and annotations 1115--intermixed throughout the source code. The different text types are distinguished from each other by programming conventions and by strategically placing various different markup symbols 1120 throughout the source code.

Some of the text in the source code 1100 can serve multiple purposes. For example, the program instructions 1105 in the source code 1100 are compiled into the examples' binary

executable files 1125. These program instructions include calls to the OLH facility to display the corresponding annotation at the appropriate point during execution of the example. When an example is run by the end-user, these OLH calls cause the text window to display the appropriate annotation automatically to describe what just happened in the example.

Portions of these same program instructions 1105 also will be extracted to serve as a clean copy of the examples' source code, which can be displayed to the user in an editing environment. Similarly, descriptive text that serves as program comments 1110 (unprocessed programming explanations directed at the Informix® byExample project developers) also can serve as annotations 1115 (programming explanations displayed to end-users of Informix® byExample at runtime).

The markup symbols 1120 delineate the various types of text in the source code and specify how they are to be handled when the interactive examples and the Informix® byExample annotations are built. Fig. 12 shows a sample of NewEra™ source code which includes several markup symbols including two instances of the "normal" symbol 1200 and 1205, an "[edit" symbol 1210 and a "] file" symbol 1215. Each of these markup symbols, along with their respective arguments, are bounded by a pair of brackets ("{ ... }") indicating that they reside in comment fields and are not to be treated as NewEra™ program instructions. Programming languages other than NewEra™ may use different conventions to delineate comment fields. In the Java programming language, for example, a start of a comment field is designated by a "/*" symbol and terminated by a "*/" symbol. In any event, the corresponding programming language compiler will ignore any text that has been designated as residing in a comment field.

The ".normal" markup symbol indicates that the text following that symbol (for example, "Since objects, ...," following symbol 1200) is to be treated as explanatory comments, and thus to be displayed to the end-user in a text window as part of the annotation text at an appropriate point during execution of a corresponding interactive example. Other markup symbols specify the name of output files, portions of the source code that are to serve as representative fragments of the examples' source code, hotspots and destinations for jumps and links, or GUI-related information concerning display characteristics and objects (windows, popups, buttons, etc.). A detailed description of the markup language is set forth in Appendix C.

Once the source code 1100 has been modified as desired, it is used to build the interactive examples and the descriptive content of the Informix® byExample application through a number of different steps. First, the source code 1100 is processed by two different scripts 1130--a PERL script (Practical Extraction and Report Language, a general purpose interpreted language often used for parsing text) and a WordBasic script. The scripts 1130 generate two basic types of output: source code files 1135 for the interactive examples, and RTF files 1140 (Rich Text Format, the format expected by the OLH compiler) which represent the descriptive and visual content (for example, annotations, source code fragments, shortcuts to source code editors, links to online reference, jumps to executable examples) of the Informix® byExample application.

The PERL script parses the source code 1100 searching for markup symbols and, based on the particular markup symbols encountered, produces several RTF file fragments and several source code files 1135, which represent various subsets of the overall source code 1100. The WordBasic Script then merges the RTF file fragments into complete RTF files 1140 which are processed by the Windows OLH compiler 1145 to produce OLH files 1150 containing the descriptive and visual content for the Informix® byExample application. At the same time, the examples' source code 1135 is compiled by the NewEra™ compiler 1155 to generate the binary executable corresponding to the interactive examples 1125.

The RTF file fragments generated by PERL script contain several different components in addition to the annotations 1115 appearing in the source code 1100. The PERL script identifies each instance of a keyword or a class name appearing in the source code extracted for the examples. For each keyword and class name detected, the PERL script creates a link in the RTF file to the corresponding entry in the online reference materials.

The PERL script also extracts fragments of representative source code for inclusion in the RTF files as text that appears along with the explanatory comments. The source code fragments are formatted as monospace unwrapped text delineated by leading and trailing blank lines whereas the explanatory comments are formatted as proportionally spaced wrapped text. For each source code fragment included in the RTF file, the PERL script also inserts in the RTF file a corresponding short-cut button which enables the end-user to launch the source code editors and view the source code at the line where the fragment starts. The PERL script also strips all of the markup symbols 1120 from the source code extracted for the

examples. This provides end-users with a clean version of the source code for viewing in the associated editor.

Other functions performed by the PERL script include automatically guaranteeing that the identifier for an annotation topic is the same in an interactive example as it is in the Windows OLH facility. That is, the PERL script reads the help topic identifiers for the Windows OLH facility and generates corresponding NewEra™ constants. The PERL script also generates modified versions of the NewEra™ makefiles (files that include computer-readable instructions for building an application) which are used to build the examples. Further details of the PERL script and its operation are set forth in Appendix B.

Although the PERL and WordBasic scripts described above operate on source code written in the NewEra™ programming language, different scripts can be used to parse other types of source code, for example, Java or Visual Basic. Generally, appropriate PERL and WordBasic scripts can be written to process virtually any type of programming language provided the programming language utilizes ASCII source code (required by PERL) and provides some sort of source code comment mechanism. Other programming language attributes that facilitate use of the Informix® byExample techniques include a mechanism for invoking the Windows OLH facility with a topic identifier (so the example can display its annotations), a mechanism for invoking the editing functions of the development environment (so the annotation can open source code files, assuming the programming language under consideration provides or requires a development environment), and an online reference in Windows OLH format (so keywords in the source code can have jumps to the online reference). Many of the Informix® byExample features described above can be implemented even if the underlying programming language lacks one or more of these other attributes, however.

PERL scripts can be modified to output files in formats other than RTF. For example, a modified PERL script can output hypertext markup language (HTML) files, which can be viewed using any available web browser (for example, Netscape Navigator).

Other variations of documentation by example are possible. For example, the annotations describing the interactive examples could be presented in a manner other than textual. Sounds, graphical symbols, pictures, movies or any other means of communication could be used as desired. Further, the selection of which interactive examples to perform

could be based on factors other than, or in addition to, designation by the user. For example, an interactive example could be launched automatically at certain points during execution of the underlying application, or at certain execution points in the help system. When the user clicks a keyword, class name or other link, an example could be launched automatically either in addition to, or instead of, displaying the textual reference information pointed to by the link.

The documentation by example methods and techniques described above are not limited to aiding users of software development systems but rather may find application as a general training and education tool for any computer-based application or utility. Moreover, the techniques described here may be implemented in hardware or software, or a combination of the two. Preferably, the techniques are implemented in computer programs executing on programmable computers that each includes a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), and suitable input and output devices. Program code is applied to data entered using an input device to perform the functions described and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (for example, CD-ROM, hard disk or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described. The system also may be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other embodiments are within the scope of the following claims.

For example, the documentation by example method and techniques may be applied in a network or web-based environment such as a local area network (LAN), an intranet, or the Internet. A web-based instruction system developed by Informix® Software, Inc., known as

Informix® byExample, dramatically enhances the ease with which developers can effectively and efficiently implement database applications. The web-based instruction system provides users with examples and instructions in web pages to teach users how to implement applications. The applications themselves may or may not be web-based applications. The web-based instruction system enables users to read formatted source code, and navigate program structure (for example, the hierarchy of file, class, and function) to access details of the application program's implementation. The web-based system also permits users to jump from a language keyword (that may be indicated as such in code fragments) to a full reference documentation pertaining to that keyword. Users are able to open source code in an editor and run programs directly from the web browser at the click of a run button in a toolbar. As the example runs, an annotation page for the current method or function in the program displays. Therefore, the user is able to view the annotation as the function or method executes to more quickly understand how to implement such an application program.

This is in contrast to prior online help systems that simply show a user how to use an application. Because some implementation changes do not alter the operation of the application, but all implementation changes, by definition, affect the implementation, this makes maintenance of the implementation documentation even more important. The web-based instruction and help system solves this problem with prior online help systems by maintaining the annotations as embedded comments within the source code and generating the annotation pages automatically whenever the source file changes.

The web-based system also provides the users with how-to documentation for accomplishing a particular task. The how-to documentation provides clear step-by-step instructions, and presents a flexible interface for experts and "newbies" alike. The how-to documentation includes useful graphics, and provides links to needed software and related demos and other technical information. Additionally, the how-to documentation helps the user to determine which products to use to accomplish a goal.

As illustrated in Fig. 13, when a user first accesses a web browser providing the web-based instruction system, a default browser configuration 1300 is displayed. The initial browser includes a framework of three separate display frames: a content frame 1305 that typically displays a web page, but may also display other relevant content such as annotation pages (which are a specific type of web page), a top frame 1310 that includes a framework

applet (for example, a Java applet) that displays a tool or navigation bar in addition to providing other services, and a table of contents (TOC) frame 1315 which may be implemented using, for example, a Java applet.

When the browser displays a byExample® HTML page, the page uses JavaScript to automatically check for the presence of the framework. If the framework is present, the JavaScript notifies the framework applet in the top frame 1310 about the name of the new page and the type of content it provides. If the framework is absent from the page, the JavaScript opens the framework with the page as the initial page (that is, the page on which the notification occurs).

Referring also to Figs. 14A and 14B, based on the name of the page 1400, the table of contents 1315 automatically selects the corresponding title 1405. The table of contents 1315 tests the available Java libraries on the client and automatically displays the table of contents hierarchy with a tree control if available. The table of contents frame 1315 is user- resizable and dismissible using a contents button 1410 in the top frame 1310: for example, in Fig. 14A, the table of contents frame 1315 takes up around 40% of the browser, while in Fig. 14B, the table of contents frame 1315 is dismissed and therefore not visible. The contents button 1410 automatically indicates the status of the TOC by displaying a dismissed icon (for example, in Fig. 14B) or an opened icon (for example, in Fig. 14A).

The user may access other links in the content frame 1305 from the TOC frame 1315 by clicking on a corresponding subtopic in the TOC tree, for example, the subtopic "Extending Database Servers" 1415. In response, as shown in Fig. 14C, the content frame 1305 displays content corresponding to that subtopic and the TOC frame 1315 highlights the selected subtopic 1415. Also evident in this particular example is that the top frame 1310 includes a home button 1420 that automatically changes its appearance (like the contents button) depending on whether the browser is displaying a home page or not.

Alternatively, the user may access links in the content frame 1305 by clicking on the corresponding link in the content frame, for example, subtopic link "Java Database Connectivity (JDBC)" 1425 shown in Fig. 14B. In response, as shown in Fig. 14D, the content frame 1305 displays content corresponding to that subtopic, including a title 1430 representing that subtopic. Furthermore, the TOC frame 1315 automatically highlights the selected subtopic 1435 in the TOC tree to indicate to the user which content is displayed in

the content frame 1305. Such synchronization between the TOC and the content helps the user to effectively and efficiently navigate through the many subtopics and topics of an application.

Referring again to Fig. 14C, when the user selects an example using either an example icon or link 1440 in the content frame 1305 or a subtopic icon or link 1445 in the TOC frame 1315, the browser displays in the content frame 1305 an annotation page 1500 that describes the corresponding example as shown in Fig. 15A. Referring also to Fig. 15B, the annotation page 1500 includes annotations 1505 descriptive of the example (also called prose), and at least one link to another annotation page 1510.

For example, when the user clicks the link for the source file "querydb.html", the content frame 1305 displays the source code file annotation page "querydb.html" 1525 as shown in the browser of Fig. 15C. The content frame displays source code fragments 1530 relating to the corresponding source code file querydb.html, as shown in Fig. 15D. The source code fragments 1530 have been imported directly from the source code files of the example under consideration. The source code file annotation page in the content frame also displays annotations 1535 describing the source code file to which the topic corresponds; such annotations 1535 may be referred to as prose.

Source code is marked up with standard HTML . Annotations 1535 are easy to maintain because the annotation comments are embedded within source code comments in the source code files. For example, referring to Fig. 15E, the source code file 1560 is shown for the querydb.html source code 1525. Annotation pages, containing the source code fragments, are generated automatically from the marked up source code files.

Referring to the flow chart 1600 of Fig. 16A, an author of an example annotates the source code file by embedding annotations in source code comments (step 1605). The author also marks up the embedded annotations with HTML tags and one or more special instruction tags for the web-based system (step 1610). Once the source code files are marked up, as shown in the flow chart 1620 of Fig. 16B, a scripting programming language such as PERL parses the source code file to determine the program structure from the programming language, and reads the annotations and special instruction tags (step 1625). The PERL script generates annotation pages that reflect the program structure of the source code file (step 1630). Moreover, the PERL script provides links between annotation pages for navigating the

program structure (step 1635), incorporates the annotation comments (step 1640), while using Java Script to notify the framework (step 1645), and highlights language keywords in source code as links to related information (step 1650). Annotations are therefore easily maintained. Referring again to Fig. 16A, the author of the example may adjust or edit the source code and the annotations at the same time and at the same location (step 1615). Therefore, annotations are regenerated automatically.

The annotation page in the content frame also may contain one or more links to related information or from one annotation page to another annotation page. Such flexibility is due to the automatic generation of annotation pages discussed above because the PERL script determines the program structure and formats references to other parts of the program as links. For example, when the user selects the "sportsHeader.tag" link in the querydb.html file 1525, the content frame 1305 displays the sportsHeader.tag source code annotation page 1540, and the TOC frame 1315 automatically highlights the corresponding subtopic link 1700, as shown in Fig. 17A.

The selectable link in the content frame annotation page may correspond to background reference material. For example, when the "MIVAR" link is selected, a separate browser 1705 is spawned that displays information about the MIVAR tag, as shown in Fig. 17B. In this case, the PERL script finds keywords in the source code and formats them as links to the reference documentation. If the user selects the "WEB HOME" link in the content window, a browser 1710 is spawned that includes a glossary, as shown in Fig. 17C.

Referring again to Fig. 15C, the user can select an edit button 1562 in the top frame 1310 to open the source file corresponding to the annotation page in the content frame 1305 as plain text in a browser window 1715, as shown in Fig. 17D. In this case, the PERL script automatically strips the annotation comments out of the source file so the user is able to edit a source file that is unencumbered by long explanations. The user could then cut and paste text between this window and another text window that displays the user's own program.

Referring again to Fig. 15A, the user can run an example using the framework applet to launch a Java example applet or application, to download an HTML page containing a Java applet or other embeddable program object, or to send a common gateway interface (CGI) request to the web server to run non-interactive programs with output redirected to a browser window. In the case of Java examples, the running example calls Java application

programming interfaces (APIs) to ask the web browser to show the annotation pages. In the case of CGI programs, the example program returns an HTML page that contains JavaScript to ask the web browser to show the annotation pages. Thus, the examples run either on the server or in the restricted sandbox of the web browser. The examples could run on a local client if the framework used the security features of the browser to ask the user for permission to run the example locally.

In Fig. 15A, the user can run the example by selecting the Run button 1570. Referring to the flow chart 1655 in Fig. 16C, upon selection of the Run button 1570, the example launches (step 1660), using any of the methods described above, in a browser window 1800, as shown in Fig. 18A. The TOC frame 1315 simultaneously and automatically synchronizes with the running example by highlighting the current annotation page for the source code that implemented the example. A PERL script generates the TOC automatically by parsing the HTML pages for links that indicate hierarchical structure much like the automatic generation of the annotation pages.

Moreover, the example simultaneously and automatically synchronizes the content frame 1305 with the running example by displaying the current annotation page, in this example, the querydb.html file 1525 discussed above. The user can then view the annotated source code or edit the source file while the example is running, and jump from keywords to the reference documentation for the keywords as described above.

When the user interacts (step 1665) with the example in the browser window 1800, (by, for example, entering a customer number and selecting the "Submit" button in the browser window 1800), the user interface changes (for example, the Customer Report corresponding to the entered customer number) in the browser window 1800, as shown in Fig. 18B (step 1670). The user interface changes correspond to the next annotation page, cust_db.html file, 1586 used in the running example. The running example automatically and simultaneously calls an annotation request module with the example, file, class and function names of the program unit for which annotations should be displayed (step 1675). The annotation request module maps that request to an annotation page and tells the browser to display the annotation page in the content frame (step 1680). For example, in Fig. 18B, the next annotation page is cust_db.html 1586. The cust_db.html subtopic is highlighted 1585 in the TOC frame 1315 and the content frame 1305 displays the cust_db.html annotation page

1586, as shown in Fig. 18B. Embedded annotation markup 1810 are shown in the cust_db.html source code file displayed in Fig. 18C. The embedded markup 1810 is used when generating the annotation page from the source code file.

Referring again to Fig. 14C, in addition to selecting an example, the user could also select a how to document icon or link 1450 in the content frame 1305 or a how to document subtopic link 1455 in the TOC frame 1315. In this case, a how to document 1900 is displayed in the content frame 1305, as shown in Fig. 19A. For clarity, the complete document 1900 is shown in Fig. 19B. In synchronization, the TOC frame 1315 highlights the displayed subtopic 1902 corresponding to the how to document 1900.

Also in synchronization, the top frame 1310 dynamically changes to reflect the selected how to document 1900. For example, in the top frame are now displayed an up button 1905, a first page button 1910, a previous page button 1915, a next page button 1920, a last page button 1925, and a list button 1930. The up button 1905, when selected, jumps to the category or subtopic that lists the how to document 1900. For example, the subtopic "Extending a Database Server" lists the "Create Web applications" how to document 1900, and upon selection of the up button 1905, the browser would display all information relating to the "Extending a Database Server" subtopic, as shown in Fig. 14C. Generally, the top frame changes to match the content type which is specified in the Java Script call that notifies the framework that a new content page has displayed.

The document 1900 contains pointers to other documents or pages, the pointers being accessed in one of several ways. The pointers may be selected directly from the content frame 1305 by clicking on a link in the content frame which is indicated by, for example, a different color, font, or style. For example, the user may click on the Application Page link 1930 or the Prepare Database link 1935 in the content frame 1305 as shown in Fig. 19B. Pointers may be selected from the TOC frame 1315 by selecting a subtopic for the Create Web applications document 1902. For example, the subtopic Prepare Database 1940 corresponds to the link Prepare Database 1935, and selection of either of these pointers would take the user to the same document. Upon selection of a pointer, the document or page corresponding to the pointer is then displayed to the user.

Referring also to Fig. 19C, if the user selects the list button 1930 in the top frame, a local table of contents window 1320 is displayed. For example, if the user selects the list

button 1930 in Fig. 19A, the window 1320 is spawned, as shown in Fig. 19C. A traditional model for navigating a document is with a hierarchical table of contents. The TOC model orients the user at all times but restricts the content, which may not fit a hierarchy. A newer model for navigating a document is the hypertext web. The web model links any topic in a document with a more detailed document that expands on the same topic. The web model removes artificial restrictions but quickly disorients the user.

In contrast, the web version of byExample® uses a local table of contents model that provides a structured hierarchical view at a local corner of an unstructured web page of links. In this model, some of the pages in the document are root pages for the local TOC. For example, the page corresponding to the Create Web applications document 1900 is a root page for the local TOC. When the user navigates to any page that is unique to the local TOC, such as the root page (which is unique), the framework applet reads the local TOC for that root page. As seen in Fig. 19D, when the user navigates to the Introduction page in the local TOC 1320, the framework applet reads the local TOC for that root page and displays the corresponding information in the TOC frame 1315 and the content frame 1305. Each page in the local TOC is unique within the local TOC. The user can step through the pages in the local TOC sequentially, see the number of the page in the local TOC sequence, or view the local TOC hierarchy with the current page selected. Thus, the user is oriented within the local topic.

The local TOC does not constrain links in pages within the local TOC. That is, pages within the local TOC can have links to pages that are not in the local TOC 1320. For example, when the user selects the Informix Web Integration Option link 1942 in the Introduction document, the browser opens an Informix Web Integration Option page 1945 that is not in the local TOC 1320, as shown in Fig. 19E.

Moreover, pages can appear within multiple local TOCs. For example, the page entitled "Create subspace" 1950 in the local TOC in Fig. 19D might appear in another local TOC relating to another document. Only the root page (in this example, the "Create Web applications" page 1900) is constrained to a single local TOC.

The user may navigate through the local TOC directly from the local TOC Window by selecting the topic of interest in the local TOC. Additionally, the user may navigate the through the local TOC via the top frame 1310 using the first page button 1910, the previous

page button 1915, the next page button 1920, or the last page button 1925. These buttons basically turn the pages in the document for the user by moving through the local TOC. For example, if the user selects the next page button 1920 while in the Create Web applications document 1900, the browser displays the next document in the Create Web applications document, which is the "Table of Contents" document 1955, as shown in Fig. 19F.

Referring to Fig. 20, a model 2000 of how the local TOCs 2005 are built is shown. The building of local TOCs imposes ordered views on the unstructured (or random) web of pages 2010, thus facilitating viewing of the web of pages 2010.

Other embodiments are within the scope of the following claims.

The web-based instruction system may support Java, Visual Basic, C, C++, HTML, Perl, JavaScript, SQL, Informix Stored Procedure Language (SPL), Embedded SQL for C(ESQL/C), SQLJ, JSP, ASP, and Informix Web DataBlade Module languages.

What is claimed is:

1. A method, performed in a web-based environment on a computer system, of helping a user learn to implement an application, the method comprising:

providing a predetermined plurality of applications;

presenting an annotation page that includes one or more annotations descriptive of a source file of a predetermined application, each annotation including keyword links, annotation links, and detail of implementation of the application;

permitting the user to select a link in an annotation;

if the user selects a keyword link, presenting reference documentation associated with that keyword; and

if the user selects an annotation link, presenting another annotation descriptive of another source file of a predetermined application.

2. The method of claim 1 further comprising performing a predetermined application and presenting one or more annotations descriptive of the performed application in coordination with performance of the predetermined application.

3. The method of claim 2 in which performing the predetermined application comprises receiving input from the user.

4. The method of claim 3 further comprising presenting another annotation page in coordination with performance of the predetermined application based on input from the user.

5. The method of claim 4 in which presenting another annotation page comprises:

automatically and simultaneously calling an annotation request module including application, file, class and function names of a program unit for which detail should be displayed;

mapping the request to an annotation; and

informing a browser window in the web-based environment to display the other annotation page.

6. The method of claim 3 in which another annotation page is presented in coordination with performance of the predetermined application.

7. The method of claim 6 further comprising automatically generating a global table of contents comprising links to annotations by parsing structured links in web pages including annotation pages.

8. The method of claim 7 in which the links in the global table of contents are synchronized with presented annotations by highlighting links corresponding to a current annotation page.

9. The method of claim 8 in which the global table of contents is presented in a first frame of a first browser window, the annotation page is presented in a second frame of the first browser window, and the predetermined application is performed in a second browser window.

10 The method of claim 2 in which performing the predetermined application comprises launching a Java applet or application.

11. The method of claim 10 in which launching the Java applet or application comprises calling a Java application programming interface to ask a web browser to show the annotation page.

12. The method of claim 2 in which performing the predetermined application comprises downloading a hyper-text markup language page containing a Java applet.

13. The method of claim 2 in which performing the predetermined application comprises sending a common gateway interface request to a web server that launches the application in a window in the web-based environment.

14. The method of claim 13 in which the application returns a hyper-text markup language page that includes JavaScript to ask a web browser to display the one or more annotations.

15. The method of claim 2 in which the annotation page is presented in a first browser window and the predetermined application is performed in a second browser window.

16. The method of claim 1 in which application implementation detail includes text descriptive of the application, fragments of source code from the application, or both.

17. The method of claim 16 in which source code fragments are imported directly from the source code file of the presented application.

18. The method of claim 1 further comprising automatically generating the annotation page descriptive of the source code file of a predetermined application.

19. The method of claim 18 in which generating the annotation page comprises:
receiving a source code file that has embedded text marked up with instructions;
parsing the source code to determine a structure of the predetermined application; and
generating one or more annotations based on the predetermined application structure and instructions.

20. The method of claim 19 in which generating the annotation page comprises:
generating one or more annotation links for navigating the annotations of the predetermined application;
generating application implementation detail based on the embedded information; and
generating one or more keyword links for reference documentation.

21. The method of claim 20 in which generating the annotation page comprises highlighting the keyword links and the annotation links in the annotation page.

22. The method of claim 19 further comprising automatically updating the annotation page descriptive of the source code file of the predetermined application when an updated source code file is received.

23. The method of claim 1 further comprising automatically generating a global table of contents by parsing the plurality of annotations for annotation links.

24. The method of claim 23 further comprising providing the global table of contents, in which the global table of contents comprises links to annotations.

25. The method of claim 23 further comprising generating a local table of contents, in which the local table of contents comprises links to web pages including annotation pages relating to an application.

26. The method of claim 25 further comprising providing the local table of contents when a local link in the global table of contents is selected.

27. The method of claim 1 in which the presented annotation page is descriptive of the performed application and the annotation page is presented in coordination with performance of the predetermined application.

28. The method of claim 1 further comprising:

generating a source code file stripped of annotation mark up, the generated source code file including source code of the application but not including text from the annotations;

presenting the stripped source code file; and

permitting the user to edit the stripped source code file.

29. A method, performed in a web-based environment on a computer system, of teaching user to implement an application, the method comprising:

providing a predetermined plurality of applications;

performing a predetermined application; and

presenting an annotation page descriptive of a performed application in coordination with performance of the predetermined application, the annotation page including detail of application implementation and links to annotations and reference documentation.

30. A method, performed in a web-based environment on a computer system, of teaching a user to implement an application, the method comprising:

automatically assembling a global table of contents based on content in the environment, the global table of contents including a plurality of links to content within the environment;

providing the global table of contents;

generating a local table of contents that includes links to content that orient the user within a local topic; and

permitting the user to select links from the local table of contents to access local topics.

31. A method, performed in a web-based environment on a computer system, of teaching a user to implement an application, the method comprising:

providing a plurality of predefined interactive examples;

performing one or more of the predefined interactive examples in response to user selection;

presenting one or more annotations descriptive of the performed interactive example in coordination with performance of the predefined interactive example; and

allowing the user to selectively explore different aspects of the performed interactive example, the annotations, or both.

32. A web-based computer system for teaching a user to implement an application, the system comprising:

one or more predefined interactive applications, a predefined interactive application selectively executable by the user of the web-based computer system; and

an annotation page including one or more annotations, in which the annotation page describes a predefined interactive application, and the annotation page further includes:

one or more links, and

detail of implementation of the application,

in which different annotations are automatically provided in the annotation page in response to selective execution of a predefined interactive application.

33. The system of claim 32 further comprising a utility through which the user can access source code associated with a predefined interactive application.

34. The system of claim 33 in which the utility enables the user to view or copy a predefined interactive application's source code.

35. The system of claim 32 in which detail of implementation of the application comprises text descriptive of the application, fragments of source code associated with the application, or both.

36. The system of claim 32 in which a link comprises a keyword link that provides the user with access to a body of reference documentation or an annotation link that provides the user with access to another annotation page.

37. The system of claim 32 further comprising a web-browser window that includes a framework that comprises:

a content frame that displays the annotations;

a framework applet that displays a navigation bar; and

a table of contents frame that displays a table of contents hierarchy of links.

38. The system of claim 37 in which the framework applet comprises a Java applet.

39. The system of claim 37 in which a Java Script automatically determines whether the framework is present in the web browser window, and if the framework is present, notifies the framework applet about the content in the framework.

40. The system of claim 39 in which the table of contents automatically highlights a link in the hierarchy based on the content in the framework.

41. The system of claim 40 in which the user accesses an annotation page by selecting a link in the table of contents hierarchy.

42. The system of claim 40 in which the user accesses an annotation page by interacting with the navigation bar.

43. The system of claim 40 in which the table of contents highlights the hierarchy based on an annotation page displayed in the content frame.

44. The system of claim 37 in which the table of contents is dismissible or resizable.

45. A web-based computer system for teaching a user to implement an application, the system comprising:

a web-browser window that includes a content frame, a framework applet, and a table of contents frame that displays a global table of contents hierarchy of links related to content in the content frame;

one or more annotations displayed in the content frame, each annotation describing a predefined interactive application and including links to other content; and

a table of contents window that displays a local table of contents hierarchy of links related to local content in the displayed annotation.

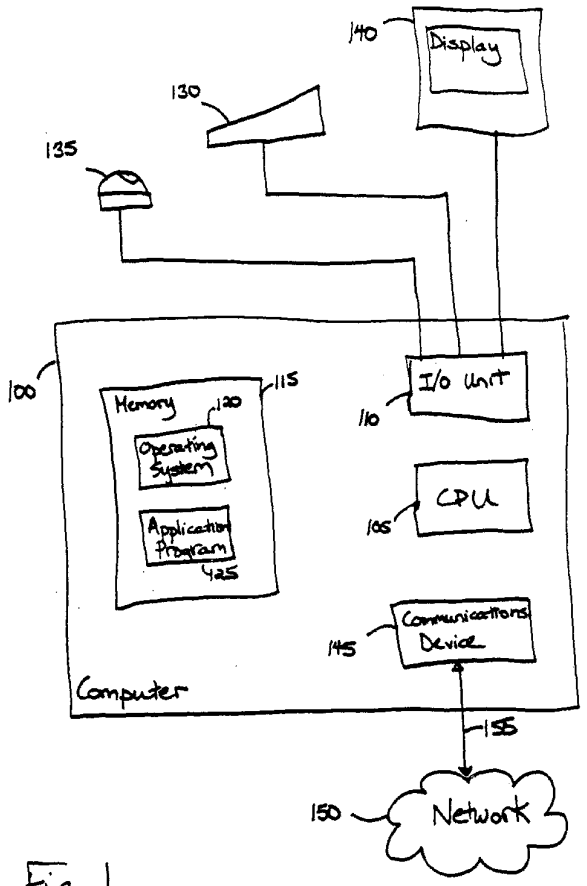

140
Display
130
135
100
Memory
115
120
Operating System
Application Program
425
I/O Unit
110
CPU
105
Communications Device
145
Computer
155
150
Network

Fig. 1

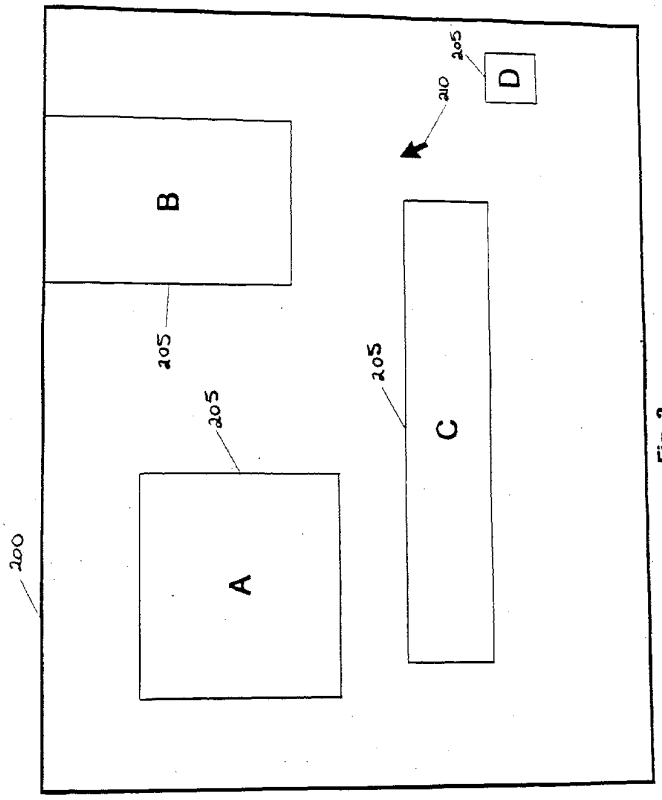


Fig. 2
PRIOR ART

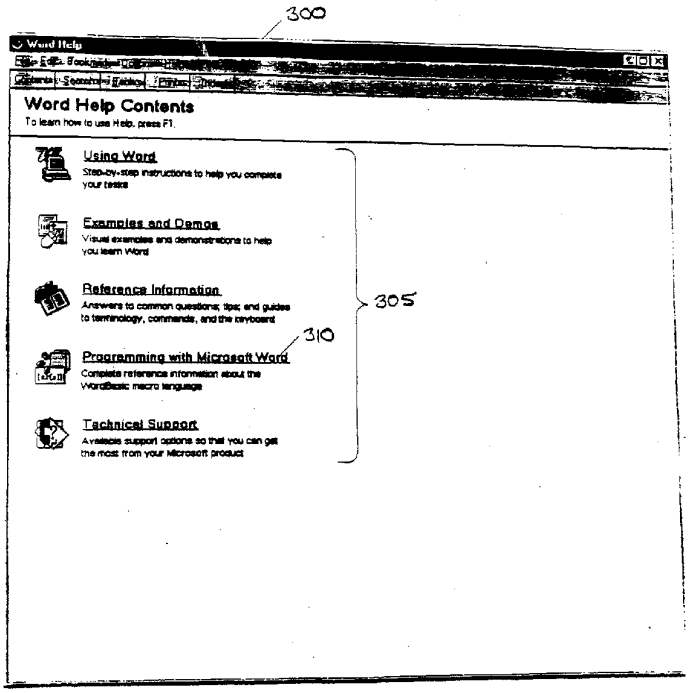


Fig. 3A
PRIOR ART

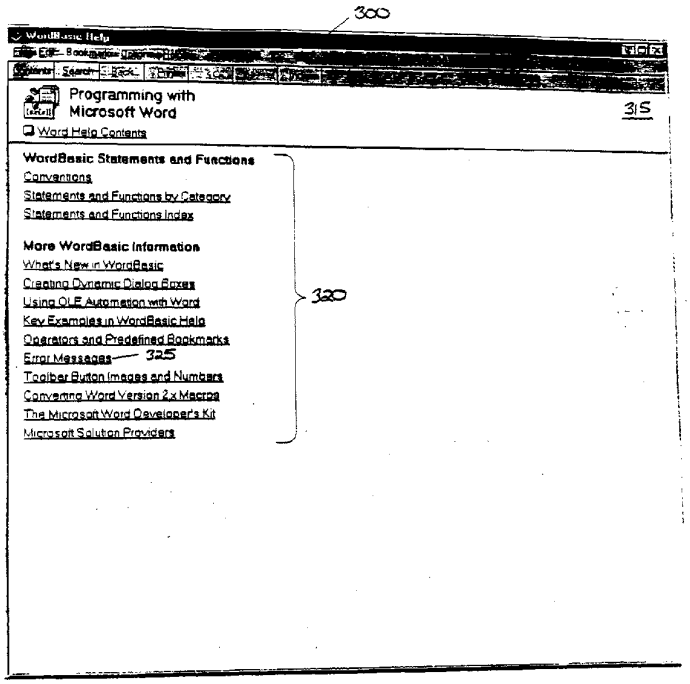


Fig. 3B
PRIOR ART

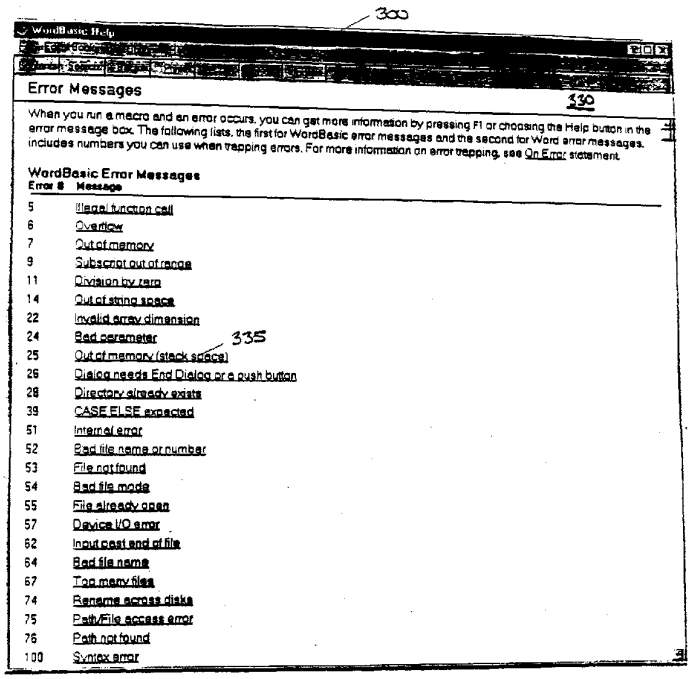


Fig. 3C
PRIOR ART

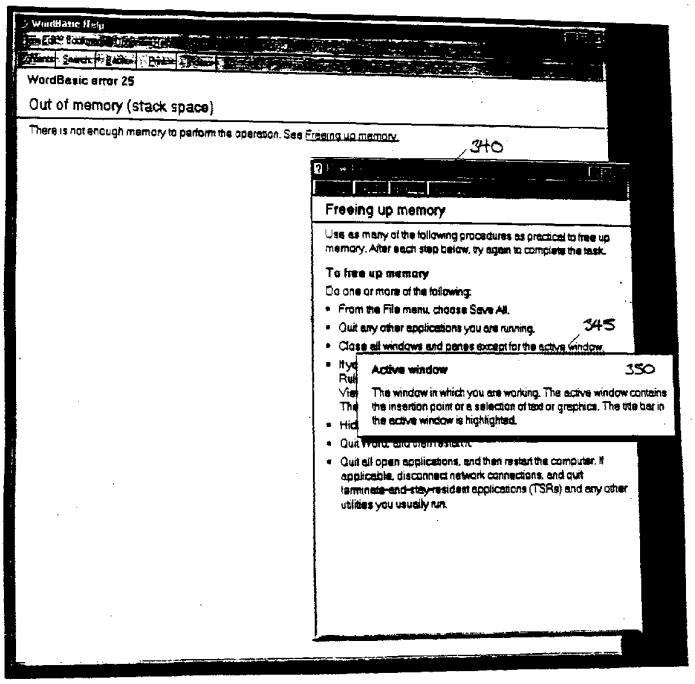


Fig. 3D
PRIOR ART

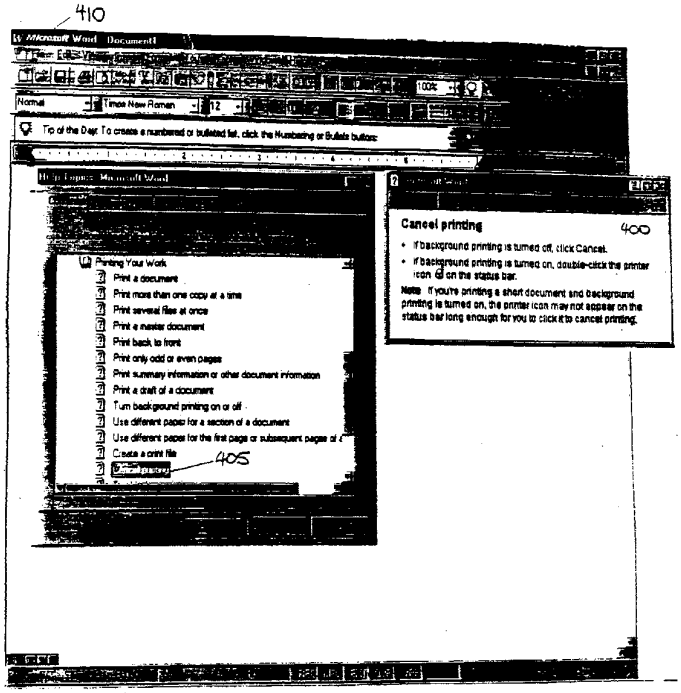


Fig. 4
PRIOR ART

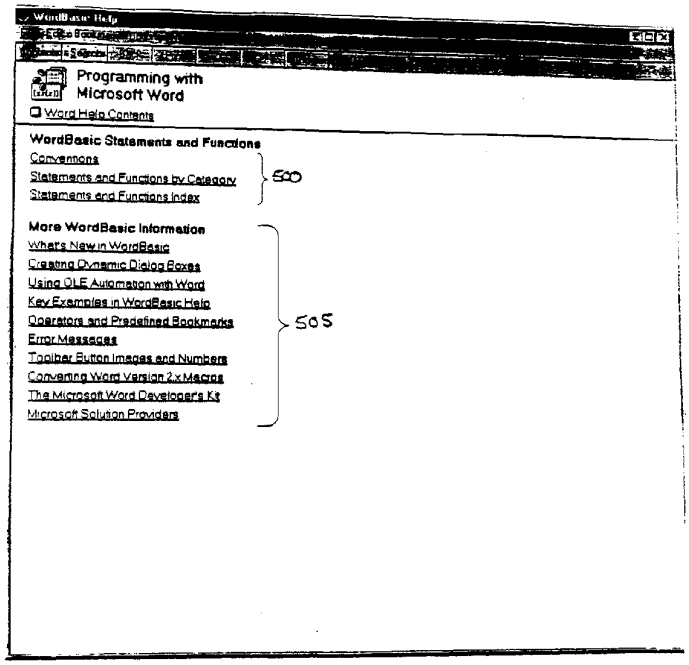
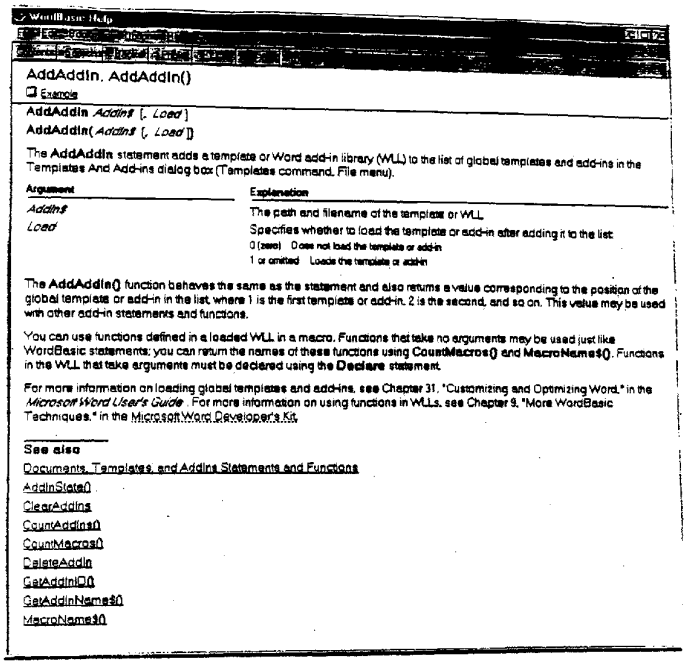


Fig. 5A
PRIOR ART



WordBasic Help

AddAddIn, AddAddIn()

Example

AddAddIn *AddIn$* [, *Load*]
AddAddIn(*AddIn$* [, *Load*])

The **AddAddIn** statement adds a template or Word add-in library (WLL) to the list of global templates and add-ins in the Templates And Add-ins dialog box (Templates command, File menu).

Argument	Explanation
AddIn$	The path and filename of the template or WLL
Load	Specifies whether to load the template or add-in after adding it to the list: 0 (zero) Does not load the template or add-in 1 or omitted Loads the template or add-in

The **AddAddIn()** function behaves the same as the statement and also returns a value corresponding to the position of the global template or add-in in the list, where 1 is the first template or add-in, 2 is the second, and so on. This value may be used with other add-in statements and functions.

You can use functions defined in a loaded WLL in a macro. Functions that take no arguments may be used just like WordBasic statements; you can return the names of these functions using **CountMacros()** and **MacroName$()**. Functions in the WLL that take arguments must be declared using the **Declare** statement.

For more information on loading global templates and add-ins, see Chapter 31, "Customizing and Optimizing Word," in the *Microsoft Word User's Guide*. For more information on using functions in WLLs, see Chapter 9, "More WordBasic Techniques," in the Microsoft Word Developer's Kit.

See also

Documents, Templates, and Addins Statements and Functions
AddInState()
ClearAddIns
CountAddIns()
CountMacros()
DeleteAddIn
GetAddInID()
GetAddInName$()
MacroName$()

Fig. 5B
PRIOR ART

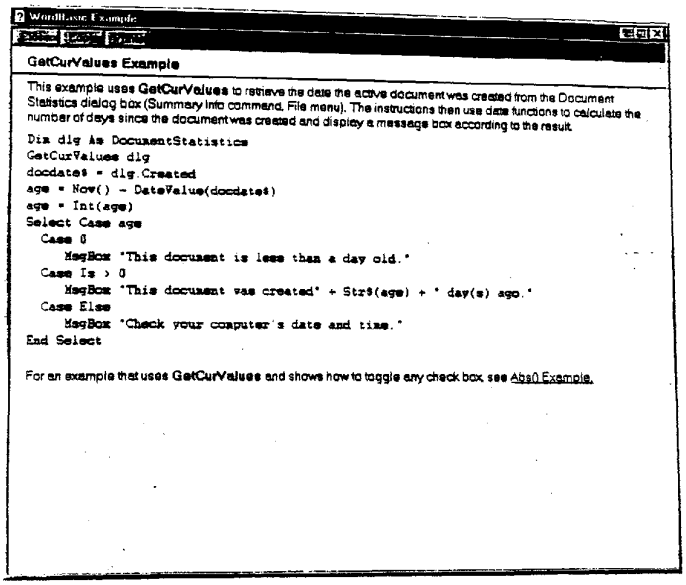


Fig. 5C
PRIOR ART

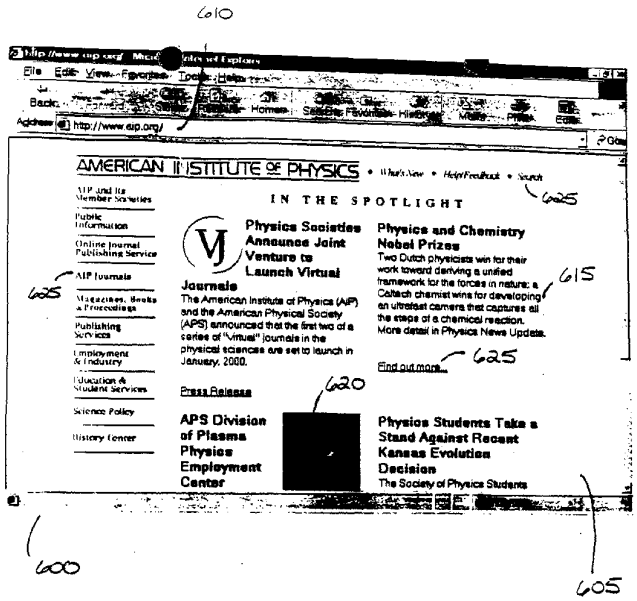

610
http://www.aip.org/
AMERICAN INSTITUTE OF PHYSICS
IN THE SPOTLIGHT
Physics Societies Announce Joint Venture to Launch Virtual Journals
The American Institute of Physics (AIP) and the American Physical Society (APS) announced that the first two of a series of "virtual" journals in the physical sciences are set to launch in January, 2000.
Press Release
Physics and Chemistry Nobel Prizes
Two Dutch physicists win for their work toward deriving a unified framework for the forces in nature; a Caltech chemist wins for developing an ultrafast camera that captures all the steps of a chemical reaction.
More detail in Physics News Update.
Find out more...
APS Division of Plasma Physics Employment Center
Physics Students Take a Stand Against Recent Kansas Evolution Decision
The Society of Physics Students
625
615
620
600
605

Fig. 6

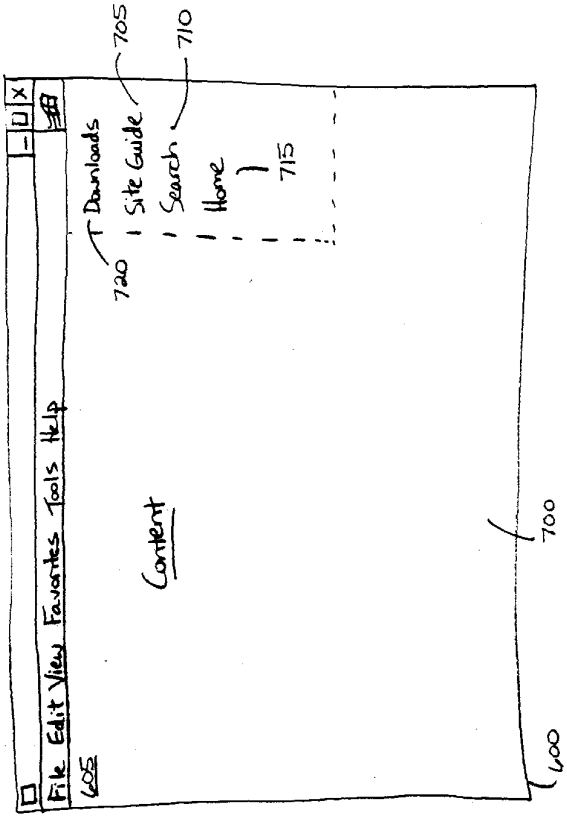

705
710
Downloads
Site Guide
Search
Home
715
720
File Edit View Favorites Tools Help
605
Content
700
600

Fig. 7

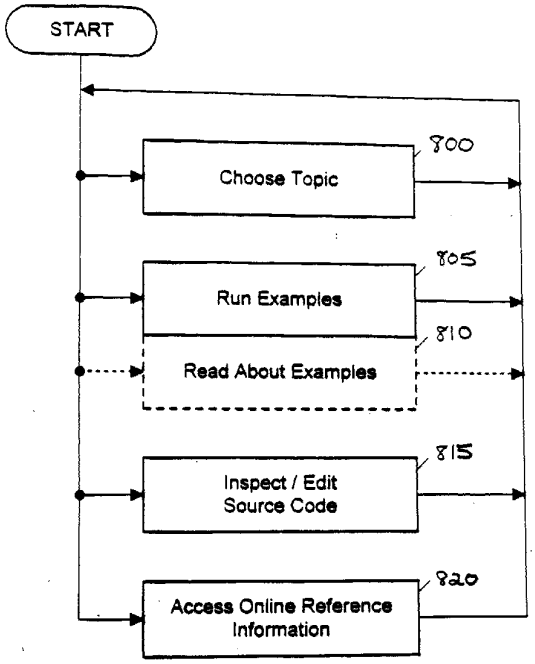

START
800
Choose Topic
805
Run Examples
810
Read About Examples
815
Inspect / Edit
Source Code
820
Access Online Reference
Information

Fig. 8

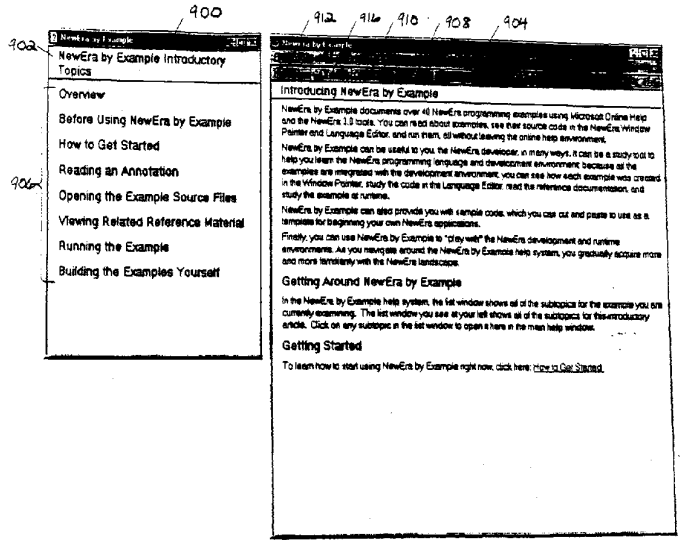

900
902
904
906
908
910
912
916
NewEra by Example Introductory Topics
Overview
Before Using NewEra by Example
How to Get Started
Reading an Annotation
Opening the Example Source Files
Viewing Related Reference Material
Running the Example
Building the Examples Yourself
Introducing NewEra by Example
Getting Around NewEra by Example
Getting Started

Fig. 9A

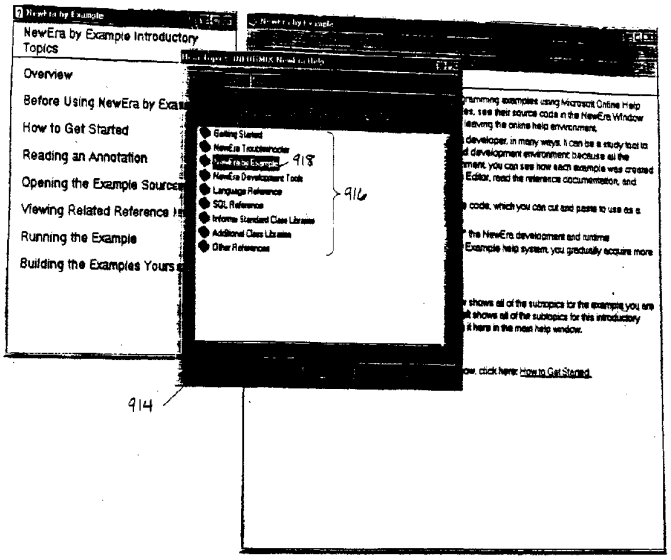

NewEra by Example Introductory Topics
Overview
Before Using NewEra by Exa
How to Get Started
Reading an Annotation
Opening the Example Source
Viewing Related Reference H
Running the Example
Building the Examples Yours
Getting Started
NewEra Troubleshooter
NewEra Examples
NewEra Development Tools
Language Reference
SQL Reference
Informix Standard Class Libraries
Additional Class Libraries
Other References
918
916
914

Fig. 9B

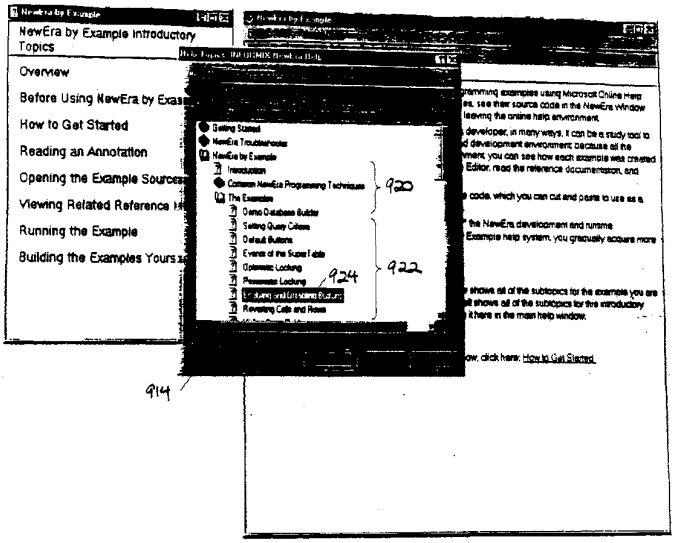

NewEra by Example Introductory Topics
Overview
How to Get Started
Reading an Annotation
Opening the Example Source
Running the Example
Building the Examples Yours
920
922
924
914

Fig. 9C

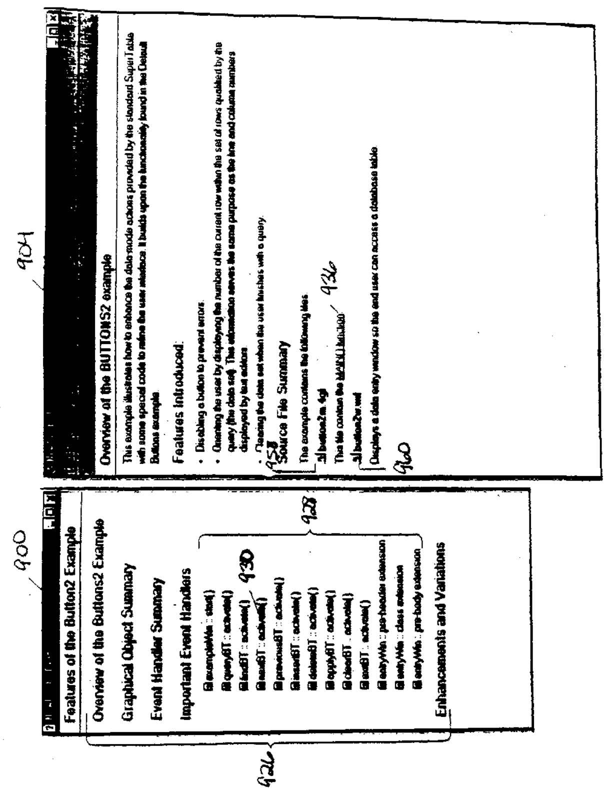

900
Features of the Button2 Example
Overview of the Buttons2 Example
Graphical Object Summary
Event Handler Summary
Important Event Handlers
Enhancements and Variations
926
928
930
904
Overview of the BUTTONS2 example
Features Introduced:
Source File Summary
936
958
960

Fig. 9D

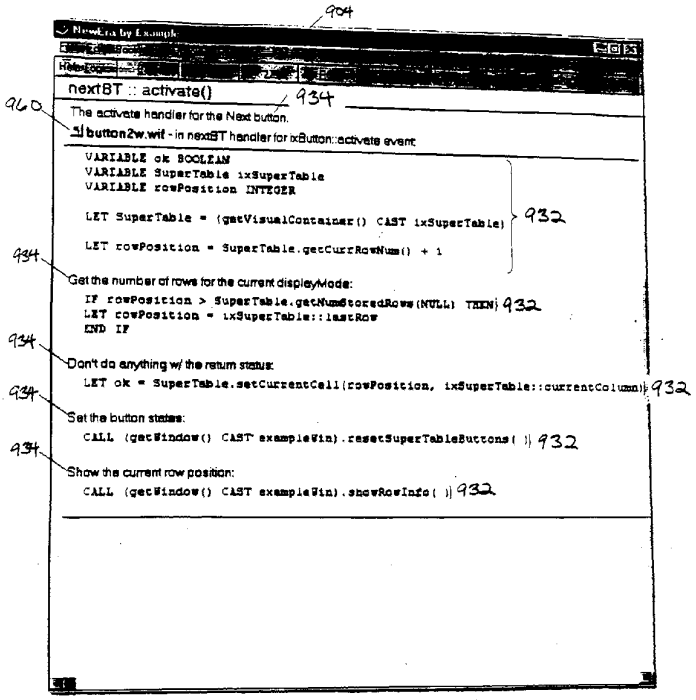

904
nextBT :: activate()
934
960
The activate handler for the Next button.
button2w.wtf - in nextBT handler for ixButton::activate event
VARIABLE ok BOOLEAN
VARIABLE SuperTable ixSuperTable
VARIABLE rowPosition INTEGER
LET SuperTable = (getVisualContainer() CAST ixSuperTable)
LET rowPosition = SuperTable.getCurrRowNum() + 1
932
934
Get the number of rows for the current displayMode:
IF rowPosition > SuperTable.getNumStoredRows(NULL) THEN
LET rowPosition = ixSuperTable::lastRow
END IF
932
934
Don't do anything w/ the return status:
LET ok = SuperTable.setCurrentCell(rowPosition, ixSuperTable::currentColumn)
932
934
Set the button states:
CALL (getWindow() CAST exampleWin).resetSuperTableButtons( )
932
934
Show the current row position:
CALL (getWindow() CAST exampleWin).showRowInfo( )
932

Fig. 4E

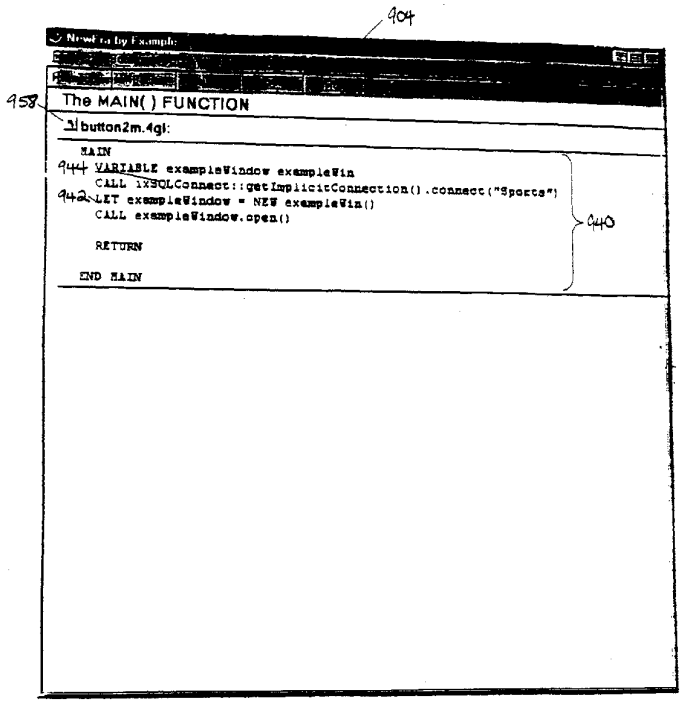

904
The MAIN( ) FUNCTION
958
button2m.4gl:
MAIN
944 VARIABLE exampleWindow exampleWin
CALL 1XSQLConnect::getImplicitConnection().connect("Sports")
942 LET exampleWindow = NEW exampleWin()
CALL exampleWindow.open()
RETURN
END MAIN
940

Fig. 9F

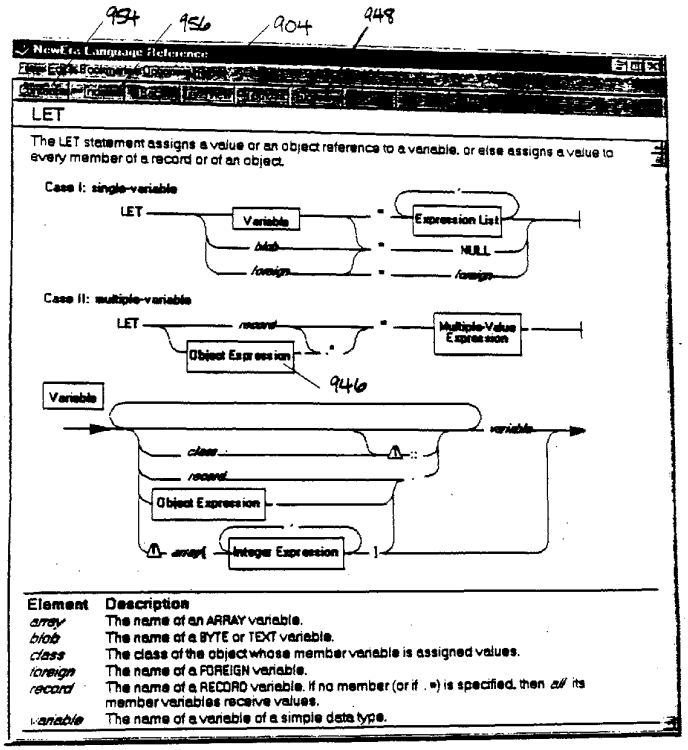

954
956
904
948
NewEra Language Reference
LET
The LET statement assigns a value or an object reference to a variable, or else assigns a value to every member of a record or of an object.
Case I: single-variable
LET
Variable
=
Expression List
blob
NULL
foreign
Case II: multiple-variable
record
Multiple-Value Expression
Object Expression
946
Variable
variable
class
::
record
Object Expression
array(
Integer Expression
)
Element Description
array The name of an ARRAY variable.
blob The name of a BYTE or TEXT variable.
class The class of the object whose member variable is assigned values.
foreign The name of a FOREIGN variable.
record The name of a RECORD variable. If no member (or if .*) is specified, then all its member variables receive values.
variable The name of a variable of a simple data type.

Fig. 9G

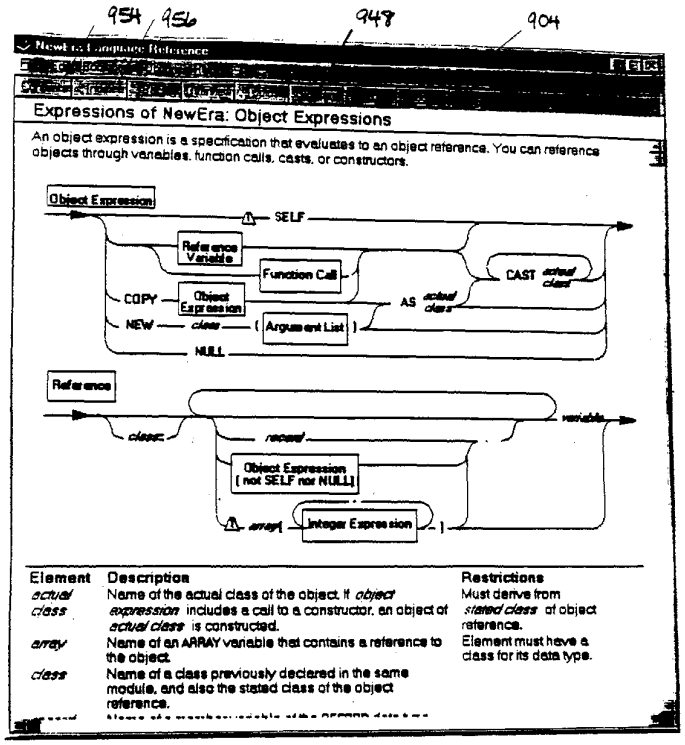

954
956
948
904
Expressions of NewEra: Object Expressions
An object expression is a specification that evaluates to an object reference. You can reference objects through variables, function calls, casts, or constructors.
Object Expression
SELF
Reference Variable
Function Call
CAST actual class
COPY
Object Expression
AS actual class
NEW
class
(Argument List)
NULL
Reference
class:
record
variable
Object Expression (not SELF nor NULL)
array[
Integer Expression
]
Element
Description
Restrictions
actual class
Name of the actual class of the object. If object expression includes a call to a constructor, an object of actual class is constructed.
Must derive from stated class of object reference.
array
Name of an ARRAY variable that contains a reference to the object.
Element must have a class for its data type.
class
Name of a class previously declared in the same module, and also the stated class of the object reference.

Fig. 9H

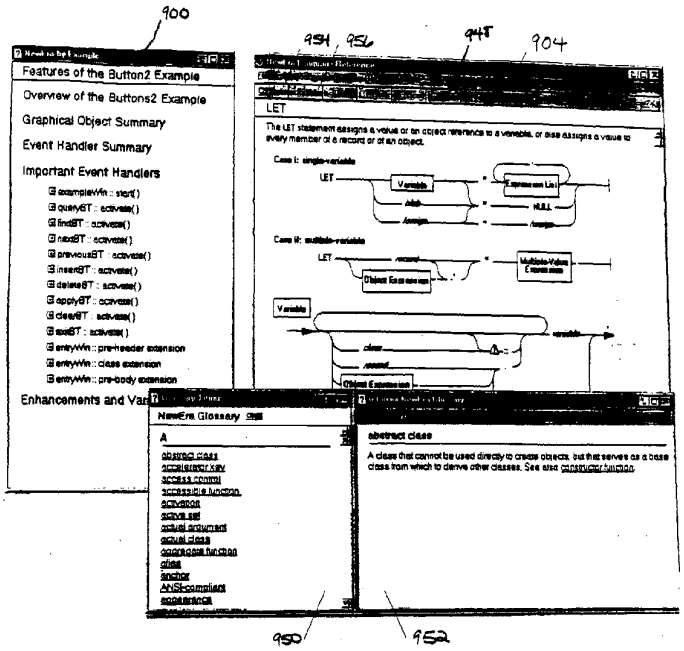

900
Features of the Button2 Example
Overview of the Buttons2 Example
Graphical Object Summary
Event Handler Summary
Important Event Handlers
LET
950
952

Fig. 9I

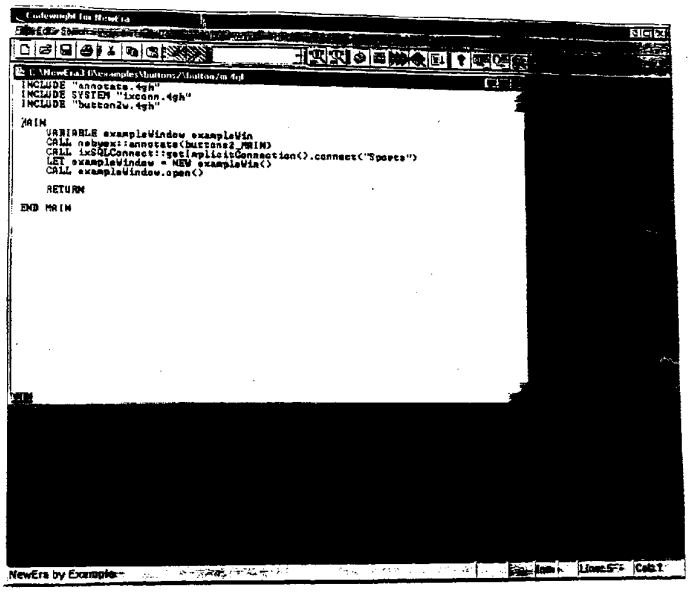

INCLUDE "annotate.4gh"
INCLUDE SYSTEM "ixconn.4gh"
INCLUDE "button2w.4gh"
MAIN
VARIABLE exampleWindow exampleWin
CALL ixSQLConnect::getImplicitConnection().connect("Spoets")
LET exampleWindow = NEW exampleWin()
CALL exampleWindow.open()
RETURN
END MAIN
NewEra by Example

Fig. 4J

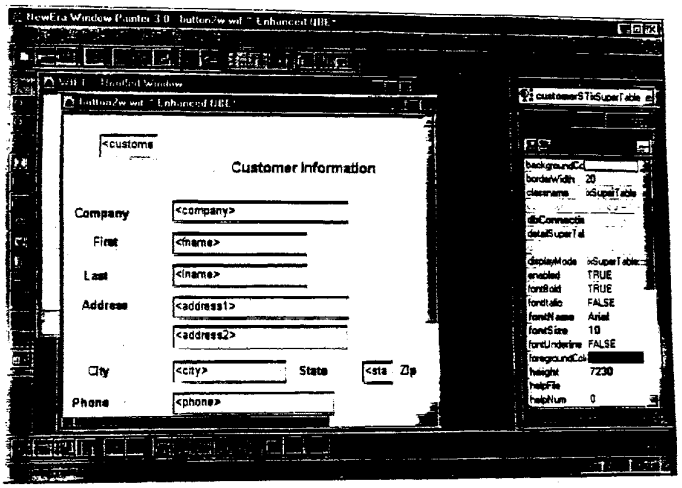

Customer Information
Company
First
Last
Address
City
State
Zip
Phone

Fig. 4K

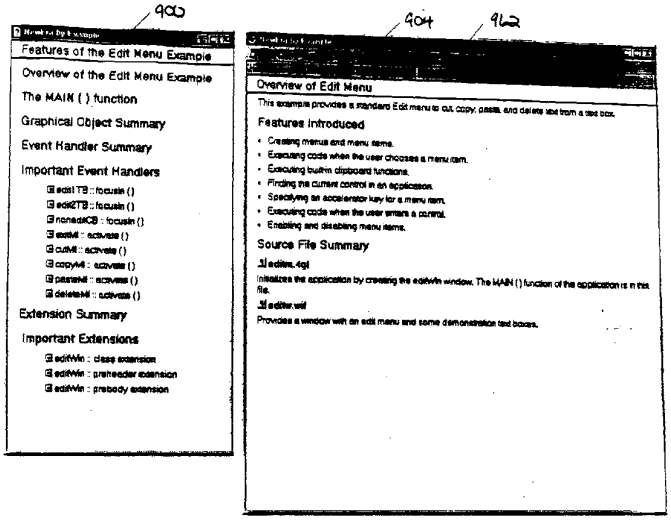

900
904
902
Features of the Edit Menu Example
Overview of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusIn ( )
edit2TB :: focusIn ( )
noneditCB :: focusIn ( )
editM :: activate ( )
cutM :: activate ( )
copyM :: activate ( )
pasteM :: activate ( )
deleteM :: activate ( )
Extension Summary
Important Extensions
editWin : class extension
editWin : preheader extension
editWin : prebody extension
Overview of Edit Menu
This example provides a standard Edit menu to cut, copy, paste, and delete text from a text box.
Features Introduced
Creating menus and menu items.
Executing code when the user chooses a menu item.
Executing built-in clipboard functions.
Finding the current control in an application.
Specifying an accelerator key for a menu item.
Executing code when the user enters a control.
Enabling and disabling menu items.
Source File Summary
editex.cpp
Initializes the application by creating the editWin window. The MAIN ( ) function of the application is in this file.
editex.wid
Provides a window with an edit menu and some demonstration text boxes.

Fig. 9L

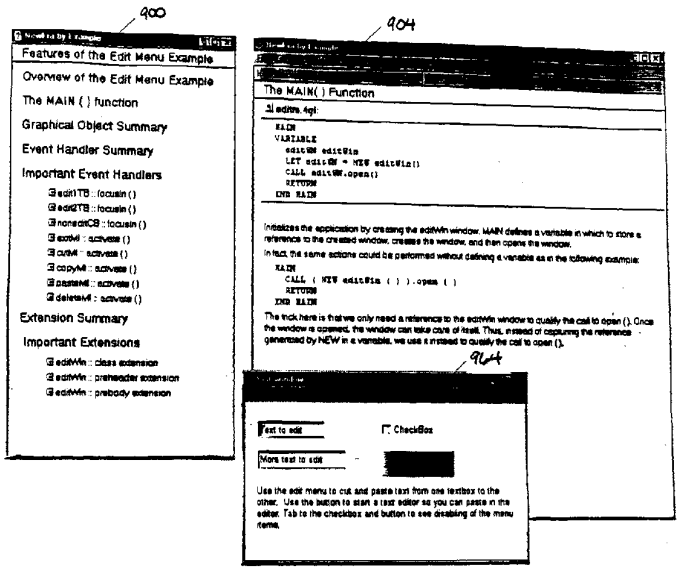

900
Features of the Edit Menu Example
Overview of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
Extension Summary
Important Extensions
904
The MAIN( ) Function
964
Text to edit
CheckBox
More text to edit

Fig. 9M

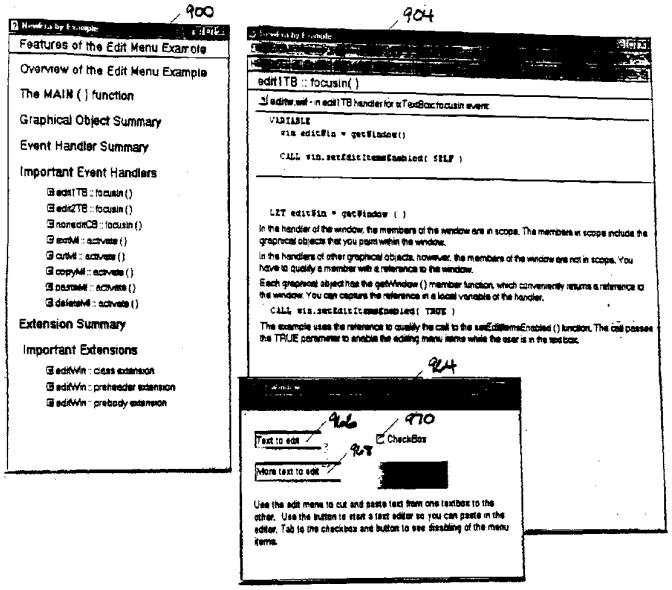

900
Features of the Edit Menu Example
Overview of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusin ( )
edit2TB :: focusin ( )
nonedit2B :: focusin ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteMI :: activate ( )
Extension Summary
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
904
edit1TB :: focusin ( )
924
926
910
CheckBox
928
Text to edit
More text to edit
Use the edit menu to cut and paste text from one textbox to the other. Use the button to start a text editor so you can paste in the editor. Tab to the checkbox and button to see disabling of the menu items.

Fig. 9N

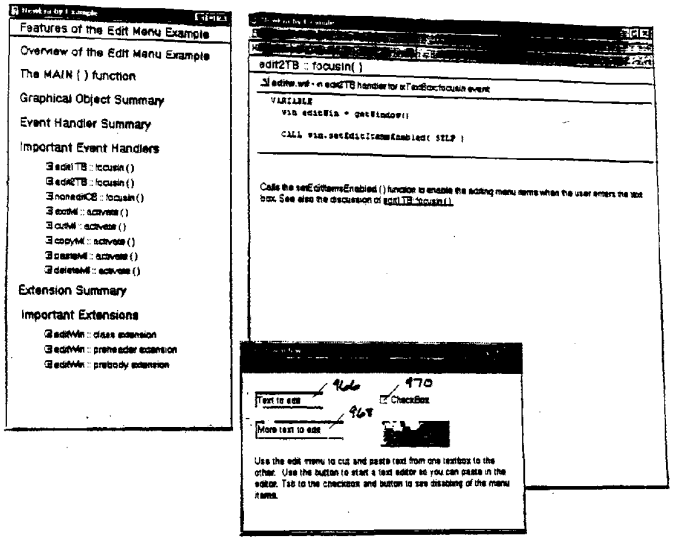

Features of the Edit Menu Example
Overview of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
Extension Summary
Important Extensions
edit2TB :: focusIn( )
Text to edit
CheckBox
More text to edit
970
968

Fig. 90

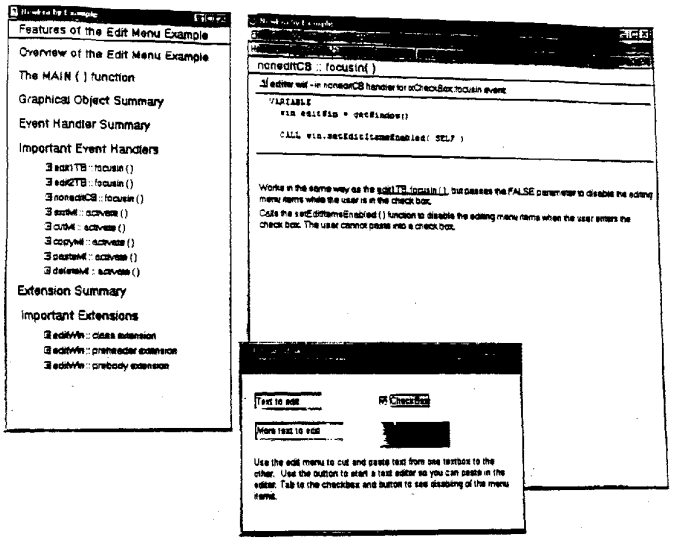

Features of the Edit Menu Example
Overview of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
Extension Summary
Important Extensions
noneditCB :: focusin( )
Text to edit
CheckBox
More text to edit

Fig. 9P

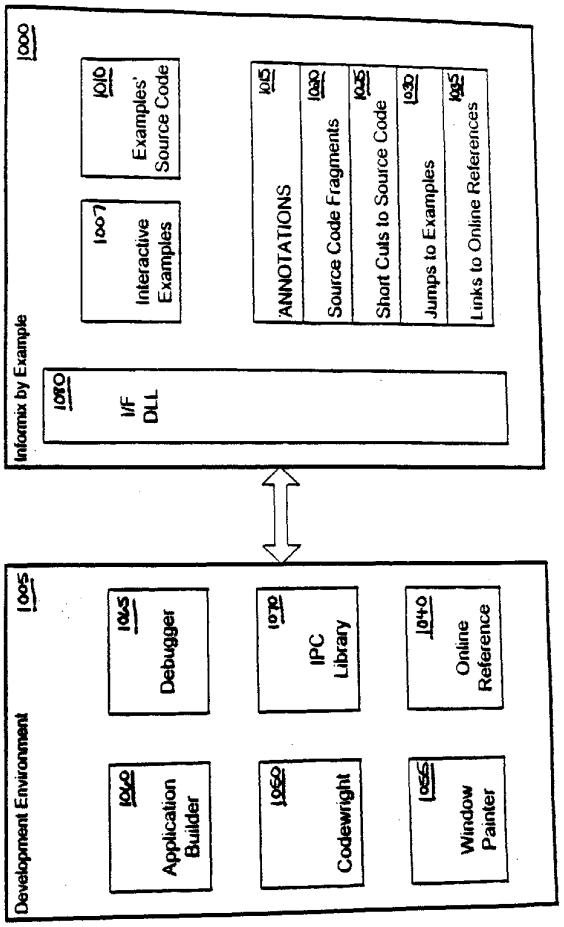

1000
Informix by Example
1010
Examples' Source Code
1007
Interactive Examples
1015
ANNOTATIONS
1020
Source Code Fragments
1025
Short Cuts to Source Code
1030
Jumps to Examples
1035
Links to Online References
1050
I/F DLL
1005
Development Environment
1045
Debugger
1070
IPC Library
1040
Online Reference
1040
Application Builder
1060
Codewright
1055
Window Painter

Fig. 10

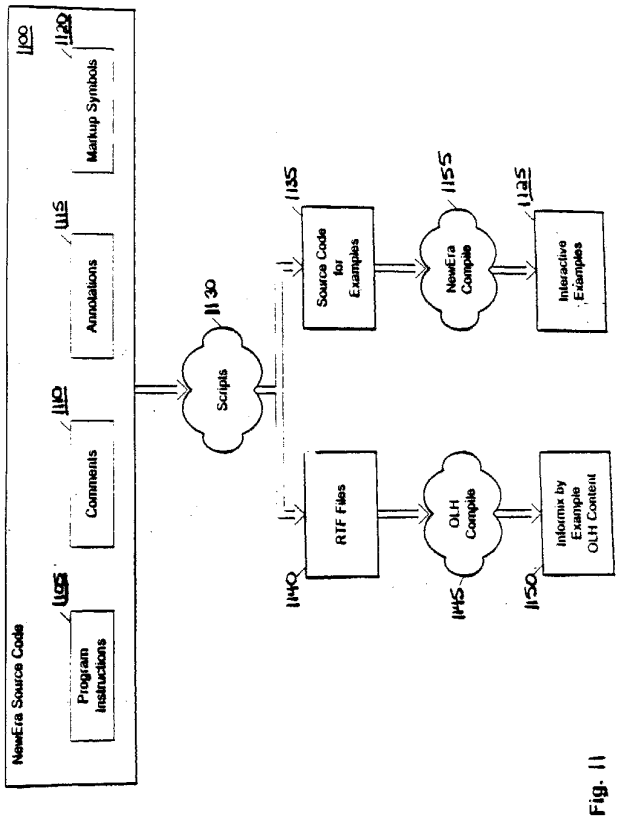

1100
NewEra Source Code
Program Instructions
1105
Comments
1110
Annotations
1115
Markup Symbols
1120
1130
Scripts
1135
Source Code for Examples
1140
RTF Files
1145
NewEra Compile
OLH Compile
1125
Interactive Examples
1150
Informix by Example OLH Content

Fig. 11

```
FUNCTION driveStockRpt( destType SMALLINT, destName CHAR(*) ) RETUR
NING VOID
{.normal
Since objects, in particular ixRow objects, cannot be passed
as arguments to the report formatter, rows of fetched data will
be unpacked into a record that matches the data types and lengths
of elements in the fetched rows.
}
VARIABLE
     stockRec RECORD
          mn CHAR(15),     -- manufact.manu_name
          sn SMALLINT,     -- stock.stock_num
          sd CHAR(15),     -- stock.description
          sp MONEY(6,2),   -- stock.unit_price
          su CHAR(4)       -- stock.unit
          END RECORD,

     stockStmt ixSQLStmt,
     stmtString CHAR(*),
     stockRow ixRow,

     errorCode INTEGER,
     logFile ixErrorLog
{.normal
Use the implicit connection object to create an SQL statement
object. The connection object must already be connected to a
database.
Checking the status of the prepare( ) call will confirm this.
}
{.[edit stmt}
LET stockStmt =
ixSQLConnect::getImplicitConnection().createStmtObject()
{.]file stmt}
```

Fig. 12

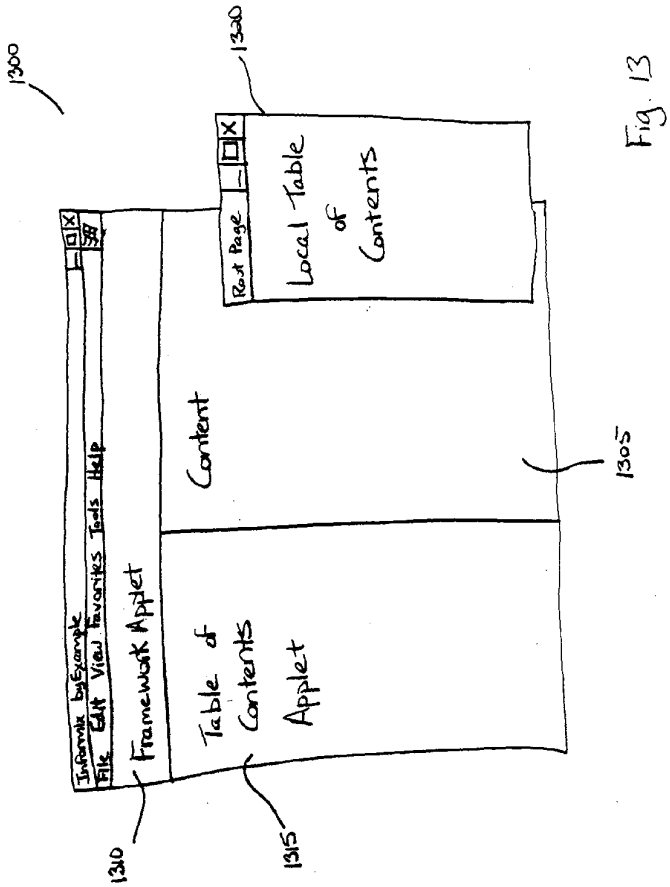

1300
1320
Root Page
Local Table of Contents
Content
1305
Framework Applet
Table of Contents Applet
1310
1315

Fig. 13

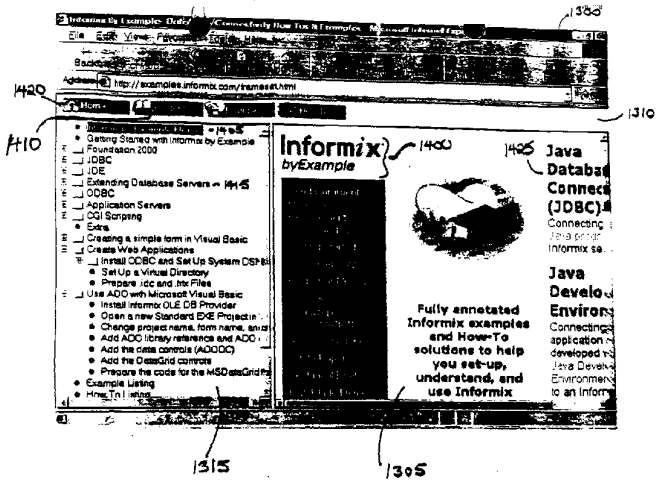

1420
1410
1310
1400
1425
Informix
byExample
Java
Fully annotated Informix examples and How-To solutions to help you set-up, understand, and use Informix
1315
1305

Fig. 14A

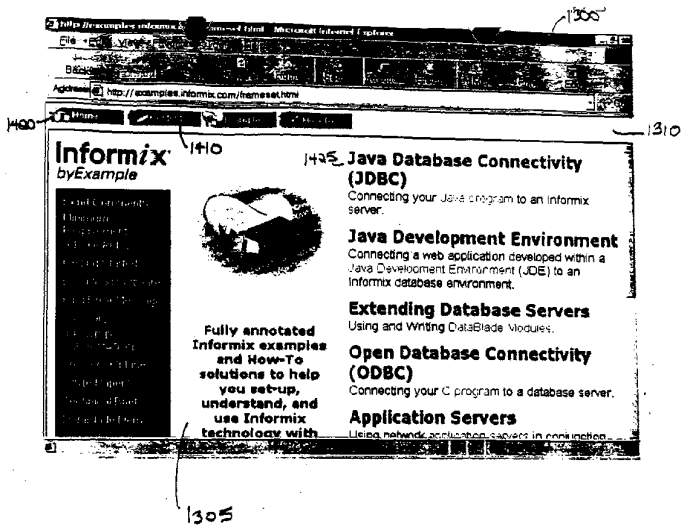

1350
1400
1410
1310
1425
Informix
byExample
Java Database Connectivity (JDBC)
Connecting your Java program to an Informix server.
Java Development Environment
Connecting a web application developed within a Java Development Environment (JDE) to an Informix database environment.
Extending Database Servers
Using and Writing DataBlade Modules.
Open Database Connectivity (ODBC)
Connecting your C program to a database server.
Application Servers
Fully annotated Informix examples and How-To solutions to help you set-up, understand, and use Informix technology with
1305

Fig. 14B

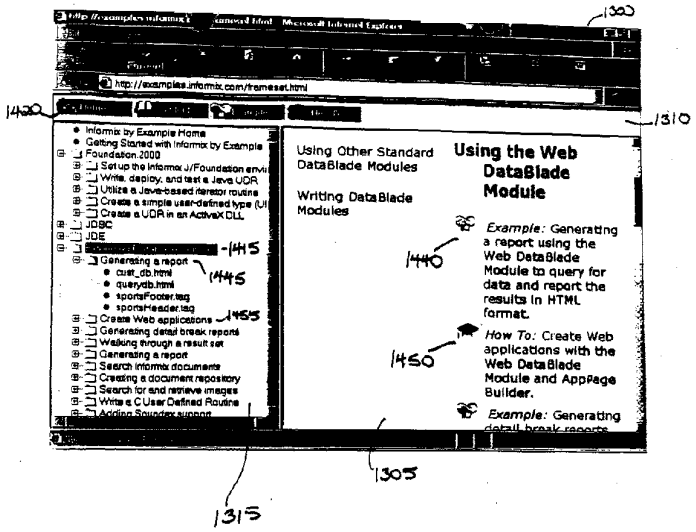

1320
http://examples.informix.com/frameset.html
1430
1310
Informix by Example Home
Getting Started with Informix by Example
Foundation.2000
JDBC
JDE
Generating a report
cust_db.html
querydb.html
sportsFooter.tag
sportsHeader.tag
Create Web applications
Generating detail break reports
Walking through a result set
Generating a report
Search Informix documents
Creating a document repository
Search for and retrieve images
Write a C User Defined Routine
1445
1455
Using Other Standard DataBlade Modules
Writing DataBlade Modules
Using the Web DataBlade Module
1440
Example: Generating a report using the Web DataBlade Module to query for data and report the results in HTML format.
1450
How To: Create Web applications with the Web DataBlade Module and AppPage Builder.
Example: Generating
1305
1315

Fig. 14C

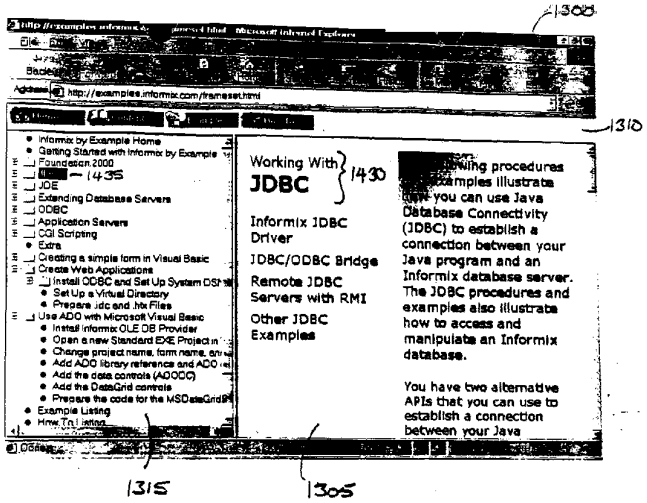

1308
http://examples.informix.com/frameset.html
1310
Informix by Example Home
Getting Started with Informix by Example
Foundation 2000
1435
JDE
Extending Database Servers
ODBC
Application Servers
CGI Scripting
Extra
Creating a simple form in Visual Basic
Create Web Applications
Install ODBC and Set Up System DSN
Set Up a Virtual Directory
Prepare .idc and .htx Files
Use ADO with Microsoft Visual Basic
Install Informix OLE DB Provider
Open a new Standard EXE Project in
Change project name, form name, and
Add ADO library reference and ADO
Add the data controls (ADODC)
Add the DataGrid controls
Prepare the code for the MSDataGrid
Example Listing
How To Listing
Working With
JDBC
1430
Informix JDBC Driver
JDBC/ODBC Bridge
Remote JDBC Servers with RMI
Other JDBC Examples
wing procedures
amples illustrate
you can use Java Database Connectivity (JDBC) to establish a connection between your Java program and an Informix database server. The JDBC procedures and examples also illustrate how to access and manipulate an Informix database.
You have two alternative APIs that you can use to establish a connection between your Java
1315
1305

Fig. 14D

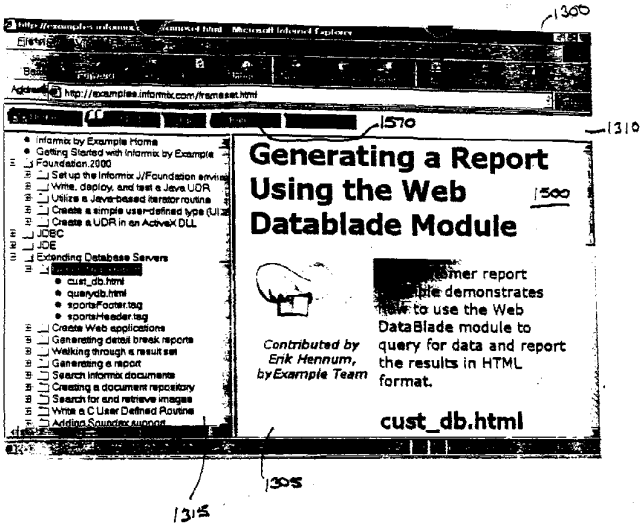

1305
1570
1310
Generating a Report Using the Web Datablade Module
1500
Contributed by Erik Hennum, byExample Team
demonstrates to use the Web DataBlade module to query for data and report the results in HTML format.
cust_db.html
1315
1305

Fig. 15A

Generating a Report Using the Web Datablade Module

1500

The customer report example demonstrates how to use the Web DataBlade module to query for data and report the results in HTML format.

1505

Contributed by Erik Hennum, byExample Team

cust_db.html – 1510

1505

This app page accepts a query and generates an HTML report

querydb.html – 1510

1505

This HTML page contains a form that invokes an app page

sportsFooter.tag – 1510

1505

The sportsFooter dynamic tag generates the footer for an app page.

sportsHeader.tag – 1510

1505

The sportsHeader dynamic tag generates the header for an app page.

Click here to view or print all of the source files for this example.

Copyright © 1999 Informix Software. All Rights Reserved.
Terms and Conditions governing the use of this website.

Fig. 15B

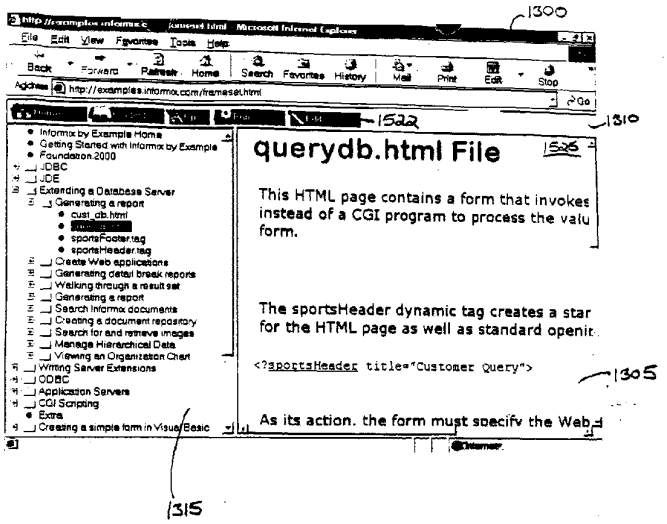

1300
File Edit View Favorites Tools Help
Back Forward Refresh Home Search Favorites History Mail Print Edit Stop
Address http://examples.informix.com/frameset.html
1522
1310
Informix by Example Home
Getting Started with Informix by Example
Foundation.2000
JDBC
JDE
Extending a Database Server
Generating a report
cust_db.html
sportsFooter.tag
sportsHeader.tag
Create Web applications
Generating detail break reports
Walking through a result set
Generating a report
Search Informix documents
Creating a document repository
Search for and retrieve images
Manage Hierarchical Data
Viewing an Organization Chart
Writing Server Extensions
ODBC
Application Servers
CGI Scripting
Extra
Creating a simple form in Visual Basic
querydb.html File
1525
This HTML page contains a form that invokes
instead of a CGI program to process the valu
form.
The sportsHeader dynamic tag creates a star
for the HTML page as well as standard openit
<?sportsHeader title="Customer Query">
1305
As its action, the form must specify the Web
1315

Fig. 15C

querydb.html File

1525

This HTML page contains a form that invokes an app page instead of a CGI program to process the values in the form.

The sportsHeader dynamic tag creates a standard header for the HTML page as well as standard opening text.

1535

```
<?sportsHeader title="Customer Query">
```

1540

1530

As its action, the form must specify the Web Driver utility.

```
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR" METHOD="GET">
```

1530

1560

To specify the app page, the form must use a hidden input component. The input component must have a name of **MIval** and a value that's the name of the app page. The input component below specifies the cust_db.html app page.

1535

```
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPE="TEXT" NAME="selectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">

</FORM>
</P>


</BODY>
</HTML>
```

1530

Copyright © 2000 Informix Software. All Rights Reserved. Terms and Conditions governing the use of this website.

Fig. 15D

```
<!-- <ibyx>
<intro>
<p><abstract>This HTML page contains a form that invokes
an app page</abstract> instead of a CGI program to process
the values in the form.
</p>
</intro>
</ibyx> -->

<!-- <ibyx>
<p>The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> -->
<?sportsHeader title="Customer Query">

<!-- <ibyx>
<p>As its action, the form must specify the Web Driver utility.
</p>
</ibyx> -->
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR>" METHOD="GET">

<!-- <ibyx>
<p>To specify the app page, the form must use a hidden input component.
The input component must have a name of <strong>MIval</strong> and
a value that's the name of the app page.  The input component below
specifies the <a href="cust_db.html">cust_db.html</a> app page.
</p>
</ibyx> -->
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPE="TEXT" NAME="selectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">

</FORM>
</P>

<?annotate>

</BODY>
</HTML>
```

1535

1535

1560

Fig. 15E

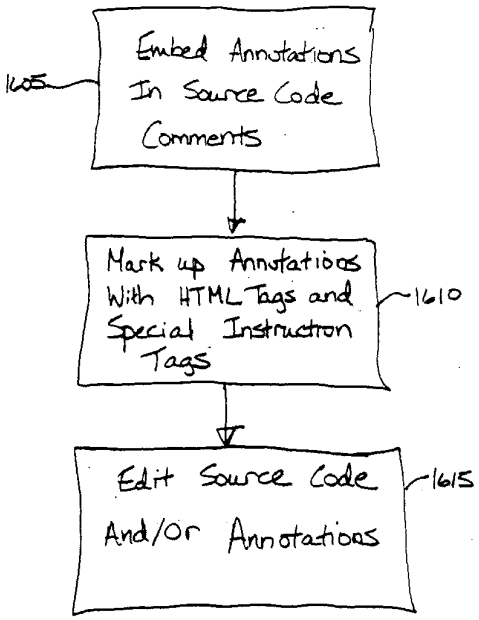

Embed Annotations In Source Code Comments
1605
Mark up Annotations With HTML Tags and Special Instruction Tags
1610
Edit Source Code And/Or Annotations
1615

Fig. 16A

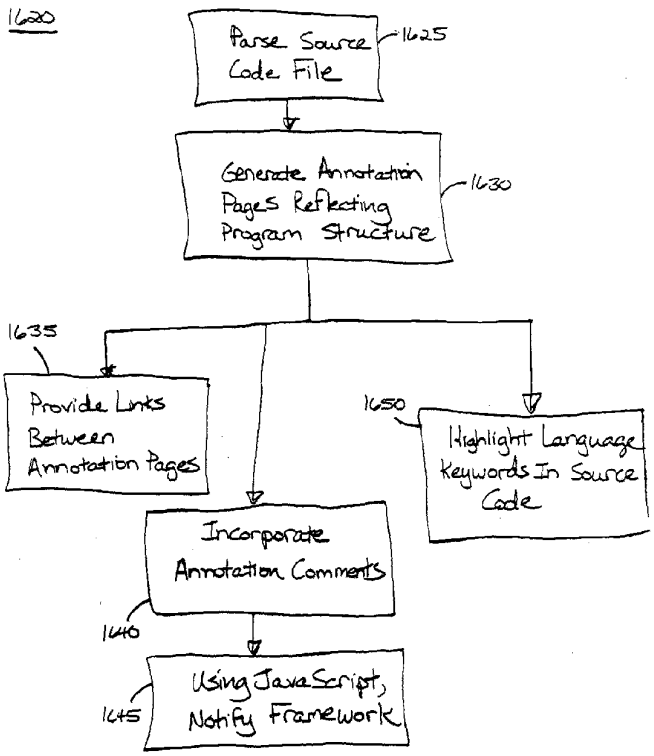
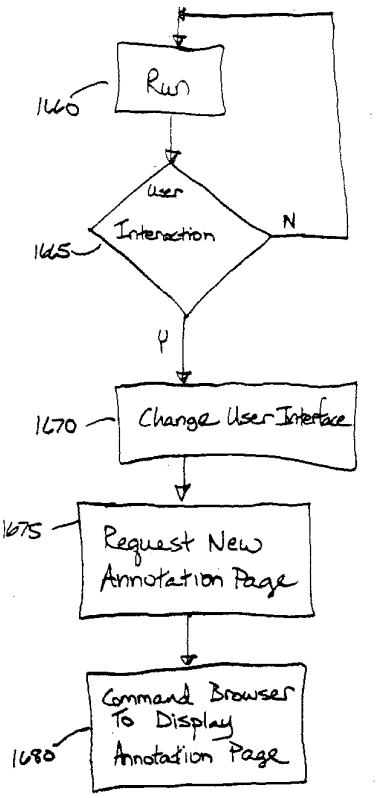

1620
Parse Source Code File
1625
Generate Annotation Pages Reflecting Program Structure
1630
1635
Provide Links Between Annotation Pages
1650
Highlight Language Keywords In Source Code
Incorporate Annotation Comments
1640
Using JavaScript, Notify Framework
1645

Fig. 16B


Run
1660
User Interaction
1665
N
Y
Change User Interface
1670
Request New Annotation Page
1675
Command Browser To Display Annotation Page
1680

1655

Fig. 16C

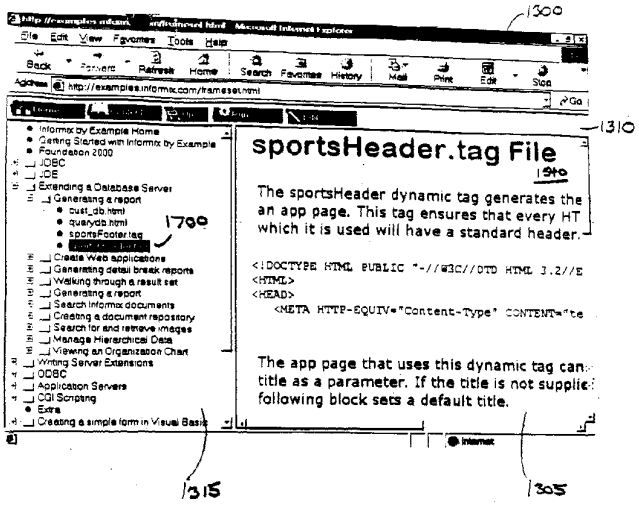

1300
1310
sportsHeader.tag File
1910
The sportsHeader dynamic tag generates the
an app page. This tag ensures that every HT
which it is used will have a standard header.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//E
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="te
The app page that uses this dynamic tag can
title as a parameter. If the title is not supplie
following block sets a default title.
1700
1315
1305

Fig. 17A

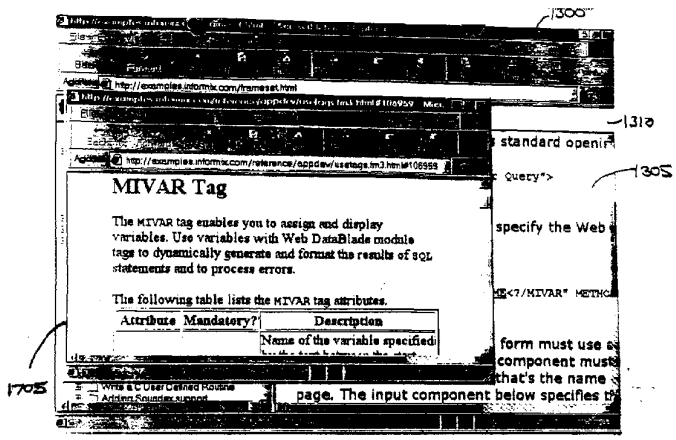

1300
1310
1305
1705
MIVAR Tag
The MIVAR tag enables you to assign and display variables. Use variables with Web DataBlade module tags to dynamically generate and format the results of SQL statements and to process errors.
The following table lists the MIVAR tag attributes.
Attribute
Mandatory?
Description
Name of the variable specified
standard openir
Query*>
specify the Web
form must use a
component must
that's the name
page. The input component below specifies th

Fig. 17B

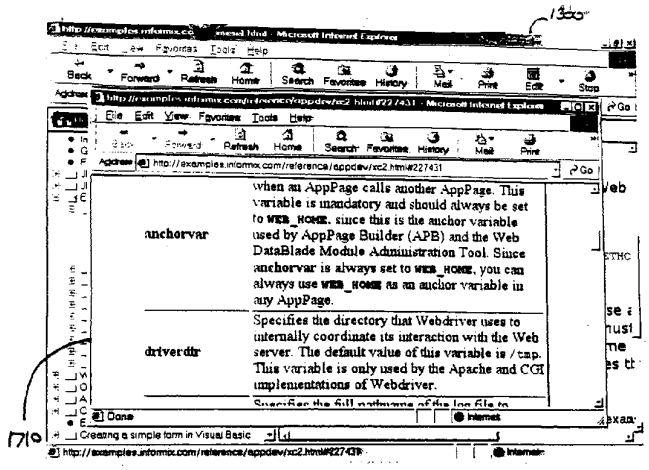

1350
1710
anchorvar
when an AppPage calls another AppPage. This variable is mandatory and should always be set to WEB_HOME, since this is the anchor variable used by AppPage Builder (APB) and the Web DataBlade Module Administration Tool. Since anchorvar is always set to WEB_HOME, you can always use WEB_HOME as an anchor variable in any AppPage.
driverdir
Specifies the directory that Webdriver uses to internally coordinate its interaction with the Web server. The default value of this variable is /tmp. This variable is only used by the Apache and CGI implementations of Webdriver.
Done
Internet
Creating a simple form in Visual Basic

Fig. 17C

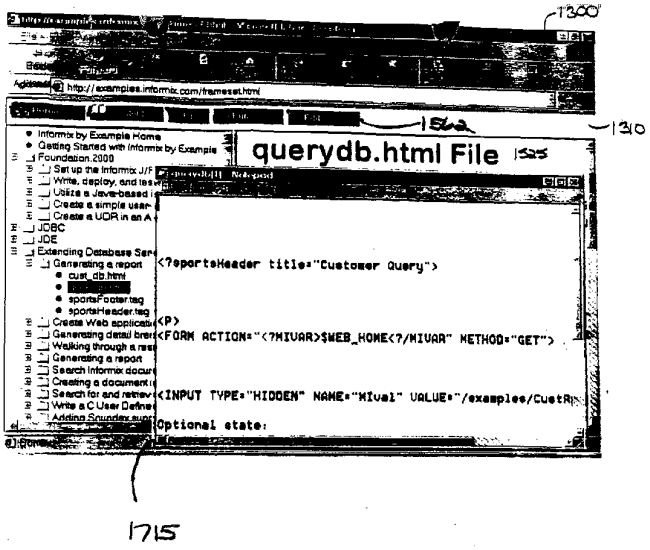

1300
1310
1320
querydb.html File
1325
<?eportsHeader title="Customer Query">
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR" METHOD="GET">
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustR
Optional state:
1715

Fig. 17D

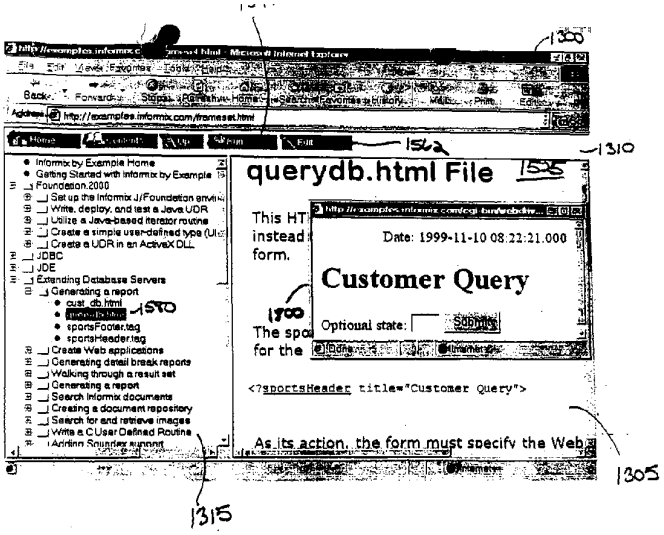

querydb.html File
Date: 1999-11-10 08:22:21.000
Customer Query
Optional state:
Submit
<?sportsHeader title="Customer Query">
As its action, the form must specify the Web
Informix by Example Home
Getting Started with Informix by Example
Foundation.2000
JDBC
JDE
Extending Database Servers
Generating a report
cust_db.html
sportsFooter.tag
sportsHeader.tag
Create Web applications
Generating detail break reports
Walking through a result set
Generating a report
Search Informix documents
Creating a document repository
Search for and retrieve images
Write a C User Defined Routine
1300
1310
1525
1500
1305
1315

Fig. 18A

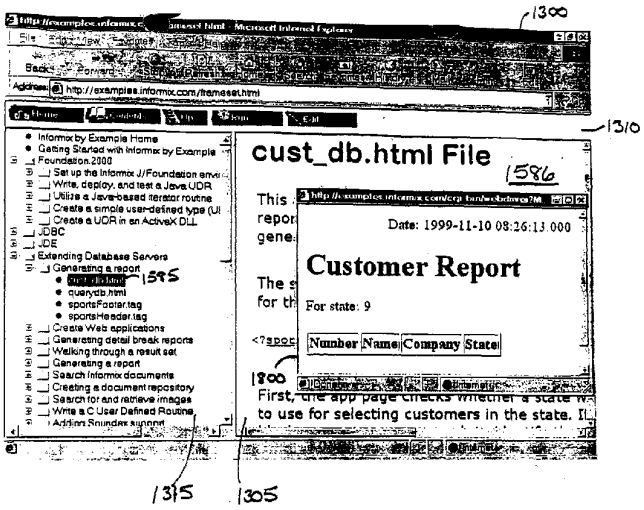

1300
1310
cust_db.html File
1586
Date: 1999-11-10 08:26:13.000
Customer Report
For state: 9
Number
Name
Company
State
1595
1800
1315
1305

Fig. 18B

cust_db.html

```
<!-- <ibyx>
<intro>
<p><abstract>This app page accepts a query and generates an HTML
report</abstract> in response.
The app page uses dynamic tags to generate the header and footer for the
HTML report.
</p>
</intro>
</ibyx> -->
<!-- <ibyx>
<p>The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> -->
<?sportsHeader title="Customer Report">

<!-- <ibyx>
<p>First, the app page checks whether a state was specified to use for
selecting customers in the state.  If so, the block generates a
paragraph to identify the state.
</p>
</ibyx> -->
<?MIVAR NAME=$WHERE_STR><?/MIVAR>
<?MIBLOCK COND="$(AND,$(XST,$selectState),$(<,0,$(STRLEN,$selectState)))">
        <?MIVAR NAME=$WHERE_STR>WHERE state='$selectState'<?/MIVAR>
        <?MIVAR><P>For state: $selectState</P><?/MIVAR>
<?/MIBLOCK>

<!-- <ibyx>
<p>Next, the app page starts the table that will contain the data.
</p>
</ibyx> -->
        <P><TABLE BORDER="1">
                <TR>
                <TH>Number</TH><TH>Name</TH><TH>Company</TH><TH>State</TH>
                </TR>

<!-- <ibyx>
<p>The MISQL block queries for customers, optionally selecting only customers
from the specified state.  Because the contents of the block are generated
for every row of data, a new table row describes each customer.

The &amp;nbsp; HTML entity is a non-breaking space.  By putting a non-breaking
space in each column, we force the Web Browser to display the column even
if the value is null.
</p>
</ibyx> -->.
<?MISQL SQL="SELECT customer_num, fname, lname, company, state FROM customer $WHERE_STR;">
                <TR>
                <TD>$1 </TD><TD>$2 $3</TD><TD>$4 </TD><TD>$5 </TD>
                </TR>
<?/MISQL>

        </TABLE></P>

<!-- <ibyx>
<p>The sportsFooter dynamic tag creates a standard footer
for the HTML page.
</p>
</ibyx> -->
<?sportsFooter>
```

180

Fig. 18C

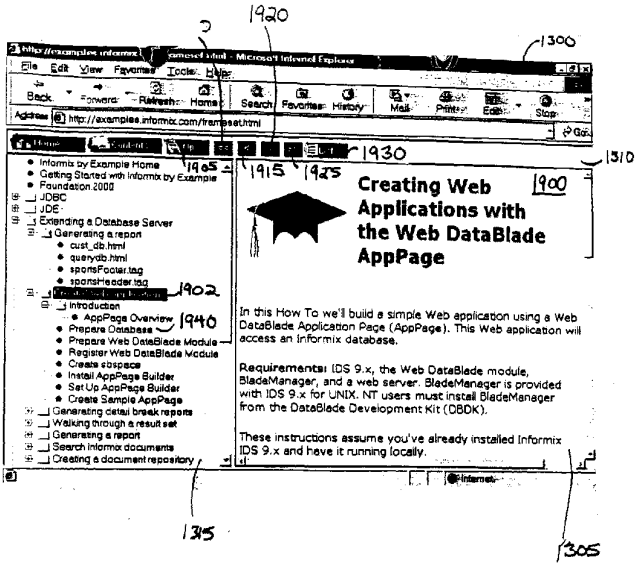

1920
1300
1930
1310
1905
1915
1925
1900
Creating Web Applications with the Web DataBlade AppPage
1902
1940
In this How To we'll build a simple Web application using a Web DataBlade Application Page (AppPage). This Web application will access an Informix database.
Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX. NT users must install BladeManager from the DataBlade Development Kit (DBDK).
These instructions assume you've already installed Informix IDS 9.x and have it running locally.
1315
1305

Fig. 19A

Creating Web Applications with the Web DataBlade AppPage

1930 1900

In this How To we'll build a simple Web application using a Web DataBlade Application Page (AppPage). This Web application will access an Informix database.

Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX. NT users must install BladeManager from the DataBlade Development Kit (DBDK).

These instructions assume you've already installed Informix IDS 9.x and have it running locally.

1935

Define a server connection and prepare a sample database.

1. Define a server connection with setnet32 (NT). Create a sample database or use the stores7 demo database. ▶ Prepare Database.
2. Prepare the Web DataBlade development environment. Install the Web DataBlade module and BladeManager. ▶ Prepare Web DataBlade Development Environment.
3. Register the Web DataBlade module in the demo database with BladeManager. ▶ Register the Web DataBlade.
4. Create a sbspace for smart large objects, like gifs. ▶ Create Smart Blob Space (sbspace).
5. Install AppPage Builder in your database. ▶ Install AppPage Builder in Your Database.
6. Setup AppPage Builder on your web server. ▶ Setup AppPage Builder on Your Web Server.
7. Create a sample AppPage. ▶ Create Sample AppPage.

Run the sample application.

8. Enter the URL `http://your_server/scripts/webdriver.exe`.

This How To has been compiled into two separate files for ease of printing. The basic file contains all of the steps you need to Create Web Applications with AppPage Builder. The secondary file contains additional detailed instructions for setting and testing database environment properties.

Copyright © 1999 Informix Software, All Rights Reserved.
Terms and Conditions governing the use of this website.

Fig. 19B

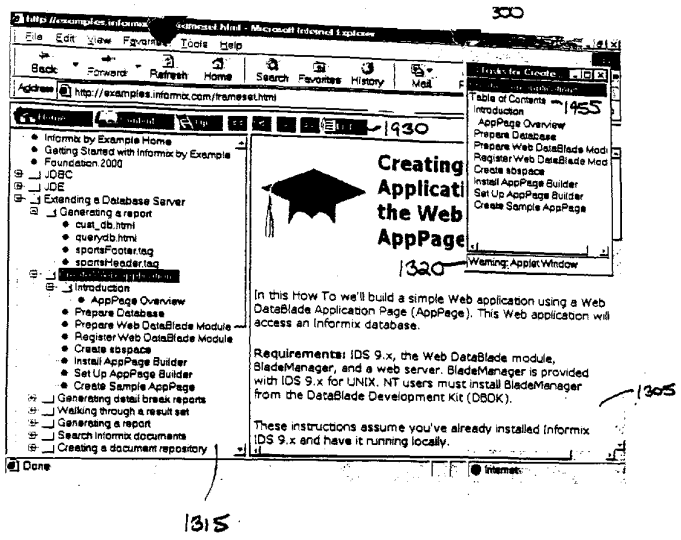

300
1930
1955
1320
1305
1315
Table of Contents
Introduction
AppPage Overview
Prepare Database
Prepare Web DataBlade Module
Register Web DataBlade Module
Create sbspace
Install AppPage Builder
Set Up AppPage Builder
Create Sample AppPage
Warning: Applet Window
Informix by Example Home
Getting Started with Informix by Example
Foundation 2000
JDBC
JDE
Extending a Database Server
Generating a report
cust_db.html
querydb.html
sportsFooter.tag
sportsHeader.tag
Introduction
AppPage Overview
Prepare Database
Prepare Web DataBlade Module
Register Web DataBlade Module
Create sbspace
Install AppPage Builder
Set Up AppPage Builder
Create Sample AppPage
Generating detail break reports
Walking through a result set
Generating a report
Search Informix documents
Creating a document repository
Creating
Applicati
the Web
AppPage
In this How To we'll build a simple Web application using a Web DataBlade Application Page (AppPage). This Web application will access an Informix database.
Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX. NT users must install BladeManager from the DataBlade Development Kit (DBDK).
These instructions assume you've already installed Informix IDS 9.x and have it running locally.
Done
Internet

Fig. 19C

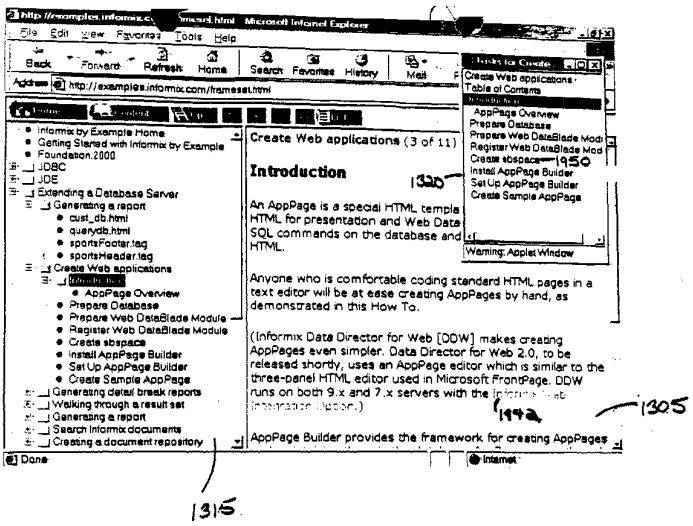

Create Web applications (3 of 11)
Introduction
1320
An AppPage is a special HTML templa
HTML for presentation and Web Data
SQL commands on the database and
HTML.
Anyone who is comfortable coding standard HTML pages in a
text editor will be at ease creating AppPages by hand, as
demonstrated in this How To.
(Informix Data Director for Web [DDW] makes creating
AppPages even simpler. Data Director for Web 2.0, to be
released shortly, uses an AppPage editor which is similar to the
three-panel HTML editor used in Microsoft FrontPage. DDW
runs on both 9.x and 7.x servers with the
1342
1305
AppPage Builder provides the framework for creating AppPages
Done
Internet
1315

Fig. 19D

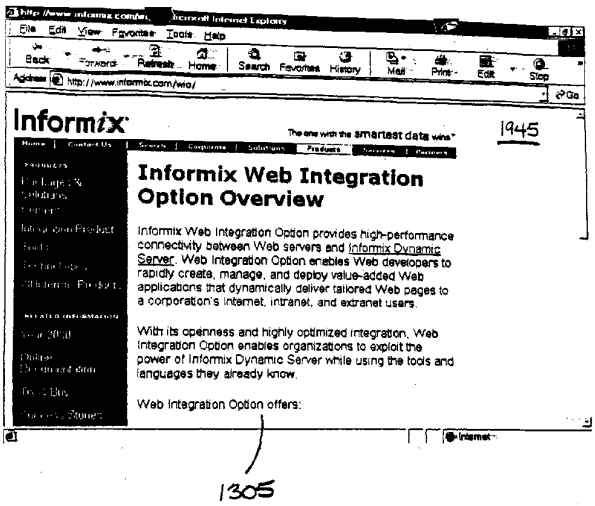

Informix
Informix Web Integration Option Overview
Informix Web Integration Option provides high-performance connectivity between Web servers and Informix Dynamic Server. Web Integration Option enables Web developers to rapidly create, manage, and deploy value-added Web applications that dynamically deliver tailored Web pages to a corporation's Internet, intranet, and extranet users.
With its openness and highly optimized integration, Web Integration Option enables organizations to exploit the power of Informix Dynamic Server while using the tools and languages they already know.
Web Integration Option offers:
1945
1305

Fig. 19E

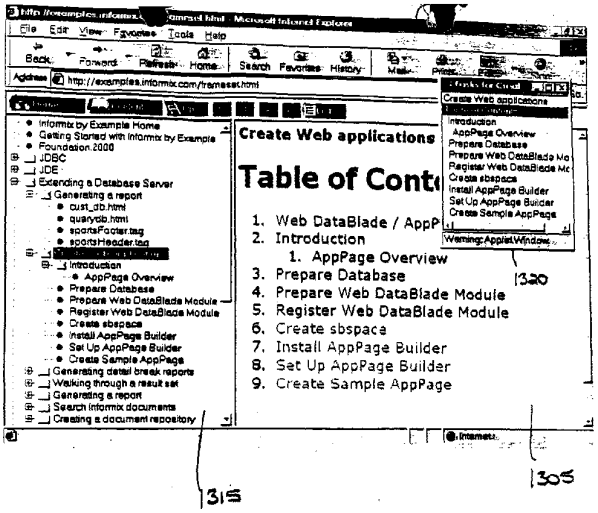

Create Web applications
Table of Cont
1. Web DataBlade / AppP
2. Introduction
1. AppPage Overview
3. Prepare Database
4. Prepare Web DataBlade Module
5. Register Web DataBlade Module
6. Create sbspace
7. Install AppPage Builder
8. Set Up AppPage Builder
9. Create Sample AppPage
315
320
305

Fig. 19F

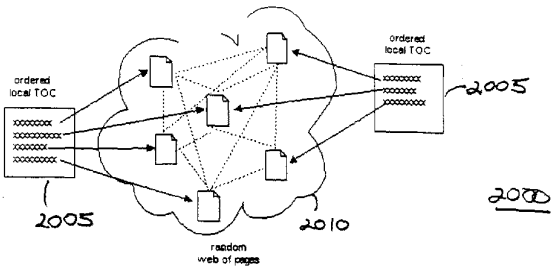

ordered
local TOC
xxxxxxxx
xxxxxxxxx
xxxxxxxx
xxxxxxxxx
2005
ordered
local TOC
xxxxxxxxx
xxxxxxxx
xxxxxxxxx
2005
2000
2010
random
web of pages

Fig. 20

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/45069 A2

(51) International Patent Classification⁷: G09B 7/00

(21) International Application Number: PCT/US00/34013

(22) International Filing Date:
15 December 2000 (15.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/172,134 17 December 1999 (17.12.1999) US
09/728,073 4 December 2000 (04.12.2000) US

(71) Applicant: INFORMIX SOFTWARE, INC. [US/US]; 4100 Bohannon Drive, Menlo Park, CA 94025 (US).

(72) Inventor: HENNUM, Erik; 78 St. Mary's Avenue, San Francisco, CA 94112 (US).

(74) Agents: DIBERARDINO, Diana; Fish & Richardson P.C., 601 Thirteenth Street N.W., Washington, DC 20005 et al. (US).

(81) Designated States (national): AU, BR, CA, JP, MX.

(84) Designated States (regional): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— without international search report and to be republished upon receipt of that report

(48) Date of publication of this corrected version:
29 November 2001

(15) Information about Correction:
see PCT Gazette No. 48/2001 of 29 November 2001, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/45069 A2

(54) Title: WEB-BASED INSTRUCTION

(57) Abstract: A method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/45069 A2

(51) International Patent Classification[7]: G09B 7/00

(21) International Application Number: PCT/US00/34013

(22) International Filing Date:
15 December 2000 (15.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/172,134 17 December 1999 (17.12.1999) US
09/728,073 4 December 2000 (04.12.2000) US

(71) Applicant: **INFORMIX SOFTWARE, INC.** [US/US]; 4100 Bohannon Drive, Menlo Park, CA 94025 (US).

(72) Inventor: **HENNUM, Erik**; 78 St. Mary's Avenue, San Francisco, CA 94112 (US).

(74) Agents: **DIBERARDINO, Diana**; Fish & Richardson P.C., 601 Thirteenth Street N.W., Washington, DC 20005 et al. (US).

(81) Designated States *(national)*: AU, BR, CA, JP, MX.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *without international search report and to be republished upon receipt of that report*

(48) Date of publication of this corrected version:
23 May 2002

(15) Information about Corrections:
see PCT Gazette No. 21/2002 of 23 May 2002, Section II
Previous Correction:
see PCT Gazette No. 48/2001 of 29 November 2001, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/45069 A2

(54) Title: WEB-BASED INSTRUCTION

(57) Abstract: A method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

WEB-BASED INSTRUCTION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Application No. 60/172,134, filed December 17, 1999 and U.S. Serial No. 08/888,925, filed July 7, 1997, the entire disclosures of which are incorporated herein by reference.

TECHNICAL FIELD

This application relates to web-based documentation and instruction.

BACKGROUND

A typical computer system as shown in Fig. 1 includes a computer 100 having a central processing unit 105, an input/output unit 110 and a memory 115 containing various programs used by the computer 100 such as an operating system 120 and one or more application programs 125. An end-user of the computer system communicates with the computer 100 by means of various input devices (keyboard 130, mouse 135) which transfer information to the computer 100 via input/output unit 110. The computer 100 replies to this input data, among other ways, by providing responsive output to the end-user, for example, by displaying appropriate text and images on the screen of a display monitor 140.

The operating system 120 may include a graphical user interface (GUI) by which the operating system and any applications it may be running (for example, a word-processing program) can communicate with a user of the computer system. A commonly used GUI implementation employs a desktop metaphor in which the screen of the monitor is regarded as a virtual desktop. The desktop is an essentially two-dimensional working template area supporting various graphical objects, including one or more display regions. As shown in Fig. 2, information generated by application programs or the operating system can be displayed on the desktop 200 within display regions 205 (for example, windows, dialog boxes, pop-up menus, pull-down menus, drop-down lists, icons). The user can interact with the operating system, and any applications it may be running, by manipulating the cursor 210

appropriately within the display regions and by entering information with the keyboard or other input device.

The computer 100 also includes some sort of communications card or device 145 (for example, a modem or network adapter) for exchanging data with a network 150 via a communication link 155 (for example, a telephone line). The network 150 may be, for example, a local area network (LAN), an intranet, or the Internet. A service provider provides access to the network and may additionally provide various utilities or services (such as electronic mail) associated with the network. Examples of service providers include Internet service providers (ISPs) such as AT&T WorldNet or online service providers (OSPs) such as America Online and CompuServe.

Developers need to know programming concepts to implement the application program. Therefore, a description of the implementation of the application (and not only the operation of the application) would be helpful.

Most computer applications provide an online help / documentation facility which aids in the use of the application. A typical online help system such as shown in Fig. 3A is accessed through a GUI in which screens of textual and graphical information are displayed to the user in a help window 300. The user can then read the screens of help text to get a better understanding of the application and its various features.

The user invokes the help system with a key sequence (for example, pressing the F1 key on the keyboard) or by clicking the mouse on an appropriate graphical icon or menu item. In response, the help system may display a table of contents 305 listing the available help topics and subtopics which can be accessed and viewed by the user as desired. The user can browse through the table of contents 305 and click a help topic of interest to cause its corresponding body of information to be displayed in a help window. In the help window 300 shown in Fig. 3A, the user has clicked the "Programming with Microsoft Word" topic 310 to cause the corresponding help screen 315 to be displayed in window 300 as shown in Fig. 3B.

The "Programming with Microsoft Word" topic 310 shown in Fig. 3B includes several subtopics 320, each represented by a separate "link." When the user clicks the mouse on one of these links--for example, the "Error Messages" link 325--the text for the corresponding help topic is displayed automatically in the help window 300, as shown in Fig. 3C. In this example, the "Error Messages" topic 330 includes several links to further subtopics relating to

specific types of error messages. As shown in Fig. 3D, when the user clicks one of these links, for example, the "Out of memory (stack space)" link 335, a new help window 340 is spawned to display the corresponding help information ("Freeing up memory") for the selected topic. The help information displayed in window 340 includes yet another link 345 for another subtopic, "active window," which when clicked by the user causes corresponding help text to appear in a pop-up dialog box 350. Virtually any level of such nested help displays is possible. The quantity and types of display regions (windows, dialog boxes, etc.) used to display help information is largely a matter of design choice based on the preferences of the help system developer.

A help system may provide "context-sensitive" help information, meaning that the help system automatically displays help information specifically relevant to the application's current task, rather than simply displaying all available help topics and forcing the user to identify and call-up the appropriate help topic manually. A context-sensitive help system decides which help information to display based on factors such as the current state of the application (for example, the particular function being invoked by the user) and the current cursor position.

The information provided by most online help systems relates to the mechanics of using features of an application. In Fig. 4, for example, the text 400 corresponding to the chosen help topic 405, "Cancel printing," describes how to control the print feature provided by the application 410 (Microsoft Word).

A help system also may provide substantive information on how to make use of the application to achieve a desired goal. In Fig. 5A, for example, the online help system provides two types of substantive information: reference material 500 for the WordBasic programming language and practical explanations 505 of how to use WordBasic to write useful programs. The reference material 500 includes textual annotations describing the syntax and meaning of various WordBasic statements, such as the AddAddIn statement, the help text for which is shown in Fig. 5B. The practical explanations 505 can include static examples of program code which the user can study to gain a better understanding of the WordBasic programming language. Fig. 5C shows an example of a program code that makes use of the GetCurValues WordBasic statement.

Online help systems typically are "built" (that is, processed into a form that facilitates run-time operation) by compiling several different help source files containing help information that has been composed by technical writers. In general, these help source files are maintained as a separate body of information apart from the application to which the help system corresponds. Consequently, when the application developers change or update the functionality of the application, the technical writers must make corresponding changes to the help source files to ensure that the online help system accurately describes the operation of the application. In general, however, online help systems fail to describe the implementation of the application.

A help system may be implemented in a network environment using a "browser", which enables users to access and view electronic content stored in the network environment. A browser typically is used for displaying documents described in Hyper-Text Markup Language (HTML) and stored on servers connected to a network such as the Internet. Fig. 6 is a screen shot of a browser application 600 (in this case, Internet Explorer) displaying a typical HTML document, or web page 605. A user instructs the browser 600 to access the web page 605 by specifying a network address 610 -- or Uniform Resource Locator (URL) -- at which a desired document resides. In response, the browser 600 contacts the corresponding server hosting the requested web page, retrieves the one or more files that make up the web page, and then displays the web page in the computer display 140.

A single web page may be composed of several different files potentially of different data types (for example, text 615, images 620, virtual worlds, sounds, or movies). In addition, a web page can include links 625, or pointers, to other resources (for example, web pages, individual files, or downloadable files) available on the network. Each link has an associated URL pointing to a location on the network. When a user clicks on, or otherwise selects a displayed link, the browser will retrieve the web page or other resource corresponding to the link's associated URL and display it to, or execute it for, the user.

Referring to Fig. 7, a web page 605 may provide, in addition to content 700, a site guide 705 that helps the user navigate through all the links associated with that web page. The site guide 705 is similar to a table of contents and typically resembles a tree structure. Likewise, the web page 605 could include a search facility 710 that enables the user to search for particular key words that appear within the links associated with that web page. The web

page may provide a "Home" link 715 that sends the user back to a main web page from which all content and links can be accessed. The web page may provide a download link 720 that, when accessed, transmits a file from another web page or computer to the user's computer.

According to one aspect of the invention, a method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a source file of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

Embodiments may include one or more of the following features. For example, a predetermined application may be performed and one or more annotations descriptive of the performed application may be presented in coordination with performance of the predetermined application. Performing the predetermined application may include receiving input from the user. Another annotation page may be presented in coordination with performance of the predetermined application based on input from the user.

Presenting the other annotation page may include automatically and simultaneously calling an annotation request module including application, file, class and function names of a program unit for which detail should be displayed. Presenting the other annotation page may also include mapping the request to an annotation, and informing a browser window in the web-based environment to display the other annotation page.

Another annotation page may be presented in coordination with performance of the predetermined application. A global table of contents that includes links to annotations may be automatically generated by parsing structured links in web pages including annotation pages. Generation of links in the global table of contents may be synchronized with presentation of annotations by highlighting links corresponding to a current annotation page. The global table of contents may be presented in a first frame of a first browser window, the annotation page may be presented in a second frame of the first browser window, and the predetermined application may be performed in a second browser window.

Performing the predetermined application may include launching a Java applet or application, which may include calling a Java application program interface to ask a web browser to show the annotation page. Performing the predetermined application may include downloading a hyper-text markup language page containing a Java applet.

Performing the predetermined application may include sending a common gateway interface request to a web server that launches the application in a window in the web-based environment. The application may return a hyper-text markup language page that includes JavaScript to ask a web browser to display the one or more annotations.

The annotation page may be presented in a first browser window and the predetermined application may be performed in a second browser window. The application implementation detail may include text descriptive of the application, fragments of source code from the application, or both. The source code fragments may be imported directly from the source code file of the presented application.

The annotation page may be automatically generated. This generation may include receiving a source code file that has embedded text marked up with instructions. Additionally, the source code may be parsed to determine a structure of the predetermined application, and one or more annotations may be generated based on the predetermined application structure and instructions. Generation of the annotation page may include generating one or more annotation links for navigating the annotations of the predetermined application. Additionally, application implementation detail may be generated based on the embedded information, and one or more keyword links may be generated for reference documentation. Generating the annotation page may also include highlighting the keyword links and the annotation links in the annotation page. The annotation page may be automatically updated when an updated source code file is received.

A global table of contents may be automatically generated by parsing the one or more annotations for annotation links. The global table of contents may be provided, and may include links to annotations. Alternatively, the global table of contents may be generated, and may include links to web page including annotation pages relating to an application. The local table of contents may be provided when a local link in the global table of contents is selected.

The presented annotation page may be descriptive of the performed application, and the annotation page may be presented in coordination with performance of the predetermined application.

A source code file, which is stripped of annotation mark up and includes source code of the application but does not include text from the annotations, may be generated. The stripped source code file may be presented and the user may be permitted to edit the stripped source code file.

According to another aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes providing a predetermined plurality of applications. A predetermined application is performed, and an annotation page descriptive of the performed application is presented in coordination with performance of the predetermined application. The annotation page includes detail of application implementation and links to annotations and reference documentation.

According to another aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes automatically assembling and providing a global table of contents based on content in the environment. The global table of contents includes a plurality of links to content within the environment. A local table of contents that includes links to content that orient the user within a local topic, is generated. The user is permitted to select links from the local table of contents to access local topics.

According to a further aspect of the invention, a method, performed in a web-based environment on a computer system, of teaching a user to implement an application includes providing a plurality of predefined interactive examples. One or more of the predefined interactive examples is performed in response to user selection, and one or more annotations descriptive of the performed interactive example are presented in coordination with performance of the predefined interactive example. The user is permitted to selectively explore different aspects of the performed interactive example, the annotations, or both.

According to another aspect of the invention, a web-based computer system for teaching a user to implement an application includes one or more predefined interactive applications, and an annotation page including one or more annotations. A predefined

interactive application is selectively executable by the user of the web-based computer system. The annotation page describes a predefined interactive application. The annotation page also includes one or more links, and detail of implementation of the application. Different annotations are automatically provided in the annotation page in response to selective execution of a predefined interactive application.

According to a further aspect of the invention, a web-based computer system for teaching a user to implement an application includes a web-browser window that includes a content frame, a framework applet, and a table of contents frame that displays a global table of contents hierarchy of links related to content in the content frame. The system also includes one or more annotations displayed in the content frame, where each annotation describes a predefined interactive application and includes links to other content. The system includes a table of contents window that displays a local table of contents hierarchy of links related to local content in the displayed annotation.

The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects and advantages will be apparent from the description, the drawings, and the claims.

DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a prior art computer system.

Fig. 2 shows display regions in a graphical user interface as used in the computer system of Fig. 1.

Figs. 3A – 5C are screen shots from a prior art online help and documentation system.

Fig. 6 is a screen shot of a browser application.

Fig. 7 shows a display region in a browser application.

Fig. 8 is a flow diagram showing the options available to a user of the Informix® byExample application.

Figs. 9A – 9P are screen shots from the Informix® byExample application and from the NewEra™ development environment.

Fig. 10 is a block diagram of the NewEra™ architecture.

Fig. 11 is a block diagram showing how the Informix® byExample application is built.

Fig. 12 is a sample of NewEra™ source code.

Fig. 13 shows display regions in a web-based instruction system.

Figs. 14A – 15A, 15C, 17A-17D, 18A, 18B, 19A and 19C-19F are screen shots from the Informix® byExample web-based application.

Figs. 15B, 15D and 19B show content in a display region of the Informix® byExample web-based application.

Fig. 15E and 18C are source code files with embedded annotations.

Fig. 16A is a flow diagram showing steps taken by an author of annotation pages.

Fig. 16B is a flow diagram showing steps taken by a scripting program to automatically generate links in the annotation pages.

Fig. 16C is a flow diagram showing steps taken by an example program.

Fig. 20 is a block diagram of a local table of contents model used in the web-based application.

DETAILED DESCRIPTION

The help information provided by conventional online help systems has proven useful in aiding users to make effective use of application programs. However, because these conventional online help systems essentially are limited to providing static textual or graphical information, their effectiveness is diminished considerably. Users of conventional online help systems gain instruction by reading and carefully studying the textual and graphical information provided by the help system and then applying its teachings to the problem to be solved. As a result, learning to use applications of any complexity often is a painstaking and time consuming process.

An online help and instruction system developed by Informix® Software, Inc., known as NewEra™ byExample, dramatically enhances the ease with which users can absorb information and learn to use new applications. NewEra™ byExample is an online documentation facility for NewEra™, an object-oriented application development environment from Informix® Software, Inc. which runs under the Windows 95/NT operating systems. A copy of NewEra™ byExample's online description of its use and operation is attached as Appendix A.

NewEra™ byExample (or more generally, Informix® byExample, which covers the example-based instruction systems provided for the NewEra™, Visual Basic and Java development environments) is a specific implementation of a more general concept referred to as "documentation by example" in which users are provided with dynamic, interactive examples demonstrating how to accomplish a given task. Annotations describing various aspects of the examples accompany the examples as they are being executed. Documentation by example is based in part on the premise that users learn best by doing something (for example, participating in an activity and observing or influencing its outcome) rather than by merely reading about the topic.

As illustrated in the flow diagram of Fig. 8, an Informix® byExample user has several different options for obtaining information including selecting among various different topics (step 800); running examples while the application for which help is sought remains active (step 805); reading about the examples, either concurrently while running the example or independent of the example (step 810); inspecting the examples' source code in different editor utilities (step 815); and accessing online background reference materials that help the user to understand the examples (step 820)--all without leaving the help environment. While in step 815, the source code for the examples can be used as sample program code which can be cut-and-pasted for use as a template in the NewEra™ development environment in creating new applications. Moreover, Informix® byExample enables users to learn through experimentation, for example, by selectively changing the examples or their parameters and observing how the changes affect the examples' outcomes.

Specific features of Informix® byExample are described in detail with reference to Figs. 9A-9P, which are exemplary screen shots taken from the Informix® byExample application.

When a user first launches Informix® byExample, the default screen configuration shown in Fig. 9A is displayed. This initial screen includes two separate display windows, a list (or "table-of-contents") window 900 showing the subtopics presently available to the user under the current topic 902, and a text window 904 which displays the help information corresponding to the topic or subtopic selected from the list window 900. As the user clicks different ones of the eight subtopics 906 displayed in the list window 900, the information in the text window 904 is updated automatically to correspond to the chosen subtopic 906. The

user can move to different pages within the current topic by clicking the forward (">>") button 908 or the backward ("<<") button 910 as desired.

In the example of Fig.9A, the subtopics shown in the list window 904 relate to the topic "NewEra™ byExample Introductory Topics." To switch to another help topic, and thereby make available a different subset of the online help documentation, the user clicks the Help Topics button 912 which brings up a window containing the Help Topics menu 914 with a list 916 of nine different help topics, as shown in Fig. 9B. At any point in the Informix® byExample application, the user can jump to any other portion of the online help system by bringing up the Help Topics menu 914 and clicking the desired topic. The user can return to a previous topic by pressing the Back button 916 an appropriate number of times.

Each of the help topics in the list 916 in Fig. 9B can be expanded to reveal a hierarchy of multiple levels of subtopics. When the user clicks, for example, on topic 918 ("NewEra™ byExample"), it expands to reveal two additional levels of subtopics as shown in Fig. 9C--a first level 920 including the subtopics "Introduction," "Common NewEra™ Programming Techniques," and "The Examples," and a second level 922 under "The Examples" subtopic which includes the 43 interactive examples.

When the user clicks one of the examples, for example, the "Enabling and Disabling Buttons" example 924, the list window 900 is updated as shown in Fig. 9D to display the annotation segments 926 ("Overview of Buttons2 Example," Graphical Object Summary," "Event Handler Summary," "Important Event Handlers," and "Enhancements and Variations") associated with the selected example. The annotation segments 926 collectively describe the corresponding example and include descriptions of the example's window, its graphical objects, and its event handlers. In addition to the prose descriptions of the example, an annotation segment usually also includes a source code fragment of particular interest which has been imported directly from the source code of the example under consideration.

As shown in Fig. 9D, one of the annotation segments ("Important Event Handlers") includes 13 topics 928-- a list of the primary event handlers used in the BUTTONS2 example. Each event handler topic 928 includes source code fragments and prose explanations describing the event handler to which the topic corresponds. For example, when the user clicks event handler topic 930, the text window 904 displays source code fragments 932

relating to the corresponding event handler (*nextBT :: activate()*) along with annotations 934 describing the code's operation, as shown in Fig. 9E.

The text window also may contain one or more links to related information, for example, background reference material, which in turn may include still further links to additional background information and so on in hierarchical fashion, each successive level in the hierarchy providing information about the example in greater detail and at a lower level of abstraction. By providing a hierarchy of links to increasingly detailed documentation in this manner, Informix® byExample supplies context-appropriate information in a helpful and efficient manner to all users, regardless of their varying levels of experience and sophistication. A user can traverse down the hierarchical links of descriptive information selectively until a level of understanding is reached that is commensurate with the user's needs and background. This arrangement provides novice users with easy access to detailed descriptive information while, at the same time, experienced users seeking help on a specific point are protected from having to navigate through large volumes of unneeded information.

An example of hierarchical linking is shown in Fig. 9D in which text window 904 includes a link 936 (displayed as green, underlined text) to the *MAIN()* function, one of functions in the BUTTONS2 example. When the user clicks the *MAIN()* function link 936, the text window 904 displays the source code 940 for that function, as shown in Fig. 9F. The source code 940 includes further links to related information such as an online language reference manual containing descriptions of keywords and object classes. When the user clicks one of these links--for example, the keyword link 942 for the LET statement--the text window 904 changes to display the corresponding online language reference entry as shown in Fig. 9G. Similarly, if the user had clicked the object class link 944, the text window 904 would have displayed information about the *ixSQLConnect* class. In Fig. 9G, the user can follow links to still further background information, for example, by clicking the Object Expression box 946 to cause the text window 904 to appear as in Fig. 9H. Subsequently, or alternatively, the user can click the Glossary button 948 to bring up an online glossary in a pair of windows--a glossary table of contents window 950 and a glossary text window 952--as shown in Fig. 9I. Clicking a term in the glossary table of contents window 950 causes its definition to appear in the glossary text window 952.

After studying an example's annotation, its source code fragments, corresponding language reference entries, the glossary, or a combination thereof, the user can jump selectively to any other location in the help system by clicking the Contents button 954, which brings up the Help Topics menu 914 shown in Fig. 9B (or the Index button 956, which presents the available help topics in a searchable-indexed form), and then selecting the desired topic in the manner described in connection with Figs. 9B and 9C.

Keyword links and class name links, such as the LET statement link 942 and the *ixSQLConnect* class link 944, respectively, in Fig. 9F are represented in visually unique manners (for example, blue uppercase text for keywords, blue upper and lowercase text for class names) so that they may be distinguished easily from each other and from other types of links such as the *MAIN()* function link 936 in Fig. 9D (green, underlined text). By using different styles for different types of links, Informix® byExample provides the user with intuitive and useful information concerning the nature of the online information available and the interrelationships between the different components (annotations, source code fragments, language references, etc.) of the examples. Virtually any number of different link types may be represented by different styles according to the preferences of the system designer.

For each of the source code fragments included in an example's annotation, a user can invoke an appropriate editing utility from within Informix® byExample to inspect, edit or copy the example's source code. This allows users to view a source code fragment in the context of the larger program from which it was taken.

Informix® byExample includes source code fragments from two different types of source code--textual program code in the NewEra™ programming language (as indicated by a 4GL or 4GH file suffix), and windows interface definition files (files having the suffix WIF) which define how the GUI will appear to, and interact with, the end-user of the application undergoing development. To view either type of source code fragment, the user clicks a short-cut arrow next to a code fragment, for example, one of the short-cut arrows 958 and 960 shown in Figs. 9D-9F, and Informix® byExample responds by launching an editor that corresponds to the type of source code under consideration. When the user clicks a short-cut arrow next to a 4GH or 4GL file, such as short-cut arrow 958 in Figs. 9D and 9F, Informix® byExample automatically launches the appropriate editor--NewEra™ Codewright--to view the source code file from which the code fragment was taken, as shown in Fig. 9J. Similarly,

when the user clicks a short-cut arrow next to a WIF file, such as short-cut arrow 960 in Figs. 9D and 9E, Informix® byExample automatically launches the appropriate editor--NewEra™ Window Painter 3.0--to view the WIF file from which the code fragment was taken, as shown in Fig. 9K.

Selectively launching an appropriate one of multiple different editors in this manner reflects the standard editing behavior of the NewEra™ development environment. Both the NewEra™ development environment and the Informix® byExample documentation system make use of the same editors in the same manner. As a result, users gain familiarity with the application for which help is sought (that is, the NewEra™ development environment) through normal interaction with the online help system (that is, Informix® byExample).

Once the user has opened up the source code for an example, the user simply can study the code or can cut-and-paste portions of the code, whether visual objects from a WIF file or program statements in a 4GH or 4GL file, into the user's own source files. Alternatively, the user can perform a "Save As..." operation and thereby save the source code for the example under a new file name. The user then can edit or otherwise manipulate the new file as desired. In this manner, the examples provided by Informix® byExample can serve as templates for use in developing new applications in the NewEra™ development environment.

Users also may execute any or all of the 43 interactive examples provided with Informix® byExample to observe first hand how they operate. The examples are prebuilt and can be launched directly from their corresponding Informix® byExample annotations. To do so, a user first selects an example of interest from the Help Topics window 914 shown in Fig. 9C and, when the corresponding annotation appears in the text window, clicks the Run button appearing near the top of the text window. In response, the example executes and, based on the input received from the user, displays various screens to the user as if the example were a standalone application. At the same time, the text window automatically updates to display descriptive information that is pertinent to the portion of the example that was just executed by the user. With each successive operation that the user performs on the running example, the text window is updated simultaneously (or nearly so) to maintain synchronization with the state of the interactive example by displaying corresponding sections of the annotations which explain to the user what just happened in the example. By coordinating the help display with

the current state of the examples, users consistently are provided with timely and useful information (for example, the particular source code being executed by the example) that is directly relevant to the user's current topic of interest. As a result, the user's ability to comprehend and absorb information is enhanced dramatically. An example of Informix® byExample's automatically coordinated help display is illustrated in Figs. 9L-9P.

Fig. 9L shows the initial list window 900 and text window 904 that are displayed when the user selects the "Displaying an Edit Menu" example from the Help Topics menu. To run this example, the user clicks the Run button 962 which, as shown in Fig. 9M, spawns an example window 964 illustrating the basics of an edit window. At the same time, the text window 904 is updated to display information on the *MAIN()* function for the "Displaying an Edit Window" example.

As the user selectively manipulates the GUI controls in the example window 964, the information displayed in the text window 904 is updated automatically in a corresponding manner. In Fig. 9N, the user has clicked in text box 966 which causes the text window 904 to display information relating to *edit1TB :: focusln()*. Similarly, when the user clicks text box 968, text window 904 displays information relating to *edit2TB :: focusln()* as shown in Fig. 9O. When the user clicks the CheckBox 970, text window 904 displays information relating to *noneditCB :: focusln()* as shown in Fig. 9P.

Users can experiment with an example by changing its source code or modifying its parameters and observing how these changes affect the example. To do so, the user edits the desired source code file, saves it a separate working directory so as not to disturb the predefined examples, and then rebuilds the example using mechanisms provided with the NewEra™ development environment. The number and types of such experiments that can be created and performed are limited only by the imagination of the user.

Other options in running the examples are possible. For example, users can run an example without concurrently viewing annotations. Additionally, the Debugger provided with NewEra™ can be used to set breakpoints in the example source code before running the example, thereby giving the user even further insight into how an example works.

A description of the Informix® byExample architecture, and the manner in which the NewEra™ development environment and the Informix® byExample application are built, is provided with reference to Figs. 10-12.

Informix® byExample builds upon the Online Help (OLH) facility provided with the Windows 95/NT operating systems. As shown in Fig. 10; the Informix® byExample application 1000 draws both upon resources created specifically for Informix® byExample as well as resources that are native to the NewEra™ development environment 1005. The components specific to the Informix® byExample application 1000 include the interactive examples 1007, source code 1010 for the examples, and annotations 1015 describing the examples. The annotations 1015 include several different subcomponents including representative fragments 1020 of the examples' source code, short-cuts 1025 that launch an appropriate editor (for example, NewEra™ Codewright or NewEra™ Window Painter) for viewing the examples' source code, jumps 1030 to the interactive examples, and links 1035 to descriptions of specified keywords and class names contained in the NewEra™ online reference 1040.

As indicated in Fig. 10, the online reference 1040, the Codewright editor 1050 and the Window Painter editor 1055--along with other components such as Application Builder 1060, Debugger 1065 and Interprocess Communications (IPC) library 1070--exist as part of the development environment 1005 and thus are logically separated from the Informix® byExample application 1000. Consequently, when a user of the Informix® byExample application 1000 requests a resource residing in the NewEra™ development environment--either by clicking a link 1035 for a keyword or class name or by clicking a shortcut 1025 to view source code--Informix® byExample 1000 first must communicate with the NewEra™ development environment 1005 via an interface dynamic linked library (DLL) 1080 to access the requested resources. The interface DLL 1080 is a compiled library of routines that enable the Informix® byExample application 1000 to communicate with other applications such as the components of the development environment. Informix® byExample 1000 calls the appropriate DLL routines to display the requested online reference information or to launch the appropriate source code editor, depending on the nature of the request made by the user.

More specifically, when an Informix® byExample user clicks on a shortcut 1025 to a location in an example's source code 1010, the Informix® byExample application 1000 calls a function in the DLL, which in turn calls a function in the IPC library 1070 which launches the appropriate editor. As part of this function call (which is generated automatically by processing source code fragments during the build of Informix® byExample, discussed

below), the Informix® byExample application 1000 passes parameters that designate the editor to be launched (Codewright 1050 or Window Painter 1055), and that identify the line number at which the examples' source code 1010 is to be opened by the designated editor. When an Informix® byExample user clicks on a link 1025 for a keyword or class name, the Informix® byExample application 1000 calls a function in the DLL, which in turn uses the Windows OLH facility to display the corresponding definition in the online reference 1040.

Other functions provided by the interface DLL 1080 control execution of the interactive examples 1007 and coordinate the list window and the text window displays to ensure that they maintain correspondence. Further details on the interface DLL 1080 and the runtime operation of the Informix® byExample application 1000 are set forth in Appendix B.

The manner in which the Informix® byExample application 1000 and its components (for example, examples 1007, examples' source code 1010 and annotations 1015) are generated realizes a high degree of code "maintainability"--a measure of the efficiency and ease with which an application can be modified. The high degree of code maintainability is achieved by incorporating all of the information used to generate both the interactive examples and the corresponding annotative components of Informix® byExample into a unified logical entity--namely, the source code for the interactive examples themselves. As a result, only one central source of information need be maintained. Any changes or updates made to that central information source will be incorporated automatically both into the examples and into the documentation / instruction / help facility (Informix® byExample) for the examples. This automated build procedure ensures that the examples and the corresponding Informix® byExample annotations are kept in synchronization regardless of the number and frequency of modifications made to the underlying source code.

As shown in Fig. 11, the NewEra™ byExample source code 1100 can be thought of as a single logical entity, although physically it is formed of a collection of interdependent files. The source code 1100 contains three basic types of text--program instructions 1105, program comments 1110 and annotations 1115--intermixed throughout the source code. The different text types are distinguished from each other by programming conventions and by strategically placing various different markup symbols 1120 throughout the source code.

Some of the text in the source code 1100 can serve multiple purposes. For example, the program instructions 1105 in the source code 1100 are compiled into the examples' binary

executable files 1125. These program instructions include calls to the OLH facility to display the corresponding annotation at the appropriate point during execution of the example. When an example is run by the end-user, these OLH calls cause the text window to display the appropriate annotation automatically to describe what just happened in the example.

Portions of these same program instructions 1105 also will be extracted to serve as a clean copy of the examples' source code, which can be displayed to the user in an editing environment. Similarly, descriptive text that serves as program comments 1110 (unprocessed programming explanations directed at the Informix® byExample project developers) also can serve as annotations 1115 (programming explanations displayed to end-users of Informix® byExample at runtime).

The markup symbols 1120 delineate the various types of text in the source code and specify how they are to be handled when the interactive examples and the Informix® byExample annotations are built. Fig. 12 shows a sample of NewEra™ source code which includes several markup symbols including two instances of the "normal" symbol 1200 and 1205, an "[edit" symbol 1210 and a "] file" symbol 1215. Each of these markup symbols, along with their respective arguments, are bounded by a pair of brackets ("{ ... }") indicating that they reside in comment fields and are not to be treated as NewEra™ program instructions. Programming languages other than NewEra™ may use different conventions to delineate comment fields. In the Java programming language, for example, a start of a comment field is designated by a "/*" symbol and terminated by a "*/" symbol. In any event, the corresponding programming language compiler will ignore any text that has been designated as residing in a comment field.

The ".normal" markup symbol indicates that the text following that symbol (for example, "Since objects, ...," following symbol 1200) is to be treated as explanatory comments, and thus to be displayed to the end-user in a text window as part of the annotation text at an appropriate point during execution of a corresponding interactive example. Other markup symbols specify the name of output files, portions of the source code that are to serve as representative fragments of the examples' source code, hotspots and destinations for jumps and links, or GUI-related information concerning display characteristics and objects (windows, popups, buttons, etc.). A detailed description of the markup language is set forth in Appendix C.

Once the source code 1100 has been modified as desired, it is used to build the interactive examples and the descriptive content of the Informix® byExample application through a number of different steps. First, the source code 1100 is processed by two different scripts 1130--a PERL script (Practical Extraction and Report Language, a general purpose interpreted language often used for parsing text) and a WordBasic script. The scripts 1130 generate two basic types of output: source code files 1135 for the interactive examples, and RTF files 1140 (Rich Text Format, the format expected by the OLH compiler) which represent the descriptive and visual content (for example, annotations, source code fragments, shortcuts to source code editors, links to online reference, jumps to executable examples) of the Informix® byExample application.

The PERL script parses the source code 1100 searching for markup symbols and, based on the particular markup symbols encountered, produces several RTF file fragments and several source code files 1135, which represent various subsets of the overall source code 1100. The WordBasic Script then merges the RTF file fragments into complete RTF files 1140 which are processed by the Windows OLH compiler 1145 to produce OLH files 1150 containing the descriptive and visual content for the Informix® byExample application. At the same time, the examples' source code 1135 is compiled by the NewEra™ compiler 1155 to generate the binary executable corresponding to the interactive examples 1125.

The RTF file fragments generated by PERL script contain several different components in addition to the annotations 1115 appearing in the source code 1100. The PERL script identifies each instance of a keyword or a class name appearing in the source code extracted for the examples. For each keyword and class name detected, the PERL script creates a link in the RTF file to the corresponding entry in the online reference materials.

The PERL script also extracts fragments of representative source code for inclusion in the RTF files as text that appears along with the explanatory comments. The source code fragments are formatted as monospace unwrapped text delineated by leading and trailing blank lines whereas the explanatory comments are formatted as proportionally spaced wrapped text. For each source code fragment included in the RTF file, the PERL script also inserts in the RTF file a corresponding short-cut button which enables the end-user to launch the source code editors and view the source code at the line where the fragment starts. The PERL script also strips all of the markup symbols 1120 from the source code extracted for the

examples. This provides end-users with a clean version of the source code for viewing in the associated editor.

Other functions performed by the PERL script include automatically guaranteeing that the identifier for an annotation topic is the same in an interactive example as it is in the Windows OLH facility. That is, the PERL script reads the help topic identifiers for the Windows OLH facility and generates corresponding NewEra™ constants. The PERL script also generates modified versions of the NewEra™ makefiles (files that include computer-readable instructions for building an application) which are used to build the examples. Further details of the PERL script and its operation are set forth in Appendix B.

Although the PERL and WordBasic scripts described above operate on source code written in the NewEra™ programming language, different scripts can be used to parse other types of source code, for example, Java or Visual Basic. Generally, appropriate PERL and WordBasic scripts can be written to process virtually any type of programming language provided the programming language utilizes ASCII source code (required by PERL) and provides some sort of source code comment mechanism. Other programming language attributes that facilitate use of the Informix® byExample techniques include a mechanism for invoking the Windows OLH facility with a topic identifier (so the example can display its annotations), a mechanism for invoking the editing functions of the development environment (so the annotation can open source code files, assuming the programming language under consideration provides or requires a development environment), and an online reference in Windows OLH format (so keywords in the source code can have jumps to the online reference). Many of the Informix® byExample features described above can be implemented even if the underlying programming language lacks one or more of these other attributes, however.

PERL scripts can be modified to output files in formats other than RTF. For example, a modified PERL script can output hypertext markup language (HTML) files, which can be viewed using any available web browser (for example, Netscape Navigator).

Other variations of documentation by example are possible. For example, the annotations describing the interactive examples could be presented in a manner other than textual. Sounds, graphical symbols, pictures, movies or any other means of communication could be used as desired. Further, the selection of which interactive examples to perform

could be based on factors other than, or in addition to, designation by the user. For example, an interactive example could be launched automatically at certain points during execution of the underlying application, or at certain execution points in the help system. When the user clicks a keyword, class name or other link, an example could be launched automatically either in addition to, or instead of, displaying the textual reference information pointed to by the link.

The documentation by example methods and techniques described above are not limited to aiding users of software development systems but rather may find application as a general training and education tool for any computer-based application or utility. Moreover, the techniques described here may be implemented in hardware or software, or a combination of the two. Preferably, the techniques are implemented in computer programs executing on programmable computers that each includes a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), and suitable input and output devices. Program code is applied to data entered using an input device to perform the functions described and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (for example, CD-ROM, hard disk or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described. The system also may be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other embodiments are within the scope of the following claims.

For example, the documentation by example method and techniques may be applied in a network or web-based environment such as a local area network (LAN), an intranet, or the Internet. A web-based instruction system developed by Informix® Software, Inc., known as

Informix® byExample, dramatically enhances the ease with which developers can effectively and efficiently implement database applications. The web-based instruction system provides users with examples and instructions in web pages to teach users how to implement applications. The applications themselves may or may not be web-based applications. The web-based instruction system enables users to read formatted source code, and navigate program structure (for example, the hierarchy of file, class, and function) to access details of the application program's implementation. The web-based system also permits users to jump from a language keyword (that may be indicated as such in code fragments) to a full reference documentation pertaining to that keyword. Users are able to open source code in an editor and run programs directly from the web browser at the click of a run button in a toolbar. As the example runs, an annotation page for the current method or function in the program displays. Therefore, the user is able to view the annotation as the function or method executes to more quickly understand how to implement such an application program.

This is in contrast to prior online help systems that simply show a user how to use an application. Because some implementation changes do not alter the operation of the application, but all implementation changes, by definition, affect the implementation, this makes maintenance of the implementation documentation even more important. The web-based instruction and help system solves this problem with prior online help systems by maintaining the annotations as embedded comments within the source code and generating the annotation pages automatically whenever the source file changes.

The web-based system also provides the users with how-to documentation for accomplishing a particular task. The how-to documentation provides clear step-by-step instructions, and presents a flexible interface for experts and "newbies" alike. The how-to documentation includes useful graphics, and provides links to needed software and related demos and other technical information. Additionally, the how-to documentation helps the user to determine which products to use to accomplish a goal.

As illustrated in Fig. 13, when a user first accesses a web browser providing the web-based instruction system, a default browser configuration 1300 is displayed. The initial browser includes a framework of three separate display frames: a content frame 1305 that typically displays a web page, but may also display other relevant content such as annotation pages (which are a specific type of web page), a top frame 1310 that includes a framework

applet (for example, a Java applet) that displays a tool or navigation bar in addition to providing other services, and a table of contents (TOC) frame 1315 which may be implemented using, for example, a Java applet.

When the browser displays a byExample® HTML page, the page uses JavaScript to automatically check for the presence of the framework. If the framework is present, the JavaScript notifies the framework applet in the top frame 1310 about the name of the new page and the type of content it provides. If the framework is absent from the page, the JavaScript opens the framework with the page as the initial page (that is, the page on which the notification occurs).

Referring also to Figs. 14A and 14B, based on the name of the page 1400, the table of contents 1315 automatically selects the corresponding title 1405. The table of contents 1315 tests the available Java libraries on the client and automatically displays the table of contents hierarchy with a tree control if available. The table of contents frame 1315 is user-resizable and dismissible using a contents button 1410 in the top frame 1310: for example, in Fig. 14A, the table of contents frame 1315 takes up around 40% of the browser, while in Fig. 14B, the table of contents frame 1315 is dismissed and therefore not visible. The contents button 1410 automatically indicates the status of the TOC by displaying a dismissed icon (for example, in Fig. 14B) or an opened icon (for example, in Fig. 14A).

The user may access other links in the content frame 1305 from the TOC frame 1315 by clicking on a corresponding subtopic in the TOC tree, for example, the subtopic "Extending Database Servers" 1415. In response, as shown in Fig. 14C, the content frame 1305 displays content corresponding to that subtopic and the TOC frame 1315 highlights the selected subtopic 1415. Also evident in this particular example is that the top frame 1310 includes a home button 1420 that automatically changes its appearance (like the contents button) depending on whether the browser is displaying a home page or not.

Alternatively, the user may access links in the content frame 1305 by clicking on the corresponding link in the content frame, for example, subtopic link "Java Database Connectivity (JDBC)" 1425 shown in Fig. 14B. In response, as shown in Fig. 14D, the content frame 1305 displays content corresponding to that subtopic, including a title 1430 representing that subtopic. Furthermore, the TOC frame 1315 automatically highlights the selected subtopic 1435 in the TOC tree to indicate to the user which content is displayed in

the content frame 1305. Such synchronization between the TOC and the content helps the user to effectively and efficiently navigate through the many subtopics and topics of an application.

Referring again to Fig. 14C, when the user selects an example using either an example icon or link 1440 in the content frame 1305 or a subtopic icon or link 1445 in the TOC frame 1315, the browser displays in the content frame 1305 an annotation page 1500 that describes the corresponding example as shown in Fig. 15A. Referring also to Fig. 15B, the annotation page 1500 includes annotations 1505 descriptive of the example (also called prose), and at least one link to another annotation page 1510.

For example, when the user clicks the link for the source file "querydb.html", the content frame 1305 displays the source code file annotation page "querydb.html" 1525 as shown in the browser of Fig. 15C. The content frame displays source code fragments 1530 relating to the corresponding source code file querydb.html, as shown in Fig. 15D. The source code fragments 1530 have been imported directly from the source code files of the example under consideration. The source code file annotation page in the content frame also displays annotations 1535 describing the source code file to which the topic corresponds; such annotations 1535 may be referred to as prose.

Source code is marked up with standard HTML . Annotations 1535 are easy to maintain because the annotation comments are embedded within source code comments in the source code files. For example, referring to Fig. 15E, the source code file 1560 is shown for the querydb.html source code 1525. Annotation pages, containing the source code fragments, are generated automatically from the marked up source code files.

Referring to the flow chart 1600 of Fig. 16A, an author of an example annotates the source code file by embedding annotations in source code comments (step 1605). The author also marks up the embedded annotations with HTML tags and one or more special instruction tags for the web-based system (step 1610). Once the source code files are marked up, as shown in the flow chart 1620 of Fig. 16B, a scripting programming language such as PERL parses the source code file to determine the program structure from the programming language, and reads the annotations and special instruction tags (step 1625). The PERL script generates annotation pages that reflect the program structure of the source code file (step 1630). Moreover, the PERL script provides links between annotation pages for navigating the

program structure (step 1635), incorporates the annotation comments (step 1640), while using Java Script to notify the framework (step 1645), and highlights language keywords in source code as links to related information (step 1650). Annotations are therefore easily maintained. Referring again to Fig. 16A, the author of the example may adjust or edit the source code and the annotations at the same time and at the same location (step 1615). Therefore, annotations are regenerated automatically.

The annotation page in the content frame also may contain one or more links to related information or from one annotation page to another annotation page. Such flexibility is due to the automatic generation of annotation pages discussed above because the PERL script determines the program structure and formats references to other parts of the program as links. For example, when the user selects the "sportsHeader.tag" link in the querydb.html file 1525, the content frame 1305 displays the sportsHeader.tag source code annotation page 1540, and the TOC frame 1315 automatically highlights the corresponding subtopic link 1700, as shown in Fig. 17A.

The selectable link in the content frame annotation page may correspond to background reference material. For example, when the "MIVAR" link is selected, a separate browser 1705 is spawned that displays information about the MIVAR tag, as shown in Fig. 17B. In this case, the PERL script finds keywords in the source code and formats them as links to the reference documentation. If the user selects the "WEB HOME" link in the content window, a browser 1710 is spawned that includes a glossary, as shown in Fig. 17C.

Referring again to Fig. 15C, the user can select an edit button 1562 in the top frame 1310 to open the source file corresponding to the annotation page in the content frame 1305 as plain text in a browser window 1715, as shown in Fig. 17D. In this case, the PERL script automatically strips the annotation comments out of the source file so the user is able to edit a source file that is unencumbered by long explanations. The user could then cut and paste text between this window and another text window that displays the user's own program.

Referring again to Fig. 15A, the user can run an example using the framework applet to launch a Java example applet or application, to download an HTML page containing a Java applet or other embeddable program object, or to send a common gateway interface (CGI) request to the web server to run non-interactive programs with output redirected to a browser window. In the case of Java examples, the running example calls Java application

programming interfaces (APIs) to ask the web browser to show the annotation pages. In the case of CGI programs, the example program returns an HTML page that contains JavaScript to ask the web browser to show the annotation pages. Thus, the examples run either on the server or in the restricted sandbox of the web browser. The examples could run on a local client if the framework used the security features of the browser to ask the user for permission to run the example locally.

In Fig. 15A, the user can run the example by selecting the Run button 1570. Referring to the flow chart 1655 in Fig. 16C, upon selection of the Run button 1570, the example launches (step 1660), using any of the methods described above, in a browser window 1800, as shown in Fig. 18A. The TOC frame 1315 simultaneously and automatically synchronizes with the running example by highlighting the current annotation page for the source code that implemented the example. A PERL script generates the TOC automatically by parsing the HTML pages for links that indicate hierarchical structure much like the automatic generation of the annotation pages.

Moreover, the example simultaneously and automatically synchronizes the content frame 1305 with the running example by displaying the current annotation page, in this example, the querydb.html file 1525 discussed above. The user can then view the annotated source code or edit the source file while the example is running, and jump from keywords to the reference documentation for the keywords as described above.

When the user interacts (step 1665) with the example in the browser window 1800, (by, for example, entering a customer number and selecting the "Submit" button in the browser window 1800), the user interface changes (for example, the Customer Report corresponding to the entered customer number) in the browser window 1800, as shown in Fig. 18B (step 1670). The user interface changes correspond to the next annotation page, cust_db.html file, 1586 used in the running example. The running example automatically and simultaneously calls an annotation request module with the example, file, class and function names of the program unit for which annotations should be displayed (step 1675). The annotation request module maps that request to an annotation page and tells the browser to display the annotation page in the content frame (step 1680). For example, in Fig. 18B, the next annotation page is cust_db.html 1586. The cust_db.html subtopic is highlighted 1585 in the TOC frame 1315 and the content frame 1305 displays the cust_db.html annotation page

1586, as shown in Fig. 18B. Embedded annotation markup 1810 are shown in the cust_db.html source code file displayed in Fig. 18C. The embedded markup 1810 is used when generating the annotation page from the source code file.

Referring again to Fig. 14C, in addition to selecting an example, the user could also select a how to document icon or link 1450 in the content frame 1305 or a how to document subtopic link 1455 in the TOC frame 1315. In this case, a how to document 1900 is displayed in the content frame 1305, as shown in Fig. 19A. For clarity, the complete document 1900 is shown in Fig. 19B. In synchronization, the TOC frame 1315 highlights the displayed subtopic 1902 corresponding to the how to document 1900.

Also in synchronization, the top frame 1310 dynamically changes to reflect the selected how to document 1900. For example, in the top frame are now displayed an up button 1905, a first page button 1910, a previous page button 1915, a next page button 1920, a last page button 1925, and a list button 1930. The up button 1905, when selected, jumps to the category or subtopic that lists the how to document 1900. For example, the subtopic "Extending a Database Server" lists the "Create Web applications" how to document 1900, and upon selection of the up button 1905, the browser would display all information relating to the "Extending a Database Server" subtopic, as shown in Fig. 14C. Generally, the top frame changes to match the content type which is specified in the Java Script call that notifies the framework that a new content page has displayed.

The document 1900 contains pointers to other documents or pages, the pointers being accessed in one of several ways. The pointers may be selected directly from the content frame 1305 by clicking on a link in the content frame which is indicated by, for example, a different color, font, or style. For example, the user may click on the Application Page link 1930 or the Prepare Database link 1935 in the content frame 1305 as shown in Fig. 19B. Pointers may be selected from the TOC frame 1315 by selecting a subtopic for the Create Web applications document 1902. For example, the subtopic Prepare Database 1940 corresponds to the link Prepare Database 1935, and selection of either of these pointers would take the user to the same document. Upon selection of a pointer, the document or page corresponding to the pointer is then displayed to the user.

Referring also to Fig. 19C, if the user selects the list button 1930 in the top frame, a local table of contents window 1320 is displayed. For example, if the user selects the list

button 1930 in Fig. 19A, the window 1320 is spawned, as shown in Fig. 19C. A traditional model for navigating a document is with a hierarchical table of contents. The TOC model orients the user at all times but restricts the content, which may not fit a hierarchy. A newer model for navigating a document is the hypertext web. The web model links any topic in a document with a more detailed document that expands on the same topic. The web model removes artificial restrictions but quickly disorients the user.

In contrast, the web version of byExample® uses a local table of contents model that provides a structured hierarchical view at a local corner of an unstructured web page of links. In this model, some of the pages in the document are root pages for the local TOC. For example, the page corresponding to the Create Web applications document 1900 is a root page for the local TOC. When the user navigates to any page that is unique to the local TOC, such as the root page (which is unique), the framework applet reads the local TOC for that root page. As seen in Fig. 19D, when the user navigates to the Introduction page in the local TOC 1320, the framework applet reads the local TOC for that root page and displays the corresponding information in the TOC frame 1315 and the content frame 1305. Each page in the local TOC is unique within the local TOC. The user can step through the pages in the local TOC sequentially, see the number of the page in the local TOC sequence, or view the local TOC hierarchy with the current page selected. Thus, the user is oriented within the local topic.

The local TOC does not constrain links in pages within the local TOC. That is, pages within the local TOC can have links to pages that are not in the local TOC 1320. For example, when the user selects the Informix Web Integration Option link 1942 in the Introduction document, the browser opens an Informix Web Integration Option page 1945 that is not in the local TOC 1320, as shown in Fig. 19E.

Moreover, pages can appear within multiple local TOCs. For example, the page entitled "Create subspace" 1950 in the local TOC in Fig. 19D might appear in another local TOC relating to another document. Only the root page (in this example, the "Create Web applications" page 1900) is constrained to a single local TOC.

The user may navigate through the local TOC directly from the local TOC Window by selecting the topic of interest in the local TOC. Additionally, the user may navigate the through the local TOC via the top frame 1310 using the first page button 1910, the previous

page button 1915, the next page button 1920, or the last page button 1925. These buttons basically turn the pages in the document for the user by moving through the local TOC. For example, if the user selects the next page button 1920 while in the Create Web applications document 1900, the browser displays the next document in the Create Web applications document, which is the "Table of Contents" document 1955, as shown in Fig. 19F.

Referring to Fig. 20, a model 2000 of how the local TOCs 2005 are built is shown. The building of local TOCs imposes ordered views on the unstructured (or random) web of pages 2010, thus facilitating viewing of the web of pages 2010.

Other embodiments are within the scope of the following claims.

The web-based instruction system may support Java, Visual Basic, C, C++, HTML, Perl, JavaScript, SQL, Informix Stored Procedure Language (SPL), Embedded SQL for C(ESQL/C), SQLJ, JSP, ASP, and Informix Web DataBlade Module languages.

What is claimed is:

CLAIMS:

1. A method, performed in a web-based environment on a computer system, of helping a user learn to implement an application, the method comprising:

providing a predetermined plurality of applications;

presenting an annotation page that includes one or more annotations descriptive of a source file of a predetermined application, each annotation including keyword links, annotation links, and detail of implementation of the application;

permitting the user to select a link in an annotation;

if the user selects a keyword link, presenting reference documentation associated with that keyword; and

if the user selects an annotation link, presenting another annotation descriptive of another source file of a predetermined application.

2. The method of claim 1 further comprising performing a predetermined application and presenting one or more annotations descriptive of the performed application in coordination with performance of the predetermined application.

3. The method of claim 2 in which performing the predetermined application comprises receiving input from the user.

4. The method of claim 3 further comprising presenting another annotation page in coordination with performance of the predetermined application based on input from the user.

5. The method of claim 4 in which presenting another annotation page comprises:

automatically and simultaneously calling an annotation request module including application, file, class and function names of a program unit for which detail should be displayed;

mapping the request to an annotation; and

informing a browser window in the web-based environment to display the other annotation page.

6. The method of claim 3 in which another annotation page is presented in coordination with performance of the predetermined application.

7. The method of claim 6 further comprising automatically generating a global table of contents comprising links to annotations by parsing structured links in web pages including annotation pages.

8. The method of claim 7 in which the links in the global table of contents are synchronized with presented annotations by highlighting links corresponding to a current annotation page.

9. The method of claim 8 in which the global table of contents is presented in a first frame of a first browser window, the annotation page is presented in a second frame of the first browser window, and the predetermined application is performed in a second browser window.

10 The method of claim 2 in which performing the predetermined application comprises launching a Java applet or application.

11. The method of claim 10 in which launching the Java applet or application comprises calling a Java application programming interface to ask a web browser to show the annotation page.

12. The method of claim 2 in which performing the predetermined application comprises downloading a hyper-text markup language page containing a Java applet.

13. The method of claim 2 in which performing the predetermined application comprises sending a common gateway interface request to a web server that launches the application in a window in the web-based environment.

14. The method of claim 13 in which the application returns a hyper-text markup language page that includes JavaScript to ask a web browser to display the one or more annotations.

15. The method of claim 2 in which the annotation page is presented in a first browser window and the predetermined application is performed in a second browser window.

16. The method of claim 1 in which application implementation detail includes text descriptive of the application, fragments of source code from the application, or both.

17. The method of claim 16 in which source code fragments are imported directly from the source code file of the presented application.

18. The method of claim 1 further comprising automatically generating the annotation page descriptive of the source code file of a predetermined application.

19. The method of claim 18 in which generating the annotation page comprises:
receiving a source code file that has embedded text marked up with instructions;
parsing the source code to determine a structure of the predetermined application; and
generating one or more annotations based on the predetermined application structure and instructions.

20. The method of claim 19 in which generating the annotation page comprises:
generating one or more annotation links for navigating the annotations of the predetermined application;
generating application implementation detail based on the embedded information; and
generating one or more keyword links for reference documentation.

21. The method of claim 20 in which generating the annotation page comprises highlighting the keyword links and the annotation links in the annotation page.

22. The method of claim 19 further comprising automatically updating the annotation page descriptive of the source code file of the predetermined application when an updated source code file is received.

23. The method of claim 1 further comprising automatically generating a global table of contents by parsing the plurality of annotations for annotation links.

24. The method of claim 23 further comprising providing the global table of contents, in which the global table of contents comprises links to annotations.

25. The method of claim 23 further comprising generating a local table of contents, in which the local table of contents comprises links to web pages including annotation pages relating to an application.

26. The method of claim 25 further comprising providing the local table of contents when a local link in the global table of contents is selected.

27. The method of claim 1 in which the presented annotation page is descriptive of the performed application and the annotation page is presented in coordination with performance of the predetermined application.

28. The method of claim 1 further comprising:

generating a source code file stripped of annotation mark up, the generated source code file including source code of the application but not including text from the annotations;

presenting the stripped source code file; and

permitting the user to edit the stripped source code file.

29. A method, performed in a web-based environment on a computer system, of teaching user to implement an application, the method comprising:

providing a predetermined plurality of applications;

performing a predetermined application; and

presenting an annotation page descriptive of a performed application in coordination with performance of the predetermined application, the annotation page including detail of application implementation and links to annotations and reference documentation.

30. A method, performed in a web-based environment on a computer system, of teaching a user to implement an application, the method comprising:

automatically assembling a global table of contents based on content in the environment, the global table of contents including a plurality of links to content within the environment;

providing the global table of contents;

generating a local table of contents that includes links to content that orient the user within a local topic; and

permitting the user to select links from the local table of contents to access local topics.

31. A method, performed in a web-based environment on a computer system, of teaching a user to implement an application, the method comprising:

providing a plurality of predefined interactive examples;

performing one or more of the predefined interactive examples in response to user selection;

presenting one or more annotations descriptive of the performed interactive example in coordination with performance of the predefined interactive example; and

allowing the user to selectively explore different aspects of the performed interactive example, the annotations, or both.

32. A web-based computer system for teaching a user to implement an application, the system comprising:

one or more predefined interactive applications, a predefined interactive application selectively executable by the user of the web-based computer system; and

an annotation page including one or more annotations, in which the annotation page describes a predefined interactive application, and the annotation page further includes:

one or more links, and

detail of implementation of the application,

in which different annotations are automatically provided in the annotation page in response to selective execution of a predefined interactive application.

33. The system of claim 32 further comprising a utility through which the user can access source code associated with a predefined interactive application.

34. The system of claim 33 in which the utility enables the user to view or copy a predefined interactive application's source code.

35. The system of claim 32 in which detail of implementation of the application comprises text descriptive of the application, fragments of source code associated with the application, or both.

36. The system of claim 32 in which a link comprises a keyword link that provides the user with access to a body of reference documentation or an annotation link that provides the user with access to another annotation page.

37. The system of claim 32 further comprising a web-browser window that includes a framework that comprises:

a content frame that displays the annotations;

a framework applet that displays a navigation bar; and

a table of contents frame that displays a table of contents hierarchy of links.

38. The system of claim 37 in which the framework applet comprises a Java applet.

39. The system of claim 37 in which a Java Script automatically determines whether the framework is present in the web browser window, and if the framework is present, notifies the framework applet about the content in the framework.

40. The system of claim 39 in which the table of contents automatically highlights a link in the hierarchy based on the content in the framework.

41. The system of claim 40 in which the user accesses an annotation page by selecting a link in the table of contents hierarchy.

42. The system of claim 40 in which the user accesses an annotation page by interacting with the navigation bar.

43. The system of claim 40 in which the table of contents highlights the hierarchy based on an annotation page displayed in the content frame.

44. The system of claim 37 in which the table of contents is dismissible or resizable.

45. A web-based computer system for teaching a user to implement an application, the system comprising:

a web-browser window that includes a content frame, a framework applet, and a table of contents frame that displays a global table of contents hierarchy of links related to content in the content frame;

one or more annotations displayed in the content frame, each annotation describing a predefined interactive application and including links to other content; and

a table of contents window that displays a local table of contents hierarchy of links related to local content in the displayed annotation.

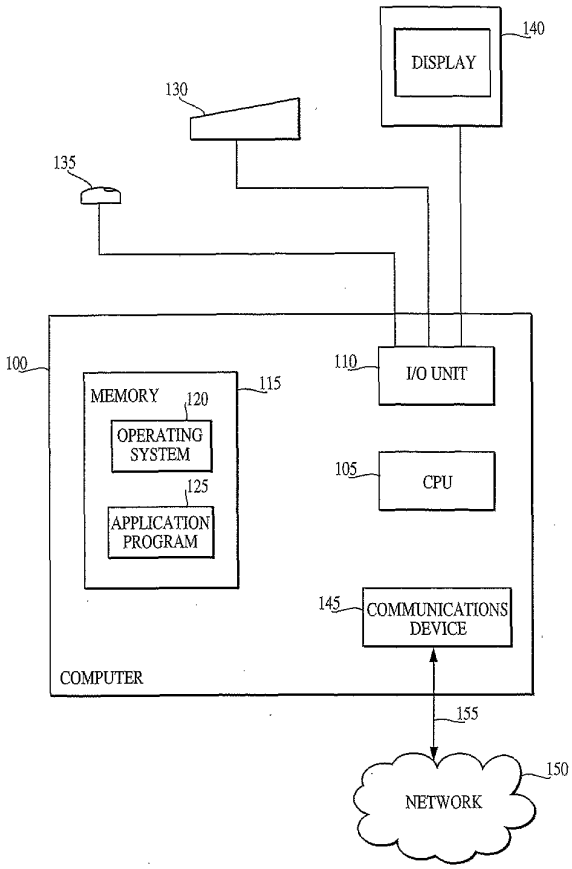

DISPLAY
140
130
135
100
MEMORY
120
115
OPERATING SYSTEM
125
APPLICATION PROGRAM
110
I/O UNIT
105
CPU
145
COMMUNICATIONS DEVICE
COMPUTER
155
150
NETWORK

FIG. 1

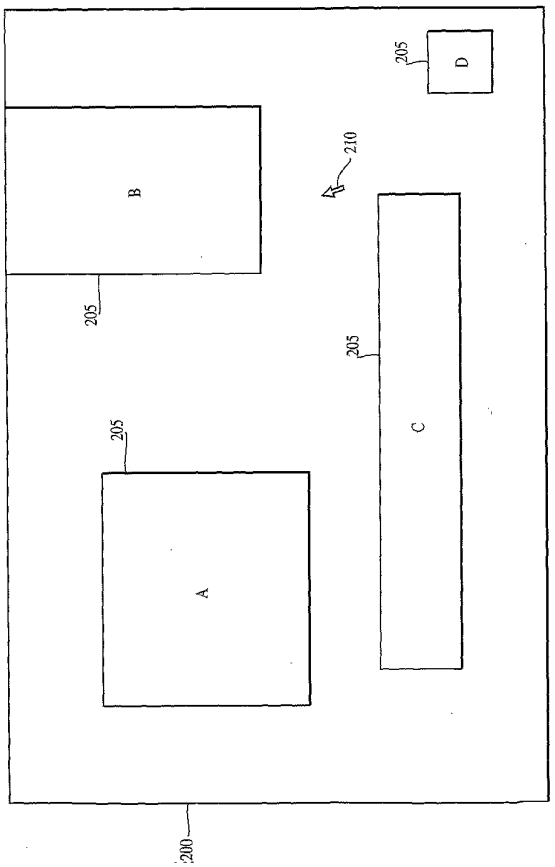


FIG. 2
PRIOR ART

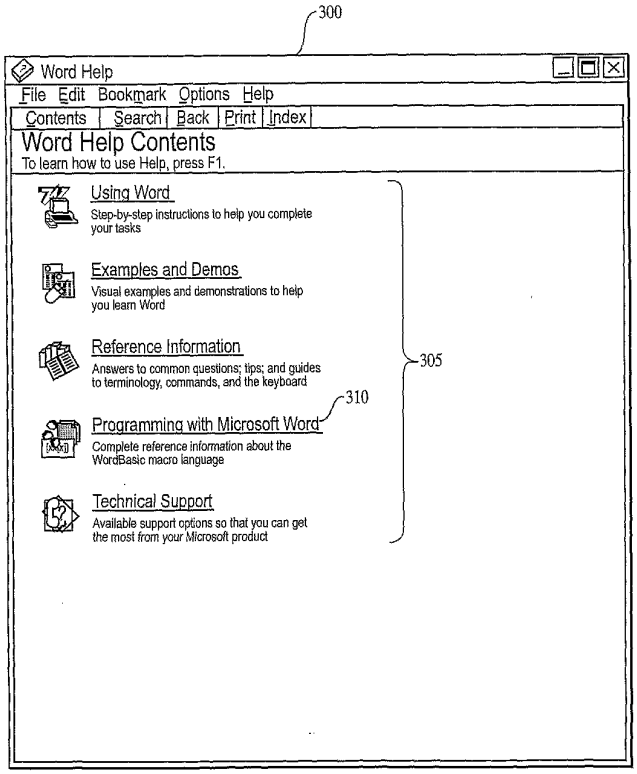


FIG. 3A
PRIOR ART

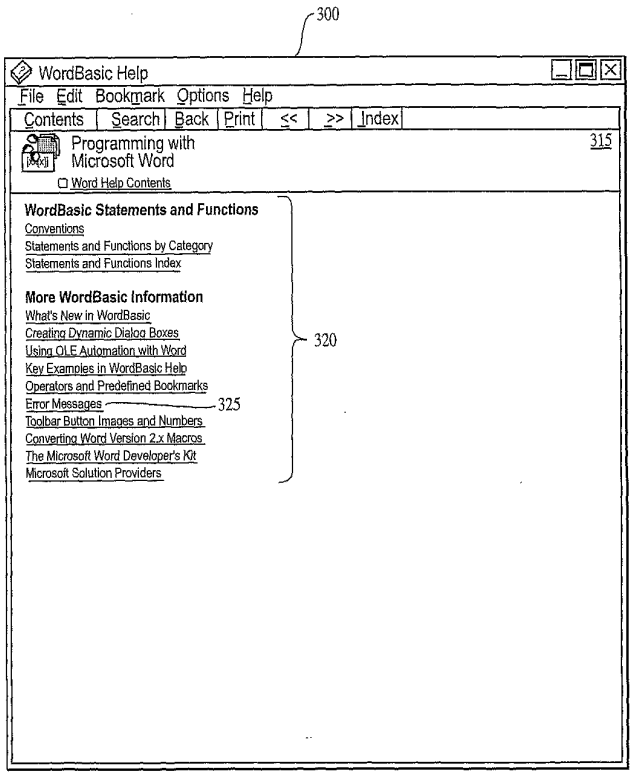


FIG. 3B
PRIOR ART

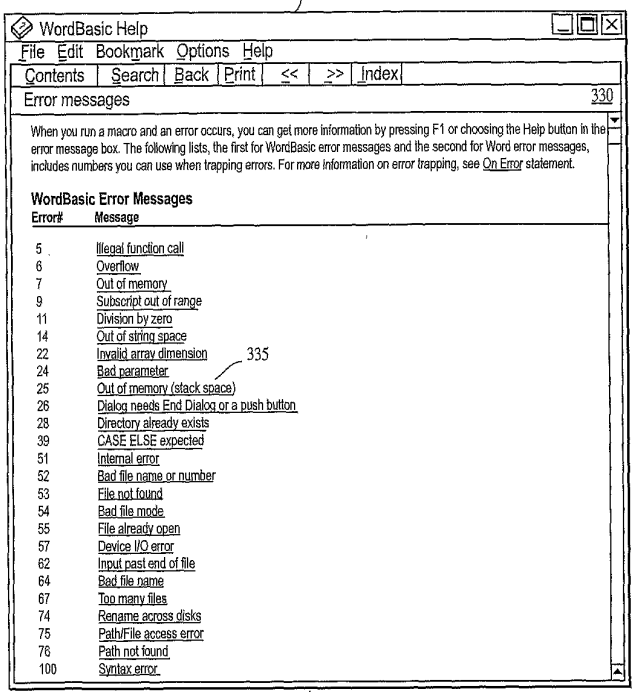


FIG. 3C
PRIOR ART

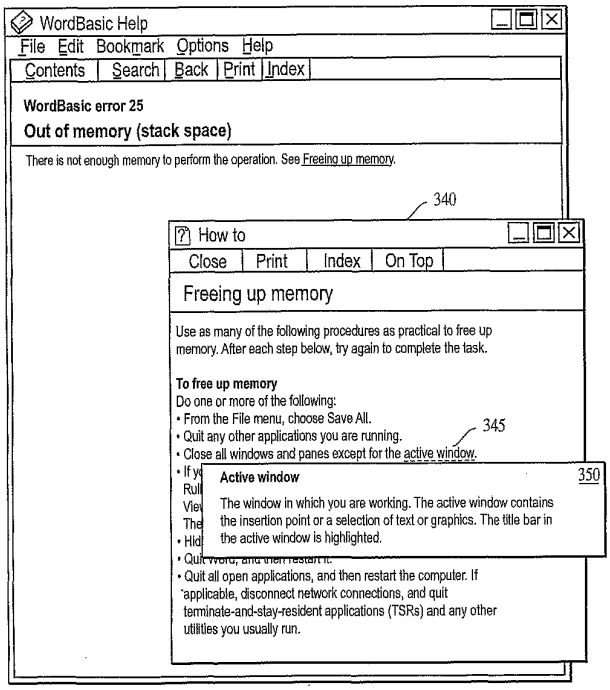


FIG. 3D
PRIOR ART

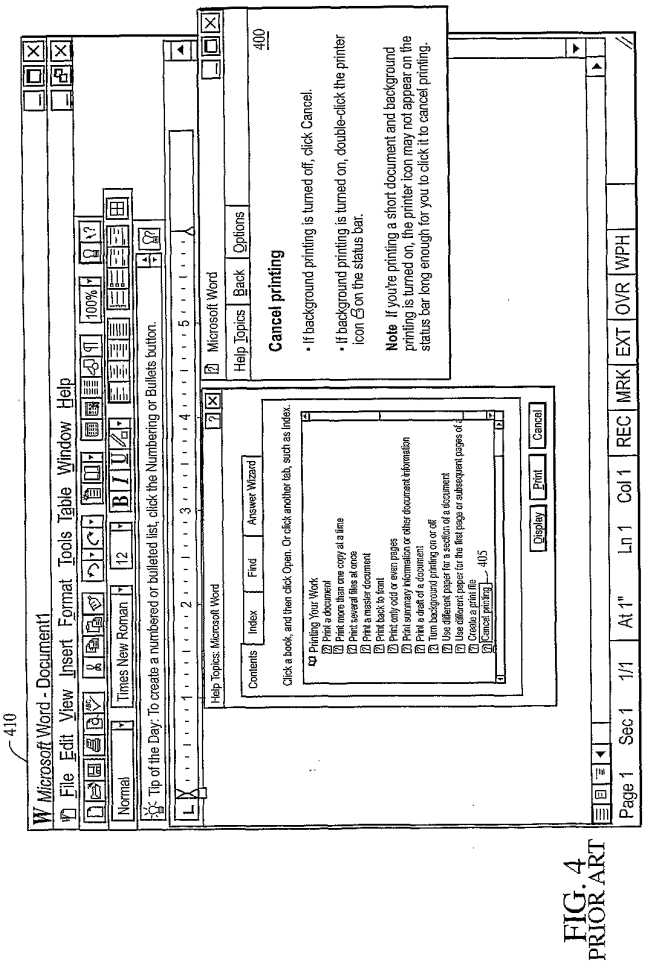


FIG. 4
PRIOR ART

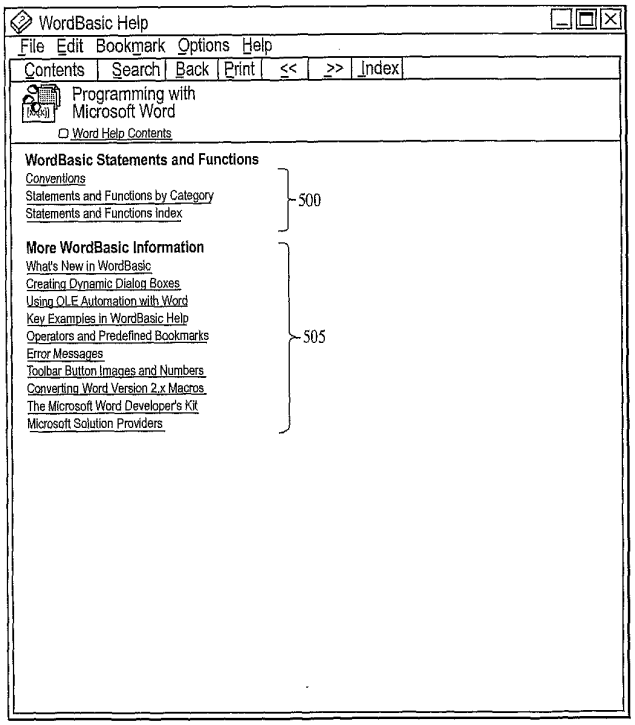


FIG. 5A
PRIOR ART

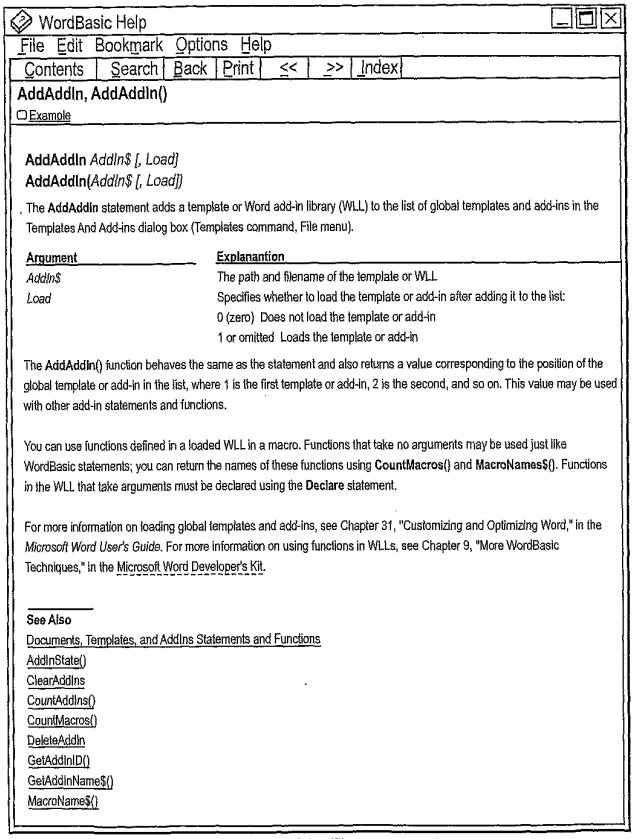


FIG. 5B
PRIOR ART

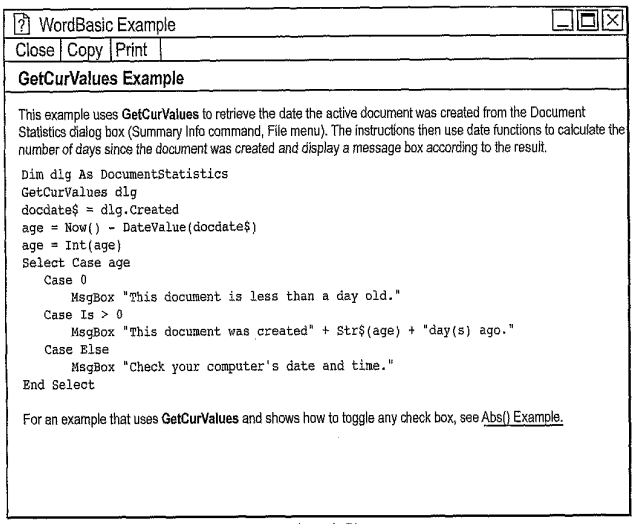


FIG. 5C
PRIOR ART

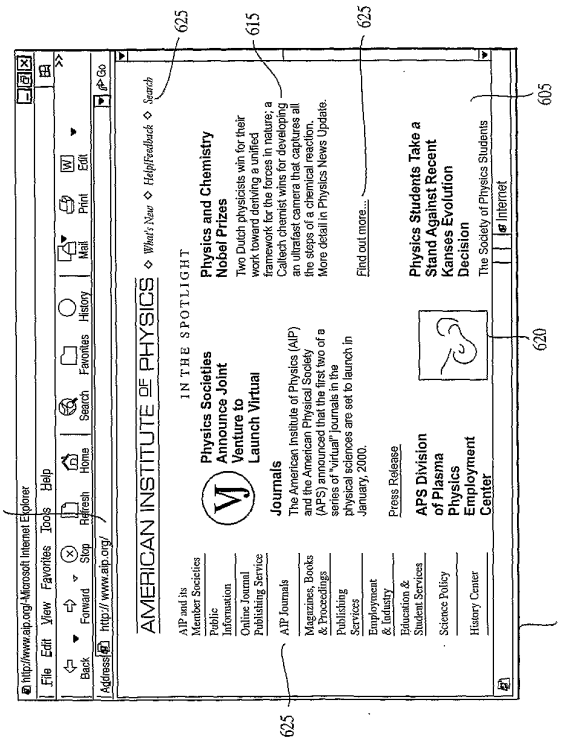

610
625
615
625
605
620
600
625
AMERICAN INSTITUTE OF PHYSICS
IN THE SPOTLIGHT
Physics Societies Announce Joint Venture to Launch Virtual Journals
Physics and Chemistry Nobel Prizes
Two Dutch physicists win for their work toward deriving a unified framework for the forces in nature; a Caltech chemist wins for developing an ultrafast camera that captures all the steps of a chemical reaction. More detail in Physics News Update.
Find out more...
Press Release
APS Division of Plasma Physics Employment Center
Physics Students Take a Stand Against Recent Kansas Evolution Decision
The Society of Physics Students

FIG. 6

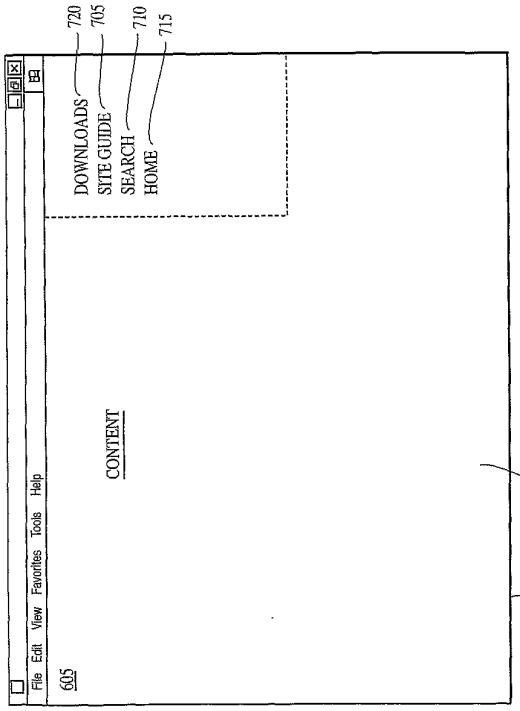

720
705
710
715
DOWNLOADS
SITE GUIDE
SEARCH
HOME
CONTENT
700
600
605
File
Edit
View
Favorites
Tools
Help

FIG. 7

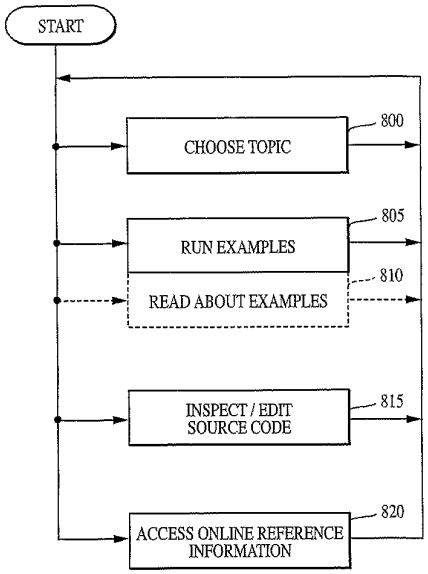

START
CHOOSE TOPIC
800
RUN EXAMPLES
805
READ ABOUT EXAMPLES
810
INSPECT / EDIT SOURCE CODE
815
ACCESS ONLINE REFERENCE INFORMATION
820

FIG. 8

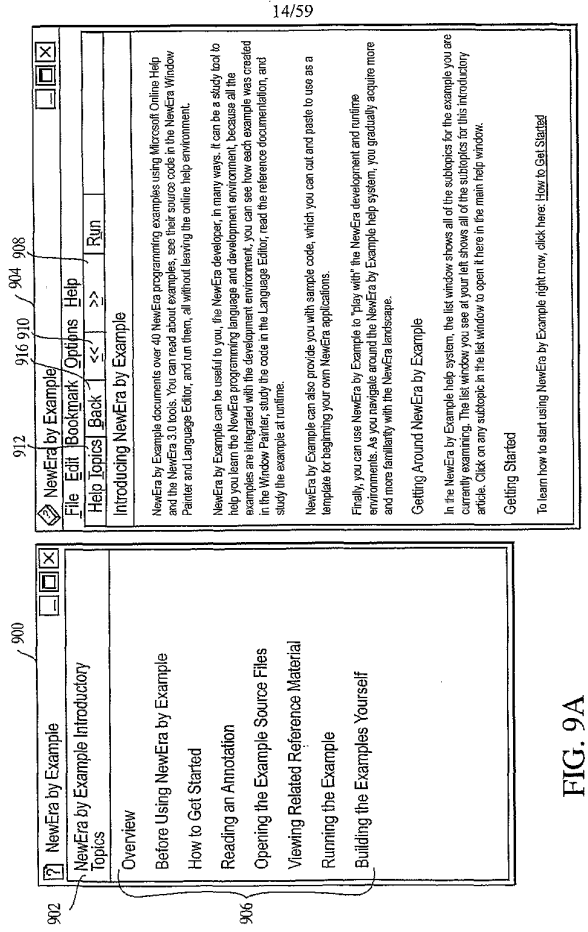

900
902
904
906
908
910
912
916
NewEra by Example
NewEra by Example Introductory Topics
Overview
Before Using NewEra by Example
How to Get Started
Reading an Annotation
Opening the Example Source Files
Viewing Related Reference Material
Running the Example
Building the Examples Yourself
File Edit Bookmark Options Help
Help Topics Back << >> Run
Introducing NewEra by Example
NewEra by Example documents over 40 NewEra programming examples using Microsoft Online Help and the NewEra 3.0 tools. You can read about examples, see their source code in the NewEra Window Painter and Language Editor, and run them, all without leaving the online help environment.
NewEra by Example can be useful to you, the NewEra developer, in many ways. It can be a study tool to help you learn the NewEra programming language and development environment, because all the examples are integrated with the development environment, you can see how each example was created in the Window Painter, study the code in the Language Editor, read the reference documentation, and study the example at runtime.
NewEra by Example can also provide you with sample code, which you can cut and paste to use as a template for beginning your own NewEra applications.
Finally, you can use NewEra by Example to "play with" the NewEra development and runtime environments. As you navigate around the NewEra by Example help system, you gradually acquire more and more familiarity with the NewEra landscape.
Getting Around NewEra by Example
In the NewEra by Example help system, the list window shows all of the subtopics for the example you are currently examining. The list window you see at your left shows all of the subtopics for this introductory article. Click on any subtopic in the list window to open it here in the main help window.
Getting Started
To learn how to start using NewEra by Example right now, click here: How to Get Started

FIG. 9A

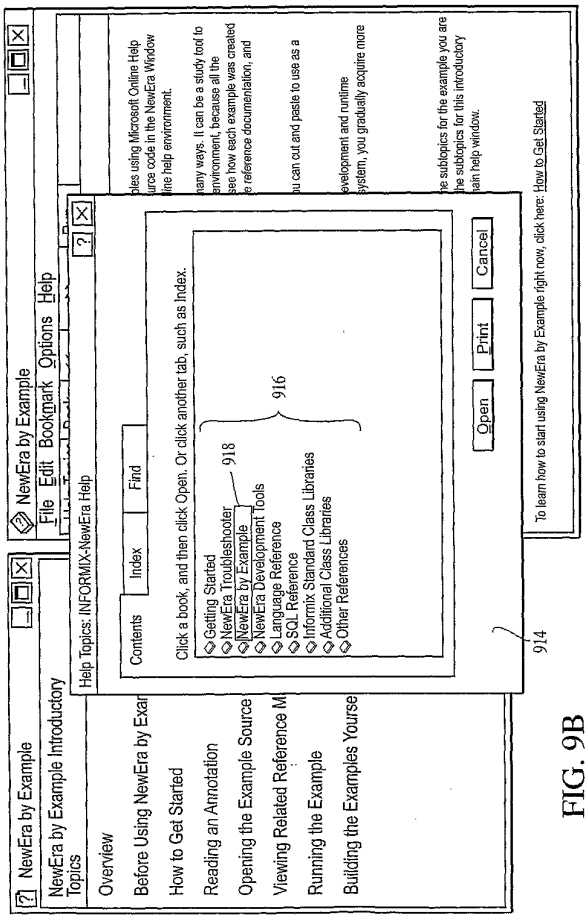

NewEra by Example
File Edit Bookmark Options Help
NewEra by Example Introductory Topics
Overview
Before Using NewEra by Exar
How to Get Started
Reading an Annotation
Opening the Example Source
Viewing Related Reference M
Running the Example
Building the Examples Yourse
Help Topics: INFORMIX-NewEra Help
Contents
Index
Find
Click a book, and then click Open. Or click another tab, such as Index.
Getting Started
NewEra Troubleshooter
NewEra by Example
NewEra Development Tools
Language Reference
SQL Reference
Informix Standard Class Libraries
Additional Class Libraries
Other References
916
918
914
Open
Print
Cancel
To learn how to start using NewEra by Example right now, click here: How to Get Started

FIG. 9B

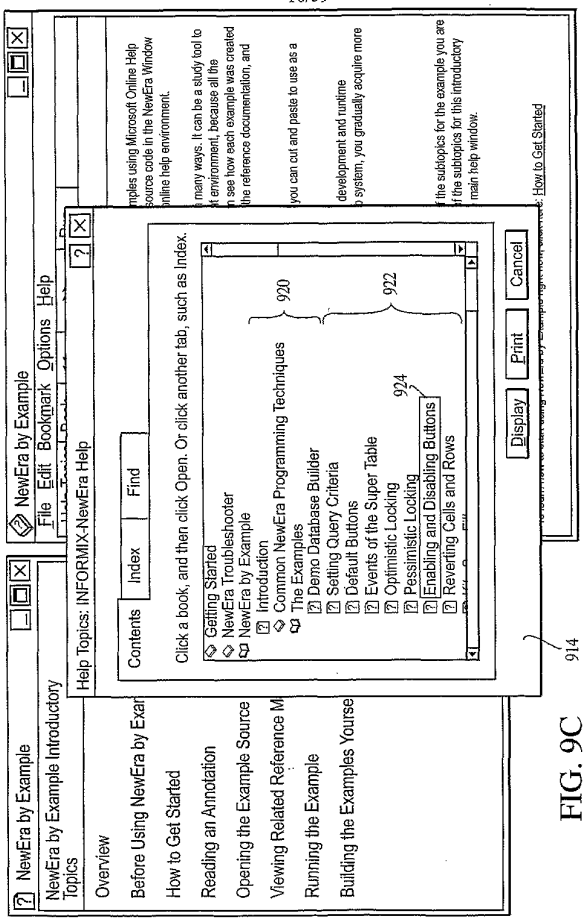

NewEra by Example
NewEra by Example Introductory Topics
Overview
Before Using NewEra by Exar
How to Get Started
Reading an Annotation
Opening the Example Source
Viewing Related Reference M
Running the Example
Building the Examples Yourse
NewEra by Example
File Edit Bookmark Options Help
Help Topics: INFORMIX-NewEra Help
Contents
Index
Find
Click a book, and then click Open. Or click another tab, such as Index.
Getting Started
NewEra Troubleshooter
NewEra by Example
Introduction
Common NewEra Programming Techniques
The Examples
Demo Database Builder
Setting Query Criteria
Default Buttons
Events of the Super Table
Optimistic Locking
Pessimistic Locking
Enabling and Disabling Buttons
Reverting Cells and Rows
920
922
924
914
Display
Print
Cancel
How to Get Started

FIG. 9C

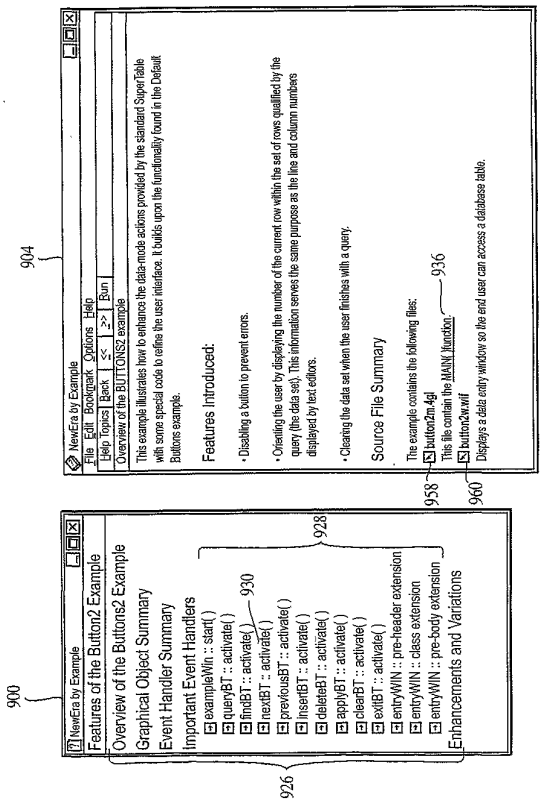

900
NewEra by Example
Features of the Button2 Example
Overview of the Buttons2 Example
Graphical Object Summary
Event Handler Summary
Important Event Handlers
exampleWin :: start( )
queryBT :: activate( )
findBT :: activate( )
nextBT :: activate( )
previousBT :: activate( )
insertBT :: activate( )
deleteBT :: activate( )
applyBT :: activate( )
clearBT :: activate( )
exitBT :: activate( )
entryWIN :: pre-header extension
entryWIN :: class extension
entryWIN :: pre-body extension
Enhancements and Variations
926
928
930
904
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
Overview of the BUTTONS2 example
This example illustrates how to enhance the data-mode actions provided by the standard SuperTable with some special code to refine the user interface. It builds upon the functionality found in the Default Buttons example.
Features Introduced:
• Disabling a button to prevent errors.
• Orienting the user by displaying the number of the current row within the set of rows qualified by the query (the data set). This information serves the same purpose as the line and column numbers displayed by text editors.
• Cleaning the data set when the user finishes with a query.
Source File Summary
The example contains the following files:
button2m.4gl
This file contain the MAIN function
button2w.per
Displays a data entry window so the end user can access a database table.
958
960
936

FIG. 9D

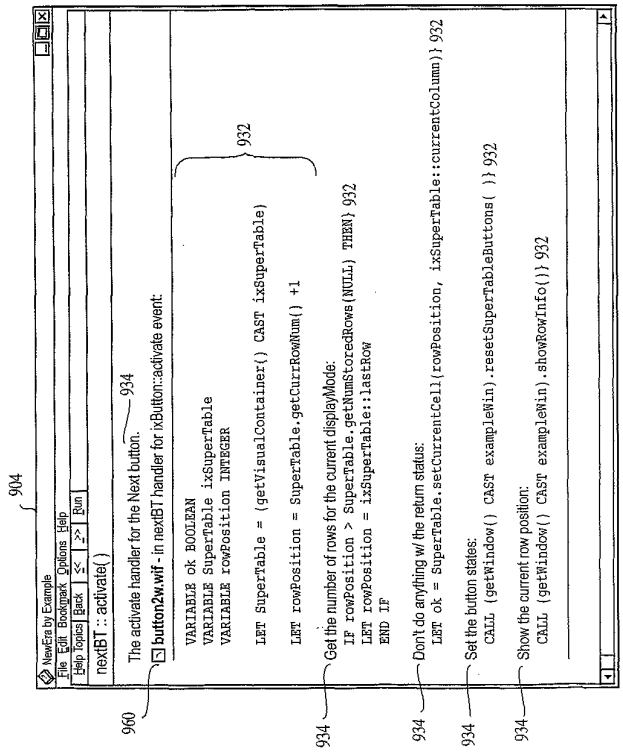

904
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
nextBT :: activate( )
The activate handler for the Next button:
934
960
button2w.wif - in nextBT handler for ixButton::activate event:
VARIABLE ok BOOLEAN
VARIABLE SuperTable ixSuperTable
VARIABLE rowPosition INTEGER
LET SuperTable = (getVisualContainer( ) CAST ixSuperTable)
LET rowPosition = SuperTable.getCurrRowNum( ) +1
932
Get the number of rows for the current displayMode:
IF rowPosition > SuperTable.getNumStoredRows(NULL) THEN} 932
LET rowPosition = ixSuperTable::lastRow
END IF
Don't do anything w/ the return status:
LET ok = SuperTable.setCurrentCell(rowPosition, ixSuperTable::currentColumn)} 932
Set the button states:
CALL (getWindow( ) CAST exampleWin).resetSuperTableButtons( )} 932
Show the current row position:
CALL (getWindow( ) CAST exampleWin).showRowInfo( )} 932

FIG. 9E

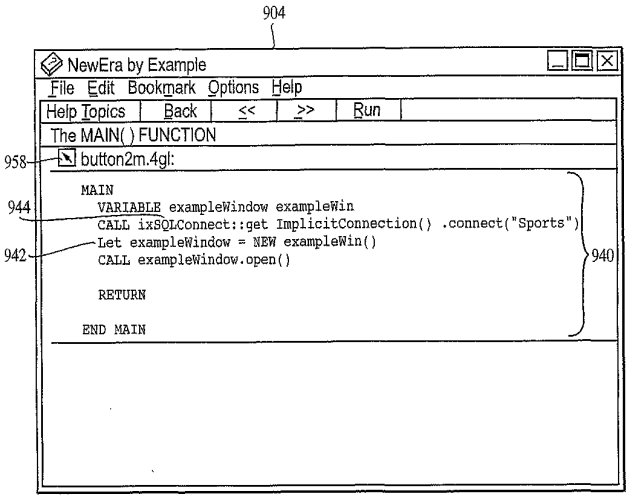

904
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
The MAIN( ) FUNCTION
958
button2m.4gl:
MAIN
944
VARIABLE exampleWindow exampleWin
CALL ixSQLConnect::get ImplicitConnection() .connect("Sports")
942
Let exampleWindow = NEW exampleWin()
CALL exampleWindow.open()
RETURN
END MAIN
940

FIG. 9F

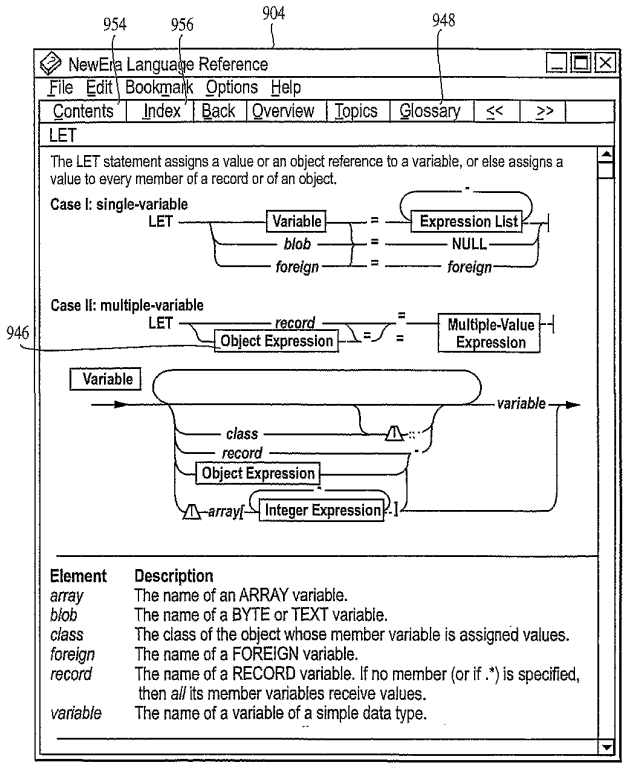

954
956
904
948
NewEra Language Reference
File Edit Bookmark Options Help
Contents Index Back Overview Topics Glossary << >>
LET
The LET statement assigns a value or an object reference to a variable, or else assigns a value to every member of a record or of an object.
Case I: single-variable
LET
Variable
blob
foreign
Expression List
NULL
foreign
Case II: multiple-variable
946
LET
record
Object Expression
Multiple-Value Expression
Variable
variable
class
record
Object Expression
array[
Integer Expression
Element
Description
array The name of an ARRAY variable.
blob The name of a BYTE or TEXT variable.
class The class of the object whose member variable is assigned values.
foreign The name of a FOREIGN variable.
record The name of a RECORD variable. If no member (or if .*) is specified, then all its member variables receive values.
variable The name of a variable of a simple data type.

FIG. 9G

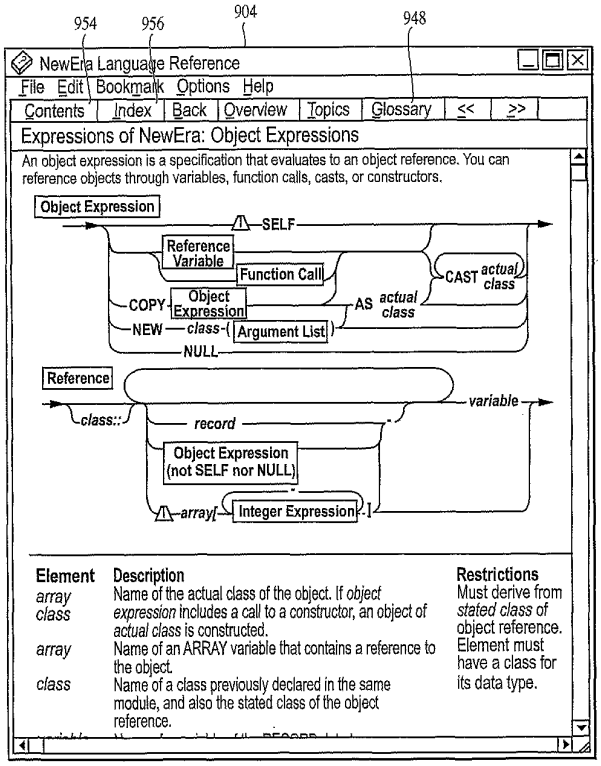

954
956
904
948
NewEra Language Reference
File Edit Bookmark Options Help
Contents Index Back Overview Topics Glossary << >>
Expressions of NewEra: Object Expressions
An object expression is a specification that evaluates to an object reference. You can reference objects through variables, function calls, casts, or constructors.
Object Expression
SELF
Reference Variable
Function Call
CAST actual class
COPY Object Expression
AS actual class
NEW class (Argument List)
NULL
Reference
variable
class::
record
Object Expression (not SELF nor NULL)
array[Integer Expression]
Element
Description
Restrictions
array class
Name of the actual class of the object. If object expression includes a call to a constructor, an object of actual class is constructed.
Must derive from stated class of object reference.
array
Name of an ARRAY variable that contains a reference to the object.
Element must have a class for its data type.
class
Name of a class previously declared in the same module, and also the stated class of the object reference.

FIG. 9H

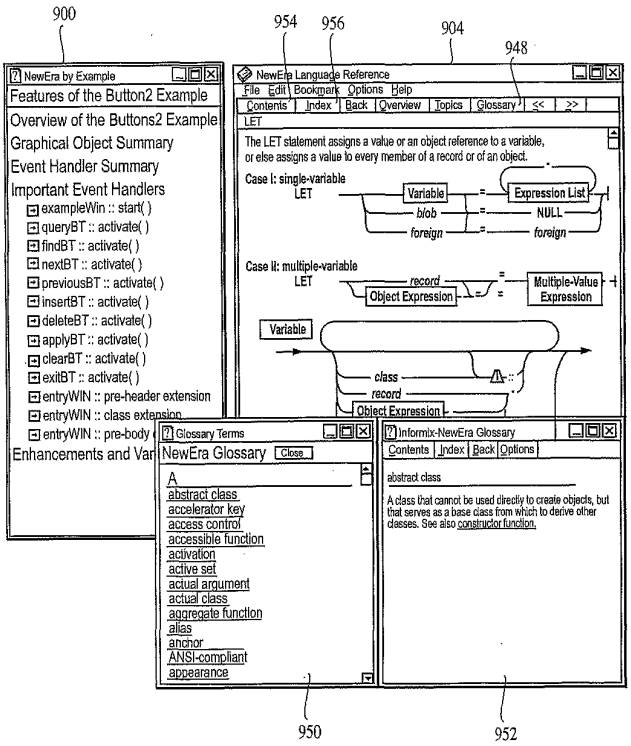

900
954
956
904
948
NewEra by Example
Features of the Button2 Example
Overview of the Buttons2 Example
Graphical Object Summary
Event Handler Summary
Important Event Handlers
exampleWin :: start( )
queryBT :: activate( )
findBT :: activate( )
nextBT :: activate( )
previousBT :: activate( )
insertBT :: activate( )
deleteBT :: activate( )
applyBT :: activate( )
clearBT :: activate( )
exitBT :: activate( )
entryWIN :: pre-header extension
entryWIN :: class extension
entryWIN :: pre-body
Enhancements and Var
NewEra Language Reference
File Edit Bookmark Options Help
Contents Index Back Overview Topics Glossary << >>
LET
The LET statement assigns a value or an object reference to a variable, or else assigns a value to every member of a record or of an object.
Case I: single-variable
LET
Variable
Expression List
blob
NULL
foreign
foreign
Case II: multiple-variable
LET
record
Object Expression
Multiple-Value Expression
Variable
class
record
Object Expression
Glossary Terms
NewEra Glossary
Close
A
abstract class
accelerator key
access control
accessible function
activation
active set
actual argument
actual class
aggregate function
alias
anchor
ANSI-compliant
appearance
Informix-NewEra Glossary
Contents Index Back Options
abstract class
A class that cannot be used directly to create objects, but that serves as a base class from which to derive other classes. See also constructor function.
950
952

FIG. 9I

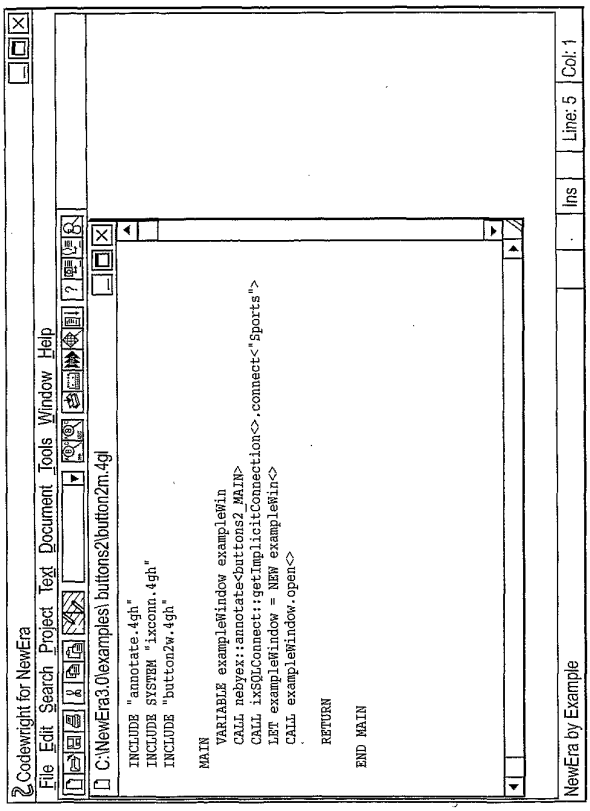

Codewright for NewEra
File Edit Search Project Text Document Tools Window Help
C:\NewEra3.0\examples\buttons2\button2m.4gl
INCLUDE "annotate.4gh"
INCLUDE SYSTEM "ixconn.4gh"
INCLUDE "button2w.4gh"
MAIN
VARIABLE exampleWindow exampleWin
CALL nebyex::annotate<buttons2_MAIN>
CALL ixSQLConnect::getImplicitConnection<>.connect<"Sports">
LET exampleWindow = NEW exampleWin<>
CALL exampleWindow.open<>
RETURN
END MAIN
NewEra by Example
Ins
Line: 5
Col: 1

FIG. 9J

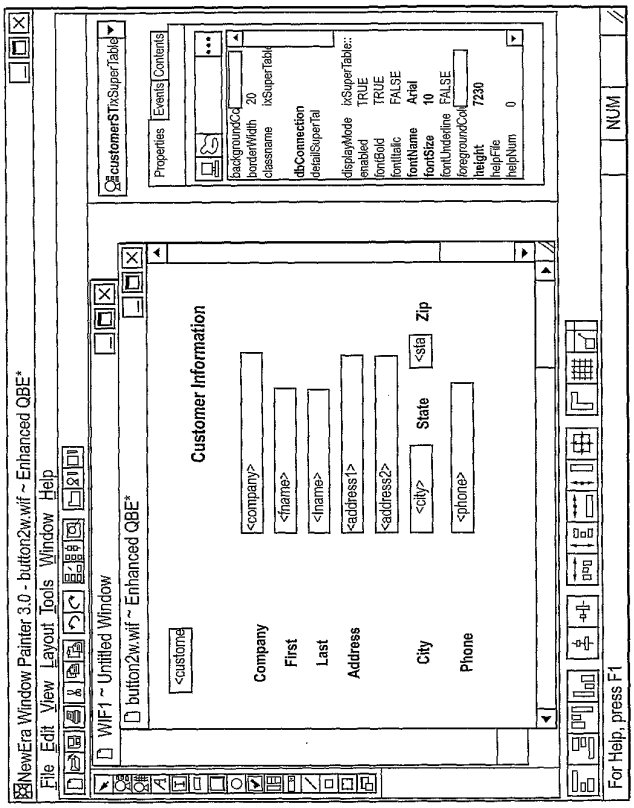

NewEra Window Painter 3.0 - button2w.wif ~ Enhanced QBE*
File Edit View Layout Tools Window Help
WIF1 ~ Untitled Window
button2w.wif ~ Enhanced QBE*
Customer Information
Company
First
Last
Address
City
State
Zip
Phone
<company>
<fname>
<lname>
<address1>
<address2>
<city>
<phone>
customerSTtxSuperTable
Properties
Events
Contents
dbConnection
For Help, press F1
NUM

FIG. 9K

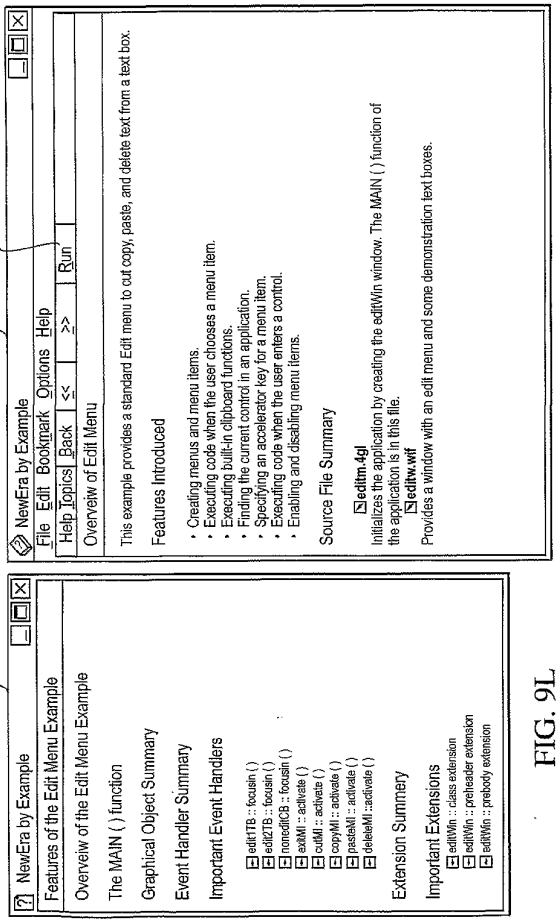

900
NewEra by Example
Features of the Edit Menu Example
Overveiw of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusin ( )
edit2TB :: focusin ( )
noneditCB :: focusin ( )
exitMI :: activate ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteMI ::activate ( )
Extension Summery
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
904
962
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
Overveiw of Edit Menu
This example provides a standard Edit menu to cut copy, paste, and delete text from a text box.
Features Introduced
Creating menus and menu items.
Executing code when the user chooses a menu item.
Executing built-in clipboard functions.
Finding the current control in an application.
Specifying an accelerator key for a menu item.
Executing code when the user enters a control.
Enabling and disabling menu items.
Source File Summary
editm.4gl
Initializes the application by creating the editWin window. The MAIN ( ) function of the application is in this file.
editw.4tf
Provides a window with an edit menu and some demonstration text boxes.

FIG. 9L

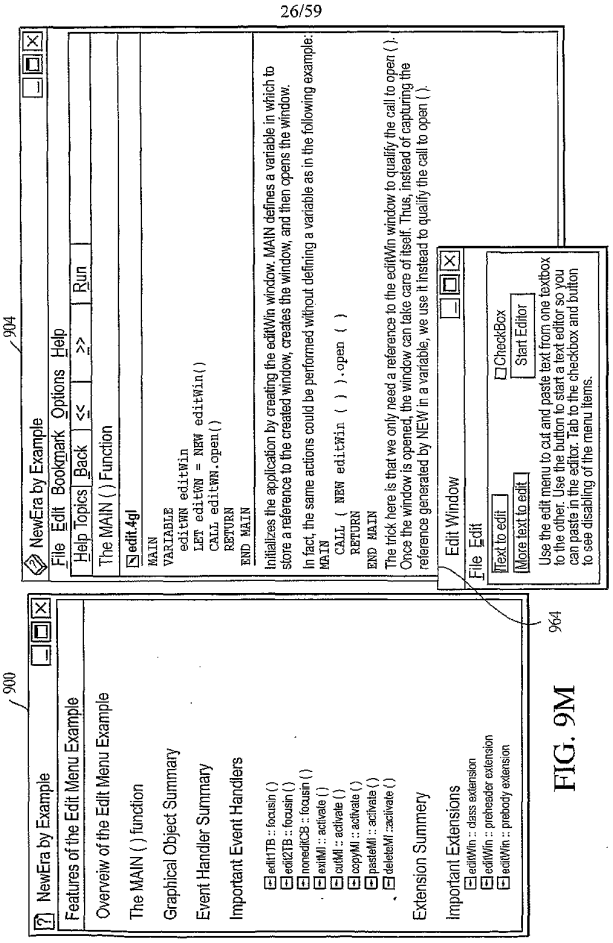

900
NewEra by Example
Features of the Edit Menu Example
Overveiw of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusin ( )
edit2TB :: focusin ( )
noneditCB :: focusin ( )
editMI :: activate ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteM :: activate ( )
Extension Summery
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
904
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
The MAIN ( ) Function
edit.4gl
MAIN
VARIABLE
editWN editWin
LET editWN = NEW editWin()
CALL editWN.open()
RETURN
END MAIN
Initializes the application by creating the editWin window. MAIN defines a variable in which to store a reference to the created window, creates the window, and then opens the window.
In fact, the same actions could be performed without defining a variable as in the following example:
MAIN
CALL (NEW editWin ( )).open ( )
RETURN
END MAIN
The trick here is that we only need a reference to the editWin window to qualify the call to open ( ). Once the window is opened, the window can take care of itself. Thus, instead of capturing the reference generated by NEW in a variable, we use it instead to qualify the call to open ( ).
964
Edit Window
File Edit
Text to edit
More text to edit
CheckBox
Start Editor
Use the edit menu to cut and paste text from one textbox to the other. Use the button to start a text editor so you can paste in the editor. Tab to the checkbox and button to see disabling of the menu items.

FIG. 9M

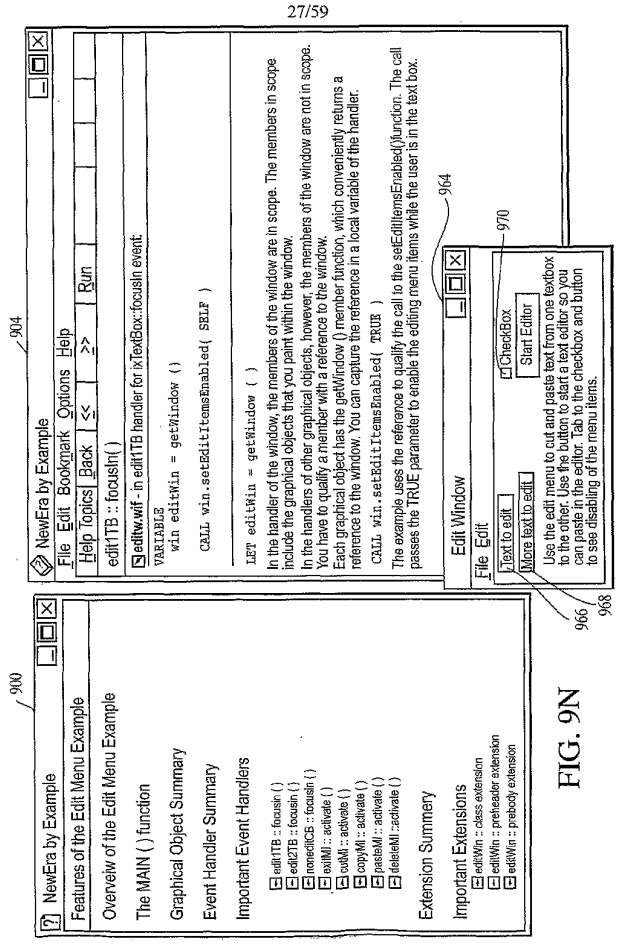

900
NewEra by Example
Features of the Edit Menu Example
Overveiw of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusIn ( )
edit2TB :: focusIn ( )
nonexistCB :: focusIn ( )
editMI :: activate ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteMI ::activate ( )
Extension Summery
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
904
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
edit1TB :: focusIn ( )
editw.wif - in edit1TB handler for ix TextBox::focusIn event
VARIABLE
win editWin = getWindow ()
CALL win.setEditItemsEnabled(SELF)
LET editWin = getWindow ( )
In the handler of the window, the members of the window are in scope. The members in scope include the graphical objects that you paint within the window.
In the handlers of other graphical objects, however, the members of the window are not in scope. You have to qualify a member with a reference to the window.
Each graphical object has the getWindow () member function, which conveniently returns a reference to the window. You can capture the reference in a local variable of the handler.
CALL win.setEditItemsEnabled(TRUE)
The example uses the reference to qualify the call to the setEditItemsEnabled()function. The call passes the TRUE parameter to enable the editing menu items while the user is in the text box.
964
Edit Window
File Edit
Text to edit
More text to edit
CheckBox
Start Editor
970
Use the edit menu to cut and paste text from one textbox to the other. Use the button to start a text editor so you can paste in the editor. Tab to the checkbox and button to see disabling of the menu items.
966
968

FIG. 9N

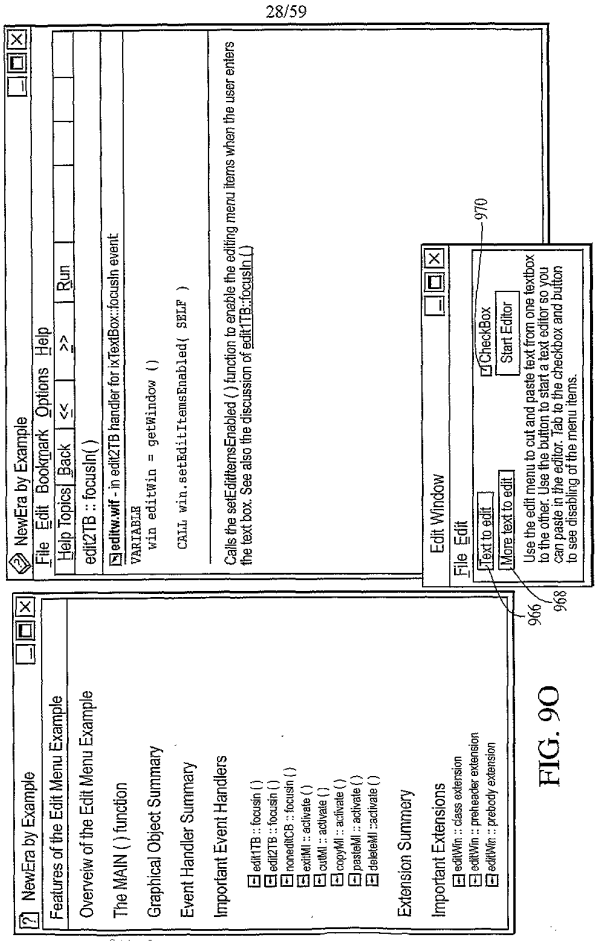

NewEra by Example
Features of the Edit Menu Example
Overveiw of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusIn ( )
edit2TB :: focusIn ( )
noneditCB :: focusIn ( )
editMI :: activate ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteMI ::activate ( )
Extension Summary
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
edit2TB :: focusIn ( )
editw.wif - in edit2TB handler for rxTextBox::focusIn event
VARIABLE
win editWin = getWindow ()
CALL win.setEditItemsEnabled(SELF)
Calls the setEditItemsEnabled ( ) function to enable the editing menu items when the user enters the text box. See also the discussion of edit1TB::focusIn ( )
Edit Window
File Edit
Text to edit
More text to edit
CheckBox
Start Editor
Use the edit menu to cut and paste text from one textbox to the other. Use the button to start a text editor so you can paste in the editor. Tab to the checkbox and button to see disabling of the menu items.
966
968
970

FIG. 9O

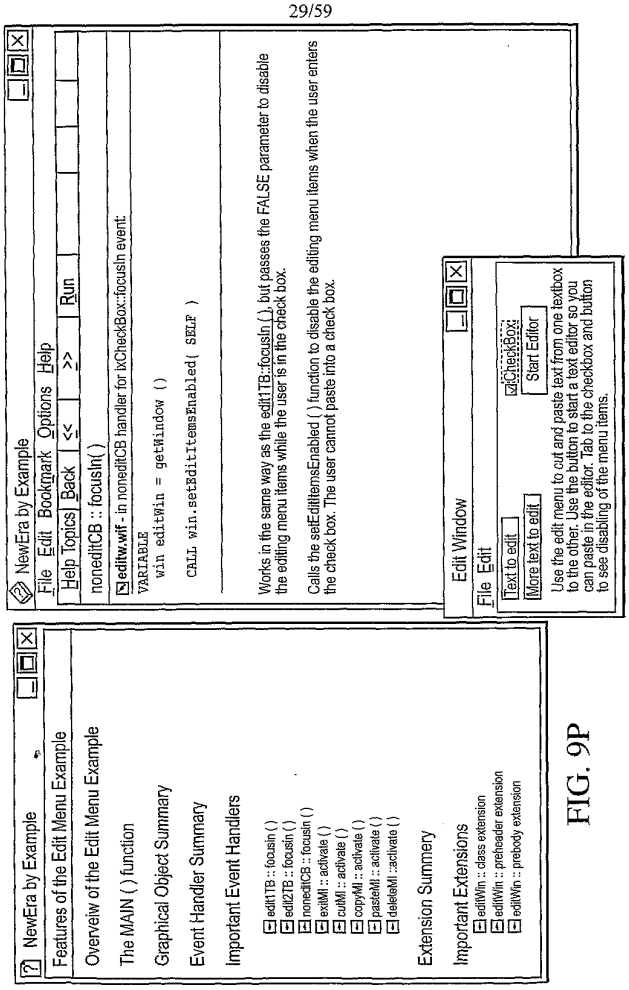

NewEra by Example
Features of the Edit Menu Example
Overveiw of the Edit Menu Example
The MAIN ( ) function
Graphical Object Summary
Event Handler Summary
Important Event Handlers
edit1TB :: focusin ( )
edit2TB :: focusin ( )
noneditCB :: focusin ( )
exitMI :: activate ( )
cutMI :: activate ( )
copyMI :: activate ( )
pasteMI :: activate ( )
deleteMI ::activate ( )
Extension Summery
Important Extensions
editWin :: class extension
editWin :: preheader extension
editWin :: prebody extension
NewEra by Example
File Edit Bookmark Options Help
Help Topics Back << >> Run
noneditCB :: focusin( )
editw.wif - in noneditCB handler for tkCheckBox::focusIn event:
VARIABLE
win editWin = getWindow ()
CALL win.setEditItemsEnabled(SELF)
Works in the same way as the edit1TB::focusIn ( ), but passes the FALSE parameter to disable the editing menu items while the user is in the check box.
Calls the setEditItemsEnabled ( ) function to disable the editing menu items when the user enters the check box. The user cannot paste into a check box.
Edit Window
File Edit
Text to edit
More text to edit
tkCheckBox
Start Editor
Use the edit menu to cut and paste text from one textbox to the other. Use the button to start a text editor so you can paste in the editor. Tab to the checkbox and button to see disabling of the menu items.

FIG. 9P

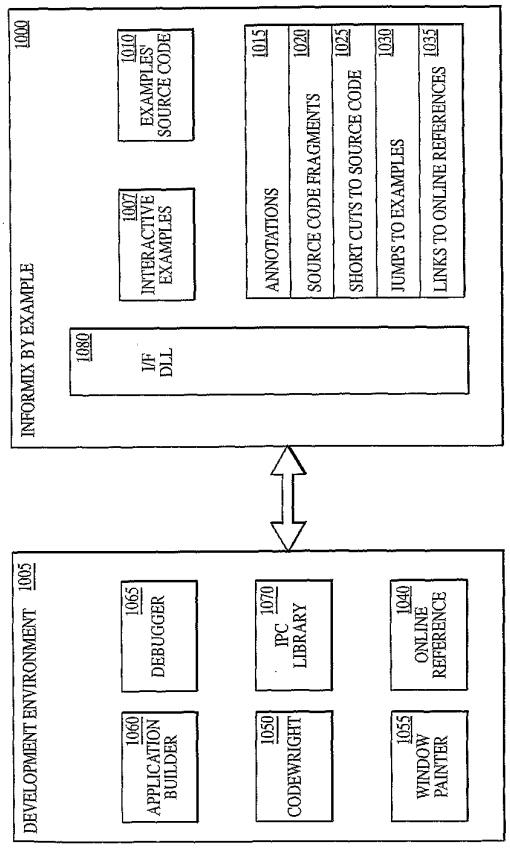

1000
INFORMIX BY EXAMPLE
1010
EXAMPLES' SOURCE CODE
1007
INTERACTIVE EXAMPLES
1080
I/F DLL
1015
ANNOTATIONS
1020
SOURCE CODE FRAGMENTS
1025
SHORT CUTS TO SOURCE CODE
1030
JUMPS TO EXAMPLES
1035
LINKS TO ONLINE REFERENCES
1005
DEVELOPMENT ENVIRONMENT
1065
DEBUGGER
1070
IPC LIBRARY
1040
ONLINE REFERENCE
1060
APPLICATION BUILDER
1050
CODEWRIGHT
1055
WINDOW PAINTER

FIG. 10

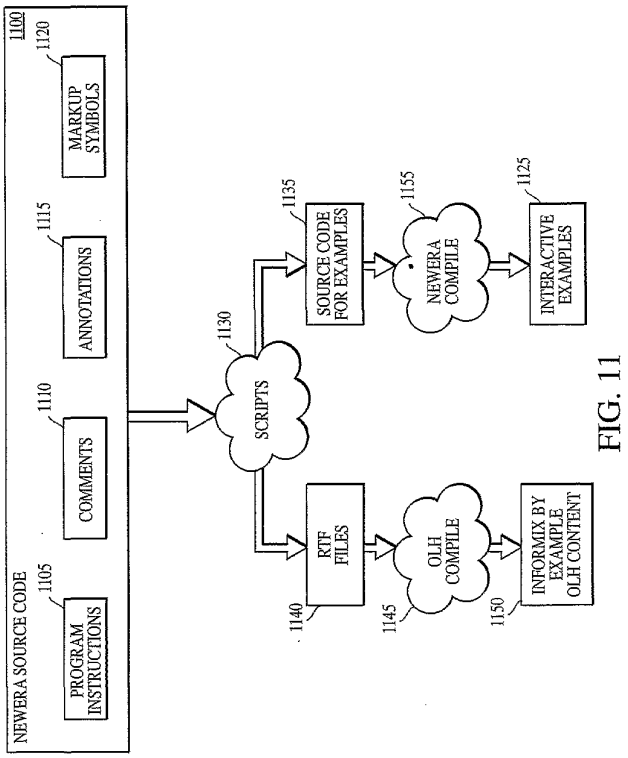

1100
NEWERA SOURCE CODE
1105
PROGRAM INSTRUCTIONS
1110
COMMENTS
1115
ANNOTATIONS
1120
MARKUP SYMBOLS
1130
SCRIPTS
1135
SOURCE CODE FOR EXAMPLES
1155
NEWERA COMPILE
1125
INTERACTIVE EXAMPLES
1140
RTF FILES
1145
OLH COMPILE
1150
INFORMIX BY EXAMPLE OLH CONTENT

FIG. 11

```
    FUNCTION driveStockRpt( destType SMALLINT, destName CHAR(*) )
1200 RETURNING VOID
    {.normal
    Since objects, in particular ixRow objects, cannot be passed
    as arguments to the report formatter, rows of fetched data will
    be unpacked into a record that matches the data types and lengths
    of elements in the fetched rows.
    }
    VARIABLE
         stockRec RECORD
              mn CHAR(15),     -- manufact.manu_name
              sn SMALLINT,     -- stock.stock_num
              sd CHAR(15),     -- stock.description
              sp MONEY(6,2),   -- stock.unit_price
              su CHAR(4)       -- stock.unit
              END RECORD

         stockStmt ixSQLStmt,
         stmtString CHAR(*),
         stockRow ixRow,

         errorCode INTEGER,
         logFile ixErrorLog
1205
    {.normal
    Use the implicit connection object to create an SQL statement
    object. The connection object must already be connected to a
    database.
    Checking the status of the prepare( ) call will confirm this.
1210 }
    {.[edit stmt}
    LET stockStmt =
    ixSQLConnect::getImplicitConnection().createStmtObject()
    {.]file stmt}
1215
```

FIG. 12

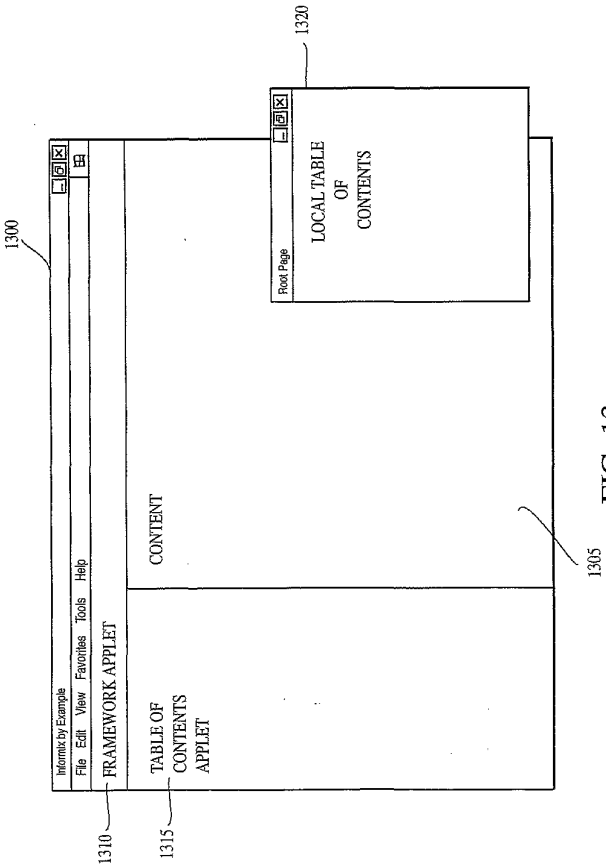

1320
LOCAL TABLE OF CONTENTS
Root Page
1300
Informix by Example
File Edit View Favorites Tools Help
FRAMEWORK APPLET
CONTENT
TABLE OF CONTENTS APPLET
1310
1315
1305

FIG. 13

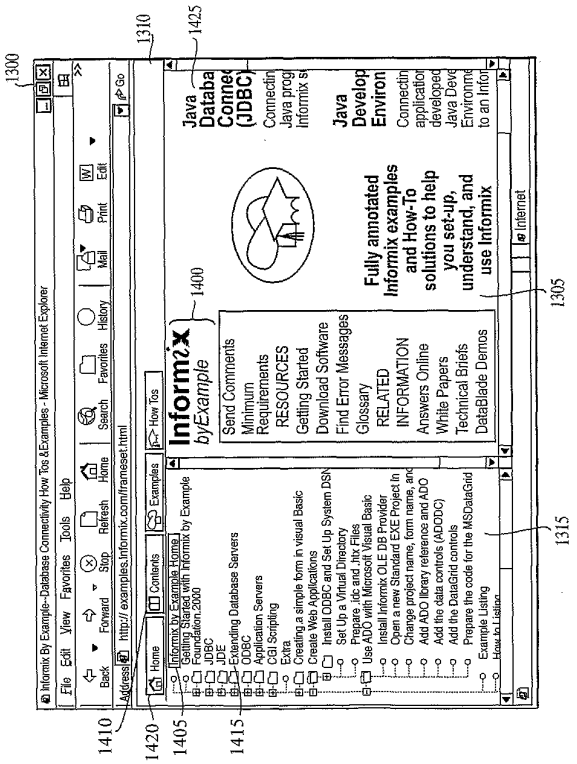

1300
1310
1425
1400
1305
1315
1410
1420
1405
1415
Informix By Example--Database Connectivity How Tos &Examples - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://examples.informix.com/frameset.html
Go
Home Contents Examples How Tos
Informix by Example Home
Getting Started with Informix by Example
Foundation.2000
JDBC
JPE
Extending Database Servers
ODBC
Application Servers
CGI Scripting
Extra
Creating a simple form in visual Basic
Create Web Applications
Install ODBC and Set Up System DSN
Set Up a Virtual Directory
Prepare .idc and .htx Files
Use ADO with Microsoft Visual Basic
Install Informix OLE DB Provider
Open a new Standard EXE Project in
Change project name, form name, and
Add ADO library references and ADO
Add the data controls (ADODC)
Add the DataGrid controls
Prepare the code for the MSDataGrid
Example Listing
How to Listing
Informix by Example
Send Comments
Minimum Requirements
RESOURCES
Getting Started
Download Software
Find Error Messages
Glossary
RELATED INFORMATION
Answers Online
White Papers
Technical Briefs
DataBlade Demos
Java Databa Connec (JDBC)
Connectin Java prog Informix s
Java Develop Environ
Connectin application developed Java Deve Environme to an Infor
Fully annotated Informix examples and How-To solutions to help you set-up, understand, and use Informix
Internet

FIG. 14A

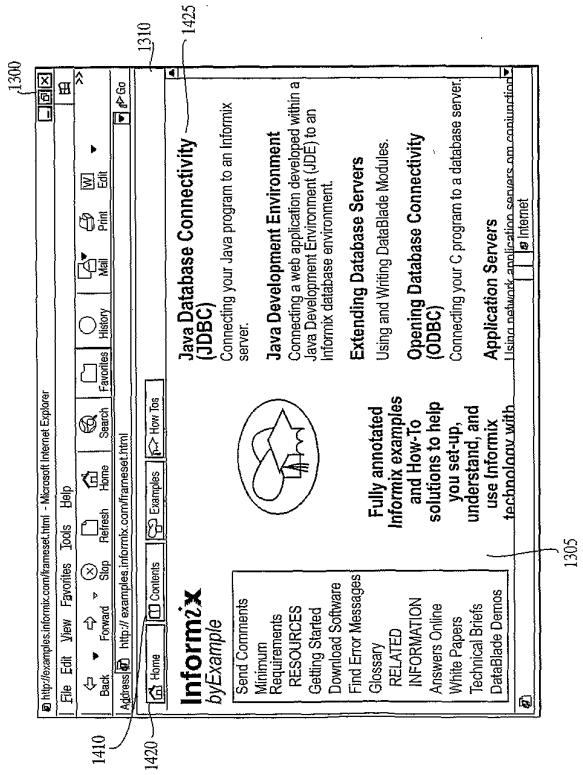

1300
1310
1425
1410
1420
1305
http://examples.informix.com/frameset.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://examples.informix.com/frameset.html
Go
Home
Contents
Examples
How Tos
Informix
byExample
Send Comments
Minimum Requirements
RESOURCES
Getting Started
Download Software
Find Error Messages
Glossary
RELATED INFORMATION
Answers Online
White Papers
Technical Briefs
DataBlade Demos
Java Database Connectivity (JDBC)
Connecting your Java program to an Informix server.
Java Development Environment
Connecting a web application developed within a Java Development Environment (JDE) to an Informix database environment.
Extending Database Servers
Using and Writing DataBlade Modules.
Opening Database Connectivity (ODBC)
Connecting your C program to a database server.
Application Servers
Using network application servers can contribution
Fully annotated Informix examples and How-To solutions to help you set-up, understand, and use Informix technology with
Internet

FIG. 14B

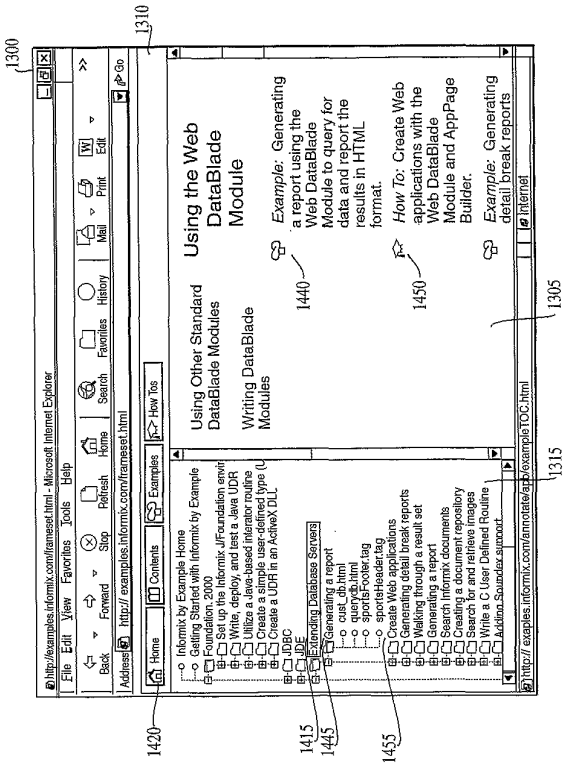

1300
1310
1420
1415
1445
1455
1440
1450
1305
1315
Using the Web DataBlade Module
Using Other Standard DataBlade Modules
Writing DataBlade Modules
Example: Generating a report using the Web DataBlade Module to query for data and report the results in HTML format.
How To: Create Web applications with the Web DataBlade Module and AppPage Builder.
Example: Generating detail break reports

FIG. 14C

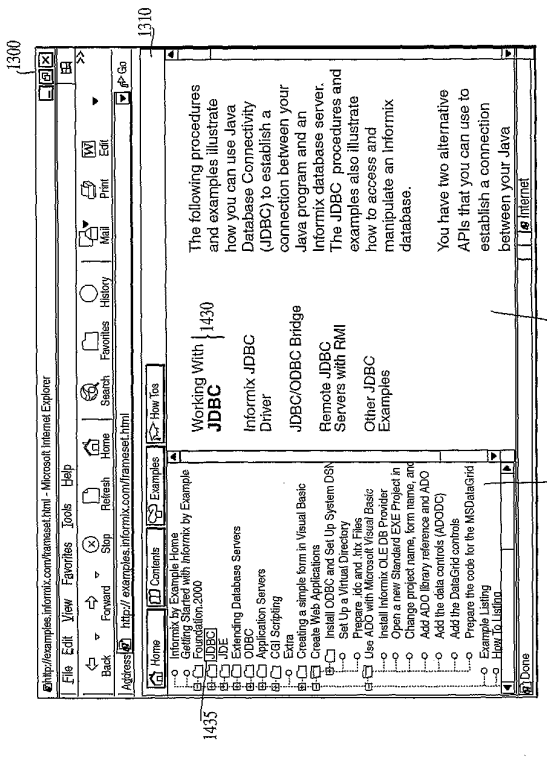

1300
1310
1435
1430
1305
1315
Working With JDBC
Informix JDBC Driver
JDBC/ODBC Bridge
Remote JDBC Servers with RMI
Other JDBC Examples
The following procedures and examples illustrate how you can use Java Database Connectivity (JDBC) to establish a connection between your Java program and an Informix database server. The JDBC procedures and examples also illustrate how to access and manipulate an Informix database.
You have two alternative APIs that you can use to establish a connection between your Java

FIG. 14D

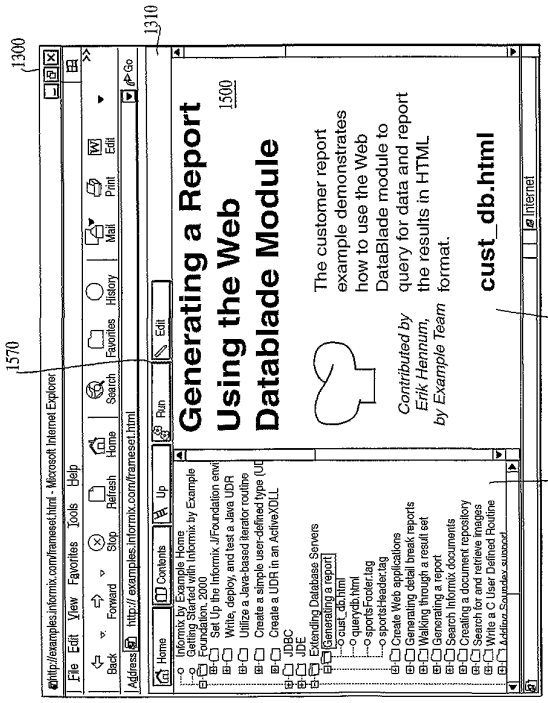

1300
1310
1570
1315
1305
1500
Generating a Report
Using the Web
Datablade Module
The customer report example demonstrates how to use the Web DataBlade module to query for data and report the results in HTML format.
Contributed by Erik Hennum, by Example Team
cust_db.html

FIG. 15A

1500

Generating a Report Using the Web Datablade Module

The customer report example demonstrates how to use the Web DataBlade module to query for data and report the results in HTML format. —1505

Contributed by Erik Hennum, byExample Team

cust_db.html —1515

1520

This app page accepts a query and generates an HTML report

1510 {

querydb.html —1515

1520

This HTML page contains a form that invokes an app page

}

sportsFooter.tag —1515

1520

The sportsFooter dynamic tag generates the footer for an app page.

sportsHeader.tag —1515

1520

The sportsHeader dynamic tag generates the header for an app page.

Click here to view or print all of the source files for this example.

FIG. 15B

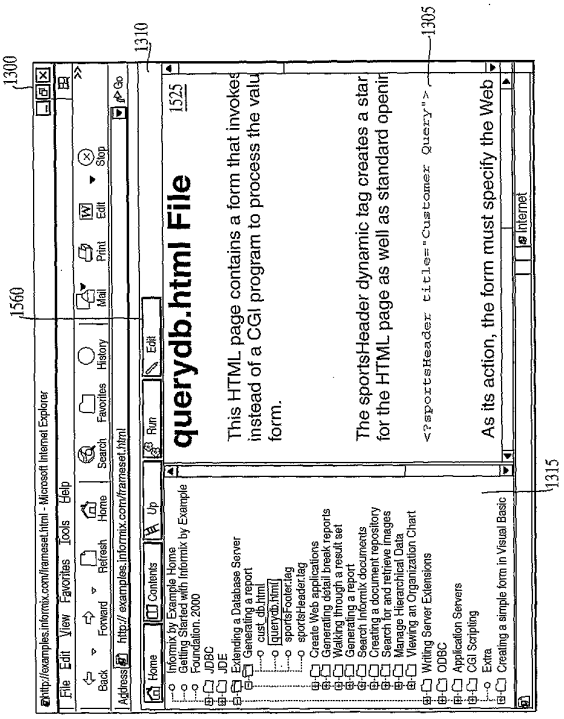

1300
1310
1560
1305
1315
1525
querydb.html File
This HTML page contains a form that invokes
instead of a CGI program to process the valu
form.
The sportsHeader dynamic tag creates a star
for the HTML page as well as standard openir
<?sportsHeader title="Customer Query">
As its action, the form must specify the Web

FIG. 15C

1525

querydb.html File

This HTML page contains a form that invokes an app page instead of a CGI program to process the values in the form.

The sportsHeader dynamic tag creates a standard header for the HTML page as well as standard opening text. } 1535

```
<?sportsHeader title="Customer Query"> } 1530
```

1540

As its action, the form must specify the Web Driver utility.

```
<P>
<FORM ACTION ="<?MIVAR>$WEB_HOME<?/MIVAR"METHOD="GET"> } 1530
```

1550

To specify the app page, the form must use a hidden input component. The input component must have a name of **MIval** and a value that's the name of the app page. The input component below specifies the cust_db.html app page. } 1535

```
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPE="TEXT" NAME="SelectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="SUBMIT" VALUE="SUBMIT">

</FORM>
</P>


</BODY>
</HTML>
```

} 1530

FIG. 15D

1560

```
<!--<ibyx>
<intro>
<p><abstract>This HTML page contains a form that invokes
an app page</abstract> instead of a CGI program to process
the values in the form.
</p>
</intro>
</ibyx> -- >

<!-- <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> -->
<?sportsHeader title="Customer Query">

<!-- <ibyx>
<p> As its action, the form must specify the Web Driver utility.
</p>
</ibyx>-->
<P>
<FORM ACTION="<?MIVAR>$WEB_HOME<?/MIVAR>" METHOD="GET">

<!-- <ibyx>
<p> To specify the app page, the form must use a hidden input component.
The input component must have a name of <strong>MIval</strong> and
a value that's the name of the app page. The input component below
specifies the <a href="cust_db.html">cust_db.html</a> app page.
</p>
</ibyx> -->
<INPUT TYPE="HIDDEN" NAME="MIval" VALUE="/examples/CustRpt/cust_db.html">

Optional state:
<INPUT TYPR="TEXT" NAME="selectState" SIZE="3" MAXLENGTH="2">
<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">

</FORM>
</P>

<?annotate>

</BODY>
</HTML>
```

1535

1535

FIG. 15E

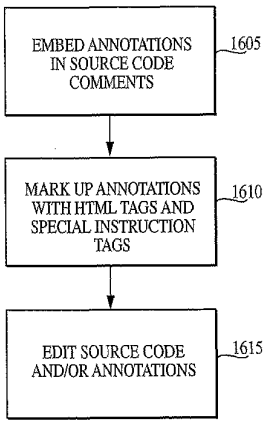

EMBED ANNOTATIONS IN SOURCE CODE COMMENTS
1605
MARK UP ANNOTATIONS WITH HTML TAGS AND SPECIAL INSTRUCTION TAGS
1610
EDIT SOURCE CODE AND/OR ANNOTATIONS
1615

FIG. 16A

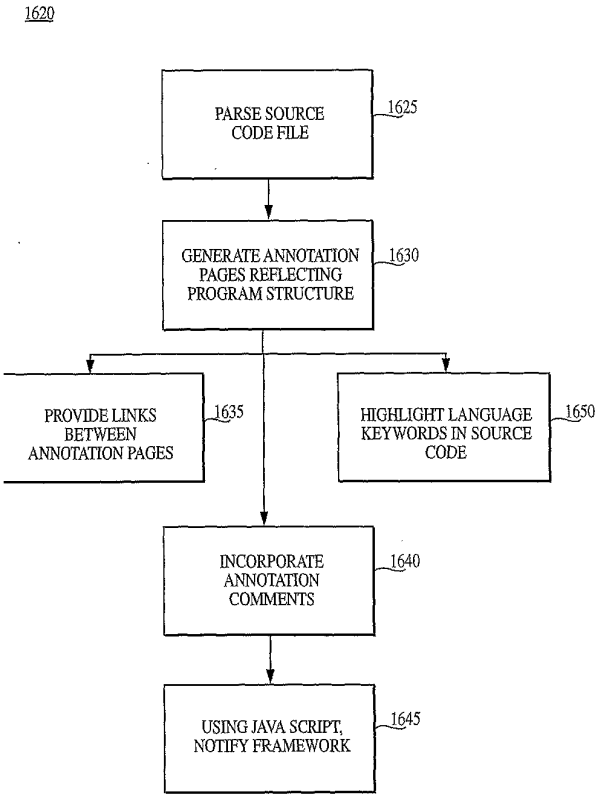

1620
PARSE SOURCE CODE FILE
1625
GENERATE ANNOTATION PAGES REFLECTING PROGRAM STRUCTURE
1630
PROVIDE LINKS BETWEEN ANNOTATION PAGES
1635
HIGHLIGHT LANGUAGE KEYWORDS IN SOURCE CODE
1650
INCORPORATE ANNOTATION COMMENTS
1640
USING JAVA SCRIPT, NOTIFY FRAMEWORK
1645

FIG. 16B

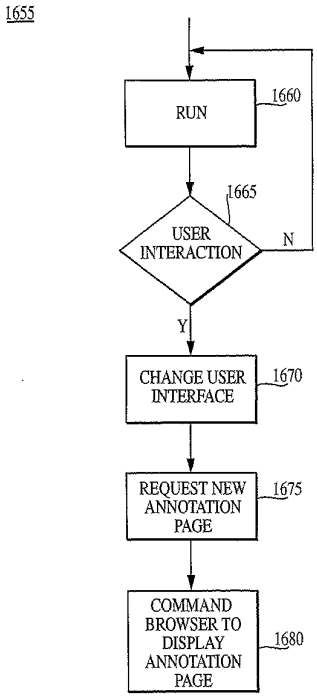

1655
RUN
1660
1665
USER INTERACTION
N
Y
CHANGE USER INTERFACE
1670
REQUEST NEW ANNOTATION PAGE
1675
COMMAND BROWSER TO DISPLAY ANNOTATION PAGE
1680

FIG. 16C

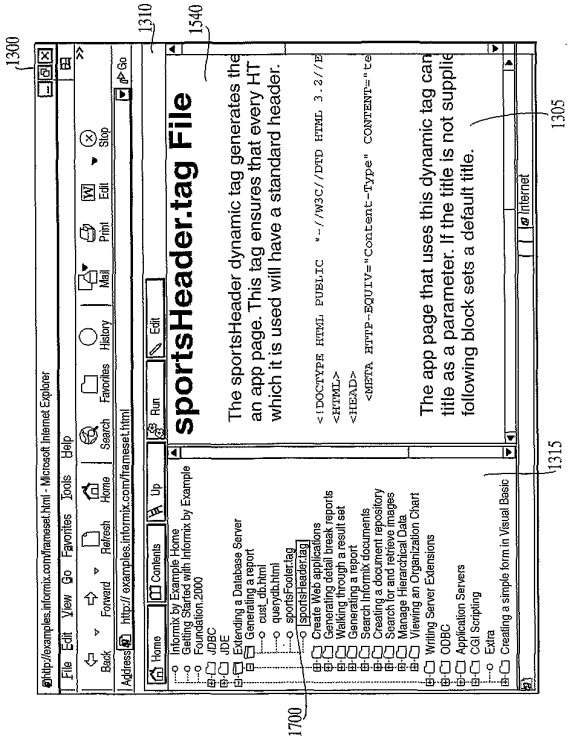

1300
1310
1540
1305
1315
1700
sportsHeader.tag File
The sportsHeader dynamic tag generates the
an app page. This tag ensures that every HT
which it is used will have a standard header.
The app page that uses this dynamic tag can
title as a parameter. If the title is not supplie
following block sets a default title.

FIG. 17A

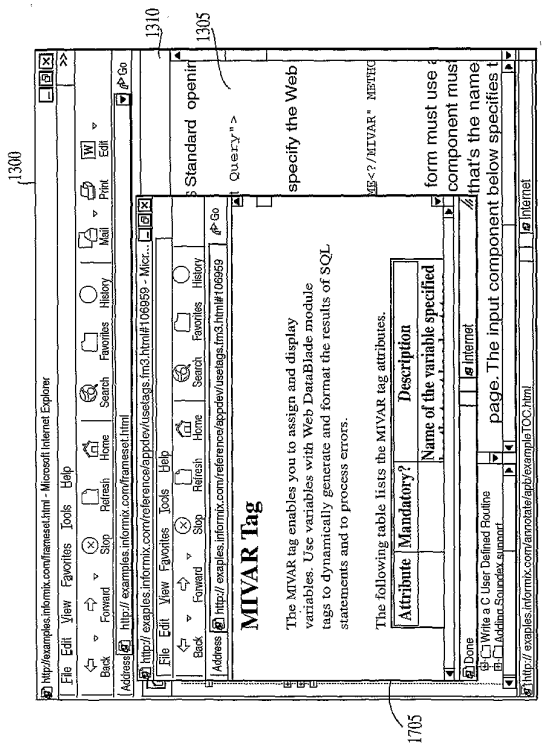

1300
1310
1305
1705
MIVAR Tag
The MIVAR tag enables you to assign and display variables. Use variables with Web DataBlade module tags to dynamically generate and format the results of SQL statements and to process errors.
The following table lists the MIVAR tag attributes.
Attribute
Mandatory?
Description
Name of the variable specified
Standard openir
Query">
specify the Web
form must use
component mus
that's the name
page. The input component below specifies t

FIG. 17B

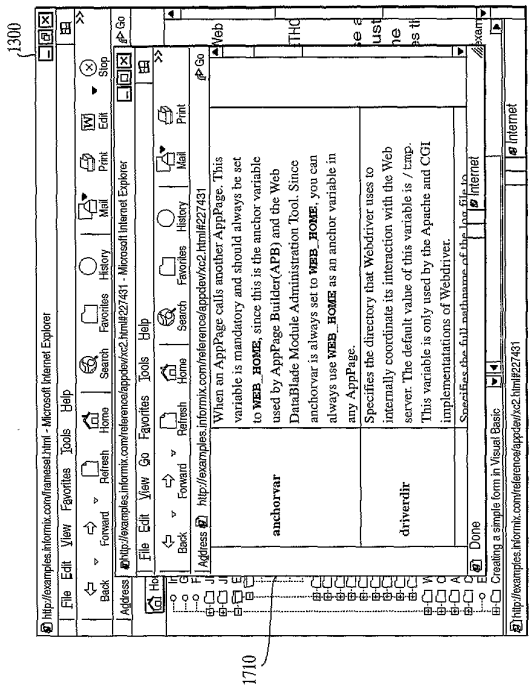

1300
1710
anchorvar
When an AppPage calls another AppPage. This variable is mandatory and should always be set to WEB_HOME, since this is the anchor variable used by AppPage Builder(APB) and the Web DataBlade Module Administration Tool. Since anchorvar is always set to WEB_HOME, you can always use WEB_HOME as an anchor variable in any AppPage.
driverdir
Specifies the directory that Webdriver uses to internally coordinate its interaction with the Web server. The default value of this variable is /tmp. This variable is only used by the Apache and CGI implementations of Webdriver.

FIG. 17C

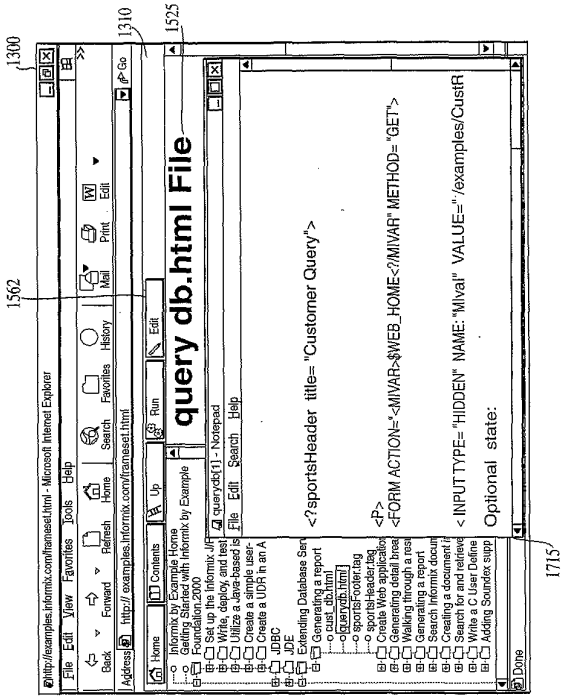

1300
1562
1310
1525
1715
query db.html File
<?sportsHeader title= "Customer Query">
<P>
<FORM ACTION="<MIVAR>$WEB_HOME</?MIVAR" METHOD= "GET">
< INPUT TYPE= "HIDDEN" NAME: "Mival" VALUE=" /examples/CustR
Optional state:

FIG. 17D

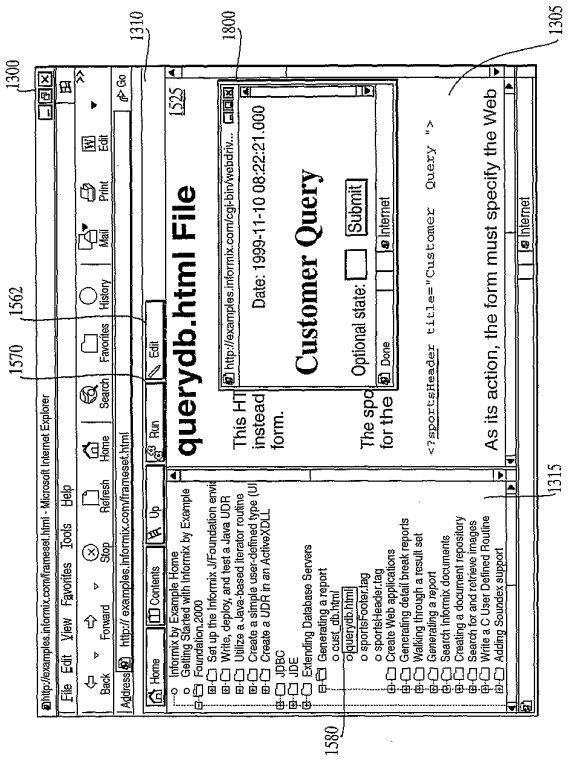

1300
1310
1800
1305
1562
1570
1525
1315
1580
querydb.html File
Customer Query
Optional state:
Submit
Date: 1999-11-10 08:22:21.000
As its action, the form must specify the Web
<?sportsHeader title="Customer Query ">
Informix by Example Home
Getting Started With Informix by Example
Foundation 2000
Set up the Informix J/Foundation envir
Write, deploy, and test a Java UDR
Utilize a Java-based iterator routine
Create a simple user-defined type (UI
Create a UDR in an ActiveXDLL
JDBC
JDE
Extending Database Servers
Generating a report
cust_db.html
querydb.html
sportsFooter.tag
sportsHeader.tag
Create Web applications
Generating detail break reports
Walking through a result set
Generating a report
Search Informix documents
Creating a document repository
Search for and retrieve images
Write a C User Defined Routine
Adding Soundex support

FIG. 18A

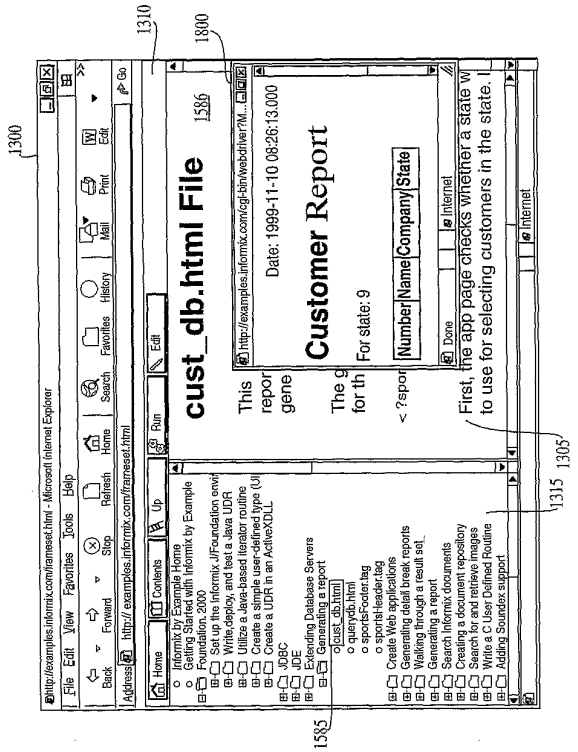

1300
1310
1800
1586
1585
1315
1305
cust_db.html File
Customer Report
Date: 1999-11-10 08:26:13.000
For state: 9
Number
Name
Company
State
Done
Internet

FIG. 18B

```
<! - - <ibyx>
<intro>
<p> <abstract>This app page accepts a query and generates an HTML
report</abstract > in response.
The app page uses dynamic tags to generate the header and footer for the
HTML report.
</p>
</intro>
</ibyx> - - >
<! - - <ibyx>
<p> The sportsHeader dynamic tag creates a standard header
for the HTML page as well as standard opening text.
</p>
</ibyx> - ->
<?sportsHeader title="Customer Report">

<! - - <ibyx>
<p> First, the app page checks whether a state was specified to use for
selecting customers in the state. If so, the block generates a
paragraph to identify the state.
</p>
</ibyx> - - >
<?MIVAR NAME=$WHERE_STR><?/MIVAR>
<?MIBLOCK COND='$(AND,$(XST,$selectState) ,  $ (<,0,$ (STRLEN, $selectState)))">
        <?MIVAR NAME= $WHERE_STR>WHERE state= ' $selectState'<?/MIVAR>
        <?MIVAR><P>For state: $selectState</P><?/MIVAR>
<?/MIBLOCK>

<! - - <ibyx>
<p>Next,  the app page starts the table that will contain the data.
</p>
</ibyx> - ->
        <P><TABLE BORDER="1">
                <TR>
                <TH>Number</TH><TH>Name</TH><TH>Company</TH><TH>State</TH>
                </TR>

<! - - <ibyx>
<p> The MISQL block queries for customers, optionally selecting only customers
from the specified state. Because the contents of the block are generated
for every row of data, a new table row describes each customer.

The &amp ;nbsp; HTML entity is a non-breaking space. By putting a non-breaking
space in each column, we force the Web Browser to display the column even
if the value is null.
</p>
</ibyx> - ->
<?MISQL SQL= 'SELECT customer_num, fname, lname, company, state FROM customer $WHERE_STR; '>
                <TR>
                <TD> $1  </TD><TD>$2  $3</TD><TD>$4 </TD><TD>$5 </TD>
                </TR>
<?/MISQL>

        </TABLE></P>

< ! - -  <ibyx>
<p> The sportsFooter dynamic tag creates a standard footer
for the HTML page.
</p>
</ibyx> - ->
<?sportsFooter>
```

1810

FIG. 18C

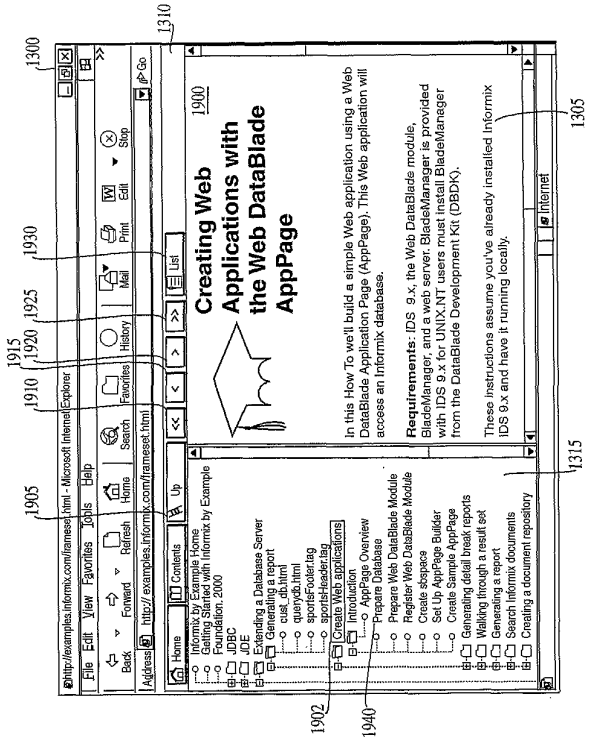

1300
1310
1930
1925
1915
1920
1910
1905
1900
1902
1940
1315
1305
Creating Web Applications with the Web DataBlade AppPage
In this How To we'll build a simple Web application using a Web DataBlade Application Page (AppPage). This Web application will access an Informix database.
Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX.NT users must install BladeManager from the DataBlade Development Kit (DBDK).
These instructions assume you've already installed Informix IDS 9.x and have it running locally.

FIG. 19A

1900

Creating Web Applications with the Web DataBlade AppPage

1930

i this How To we'll build a simple Web application using Web DataBlade Application Page (AppPage). This /eb application will access an Informix database.

lequirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is rovided with IDS 9.x for UNIX. NT users must install BladeManager from the DataBlade Development Kit)BDK)

hese instructions assume you've already installed Informix IDS 9.x and have it running locally.

1935

Define a server connection and prepare a sample database.

1. Define a server connection with setnet32 (NT). Create a sample database or use the stores7 demo database. ▷ Prepare Database.

Prepare the Web DataBlade development environment.

2. Install the Web DataBlade module and BladeManager. ▷ Prepare Web DataBlade Development Environment.

3. Register the Web DataBlade module in the demo database with BladeManager. ▷ Register the Web DataBlade.

4. Create a sbspace for smart large objects, like gifs. ▷ Create Smart Blob Space (sbspace).

5. Install AppPage Builder in your database. ▷ Install AppPage Builder in Your Database.

6. Setup AppPage Builder on your web server. ▷ Setup AppPage Builder on Your Web Server.

7. Create a sample AppPage. ▷ Create Sample AppPage.

Run the sample application.

8. Enter the URL `http:// your_server/scripts/webdriver.exe`.

This How To has been compiled into two separate files for ease of printing. The basic file contains all of the steps you need to Create Web Applications with AppPage Builder. The secondary file contains additional detailed instructions for setting and testing database environment properties.

FIG. 19B

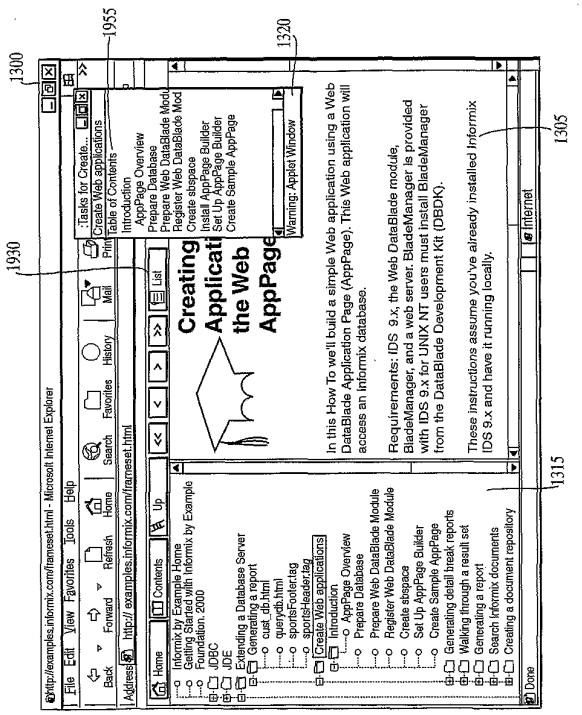

1300
1955
1930
1320
1305
1315
Creating
Applicati
the Web
AppPage
Warning: Applet Window
In this How To we'll build a simple Web application using a Web DataBlade Application Page (AppPage). This Web application will access an Informix database.
Requirements: IDS 9.x, the Web DataBlade module, BladeManager, and a web server. BladeManager is provided with IDS 9.x for UNIX NT users must install BladeManager from the DataBlade Development Kit (DBDK).
These instructions assume you've already installed Informix IDS 9.x and have it running locally.

FIG. 19C

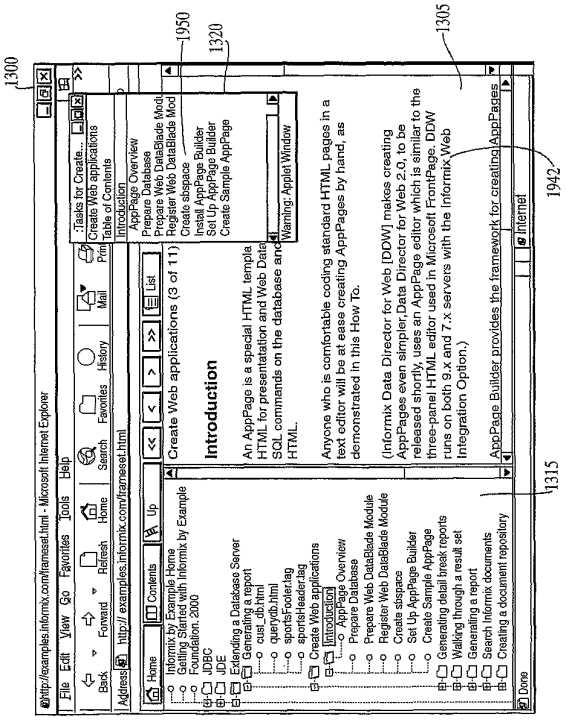

1300
1950
1320
1305
1942
1315
Create Web applications (3 of 11)
Introduction
Warning: Applet Window
Done
Internet

FIG. 19D

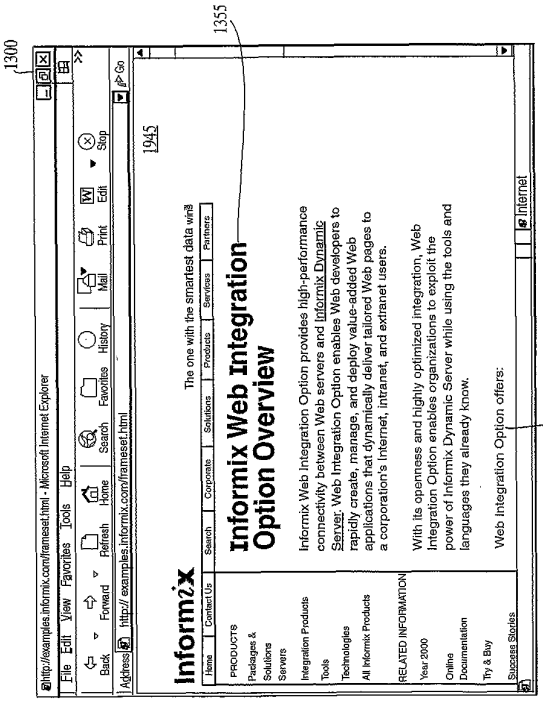

1300
1355
1945
1305
http://examples.informix.com/frameset.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Refresh Home Search Favorites History Mail Print Edit Stop
Address http://examples.informix.com/frameset.html
Go
Internet
Informix
The one with the smartest data wins
Home Contact Us Search Corporate Solutions Products Services Partners
PRODUCTS
Packages & Solutions
Servers
Integration Products
Tools
Technologies
All Informix Products
RELATED INFORMATION
Year 2000
Online Documentation
Try & Buy
Success Stories
Informix Web Integration Option Overview
Informix Web Integration Option provides high-performance connectivity between Web servers and Informix Dynamic Server. Web Integration Option enables Web developers to rapidly create, manage, and deploy value-added Web applications that dynamically deliver tailored Web pages to a corporation's Internet, intranet, and extranet users.
With its openness and highly optimized integration, Web Integration Option enables organizations to exploit the power of Informix Dynamic Server while using the tools and languages they already know.
Web Integration Option offers:

FIG. 19E

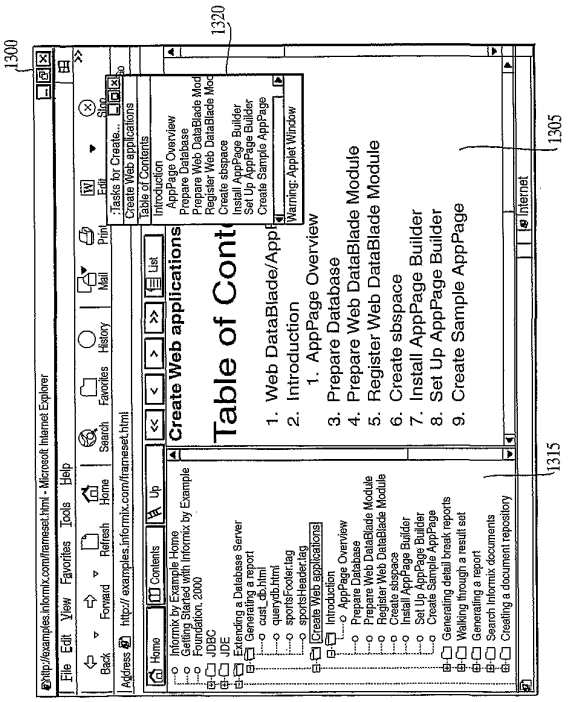

1300
1320
1315
1305
Table of Cont
1. Web DataBlade/Appl
2. Introduction
1. AppPage Overview
3. Prepare Database
4. Prepare Web DataBlade Module
5. Register Web DataBlade Module
6. Create sbspace
7. Install AppPage Builder
8. Set Up AppPage Builder
9. Create Sample AppPage

FIG. 19F

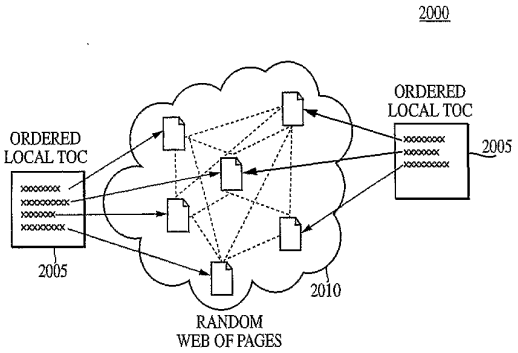

2000
ORDERED
LOCAL TOC
2005
ORDERED
LOCAL TOC
2005
2010
RANDOM
WEB OF PAGES

FIG. 20

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/045069 A3

(51) International Patent Classification⁷: G06F 9/44

(21) International Application Number: PCT/US00/34013

(22) International Filing Date:
15 December 2000 (15.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/172,134 17 December 1999 (17.12.1999) US
09/728,073 4 December 2000 (04.12.2000) US

(71) Applicant: INFORMIX SOFTWARE, INC. [US/US]; 4100 Bohannon Drive, Menlo Park, CA 94025 (US).

(72) Inventor: HENNUM, Erik; 78 St. Mary's Avenue, San Francisco, CA 94112 (US).

(74) Agents: DIBERARDINO, Diana; Fish & Richardson P.C., 601 Thirteenth Street N.W., Washington, DC 20005 et al. (US).

(81) Designated States *(national)*: AU, BR, CA, JP, MX.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*

(88) Date of publication of the international search report:
11 July 2002

(15) Information about Corrections:
Previous Corrections:
see PCT Gazette No. 21/2002 of 23 May 2002, Section II
see PCT Gazette No. 48/2001 of 29 November 2001, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/045069 A3

(54) Title: WEB-BASED INSTRUCTION

(57) Abstract: A method performed in a web-based environment on a computer system teaches a user to implement an application. The method includes providing predetermined applications and presenting an annotation page that includes one or more annotations descriptive of a predetermined application. Each annotation includes keyword links, annotation links, and detail of implementation of the application. The method includes permitting the user to select a link in an annotation. If the user selects a keyword link, reference documentation associated with that keyword is presented. If the user selects an annotation link, another annotation descriptive of another source file of a predetermined application is presented.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 00/34013

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PRIESTLEY M: "TASK ORIENTED OR TASK DISORIENTED: DESIGNING A USABLE HELP WEB" THE 16TH ANNUAL INTERNATIONAL CONFERENCE OF COMPUTER DOCUMENTATION. CONFERENCE PROCEEDINGS. SIGDOC '98 SCALING THE HEIGHTS: THE FUTURE OF INFORMATION TECHNOLOGY. QUEBEC CITY, SEPT. 23 - 26, 1998, ANNUAL INTERNATIONAL CONFERENCE OF COMPUTER DOCUMENTAT, 23 September 1998 (1998-09-23), pages 194-199, XP000869304 ISBN: 1-58113-004-X the whole document --- -/--	1-45

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 March 2002	28/03/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Brandt, J

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 00/34013

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	RINTJEMA L ET AL: "CREATING AN HTML HELP SYSTEM FOR WEB-BASED PRODUCTS" THE 16TH ANNUAL INTERNATIONAL CONFERENCE OF COMPUTER DOCUMENTATION. CONFERENCE PROCEEDINGS. SIGDOC '98 SCALING THE HEIGHTS: THE FUTURE OF INFORMATION TECHNOLOGY. QUEBEC CITY, SEPT. 23 - 26, 1998, ANNUAL INTERNATIONAL CONFERENCE OF COMPUTER DOCUMENTAT, 23 September 1998 (1998-09-23), pages 227-233, XP000869306 ISBN: 1-58113-004-X the whole document	1-45
A	US 5 877 757 A (GAITATZES ATHANASIOS GEORGE ET AL) 2 March 1999 (1999-03-02) column 4, line 10 - line 42	1-45
A	EP 0 509 947 A (IBM) 21 October 1992 (1992-10-21) column 2, line 35 -column 3, line 13	1-45

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

ormation on patent family members

Interr nal Application No
PCT/US 00/34013

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 5877757	A	02-03-1999	NONE		
EP 0509947	A	21-10-1992	CA	2060983 A1	19-10-1992
			EP	0509947 A2	21-10-1992
			JP	4326135 A	16-11-1992

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(72)発明者　ヘナム、エリック
アメリカ合衆国９４１１２　カリフォルニア州サンフランシスコ　セイント・マリーズ・アベニュー　７８

Ｆターム(参考) 5B085 BA06 BE07 BG07 CE02 CE03 CE06 CE07 CE09
5E501 AB15 BA05 CA03 CB09 CC17 EA12 FA13 FA22 FA42 FB43