



(12) 发明专利

(10) 授权公告号 CN 1655120 B

(45) 授权公告日 2010.05.26

(21) 申请号 200510008151.2

W0 01/25909 A2, 2001.04.12, 全文.

(22) 申请日 2005.02.06

US 5987247 A, 1999.11.16, 全文.

(30) 优先权数据

CN 1059981 A, 1992.04.01, 权利要求 1 -

10/779, 298 2004.02.13 US

17.

(73) 专利权人 微软公司

审查员 王学睿

地址 美国华盛顿州

(72) 发明人 R·奇特萨兹 S·罗克什

T·E·奎恩

(74) 专利代理机构 上海专利商标事务所有限公

司 31100

代理人 张政权

(51) Int. Cl.

G06F 9/44 (2006.01)

(56) 对比文件

CN 1349626 A, 2002.05.15, 全文.

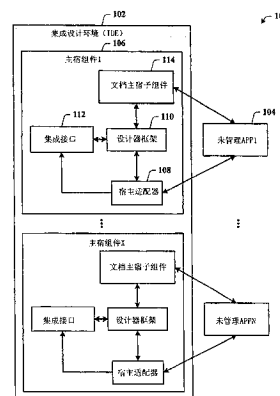
权利要求书 2 页 说明书 10 页 附图 7 页

(54) 发明名称

在集成开发环境中作为设计器主宿应用程序的方法和系统

(57) 摘要

一种将未管理应用程序作为主宿的应用程序按一致且仍可用的方式集成到设计环境中的体系结构,使得开发人员可以使用该主宿的应用程序的原有功能,同时调节由该 IDE 提供的丰富的生产效率的特征。该 IDE 包括一个主宿组件,它包括一应用程序适配器,用于提供从该未管理应用程序到该 IDE 的接口。该 IDE 还包括一设计器框架,它向用户提供对该 IDE 的设计器能力的访问。该宿主适配器直接地和 / 或通过集成接口与设计器框架接口。由此,开发人员 / 用户然后能够访问未管理应用程序的原有功能以及 IDE 的功能。这就提供了开发环境和主宿的的应用程序之间的丰富的集成性,其最后的结果是该设计器与该 IDE 交互。



1. 一种方便集成开发环境 IDE 的系统,其特征在于,包括:
 - 一未管理的应用程序,它便于处理文档;以及
 - 一宿主组件,它与所述未管理应用程序接口,使得所述未管理应用程序担当所述 IDE 中的设计器,所述宿主组件还包括:
 - 一集成接口,以便将第三方未管理应用程序作为设计器集成到所述开发环境中;
 - 一宿主适配器,它与所述未管理应用程序接口,所述适配器是应用程序专用的,以便于将所述未管理应用程序集成到所述开发环境中;以及
 - 一文档宿主子组件,它便于宿主能够在所述开发环境中操纵的文档。
2. 如权利要求 1 所述的系统,其特征在于,所述宿主组件支持对象链接和嵌入 OLE 技术。
3. 如权利要求 1 所述的系统,其特征在于,所述未管理的应用程序包括文字处理应用程序和电子表格应用程序中的至少一个。
4. 如权利要求 1 所述的系统,其特征在于,所述文档能够使用所述未管理的应用程序和所述开发环境的至少一个来操纵。
5. 如权利要求 1 所述的系统,其特征在于,所述文档是新文档和已存在文档之一。
6. 如权利要求 1 所述的系统,其特征在于,它便于宿主多个不同的未管理应用程序。
7. 如权利要求 1 所述的系统,其特征在于,它还包括一托盘组件,它便于在所述 IDE 中呈现非可视数据。
8. 如权利要求 1 所述的系统,其特征在于,所述宿主组件还包括一设计器框架,它便于将所述宿主组件与所述开发环境接口。
9. 如权利要求 1 所述的系统,其特征在于,所述未管理应用程序的原有功能在所述 IDE 中是可用的。
10. 如权利要求 1 所述的系统,其特征在于,它便于向所述文档添加控件和编辑所述控件的属性。
11. 如权利要求 1 所述的系统,其特征在于,它便于将所述未管理应用程序和所述 IDE 的菜单的合并。
12. 如权利要求 1 所述的系统,其特征在于,所述文档包含嵌入和链接的对象的至少一个。
13. 如权利要求 1 所述的系统,其特征在于,它便于呈现一显示代码在后工程项目的属性的属性窗口。
14. 如权利要求 1 所述的系统,其特征在于,所述设计器在一设计视图和一代码视图中操作。
15. 如权利要求 1 所述的系统,其特征在于,所述设计器能够以二进制格式和 XML 格式中的至少一个来处理所述文档。
16. 一种采用权利要求 1 所述的系统的计算机。
17. 一种将未管理应用程序集成到开发环境中的方法,其特征在于,包括:
 - 接收所述未管理应用程序;
 - 用一宿主适配器将所述未管理应用程序与所述开发环境接口,所述适配器专用于所述未管理应用程序,使得所述未管理应用程序可以作为所述开发环境中的设计器来访问;以

及

在所述开发环境中主宿一文档,使得所述开发环境和所述未管理应用程序的功能可用于操纵所述文档。

18. 如权利要求 17 所述的方法,其特征在于,它还包括通过以下的至少一个来操纵所述文档:

执行将一控件拖放到所述文档上的操作;
察看与所述文档相关联的数据在后文件;
在所述数据在后文件中写入已管理代码;
以设计模式和运行时模式操作所述设计器;
呈现非可视数据;以及
将已管理对象绘制到所述文档上。

19. 如权利要求 17 所述的方法,其特征在于,它还包括为所述未管理应用程序提供一专用偏好页面,使得当被集成到所述开发环境中时,所述未管理应用程序根据所述偏好而表现。

20. 如权利要求 17 所述的方法,其特征在于,它还包括展示一与所述文档相关联的代码在后文件,使得所述文件的内容能够被操纵。

21. 如权利要求 17 所述的方法,其特征在于,它还包括展示一与所述文档的每一子文档相关联的代码在后文件,使得每一文件的内容能够被操纵。

22. 如权利要求 17 所述的方法,其特征在于,它还包括提供一视图控件,它响应于一事件而触发。

23. 一种便于将一未管理应用程序集成到一开发环境中的系统,其特征在于,包括:
用于接收所述未管理应用程序的装置;

用于将所述未管理应用程序与所述开发环境接口的装置,所述开发环境具有一宿主适配器,所述适配器专用于所述未管理应用程序,使得所述未管理应用程序可以作为所述开发环境中的设计器来访问。

用于在所述开发环境中主宿一文档的装置,使得所述开发环境和所述未管理应用程序的功能可用于操纵所述文档;以及

用于操纵所述文档的装置。

24. 如权利要求 23 所述的系统,其特征在于,所述用于操纵的装置还包括用于以下的至少一个的操纵装置:

实行将控件拖放到所述文档上的操作;
察看与所述文档相关联的数据在后文件;
在所述数据在后文件中写入已管理代码;
以设计模式和运行时模式操作所述设计器;
呈现非可视数据;以及
在所述文档上绘制已管理的对象。

25. 如权利要求 23 所述的系统,其特征在于,它还包括用于展示所述文档的属性和组件以供操纵的装置。

在集成开发环境中作为设计器宿主应用程序的方法和系统

技术领域

[0001] 本发明涉及一种软件开发环境,尤其涉及便于把未管理的应用程序带入到环境中作为设计器的体系结构。

背景技术

[0002] 迄今为止,软件集成开发环境 (IDEs) 并不能将未管理的主宿的 (hosted) 应用程序的功能集成到该开发工作空间中。这样,编写丰富的设计器通常是非常复杂和费时的任务,这就必须考虑非常紧密地与该宿主的应用程序的集成。

[0003] 需要的是一种体系结构,它把未管理的应用程序按一致仍而可使用的方式集成到设计环境中,使得开发者能使用该宿主的应用程序的原有的功能并同时调节由该 IDE 提供的充足的生产效率的特征。

发明内容

[0004] 以下给出了本发明的简化概述,以提供对本发明某些概念的基本理解。这个概述并不是本发明的广泛综述。这并不旨在标识本发明的关键 / 决定性元素或者描述本发明的范围。其唯一的目的是以简化的形式给出本发明某些概念,作为以后给出的更详细描述的前言。

[0005] 此处揭示并要求保护的本发明,在其一个方面中,包括一种体系结构,它把未管理的应用程序作为宿主的应用程序按一致而可使用的方式集成到设计环境中,使得该开发者能使用该宿主的应用程序的原有的功能,同时调节 (leverage) 由该 IDE 提供的充足的生产效率的特征。在此作为支持,提供了一种具有宿主组件的 IDE,该组件包括一应用程序适配程序,它提供从未管理应用程序到该 IDE 的接口。该 IDE 包括提供对 IDE 的设计器能力的用户访问的设计器框架。宿主适配程序直接地和 / 或通过一集成接口连接到设计器框架。由此,开发者 / 用户然后能够访问未管理应用程序的原有功能以及该 IDE 功能。这就提供了开发环境与该宿主的应用程序之间的丰富的集成,其结果是设计器与该 IDE 交互。

[0006] 为了实现前述的相关目的,此处结合以下描述和附图描述了本发明的某些说明性方面。然而,这些方面仅指示了可采用本发明的原理的各种方式中的几种,并且本发明旨在包括所有这样的方面和其等效方面。当结合附图考虑时,本发明的其它优点和新颖特征将从以下本发明的详细描述中变得显而易见。

附图说明

[0007] 图 1 示出本发明的一个系统。

[0008] 图 2 示出在 IDE 中集成原有的应用程序功能的过程的流程图。

[0009] 图 3 示出依照本发明的 OLE 文档开发过程的流程图。

[0010] 图 4 所示是依照本发明的集成功能的一个示例窗口。

[0011] 图 5 所示是本发明的集成且层叠的菜单特征的窗口。

[0012] 图 6 示出了可用于执行所揭示的体系结构的计算机的框图。

[0013] 图 7 示出根据本发明的示例性的计算环境的示意性框图。

具体实施方式

[0014] 现在,本发明将参照附图来描述,贯穿附图,相同的标号用于表示相同的元素。在以下的论述中,以解释为目的,阐明了若干具体细节,以提供对本发明的彻底的理解。然而,很明显,本发明可以不采用这些具体细节来实践。在其它的例子中,以框图的形式示出了众所周知的结构和设备,以便于描述本发明。

[0015] 如本申请中所使用的,术语“组件”和“系统”意在指计算机相关的实体,它可以是硬件、硬件和软件的组合、或执行中的软件。例如,组件可以是,但不局限于,在处理器上运行的进程、处理器、对象、可执行码、执行线程、程序和 / 或计算机。作为说明,运行在服务器上的应用程序和该服务器都能够是组件。一个或多个组件可驻留在进程和 / 或执行线程之中,而且组件可定位在一个计算机上和 / 或分布在两个或多个计算机之间。

[0016] 现在参照图 1,示出了本发明的一个系统 100。系统 100 是一个增强的“集成”开发环境 (IDE) 102,它便于把未管理应用程序 104 的原有功能带入到开发环境 102 之中。此外,IDE 102 包括一宿主组件 106,它至少是通过宿主适配器 108 连接到未管理应用程序 104。宿主适配器 108 是宿主组件 102 的一部分,并且对特定的未管理应用程序 104 是专用的。即,当未管理应用程序 104 是文字处理程序时,宿主适配器提供对文字处理应用程序专用的接口能力。类似地,电子表格应用程序将具有它自己的专用宿主适配器,以便于把它的原有的功能带入 IDE 102 之中。

[0017] 宿主组件 106 还包括设计器框架 110,它与宿主适配器 108 接口,并担当宿主组件 106 和 IDE 102 之间的接口。宿主适配器 108 还通过一集成接口 112 与设计器框架接口,接口 112 担当用于第三方应用程序的接口,如果有对该第三方应用程序的宿主适配器 108,其原有的功能也能被带入到 IDE 102 中。

[0018] 宿主组件 106 还包括一文档宿主子组件 114,它与设计器框架 110 和未管理应用程序接口,以便于宿主应用程序 104 的一个相关联的文档。

[0019] 如上所述,系统 100 能够集成多个未管理应用程序 (称作为“未管理 App₁” … “未管理 App_N”) 的原有功能。由此,IDE 102 包括对应的多个宿主组件 (称作为“宿主组件₁” … “宿主组件_x”)。每一个宿主组件包括宿主适配器、设计器框架、集成接口和文档宿主子组件。注意,虽然这些组件被表示成分离的实体,然而这些实体能以被认为适合于把未管理应用程序的原有功能带入到 IDE 中的任何方式来组合。

[0020] 系统 100 尤其适用于 OLE (对象链接和嵌入) 技术,这里该 IDE 支持 OLE,并且由文档宿主子组件 114 主宿的文档是一 OLE 文档。这样,任何支持 OLE 文档接口的应用程序能够被集成到 IDE 102 之中。然而,本发明并非局限于这样一个 OLE 技术,并构想任何其它合适的技术可用于该 IDE 中。

[0021] 在 IDE 内部主宿基于文档的丰富的应用程序 (如文字处理程序和电子表格应用程序) 之后的创新是这些应用程序被作为可视的设计器来对待。主宿的应用程序是功能完整的,但利用了由 IDE 提供的丰富的功能。当在 IDE 中主宿时,主宿文档能够通过不同的工具窗口的交互被可视化地设计。例如,“工具箱”可以将控件和组件拖放到设计器上。在设

计器中可以选中文档上的控件,并且它们的属性可以在“属性”窗口中编辑。另一操作例如通过一“数据源拾取工具”窗口在应用程序中实现数据绑定功能。

[0022] 本发明将主宿应用程序转换成为可视设计器,而不会减少由该主宿应用程序提供的原有功能的任一个。这就有效地把主宿应用程序转换成为设计表面,开发人员能够在其上拖放对象以快速地创建解决方案和应用程序。而且,所揭示的方法提供了一个熟悉且一致的开发模型以开发该 IDE 中的应用程序。另外,本发明的增强的 IDE 将概念的数量保持为最小,而同时最小化设计范例的数量。

[0023] 现在参照图 2,示出了在 IDE 中集成原有应用程序功能的过程的流程图。尽管为了简化解释的目的,此处例如以流程图的方式示出的一个或多个方法被表示和描述为一系列的动作,然而应该理解和认识到,本发明并不限制于这些动作的次序,因为根据本发明,某些动作可以不同的次序和 / 或与在此示出并描述其它动作并发地发生。例如,本领域的技术人员应理解和认识到,方法可以被替换地表示为一系列内部相关的状态和事件,如以状态图表示。此外,并非所有示出的动作都被要求来实现根据本发明的方法。

[0024] 在 200,提供一 IDE。在 202,在 IDE 中提供主宿组件,以便于主宿未管理应用程序。在 204,设计器框架如此被提供并接口到主宿组件。在 206,提供对未管理应用程序专用的宿主适配器作为主宿组件的一部分。在 208,宿主适配器直接并且间接地通过用于第三方应用程序的集成接口来与设计器框架接口。在 210,文档主宿子组件被提供并与设计器框架和未管理应用程序接口。在 212,主宿文档现在已能使用设计器功能和未管理应用程序的原有功能二者来进行操纵。然后该过程达到“停止”块。

[0025] 现在参照图 3,示出了根据本发明用于 OLE 文档开发的过程的流程图。在这个具体过程实现之中,IDE 环境支持 OLE 技术。在 300,开发人员打开 IDE 环境。在 302,未管理应用程序被选中。在 304,打开一个新的文档或打开一个已存在的文档用于编辑。在 306,能够为该文档设置安全性设定,以便例如,根据文档将在何处被处理来设定各种准许或限制级别。在 308,开发人员 / 用户利用设计器框架和主宿程序的原有功能之一或二者的能力来操纵该文档。然后该处理达到“停止”块。

[0026] 宿主适配器

[0027] 当主宿应用程序作为设计器被打开时,该主宿菜单将与 IDE 的菜单合并。根据该主宿应用程序是否被聚焦到,可启用某些菜单项,而其它的将被禁用。即,如果该 IDE 被聚焦,则在主宿应用程序中禁用某些菜单项。相反,如果主宿应用程序被聚焦,则禁用 IDE 的某些菜单项。这是,但也仅仅是宿主适配器的一个功能。

[0028] 现在参照图 4,有一示例窗口 400,它示出根据本发明的集成功能。窗口 400 是主宿的电子表格应用程序(例如,与 EXCEL 品牌的电子表格应用程序有关)的表示。窗口 400 示出了与 IDE 的 IDE 菜单 404 合并的主宿菜单 402、带有附加的“View Code(察看代码)”命令 408 的宿主上下文菜单 406、示出项目结构 412 的“Solution Explorer(解决方案浏览器)”410、示出所选中的电子表格的属性 416 的“Property(属性)”窗口 414、以及具有能添加到主宿文档 422 中的控件和组件 420 的列表的“Toolbox(工具箱)”418。

[0029] 该“Property(属性)”窗口显示代码在后(code-behind)工程项目和其它设计器工程项目的属性。主宿应用程序将其属性推入“Property”窗口中。主宿应用程序能够选择向“Property”窗口传播属性。主宿应用程序提供属性列表(例如,属性名和属性类型)。

[0030] 包含嵌入或链接的对象的任何文档能够在 IDE 中打开。选中一个嵌入或链接的对象将促使菜单和工具箱对所选中的对象改变。

[0031] 如果选中“Tools(工具)”选项中的第一个单选按钮,则适配器捕获所有的加速键(accelerator)并将加速键传递到动态创建的 IDE 中的快捷键表(或键映射表)。任何没有被处理的事情由该 IDE 表向下传递回主宿应用程序以便处理。

[0032] 在设计器上右键单击将调出标准的电子表格上下文菜单。附加的菜单命令“View Code”是该上下文菜单中的第一个命令。令该“View Code”命令处于列表的顶端提供了与其它程序的上下文菜单的一致性。从电子表格单元格中选择该“View Code”命令将开发者导航到该工作表的代码编辑器。该光标将被设置到在之前它被设置到的地方。默认地,这意味着光标将被设置到编辑器的顶部。在“ViewControl(察看控件)”上右键单击将显示该“View Control”的上下文菜单。从“ViewControl”中选择“View Code”命令将开发者导航到该 IDE 的代码编辑器,并将光标设置到之前它被设置到的地方。

[0033] 在工作表格的选项卡上右键单击将调出电子表格应用程序的一个上下文菜单。该上下文菜单包含一个“View Code”命令,如果被选中,它将用户导航到 IDE 代码编辑器中该电子表格文件后的代码。代码编辑器中与对于“View Code”命令的光标放置行为与它在设计器中对于“View Code”命令是同样的。

[0034] 现在参照图 5,示出了窗口 500,它示出本发明的一个集成且层叠的特征。主宿应用程序(也称为服务器应用程序)与 IDE 的集成会导致相似地命名的菜单项之间的冲突。这里,IDE 和主宿应用程序的集成引起菜单冲突,使得至少“View(察看)”菜单项是冲突的。IDE 修改其已存在的“View”菜单 502,以包括主宿应用程序菜单 504(可通过“Server View Commands(服务器察看命令)”来访问并标记为“Server View Commands”)。该 IDE 向用户呈现了清楚地标识服务器“View”菜单选项在 IDE “View”菜单 502 中的位置。与该“Server View Commands”菜单选项相关联的扩充箭头便于访问该层叠的服务器子菜单项目。

[0035] 示例场景

[0036] 在 IDE 中,用户创建一新的电子表格工程。与电子表格相关的工程项目在该“Solution Explorer(解决方案浏览器)”和电子表格工作簿中创建,并且工作表作为设计器被显示在文档窗口中。该电子表格文档在 IDE 内部设计。所揭示的体系结构将该 IDE 的菜单和主宿应用程序合并,并且工具栏在设计器的电子表格应用程序中可用,从而在一个环境之中提供了生产效率特征和 IDE 特征的全部。该电子表格能够使用电子表格应用程序的全部逼真度来编辑和修改,并采用该 IDE 工具窗口和编辑器以利用已管理代码来定制电子表格。从 Toolbox(工具箱)中,用户能将一个已管理按钮拖放到该文档上。该按钮能够被选中并且其属性可以采用属性网格来修改。定制可以在自动生成的事件处理器的内部通过在按钮上双击并输入“Code View”来作出。

[0037] 设计器模式

[0038] 主宿应用程序以设计模式和运行时模式操作,以便在主宿应用程序内的设计动作和运行时动作之间区分。在每个模式中用户可以执行的动作集可能在各个应用程序之间不同,并依赖于主宿应用程序,而与是否在 IDE 中无关。在一个实施例中,主宿应用程序是设计时模式,并当主宿在 IDE 中,保持在设计时模式。这个体验便于拖动控件到设计器上、选

择该控件以察看和编辑其属性、并双击该控件以进入代码视图。当在 IDE 中时,用于未管理应用程序的宿主适配器将文档推入并保持在设计模式中。

[0039] 另一个模式是“运行”模式。当该应用程序正在运行或正被调试时,它就处于“运行”模式之中。在这个模式中,只有原有的应用程序功能是可用的。一旦调试开始,代码编辑器成为活动的设计器,而且所有的加速键都由 IDE 处理。

[0040] 文档视图

[0041] 由 IDE 主宿的文档至少能具有二种视图:设计视图和代码视图。一般而言,设计视图是应用程序的设计/普通视图。设计视图对支持“OLE 文档”技术的文档启用。当工作在这些文档上时,用户的体验基本上类似于使用原始的主宿应用程序的体验。这意味着,开发人员能够设计模板/文档、建立和修改文档、以及使用专用的宿主的特征,如格式化、图表化和模式映射等等。这种行为通过调节 OLE 技术和在 IDE 中主宿所主宿的应用程序来实现。

[0042] 设计视图能以主宿应用程序所支持的格式来显示新的和已存在的文档。例如,设计器将以二进制和 XML 格式来处理文档。这是通过调整 OLE 文档技术来实现的,以将文档完全集成到 IDE 中,从而取用了各种操作,诸如将控件拖放到设计器上、选择上下文、和在控件上双击以写入代码。设计器能够显示整个文档的内容,包括但并不局限于格式化、样式、模式映射和嵌入对象等。

[0043] 当在 IDE 中时,主宿的应用程序处于“设计模式”。IDE 中主宿的应用程序是全功能的,即,当在 IDE 中开发解决方案时,主宿的应用程序的菜单、命令、上下文菜单、和加速键对用户全部可用。同时,存在一些附加到宿主的原有功能中的特征和体验。如在此所指出的,例子包括与 IDE 工具窗口的交互、在设计器(文档)上右键单击和选择“Code View”命令以导航到文件之后的代码的能力、选择功能键以开始调试等等。这些特征显著地改进了 IDE 内部的开发人员体验,并在设计应用程序时帮助开发人员。

[0044] 有时存在用户在主宿应用程序中的体验不同于在 IDE 中体验的情况。例如,如果一个“Tool > Macro(工具>宏)”菜单选项不可用,则当运用某些键组合时会发生不同的动作(例如,如果选择“Alt+F11”,将载入一个“Micro(宏)”环境)。主宿的应用程序的行为不同于它在 IDE 之外时的行为的具体情况是与宿主适配器相关联的。宿主适配器通过关闭这些宿主特征或修改它们的行为来处理这些行为上的差异。这样做的目的是改进开发人员的体验,并提高主宿的应用程序和开发环境之间的集成性。

[0045] 注意,如果一个应用程序文档具有多个与它关联的“子文档”(例如,多个工作表关联于一个工作簿),那么将存在与每一个“子文档”相关联的代码视图(即,代码在后的文件关联于每一个工作表)。正在被主宿的并且在 IDE 中定制文档具有一个关联的代码视图。

[0046] 代码视图揭示一个关联于文档的代码在后的文件。与由设计器表示的文档相关联的代码窗口能够以多种方式被激活。一种方式是当设计器或相应的代码工程项目在解决方案浏览器中被选中时,通过单击该解决方案浏览器上的“View-Code”按钮或者从 VS 视图菜单中选择菜单项“View-Code”。在解决方案浏览器中选中设计器的情况下,对应于当前活动子文档的代码窗口被打开。另一种方式是通过从设计器的上下文菜单中选择“View Code”菜单(注意该光标并不在映射的元素上)。还有另一种方式是通过从解决方案浏览器中的工程项目上下文菜单中选择“View-Code”。

[0047] 在所有以上的行动中,代码窗口用窗口顶部的光标的设置来载入。当代码窗口通过在设计器中的控件上双击而调出时,代码生成(即处理程序定义)和导航然后就发生。在此情况下,光标被设置在默认的控制事件处理程序内部。如果处理程序尚不存在,则生成它。主宿的应用程序向 IDE 提供足够的数据,以了解要生成哪一事件处理程序。如果已生成了默认的事件处理程序,则在控件上双击将用户导航到代码编辑器,并且将光标放置在默认的事件处理程序中。

[0048] 视图控件

[0049] 视图控件是被聚集以支持数据绑定的对象,并提供开发人员能用于响应由视图控件引发的事件的事件模型。为了提高视图控件的可视性和可发现性,控制被安置在工具箱中,作为用户可以拖放到应用程序设计器上的“控件”。以下的讨论是关于支持发现、创建、删除和与文字处理应用程序中的视图控件一起工作的设计时工具。然而应该理解,电子表格应用程序也具有与之相关联的视图控件。

[0050] 在文字处理应用程序(和电子表格应用程序)中存在某一级别的原有支持,例如,它们可以用于创建并与视图控件一起工作。为了改进该体验,附加的工具支持被添加到视图控件周围,以增加这个特征的可视性和可发现性,同时也改进了以它们工作的整个开发人员体验。为了提供开发人员熟悉的模型,在视图控件上体验模仿与已管理的控件一起工作,例如,选择上下文和属性窗口支持、从工具箱加入视图控件以及在视图控件上双击以导航到代码视图。

[0051] 特征包括但并不局限于:对视图控件从从工具箱中拖放的常规支持、通过工具箱对所添加的视图控件的默认命名、为视图控件选择上下文(例如,在设计器中选择视图控件将显示属性窗口之中显示属性,并在属性网格中选择个视图控件将在设计器中下拉选择视图控件)、在视图控件上双击以导航到代码编辑器、默认事件处理程序生成、添加到设计器的后方代码和控件的动态同步、对通过属性窗口命名视图控件的支持、以及对代码编辑器中的视图控件的事件下拉支持。其它特征可包括对视图控件的可访问性(支持在工具箱中的“输入”和“双击”以添加视图控件)、显示书签、处理重叠的书签、对工具箱中的视图控件项目的功能键(F1)支持、工具箱可访问性(支持 Ctrl-C 和 Ctrl-V 以添加视图控件)、对删除书签的支持、向用户提供选项以将视图控件降级(demote)为常规书签(减少混乱)、在视图控件之间跳格切换(tab)、以及对形状的工具箱支持。对文字处理程序支持书签视图控件,它用于从单个字符跨越到多个段落。

[0052] 当在 IDE 中时,开发人员能使用文字处理程序的 XML 结构任务窗格来向文档应用模式。该映射过程为每个映射的元素建立 XML 结点视图控件。该 XML 结点视图控件的默认命名将与模式映射的元素相同。为确保模式映射的视图控件变量的唯一性,如果发生冲突则向该名字添加一 X(这里, X = 整数,它在附加的冲突发生时递增)。

[0053] 对于电子表格应用程序视图控件,提供一种“命名范围(Named-Range)”控件和“列表对象(ListObject)”控件。名字范围控件是一种机制,以容易地命名一个范围,并为每个电子表格维护一名字表,以确保对每个被命名的范围唯一地命名。列表对象是电子表格范围的扩充,它被设计来提供改进的数据体验,包括自动增长/收缩到它所应用数据集的能力。列表对象包括行和列,它们被展示为“列表行(ListRows)”和“列表列(ListColumns)”。

[0054] 组件托盘 (Component Tray)

[0055] 组件托盘在宿主应用程序时促进显示或可视化非可视组件的方法。一些开发组件在运行时并不具有可视化的表示,例如数据适配器和相关的组件。组件托盘提供一种用于在设计时可视地识别和操纵非可视组件的机制。开发人员可以拖放、编辑属性、和可视化这些组件。将组件添加到视图的一种方式是通过从工具箱中拖出组件并将它放到该文档上、设置它们的属性、删除它们等等。非可视组件成为该文档后的类的已命名类成员实例。

[0056] 如果在文档中存在非可视组件,则呈现该托盘。如果不存在非可视组件,则托盘完全消失。用户通过把第一非可视组件放置到该视图的主要部分以将其插入到视图上,这将促使托盘被打开,并且非可视组件被移到该托盘中。后续的非可视组件能够被放置到可视表单或直接放到托盘上。

[0057] 如果可视项目在托盘上拖动,则显示一不可放置图标,并且该放置动作执行一个无操作。当最后一个非可视组件从表单中删除时,托盘消失,并且选择返回到设计器。托盘的背景是该系统窗口背景颜色。组件被自动排列以与托盘的当前宽度相匹配,所以从不需要水平滚动。如果需要,垂直滚动条将在托盘中出现。如果添加了比托盘框架可以显示的更多的组件,则垂直滚动条被加入到托盘。如果组件被放置,使得该组件看上去超出了托盘的左或右边界,则一个水平滚动条被加入到托盘。对组件托盘的改变并不弄脏该文档。

[0058] 回顾所揭示的体系结构的某些特点,IDE 工具箱窗口允许将已管理的控件拖放或绘制到设计器的表面上。数据源窗口允许将已管理的数据组件和控件拖放到设计器的表面上。IDE 属性窗口允许操纵文档及其上的组件的属性。IDE 组件托盘窗口宿主放置在文档上的非可视元素,例如数据适配器和数据连接器。设计器使开发人员能书写在文档及其上的组件之后的已管理代码。工具 / 选项对话框提供用于各种各样的 IDE 功能的专用选项。对设计器主宿的应用程序的每一个可以有一专用页面,以使设计器能够灵活地以各种方式表现。当该未管理应用程序成为在 IDE 中的设计器时,该专用页面包括对用户期望调用的那些偏好。

[0059] 对于基于文档的设计器,有为了改进用户的体验而定制 IDE 内的附加应用程序的附加方面。这样的功能的一个例子是已修改的文档上下文菜单。另外,“ViewCode”命令被添加到某些菜单以表示该文档是“已被设计的”和可以对照其来编程。此外,为主宿的应用程序添加一专用的“设计器”工具栏以强调设计环境。

[0060] 现在参照图 6,示出一个可用于执行所揭示的体系结构的计算机的框图。为了提供本发明的各个方面的附加的环境,图 6 和以下的讨论旨在提供可在其中本发明的各方面的适用的计算环境 600 的简要的、一般性的描述。虽然在上文中,本发明是在可运行于一个或多个计算机上的计算机可执行指令的通用环境中描述的,然而本领域的技术人员将认识到,本发明也可以结合其它程序模块和 / 或作为硬件和软件的组合来实现。

[0061] 一般而言,程序模块包括例程、程序、组件、数据结构等等,它们执行特定的任务或实现特定的抽象数据类型。而且,本领域的技术人员应认识到,本发明的方法可以用其它计算机系统配置来实践,包括单处理器或多处理器计算机系统、小型计算机、大型计算机、以及个人计算机、手持计算设备、基于微处理器或可编程客户电器等等,其每一个可操作地耦合至一个或多个相关联的设备。

[0062] 所示出的本发明的方面还可以在分布式计算环境中实践,其中某些任务由通过通

信网络连接的远程计算设备完成。在分布式计算环境中,程序模块可被定位于本地和远程存储器存储设备二者之中。

[0063] 计算机通常包括各种计算机-可读介质。计算机-可读介质可以是可由计算机访问的任何可用介质,并且包括易失的和非易失介质、可移动的和不可移动的介质。作为例子而不是限制,计算机可读介质可以包括计算机存储介质和通信介质。计算机存储介质包括易失的和非易失的、可移动的和不可移动的介质,它们以任何方法和技术实现用于存储信息,如计算机可读指令、数据结构、程序模块或其它数据。计算机存储介质包括但不限于,RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字视频盘(DVD)或其它光盘存储、磁盒、磁带、磁盘存储或其它磁存储设备、或可用于存储期望的信息并可由计算机访问的任何其它介质。

[0064] 通信介质通常在诸如载波或其它传输机制等已调制数据信号中包含计算机可读指令、数据结构、程序模块或其它数据,并包括任何信息传送介质。术语“已调制数据信号”意指其一个或多个特征以在信号中编码信息的方式设置或改变的一种信号。作为例子而不是限制,通信介质包括有线介质,如有线网络或直接线路连接,以及无线介质,如声学、RF、红外和其它无线介质。上述的任一个的组合也应当包括在计算机可读介质的范围之内。

[0065] 再一次参照图 6,示出了一个示例性环境 600,它用于实现本发明的各个方面,它包括计算机 602,计算机 602 包括处理单元 604、系统存储器 606 和系统总线 608。系统总线 608 将包括但不限于系统存储器 606 的系统组件耦合到处理单元 604。处理单元 604 可以是各种可购买的存储器的任一种。双微处理器和其它多处理器体系结构也可用作处理单元 604。

[0066] 系统总线 608 可以是若干种总线结构中的任一种,它能进一步内连到存储器总线(带有或不带有存储器控制器)、外围总线、和使用多种可购买的总线结构之一的局部总线。系统存储器 606 包括只读存储器(ROM)610 和随机存取存储器(RAM)612。基本输入/输出系统(BIOS)被存储在非易失存储器 610 中,例如 ROM、EPROM、EEPROM,该 BIOS 包含了基本例程,它们诸如在启动期间帮助在计算机 602 的单元之间传输信息。RAM 612 也能包括高速 RAM,例如静态 RAM,用于高速缓存数据。

[0067] 计算机 602 还包括内部硬盘驱动器(HDD)614(例如,EIDE, SATA),该内部硬盘驱动器 614 也能被配置为在一个适当的底盘中(未示出)供外部使用、软磁盘驱动器(FDD)616(例如,对可移动磁盘 618 读出和写入)和光盘驱动器 620(例如,读 CD-ROM 盘 622,或对诸如 DVD 等其它高容量磁介质像读出和写入)。硬盘驱动器 614、磁盘驱动器 616 和光盘驱动器 620 能够分别通过硬盘驱动器接口 624、磁盘驱动器接口 626 和光盘驱动器接口 628 连接到系统总线 608。用于外部驱动器实现的接口 624 包括通用串行总线(USB)和 IEEE 1394 接口技术的中的至少一个或两者。

[0068] 驱动器和与它们相关联的计算机可读介质提供了数据、数据结构、计算机可读指令等等的非易失性存储。对于计算机 602,驱动器和介质可容纳合适的数字格式的任何数据的存储。虽然以上计算机可读介质的论述涉及 HDD、可移动磁盘和诸如 CD 或 DVD 等可移动光介质,然而本领域的技术人员应该理解,可由计算机读取的其它类型的介质,例如 ZIP 驱动器、磁盒、闪存卡、盒式磁带及类似物也能被用于该示例性操作环境之中,并且进一步,任何这样的介质可包含用于实现本发明的方法的计算机可执行指令。

[0069] 若干个程序模块可被存储在驱动器和 RAM 612 中,包括操作系统 630、一个或多个应用程序 632、其它程序模块 634 和程序数据 636。操作系统、应用程序、模块和 / 或数据的全部或部分也可被缓存在 RAM 612 中。

[0070] 应理解,本发明能用各种可购买的操作系统或操作系统的组合来实现。

[0071] 用户可通过一个或多个有线 / 无线输入设备,例如键盘 638 和定点设备 (如鼠标 640),将命令和信息输入到计算机 602 中。其它输入设备 (未示出) 可包括话筒、IR 遥控器、操纵杆、游戏垫、输入笔、触摸屏等等。这些和其它输入设备通常通过耦合到系统总线 608 的输入设备接口 642 来连接到处理单元 604,但也能由其它接口链接,如并行端口、IEEE1394 串行端口、游戏端口、USB 端口、IR 接口等。

[0072] 监视器 644 或其它类型的显示设备也通过接口,如视频适配器 646 连接到系统总线 608。除了监视器 644 之外,计算机通常包括其它外围输出设备 (未示出),例如扬声器、打印机等等。

[0073] 计算机 602 可以在使用通过有线 / 无线通信到一个或多个远程计算机,如远程计算机 648 的逻辑连接的网络环境中操作。远程计算机 648 可以是工作站、服务器计算机、路由器、个人计算机、便携式计算机、基于微处理机的消费者电子产品、对等设备或其它公用网络节点,并通常包括相关于计算机 602 所描述的很多或全部元件,虽然为了简单、明了的目的,只有存储器存储设备 650 被示出。所描述的逻辑连接包括到局域网 (LAN) 652 和 / 或更大型网络,如广域网 (WAN) 654 的有线 / 无线连接。这种 LAN 和 WAN 网络环境常见于办公室、公司,并方便企业范围计算机网络,如内联网,所有这些都被连接到全球通信网络,如因特网。

[0074] 当在 LAN 网络环境中使用时,计算机 602 通过有线和 / 或无线通信网络接口或适配器 656 连接到局域网 652。适配器 656 可方便到 LAN 652 的有线或无线通信,它还可包括放置于其上的无线接入点,用于与无线适配器 656 通信。当在 WAN 网络环境中使用时,计算机 602 可包括调制解调器 658、或者连接到 LAN 上的通信服务器、或者具有用于通过 WAN 654,例如通过因特网建立通信的其它装置。调制解调器 658 可以是内置的也可以外置的,并且有线或无线设备通过串行端口接口 642 连接到系统总线 608。在网络环境中,相对于计算机 602 所描述的程序模块或其部分可存储在远程存储器 / 存储设备 650 中。应理解,所示的网络连接是示例性的,也可使用在计算机之间建立通信链路的其它装置。

[0075] 计算机 602 可用于与可操作地布置在无限通信中的任何无线设备或实体通信,例如打印机、扫描仪、台式和 / 或便携计算机、便携数据助理、通信卫星、与无线地可探测的标记相关联的任何一部设备或场所 (如公用电话亭、报亭、休息室)、以及电话。这至少包括 Wi-Fi 和 Bluetooth™ (蓝牙) 无线技术。这样,通信可以是对常规网络的一种预定义的结构,或者简单地至少二个设备之间的特别 (ad hoc) 通信。

[0076] Wi-Fi 或无线保真允许在家中的睡椅边、在旅馆客房的床边或在工作时从会议室连接到因特网,而不要电线。Wi-Fi 是一种类似于蜂窝电话的无线技术,它允许这类设备,如计算机,在室内和室外发送和接收数据;在一个基站范围内的任何地方。Wi-Fi 网络使用被称为 IEEE 802.11 (a,b,g 等) 的无线电技术,以提供安全、可靠、快速的无线连接。Wi-Fi 网络能用于把计算机相互连接、连接到因特网、以及连接到有线网络 (它使用 IEEE 802.3 或以太网)。Wi-Fi 网络以未许可的 2.4 和 5GHz 无线电频带操作,具有 11Mbps (802.11b)

或 54Mbps (802. 11a) 的数据速率, 或者具有包含两个频带 (双频带) 的产品, 从而该网络能够提供类似于许多办公室中使用的基本 10BaseT 有线以太网的真实世界的性能。

[0077] 现在参照图 7, 示出了一个根据本发明的示例性计算环境 700 的示意性框图。系统 700 包括一个或多个客户机 702。客户机 702 可以是硬件和 / 或软件 (如线程、进程、计算设备)。客户机 702 能例如通过采用本发明来容纳 cookie 和 / 或关联的上下文信息。系统 700 还包括一个或多个服务器 704。服务器 704 也可以是硬件和 / 或软件 (如线程、进程、计算机设备)。服务器 704 能够例如通过采用本发明来容纳线程以实现传输。在客户机 702 和服务器 704 之间的一种可能的通信可以是适用于在两个或多个计算机进程之间传输的数据分组的格式。该数据分组可包括例如 cookie 和 / 或相关的上下文信息。系统 700 包括通信框架 706 (如, 诸如因特网等全球通信网络), 它能被采用来以便于客户机 702 和服务器 704 之间的通信。

[0078] 通信可以通过有线 (包括光纤) 和 / 或无线技术来促进。客户机 702 可操作地连接到用于储存对客户机 702 本地的信息 (如 cookie 和 / 或关联的上下文信息) 的一个或多个客户机数据存储 708。类似地, 服务器 704 可操作地连接到用于储存对服务器 704 本地的信息的一个或多个服务器数据存储 710。

[0079] 以上所描述的包括本发明的例子。当然, 不可能为了描述本发明的目的而描述组件或方法的每一可构想的组合, 但本领域的技术人员可以认识到, 本发明的很多其它组合和置换都是可能的。因此, 本发明旨在包含落在所附权利要求书的精神和范围之内所有这样的变换、修改和变化。此外, 在详细描述或者权利要求之中使用了术语“包括”的意义上, 这一术语以类似于术语“包含”的方式意味着包含性的, 如术语“包含”在权利要求书中被用作过渡词时所解释的。

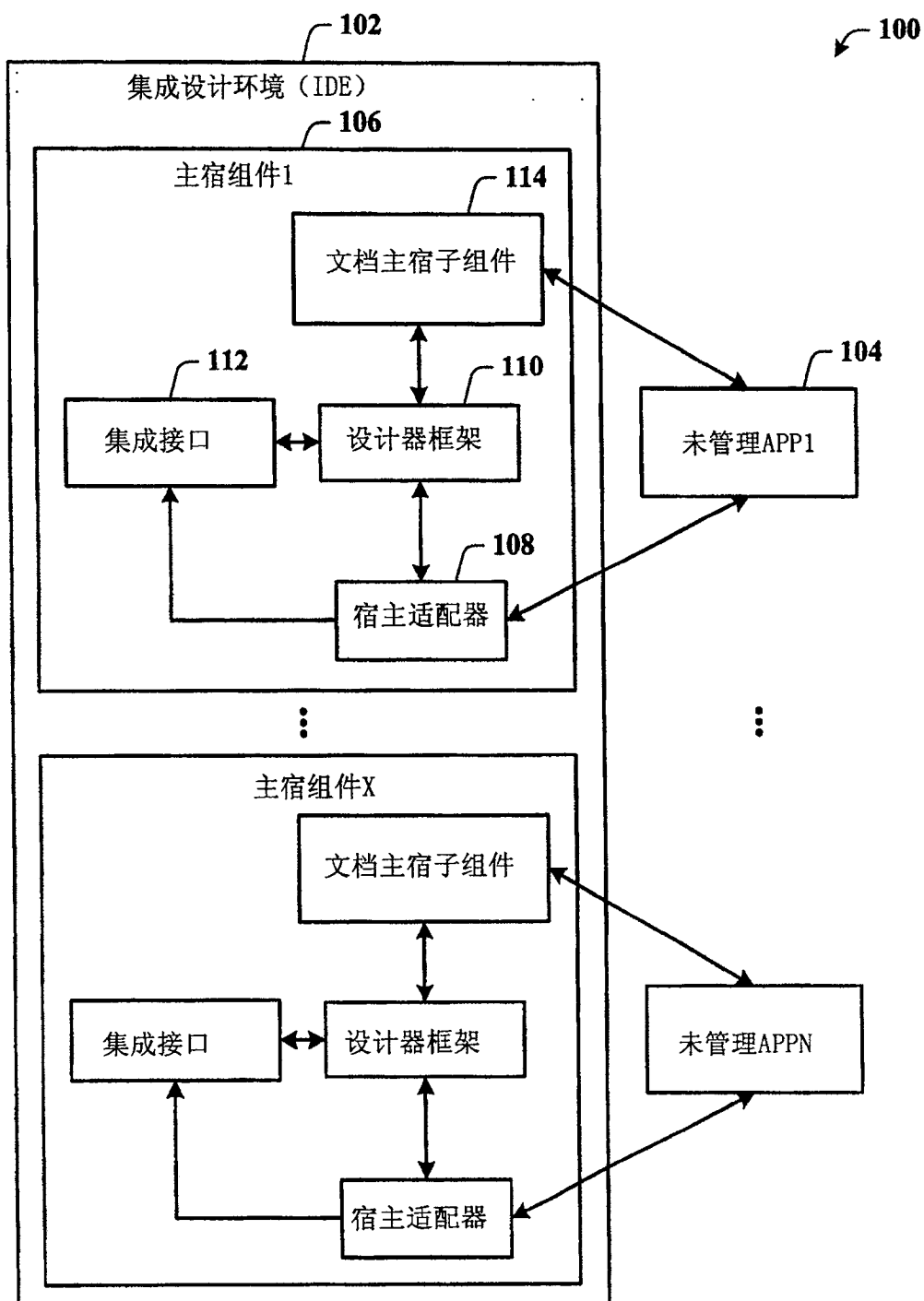


图 1

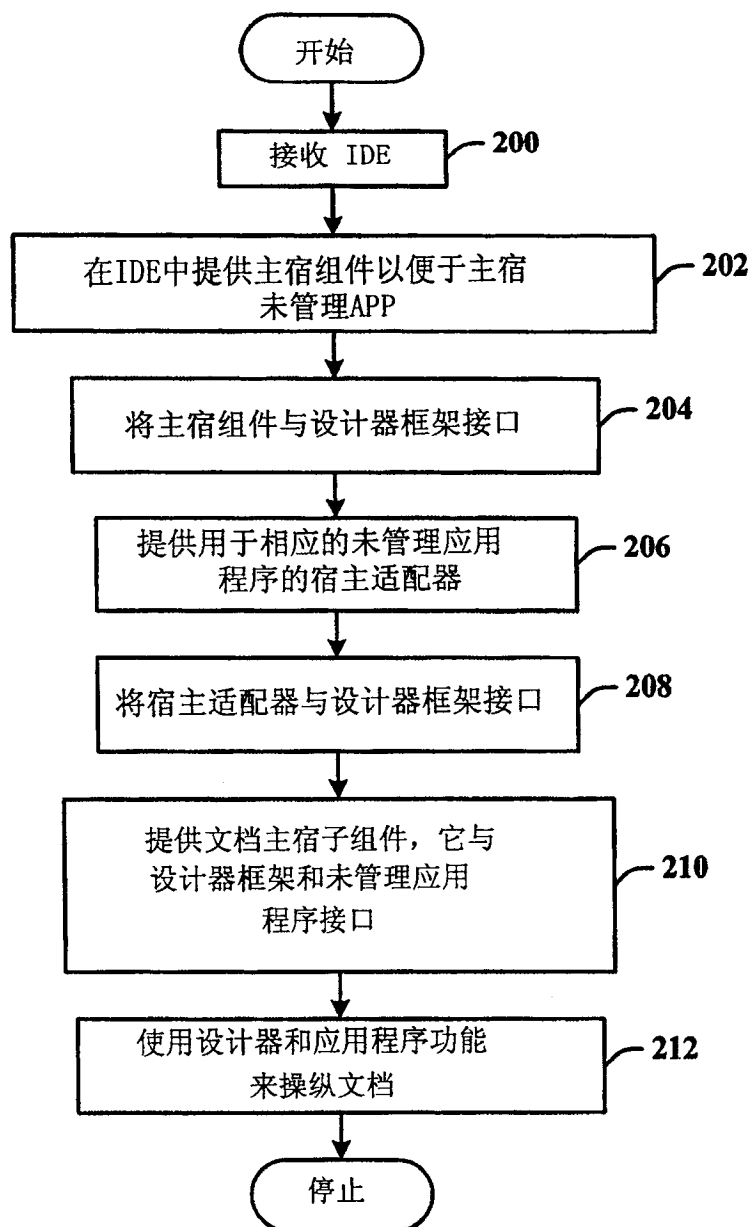


图 2

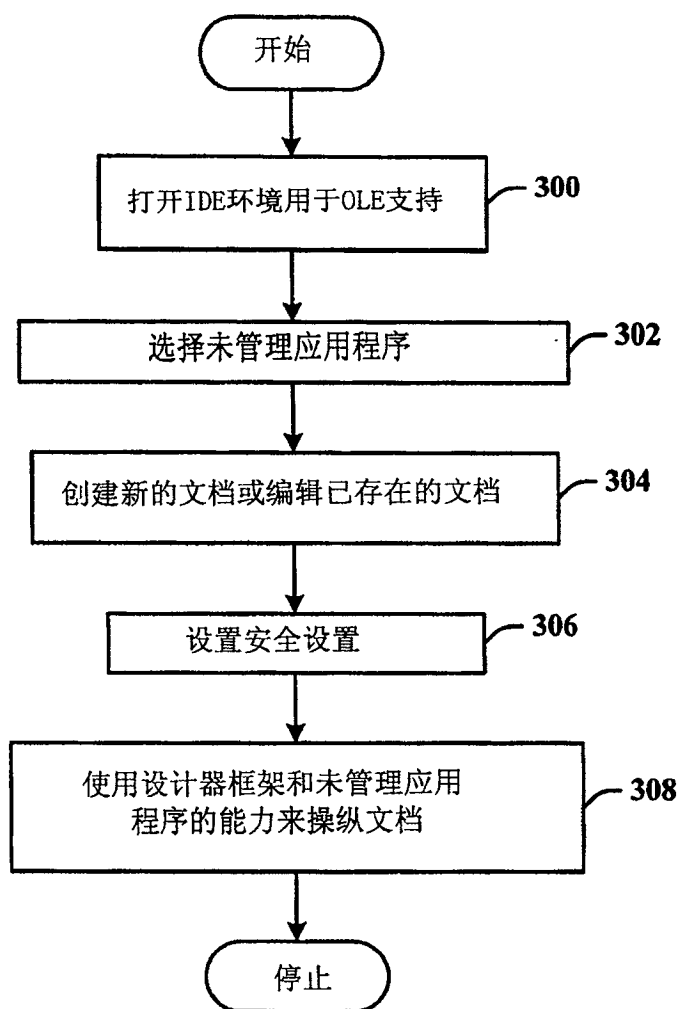


图 3

400

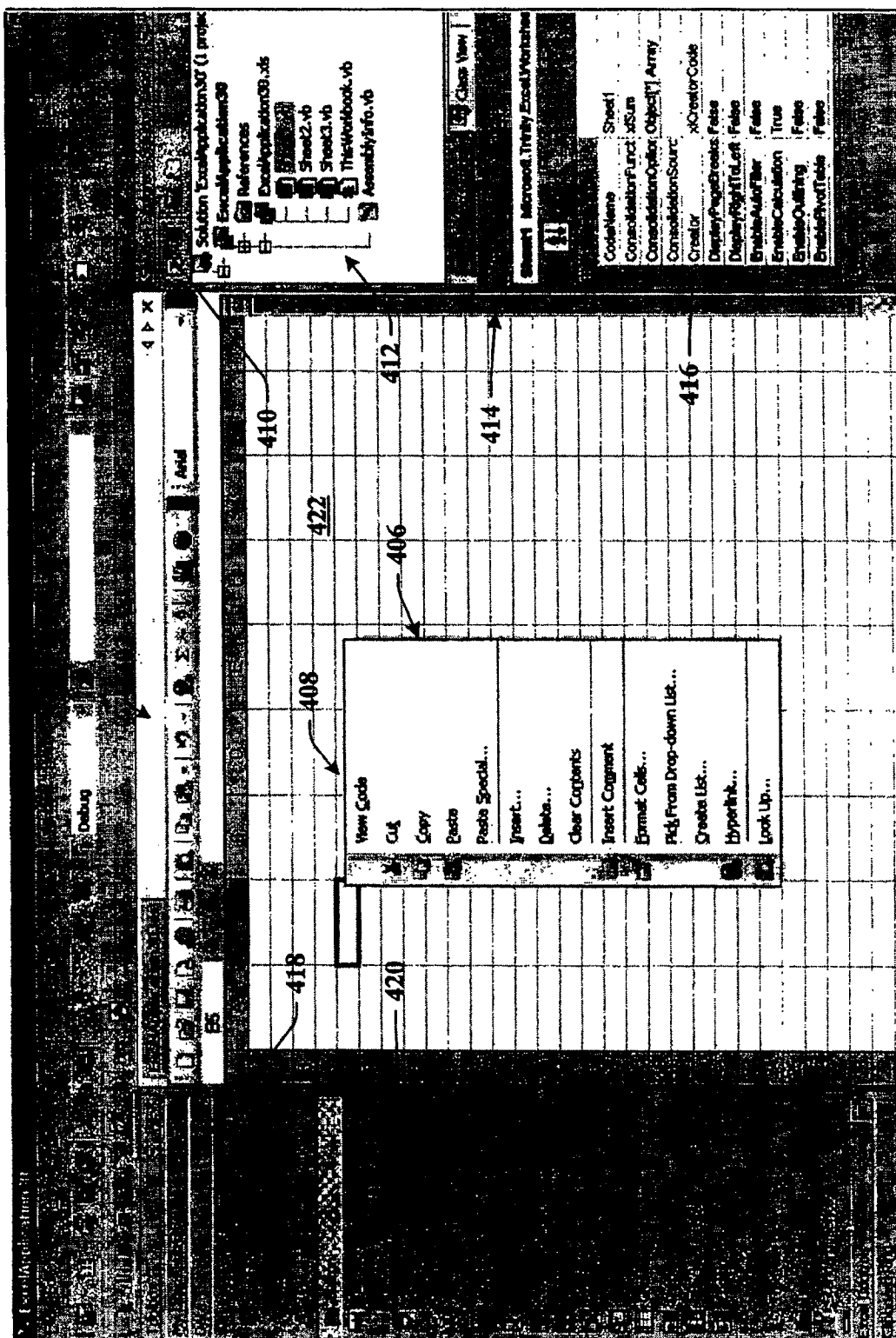


图 4

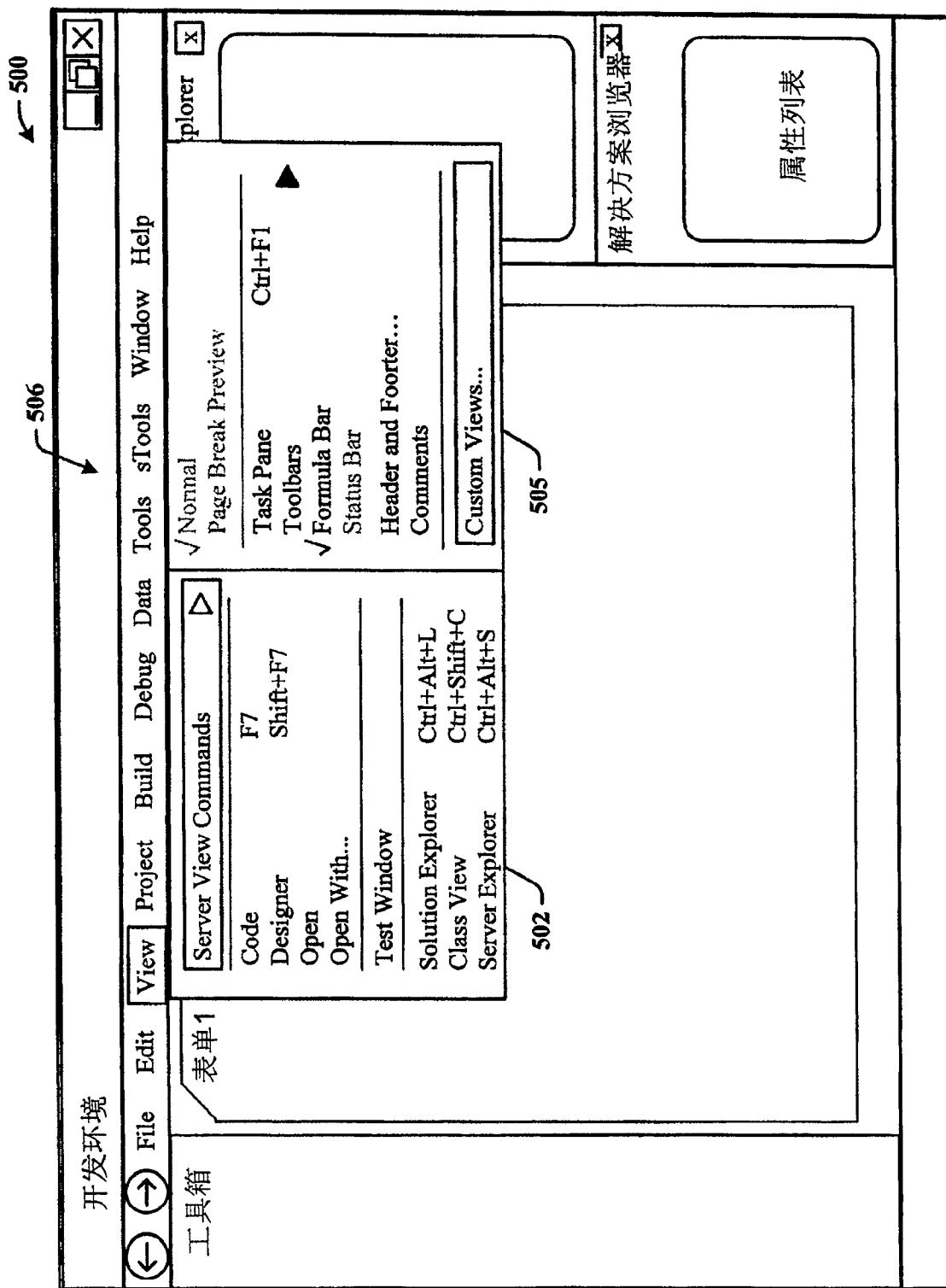


图 5

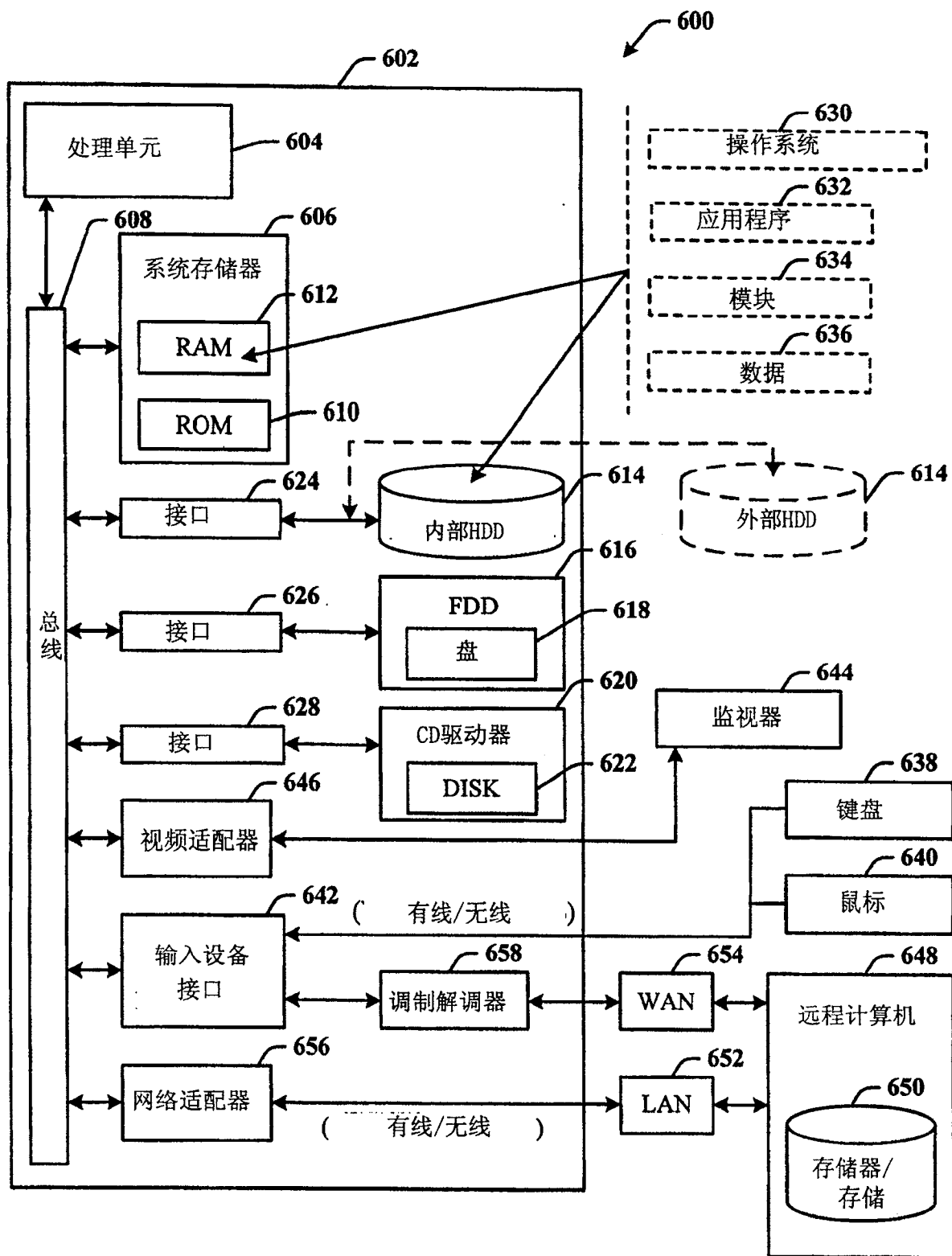


图 6

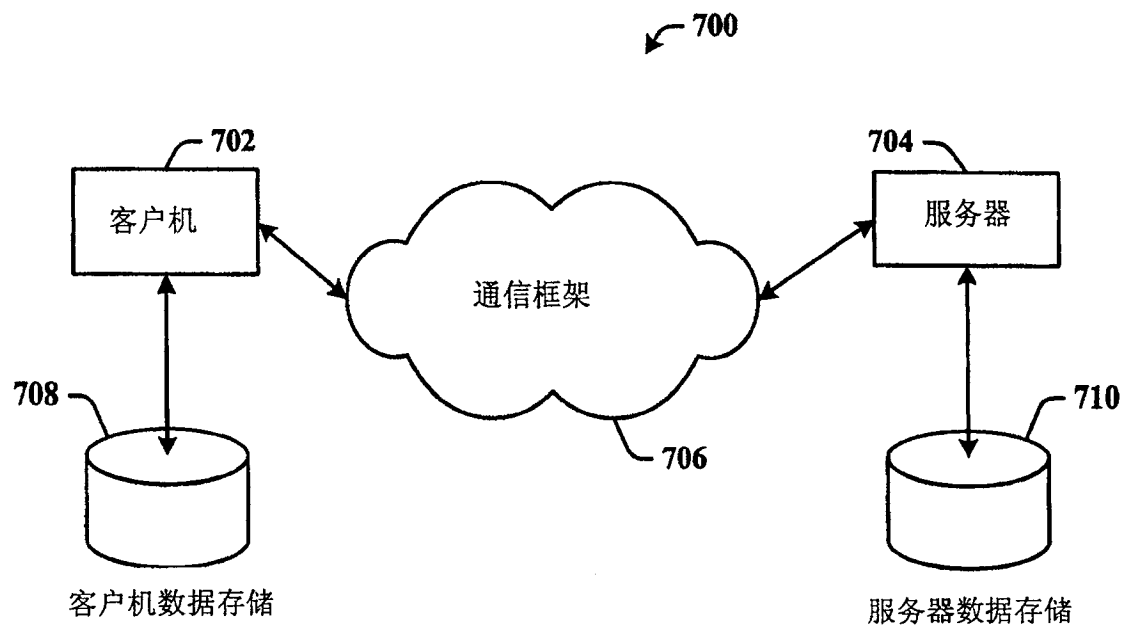


图 7