[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 12/00 (2006.01)
H04L 12/28 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200510081399.1

[43] 公开日 2006 年 3 月 22 日

[11] 公开号 CN 1750470A

[22] 申请日 2005.6.30
[21] 申请号 200510081399.1
[30] 优先权
[32] 2004. 9.17 [33] GB [31] 0420673.6
[71] 申请人 国际商业机器公司
地址 美国纽约阿芒克
[72] 发明人 布伦登·阿瑟斯
斯坦利·K·杰拉德 - 邓恩
基兰·J·奥马霍恩
查尔斯·R·E·史密斯
[74] 专利代理机构 北京市柳沈律师事务所
代理人 黄小临 王志森

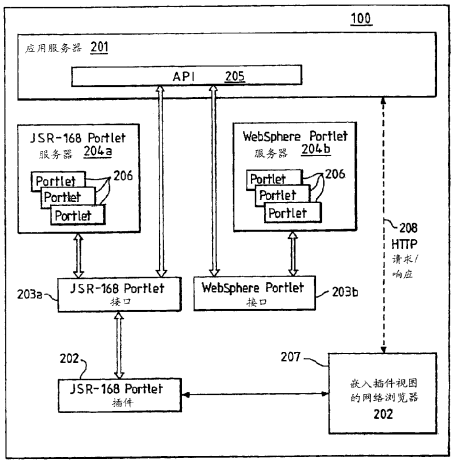
权利要求书 5 页 说明书 25 页 附图 14 页

[54] 发明名称

数据共享系统、方法以及软件工具

[57] 摘要

本发明提供一种方法、系统或软件工具，用来在客户端计算机系统中组件之间传送数据。 该客户端计算机系统具有用户界面，用来显示包含由多个组件呈现的内容的页面。 提供属性中介方，其维护客户端计算机系统的组件所链接的源与目的数据字段的注册表。 当向属性中介方注册的组件检测到用户请求时，通知该属性中介方，然后通知向属性中介方注册的一组组件通信阶段的开始。 由动作所针对的组件处理用户请求，并且将在用户请求处理过程中更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件。 当通信阶段的结束时，属性中介方通知所述组件组，然后至少一个组件呈现内容以显示。



1. 一种在客户端计算机系统中组件之间传送数据的方法，该客户端计算机系统具有用户界面，用来显示包含由多个组件呈现的内容的页面，所述方法包含以下步骤:

提供属性中介方，其维护客户端计算机系统的组件所链接的源与目的的数据字段的注册表;

当向属性中介方注册的组件检测到用户请求时，通知该属性中介方;

通知向属性中介方注册的一组组件通信阶段的开始;

由动作所针对的组件处理用户请求;

将在用户请求处理过程中更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件;

属性中介方通知所述组件组通信阶段的结束；以及

由至少一个组件呈现已更新内容。

2. 根据权利要求 1 所述的方法，其中向其传送所述已更新值的组件的目的数据字段为链接到一或多个其他组件的数据字段的源数据字段，该方法还包含：将已更新数据值传送给每个所述的其他组件。

3. 根据权利要求 1 或 2 所述的方法，其中目的组件处理更新的值，其中向该目的组件传递该更新的值。

4. 根据权利要求 3 所述的方法，其中如果目的组件对已更新值的处理更新目的组件源数据字段的值，则该方法还包含：将已更新源数据字段值传送给与其链接的每个目的数据字段。

5. 根据上述权利要求任一项所述的方法，还包含：检测引起组件动作的用户请求。

6. 根据权利要求 5 所述的方法，还包含：显示包含多个视图的页面，其中每个视图由客户端计算机系统中一或多个组件提供。

7. 根据权利要求 6 所述的方法，其中通过检测与由那个组件所提供的视图的用户交互，检测引起该组件动作的用户请求。

8. 根据权利要求 5、6 或 7 所述的方法， 其中所述检测用户请求包含：在组件中设置事件监听器。

9. 根据上述权利要求任一项所述的方法，其中被通知通信阶段开始与结

束的组件组为那些具有在显示页面上显示的视图的组件组。

10. 根据上述权利要求任一项所述的方法，其中在发送有关通信阶段结束的通知之前，动作所针对的组件呈现其内容。

11. 根据上述权利要求任一项所述的方法，其中具有在通信阶段过程中被更新的数据字段的每个组件呈现已更新的内容。

12. 根据上述权利要求任一项所述的方法，还包含：向用户显示已更新页面。

13. 根据权利要求 6 至 12 中任一项所述的方法，其中至少一个具有在显示页面上显示的视图的组件为 portlet。

14. 根据权利要求 13 所述的方法，还包含：在客户端计算机系统中提供 portlet 服务器，以处理对于 portlet 内容的请求，该 portlet 服务器在其上安装有一或多个 portlet。

15. 根据权利要求 14 所述的方法，还包含：在显示页面上提供视图窗口，并且运行其中的网络浏览器以显示 portlet 内容。

16. 根据权利要求 15 所述的方法，还包含：通过网络浏览器，从安装在 portlet 服务器上的 portlet 请求网络内容；以及

在所提供的视图窗口中显示在显示页面上收到的网络内容。

17. 根据上述权利要求任一项所述的方法，还包含：在客户端计算机系统上运行客户端平台，所述用户界面允许显示由在该客户端平台中注册的插件组件提供的视图。

18. 根据权利要求 14、15 或 16 所述的方法，还包含：提供与在 portlet 服务器上安装的每个 portlet 相关联的 portlet 视图工具。

19. 根据权利要求 18 所述的方法，还包含：portlet 视图工具截获对于来自 portlet 的对于网络内容的用户请求。

20. 根据权利要求 19 所述的方法，还包含：当截获用户请求时，通知属性中介方。

21. 一种计算机系统，包含：

多个组件，每个组件都具有一或多个数据字段，并且能够用来呈现内容；以及

属性中介方，包含注册表，其将向属性中介方注册的源组件的源数据字段映射到目的组件的目的数据字段；

其中：

第一组件用来接收用户请求，向属性中介方发送关于收到该请求的通知，并且处理该用户请求；

响应于从第一组件收到所述通知，属性中介方用来通知一组组件通信阶段的开始，并且通知所述组件组通信阶段的结束；

在用户请求处理过程中已更新的每个源数据字段的值被传送给具有所链接的目的数据字段的每个组件。

22. 根据权利要求 21 所述的计算机系统，还包含：通信部件，当目的组件的目的数据字段也是链接到一或多个其他组件的源数据字段时，用来将所述已更新数据值传送给所述一或多个其他组件。

23. 根据权利要求 21 或 22 所述的计算机系统，其中属性中介方用来将在用户请求处理过程中已更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件。

24. 根据权利要求 21 至 23 中任一项所述的计算机系统，还包含：用来处理目的组件所接收的已更新值的处理部件。

25. 根据权利要求 24 所述的计算机系统，还包含：如果由目的组件对收到的已更新值的处理更新目的组件源数据字段的值，则用来将源数据字段已更新值传送给与其链接的每个目的数据字段的部件。

26. 根据权利要求 21 至 25 中任一项所述的计算机系统，还包含：用来检测引起组件动作的用户请求的部件。

27. 根据权利要求 21 至 26 中任一项所述的计算机系统，还包含：显示部件，用来显示包含多个视图窗口的页面，其中每个视图窗口包含由客户端计算机系统的一或多个组件提供的内容。

28. 根据权利要求 27 所述的计算机系统，还包含：用来检测与由组件所提供的视图窗口的用户交互的部件。

29. 根据权利要求 25 或 27 所述的计算机系统，其中检测用户请求包含：在组件中设置事件监听器。

30. 根据权利要求 21 至 29 中任一项所述的计算机系统，其中属性中介方用来向其发送有关通信阶段开始与结束的通知的组件组为那些具有在显示页面上显示的视图的组件组。

31. 根据权利要求 21 至 30 中任一项所述的计算机系统，还包含：

用来显示包含一或多个视图窗口的页面的用户界面，其中视图窗口显示由该计算机系统的组件提供的内容。

32. 根据权利要求 31 所述的计算机系统，还包含:

portlet 服务器，用来处理对于 portlet 内容的请求，该 portlet 服务器在其上安装有一或多个 portlet；以及

portlet 视图工具，与在 portlet 服务器上安装的 portlet 相关联，用来控制相关联的 portlet 并且用来运行网络浏览器，以在视图窗口中显示来自与其相关联的 portlet 的内容。

33. 根据权利要求 32 所述的计算机系统，还包含：多个 portlet 视图工具，每个都与在 portlet 服务器上安装的 portlet 相关联。

34. 根据权利要求 32 或 33 所述的计算机系统，其中 portlet 视图工具包含与 portlet 相关联的、并且能够被配置来向该计算机系统提供视图的 portlet 视图插件。

35. 根据权利要求 32 、33 或 34 所述的计算机系统，其中包括多个 portlet 服务器，每个服务器具有一或多个安装在其中的 portlet。

36. 根据权利要求 35 所述的计算机系统，其中 portlet 视图工具包含 portlet 接口，用来使能与相关 portlet 服务器上相关联的 portlet 的交互。

37. 根据权利要求 21 至 36 中任一项所述的计算机系统，其中 portlet 视图工具能够被配置来提供用于显示 portlet 内容的视图窗口的框。

38. 根据权利要求 21 至 37 中任一项所述的计算机系统，还包含用来运行客户端平台的部件，以及用来允许显示由在该客户端平台中注册的插件组件所提供的视图的用户界面。

39. 根据权利要求 32 至 34 中任一项所述的计算机系统，其中 portlet 视图工具包含用来确定待由网络浏览器请求以在 portlet 视图窗口中显示的 portlet 内容的 URL 的部件。

40. 一种在客户端计算机系统中组件之间传送数据的软件工具，该客户端计算机系统具有用户界面，用来显示包含由多个组件呈现的内容的页面，所述软件工具用来:

维护客户端计算机系统的组件所链接的源与目的数据字段的注册表;

通过计算机系统的组件接收关于检测用户请求的通知;

通知一组组件通信阶段的开始;

将在用户请求处理过程中更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件；以及

当通信阶段结束时，通知所述组件组。

41. 根据权利要求 40 所述的软件工具，还用来：当目的组件的目的数据字段也是链接到一或多个其他组件的源数据字段时，将所述已更新数据值传送给所述一或多个其他组件。

42. 根据权利要求 40 或 41 所述的软件工具，还用来：将在用户请求处理过程中已更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件。

43. 根据权利要求 40 至 42 中任一项所述的软件工具，还用来：如果由目的组件对收到的已更新值的处理更新目的组件源数据字段的值，则将源数据字段已更新值传送给与其链接的每个目的数据字段。

44. 一种计算机程序产品，包含用来执行权利要求 1 至 20 中任一项所述的方法的计算机可实现指令。

45. 一种数据处理系统，包括权利要求 40 至 43 中任何一项所述的软件工具。

46. 一种方法、系统或软件工具，用来在计算机系统的组件之间共享数据，基本如此处参照附图所述。

数据共享系统、方法以及软件工具

技术领域

本发明涉及数据处理领域，具体而言，涉及用于在客户端平台上运行的portlet(小门户)之间数据共享的方法、系统以及软件工具。

背景技术

万维网为一种因特网多媒体信息检索系统。在网络环境中，客户端机器利用超文本传送协议（HTTP）与网络服务器通信。网络服务器利用称为超文本标记语言（HTML）的标准页面描述语言向用户提供至诸如文本、图片、图像、声音、视频等文件的访问。HTML提供基本文档格式，并且允许开发者制定至其他服务器与文件的、称为超链接的连接。在因特网模式中，至服务器的网络路径由统一资源定位符（URL）标识，URL具有用于定义网络连接的特殊语法。所谓的网络浏览器，例如Netscape Navigator(Netscape Navigator为Netscape Communication Corporation的注册商标)或者Microsoft Internet Explorer(Microsoft与Internet Explorer为Microsoft Corporation的注册商标)，其为在客户端机器上运行的应用程序，使用户能够通过借助URL指明链接来访问信息，并且能够在不同的HTML页面之间导航。

当网络浏览器的用户选择链接时，客户端机器向命名服务发出请求，以将主机名称（在URL中）映射到服务器机器所在的特定网络IP（互连网协议）地址。该命名服务返回可能相应于该请求的IP地址。利用该IP地址，网络服务器建立到服务器机器的连接。如果服务器机器可用，则其返回网页。为了进一步促进站点内的导航，网页一般包含一或多个称为“锚链”（anchor）或“链接”的超文本引用。

门户一般为聚合来自各种不同来源的内容、并且在门户网页内呈现该内容的网络应用，并且可能具有复杂的个性化特征，以向用户提供定制内容。门户应用可以提供至一或多个后端软件应用的网关，并且一般配备在分离的门户服务器上。

门户服务器一般将网页安排在包含一或多个 portlet 的门户页面内。

portlet 为网络组件，由 portlet 容器管理，其处理并生成动态网络内容。该内容一般称为片断（fragment），可以由门户与来自其他 portlet 的内容聚集，以形成门户页面。依赖于 portlet 的用户配置，由 portlet 生成的内容可能对各个用户都不相同。

门户提供了一种导航框架，用于按层次在服务器上排列的一组网页。该框架提供了用户界面，其允许导航通过服务器上的可用页面层次。提供该导航的用户界面称为主题（theme）。每个页面可能包含零或多个 portlet，利用设计或管理工具预先确定并且构造页面排列。

对于标准的服务器端门户，使用客户端网络服务器来在单个页面上观看几个 portlet 的聚集后的输出。用户与由 portlet 产生的内容交互，例如通过提交表单或者跟踪链接，从而使 portlet 动作被门户接收。当门户服务器收到 HTTP 请求时，其确定该请求是否包含目标为与门户页面相关联的任一 portlet 的动作。如果有目标为 portlet 的动作，则门户请求 portlet 容器启动该 portlet 以处理该动作。portlet 处理该动作，并且生成内容片断以包含在新的门户页面中。另外，所请求的门户页面上所有其他 portlet 都更新并且将内容片断传递给门户。门户将每个 portlet 内容片断在 portlet 窗口中打包，向每个窗口添加标题与控件按钮。有时这称为“包裹”（wrapping）每个所提供的内容片断，并且用来包裹所述片断的附加标记被称为“皮肤”(skin)。皮肤可能包含控件按钮，可以使用这些控件按钮来（例如）将 portlet 置于特定模式下，例如编辑或配置，或者用来将 portlet 的显示状态变为最大化或最小化视觉状态，就像人们在一般窗口系统中看到的一样。然后，门户将 portlet 窗口聚集到完整的门户页面中，以发送给客户端。网络浏览器将该代码呈现在客户端的显示屏幕上。

如果用户开始于至门户主页面的 URL、或者借助至特定页面的 URL、或者借助至 portlet 实例的 URL 链接，则可以通过主题所提供的导航框架导航至门户所提供的页面；在后一种情况下，门户提供包含该 portlet 的完整页面。

许多公司都对开发满足其业务需求的 portlet 进行了大量投入。目前这些都是纯服务器端应用，只能使用客户端网络浏览器访问，并且只有在通过某种形式的网络连接到门户服务器时才能访问。如果在不连接到网络时也能继续使用这些相同的 portlet，则很有用，这可以从通过将 portlet 重置为在客户端展开的独立程序实现。然而，这会需要修改所有 portlet 以允许它

们作为独立应用程序运行，这将是一项费钱费时的工作，可能会不利于从服务器向客户端转移。

与本专利申请同时提交的、名为“Method and Software Tool for the Installation of Portlets into a Client Platform”的未决英国专利申请，以及具有 portlet 视图的客户端计算机系统（Cleint Computer System with Portlet Views），目的就在于处理这些问题，并且允许将 portlet 迁移到客户端计算机系统。该系统允许以对用户透明的方式将 portlet 与其他客户端应用一道集成。

不采用门户聚集模型，而是每个 portlet 都与独立视图窗口相关联，其中在这些视图窗口的每一个中运行网络浏览器。指示浏览器导航至与 portlet 相关联的页面，由此显示由该 portlet 输出的内容。然后利用 HTTP，借助嵌入式浏览器，进行用户与 portlet 之间的所有通信。可以通过 portlet 视图工具，进行改变 portlet 状态或模式的请求，并且通过将与该 portlet 相关联的浏览器重定向到特定 URL 来将这些请求传递给给 portlet。

客户端计算机的优选实施方式允许未经修改的 portlet 在富客户端平台（也称为智能客户端、胖客户端或厚客户端）上的窗口内运行，该丰富客户端平台为以下客户端：其进行大部分处理，而服务器计算机系统执行少量或不执行处理。对 portlet 的控制从 portlet 标记转移到客户端 UI。从用户的观点看，portlet 与任何其他客户端应用之间没有差别。用户能够以与 portlet 被部署为服务器 portlet 时与该 portlet 交互完全相同的方式与其交互。

在该情况下，如果 portlet 能够共享数据并且能够与其他客户端应用合作，则可大大提高在客户端上保持 portlet 的价值。本发明目的在于提供一种对具有 portlet 视图的客户端计算机系统的改进，其具有 portlet 共享数据的功能。

发明内容

本发明的各个方面提供了一种用来在客户端计算机系统中组件之间传送数据的方法、软件工具、以及系统。该客户端计算机系统具有用户界面，用来显示包含由多个组件呈现的内容的页面。提供属性中介方，其维护客户端计算机系统的组件所链接的源与目的数据字段的注册表。当向属性中介方注

册的组件检测到用户请求时，通知该属性中介方，然后通知向属性中介方注册的一组组件通信阶段的开始。由动作所针对的组件处理用户请求，并且将在用户请求处理过程中更新的每个源数据字段的值传送给具有所链接的目的数据字段的每个组件。当通信阶段的结束时，属性中介方通知所述组件组，然后至少一个组件呈现内容以显示。

在名称为"Portlet Data Sharing System, Mehtod and Program Product"的 US20040090969（其内容融入此文作为参考）中描述的属性中介方体系结构允许 portlet 开发人员定义在标准门户中接收和/或与其他 portlet 共享数据的数据字段。该属性中介方概念用于本发明，并且得到扩展，以提供客户端 portlet 与在客户端计算机系统中运行的其他应用之间的异构通信机制。

在 US20040090969 中描述的属性中介方数据共享模型的要素在于改变 portlet 的生命周期，以在要开始动作阶段（其中进行属性中介方通信）时，以及再次在该阶段要结束时，要求通知。本发明优选实施方式避免了修改 portlet 生命周期模型的需求，由此将对 portlet API 的修改最小化。

在本发明优选实施方式中，关于属性中介方通信的开始与结束的通知由属性中介方直接转发给合作的组件，而属性中介方不知道当前组件的确切性质。通过这种方式，所有属性提供者都以一致的方式发挥作用，并且所有属性提供者都以相同的语义接收相同的通知。在不修改 portlet 容器的前提下之所以可以如此的原因在于客户端门户的性质，其中客户端平台完全控制了每个 portlet 的用户交互与显示，因此在必要时能够发出正确的事件。

目前存在大量基于 IBM WebSphere 技术的 portlet，还有越来越多的 portlet 基于最近的 JSR(Java 规格请求) -168 Portlet API 标准。尽管目的相同，这些技术在实现上有很大差异，优选实施方式修改了属性中介方的关键思路，从而该系统可与多种类型的 portlet 以及客户端应用一起工作，而不用修改现有 API.

更优选的特征为如下配置该机制，使得任意两个客户端组件之间的通信以完全独立于组件性质（不管其为 portlet 还是其他）的方式进行。

附图说明

现在仅作为例子参照附图描述本发明的优选实施方式，其中：

图 1 显示客户端平台显示的例子；

图 2A 显示根据本发明优选实施方式的系统的组件之间交互的概图;

图 2B 显示其中可以实现图 2A 的系统的客户端插件体系结构及其功能组件;

图 2C 显示根据优选实施方式的客户端计算机系统的扩展点与扩展之间相互关系;

图 3A、3B、3C 显示根据本发明优选实施方式的可能显示的显示页面或部分显示页面屏幕;

图 4A 显示根据本发明优选实施方式的在设计显示页面时所涉及的步骤;

图 4B 显示根据本发明优选实施方式的在显示所存储的显示页面时所涉及的步骤;

图 5 显示根据本发明优选实施方式的在安装 portlet 时所涉及的步骤;

图 6 显示根据本发明优选实施方式的连线工具的元素;

图 7A、7B、7C、7D 显示根据本发明一种实施方式的透视图的部分屏幕;

图 8A、8B、8C 显示根据本发明一种实施方式的连线工具屏幕; 和

图 9 显示根据本发明一种实施方式的、当检测用户动作时所采取的步骤的流程图。

具体实施方式

在以下描述中，参照构成本发明描述一部分的附图并且说明一种实现。应该理解可以利用其他实现，并且在不脱离本发明的范围的前提下，可以进行结构与操作上的修改。

在客户端途径中，分离的视图窗口与每个 portlet 相关联，并且在每个这些视图窗口内运行网络浏览器。引导 portlet 视图内的浏览器导航至与该 portlet 相关联的页面，从而显示该 portlet 的输出。然后，使用 HTTP，通过嵌入式浏览器，进行用户与 portlet 之间的所有通信。可以通过客户端平台 UI，进行改变 portlet 模式或状态的请求，并且通过将与该 portlet 相关联的浏览器重定向到特定 URL，将这些请求传递给该 portlet。

本发明的优选实施方式利用基于 Eclipse 技术的客户端平台，其提供源代码构造块、插件框架以及运行实例，这些有利于应用工具开发。其为用于创建、集成、以及部署模块（所谓的插件）的环境，其可以是跨越各种计算环境的工具。

Eclipse 平台定义了工作台（Workbench）用户界面（UI）以及一组公用的、不依赖于领域的用户交互范例，可将工具构造者可以插入其中以添加新功能。该平台具有一组可以由工具构造者扩展的标准视图。工具构造者既可以添加新视图，也可以将新的领域特定功能插入现有视图中。

工具被编写为插件，其操作 Eclipse 文件空间（称为 Workspace（工作空间））中的文件，并且在工作台中呈现其工具特定的 UI。当启动 Eclipse 平台时，向用户呈现工作台窗口显示页面，其提供由一组可用插件构成的集成开发环境。

Eclipse 将在多种操作系统上运行，包括 Linux 与 Windows((Microsoft 与 Windows 为 Microsoft Corporation 在美国、其他国家、或者所有国家的注册商标，Linux 为 Linus Torvalds 在美国、其他国家、或者所有国家的注册商标)。

在 Object Technology International Inc. 的 Eclipse Platform Technical Overview 中描述了 Eclipse 平台，其可以（例如）在 www.eclipse.org 上找到，其融入此文作为参考。

Eclipse 平台包含插件文件夹，其中部署了各个插件。每个插件安装在插件文件夹下其自身的文件夹中。插件以 XML 清单文件描述，称为 plugin.xml，其驻留在插件的文件夹中。清单文件声明该插件与其他插件的互连，并且告诉 Eclipse 平台其需要知道哪些以激活该插件。可以通过插件注册表 API，以编程得到插件清单文件的分析后内容，并且在称为插件注册表的主存内库中缓存分析后的插件规格。

Eclipse 平台提供插件管理内核以及存在于每个 Eclipse 部署的核心插件。这些核心插件的标识被硬编码在 Eclipse 平台中，并且该平台知道在 Eclipse 的每个运行实例中激活这些插件。在另一方面，当其他插件需要时，激活非核心插件。

插件之间的互连模型是简单的：插件声明其他插件能够对其作出提供的任意数目的被命名的扩展点，以及其对其他插件的或多个扩展点作出提供的任意数目的扩展。其他插件可以扩展插件的扩展。扩展点可以具有相应的 API 接口。通过对该扩展点的扩展，其他插件提供该接口的实现。

作为附件 1 融入此文的 Eclipse 平台插件清单规格在文档中记录了定义插件时使用的 XML 元素与属性。每个插件具有唯一的标识符（XML 属性 id），

其用来在其他相关插件清单文件内指示插件。也可以在供应者提供的插件代码内使用唯一标识符，以访问插件的运行实例。

在 Eclipse 平台上安装插件包含两个阶段。第一阶段包含：将构成插件的资源（清单文件、jar 或 war 文件以及其他资源）拷贝到该平台的插件目录下该插件的相应文件夹中。第二阶段包含：通过向插件注册表添加来自插件清单文件的数据（例如分析后插件规格），注册该插件。一旦完成了这一步，当需要插件执行某功能时，Eclipse 平台就可以激活该插件。

Eclipse 平台用户界面（UI）模式基于视图与透视图。视图为显示有关对象的信息的窗口；透视图提供其上可以组织视图的显示页面或画布。插件可以向平台提供视图，并且视图可以被组织为不同的透视图。

Eclipse 工作台提供的多个扩展点，包含视图扩展点。借助该视图扩展点，通过向该视图扩展点提供一或多个扩展，其他插件可以向平台提供视图。

参照图 1，其中显示客户端平台的显示器的简单例子，其具有三个不同的透视图 102a、102b、102c。透视图 102a 在最上层显示，并且包含五个视图 104，其中三个视图显示 portlet 内容，两个非 portlet 视图显示其他客户端平台组件。由 portlet 视图插件提供每个显示 portlet 的视图，作为 Eclipse 视图扩展点的扩展。以后将描述向 Eclipse 透视图添加 portlet 视图的方法。

如上所述，portlet 一般在 portlet 服务器内运行，portlet 服务器在页面上聚集来自所有 portlet 的内容，并且返回所聚集的内容。在本发明中，portlet 服务器不将来自一 portlet 的内容与来自另一 portlet 的内容聚集，而是一次返回由单个 portlet 提供的网络内容。由此，此处使用名词“portlet 服务器”来指示已经修改了的标准 portlet 服务器进程。此处使用的名词“portlet 服务器”表示以下服务器进程：当被请求时，其利用诸如 HTTP 等协议来提供来自单个 portlet 的标记文档，以及任意相关的文件与脚本。本发明可以用单个 portlet 服务器实现。然而，当前对于编写 portlet 存在几个实际的标准。优选实施方式支持这些标准中的两个，即 WebSphere portlet 以及 Java 规格请求（JSR）-168（StandardWebSphere 为 International Business Machines Corp. of Armonk, NY. 的注册商标）。

优选实施方式提供利用 WebSphere 门户服务器与 JSR-168 门户服务器的修改版本的 portlet 服务器。服务器进程提供来自单个 portlet 的标记文档，

即没有对来自不同 portlet 的内容的聚集。URL 指向单个的 portlet，并且仅返回该 portlet 自身的标记片断。消减了门户服务器的代码，从而不提供任何方案、或者皮肤、或者进行 portlet 内容片断的包裹。门户页面替换为 Eclipse 透视图，其具有在 Eclipse 中改变透视图以导航通过显示页面的功能，并且通过实现为至 Eclipse 平台的插件的 portlet 视图工具，提供用于在显示页面上显示的 portlet 内容片断的皮肤，如下所述。

另外，Tomcat 应用服务器代码、WebSphere 与 JSR-168 门户服务器代码被尽可能地消减，以不包含客户端计算机系统中不需要的不必要的功能，例如服务器安全功能（登录/验证等等）。

配置 portlet 服务器以允许安装 portlet，并且支持 portlet 生命周期（即，Init、Service、以及 Destroy 方法）以及模式改变。Init 方法允许 portlet 被初始化；当需要 portlet 呈现其内容时，调用 Service 方法；Destroy 方法允许 portlet 被停止服务并销毁。

优选实施方式利用嵌入式版本的 Tomcat Java 2 Enterprise Edition(J2EE)应用服务器来保持 portlet 服务器。Tomcat 为在 Apache Software Foundation 的 Jakarta 项目下开发的、JavaServlet 与 JavaServer Pages 技术的开源实现(Java 与所有基于 Java 的商标为 Sun Microsystems, Inc.在美国、其他国家、或者所有国家的商标)。另外，Tomcat 应用服务器提供对 J2EE 服务的支持，例如数据库连接与 Enterprise JavaBeans(EJBs)。本领域技术人员可以理解：同样可以使用其他应用服务器，例如 WebSphere 应用服务器。

图 2a 显示根据优选实施方式的系统的组件之间互连的概况。这些组件包含：Tomcat 应用服务器 201 与应用服务器接口（API）205；对于所支持的每种 portlet 类型的 portlet 服务器，此处为 WebSphere portlet 服务器 204b 与 JSR-168 portlet 服务器 204a；portlet 接口 203a、203b(其实现为 portlet 服务器的一部分)；JSR-168 portlet 视图插件 202，其与在 portlet 服务器 204a 上安装的 JSR-168 portlet 206 之一相关联；以及网络浏览器 207，其在由 portlet 视图插件 202 提供的客户端视图（未在图 2a 中显示）内运行。该网络浏览器使用 HTTP 208 与 Tomcat 服务器 201 通信。

图 2b 显示优选实施方式的 Eclipse 插件体系结构，其中提供 portlet 视图工具作为 portlet 视图插件。如图所示，客户端系统 100 包含多个 portlet

视图插件 202（显示了两个）以及项目插件 110。项目插件包含用于实现该系统、并且包含多个 UI 组件 220 与多个服务器组件 230 的代码。（虽然在图 2b 中这些被显示为形成单个插件 110，但是可以根据偏好将这些实现于许多独立的插件中。）UI 组件 220 包含安装向导 209，具有 UI 动作处理器 212 的标准 portlet 视图 211，以及门户模板 213。标准 portlet 视图代码 211 由每个 portlet 视图插件使用，以将其 portlet 内容包裹到视图窗口中、提供控件图标、当要显示其相关联的 portlet 的视图时运行网络浏览器等等。

portlet 安装向导 209 提供 portlet 安装菜单，以后将描述安装向导的实现，其创建 portlet 视图插件 202。

标准 portlet 视图 211 提供每个 portlet 视图插件实现其视图所需的代码。这使 portlet 视图工具能够提供用于 portlet 视图的框、在视图框上使用的 portlet 控件图标，并且在每个 portlet 视图中运行网络浏览器。UI 动作处理器 212 包含由 portlet 视图工具响应于用户点击 portlet 控件图标而执行的代码，用来表示控件动作。

门户模板组件 213 提供缺省初始透视图，用户可以从此选择并打开 portlet 安装向导 209。

安装向导由安装到客户端上的每个 portlet 都配备有其自身的相关联的 portlet 视图插件 202。以下问题依赖于所安装的 portlet 的类型，即其为 JSR-168 还是 WebSphere portlet：portlet 安装在哪个 portlet 服务器上，以及向导所提供的 portlet 接口扩展 214 指向哪个 portlet 接口 203a、b。

扩展了图 2b 左侧的 portlet 视图插件 202，以显示其内容，其为两个扩展：即 portlet 接口扩展 214，其通过用户定义的服务器扩展点 232 连接到服务器；以及视图扩展 210，其扩展 Eclipse 视图扩展点。视图 210 引用标准 portlet 视图 211，其从标准 portlet 视图 211 获得运行网络浏览器 207 所需的代码。网络浏览器 207 可以用来借助 HTTP 从该插件的相关 portlet 实例所在的 portlet 服务器接收网络内容。portlet 接口扩展 214 包含对于该 portlet 服务器的访问数据，并且允许视图 210 与有关门户服务器上相关 portlet 实例通信。portlet 接口扩展 214 包含其相关 portlet 的名称与 ID，并且使用这些来指向有关 portlet 接口 203（a 或 b）。

配置 portlet 服务器 204a 与 204b，从而当收到对于 portlet 内容的请求时，有关服务器（即对于 WebSphere portlet 的 WebSphere portlet 服务

器，以及对于 JSB-168 portlet 的 JSB-168 portlet 服务器）只返回该单个 Portlet 的内容，而不将该内容与来自其他 portlet 的内容聚集，并且不将该内容打包到具有标题条与控件按钮等等的 portlet 窗口中。

portlet 的视图插件使用网络浏览器以在视图窗口内显示该内容，根据 portlet 的配置数据，该插件提供该视图窗口。

如图 2c 所示，客户端计算机系统 100 包含：多个 UI 扩展，其插入 Eclipse 平台的工作台扩展点 222；以及服务器扩展，其插入项目插件 110 所声明的服务器扩展点 232。

参照图 2a 与 2b，现在描述平台的运行时间过程。首先，Eclipse 平台启动；读取其已注册插件的清单文件，构造其主存内插件注册表；并且初始化其已注册插件。由此，项目插件 110 被初始化，并且由此可以通过 Eclipse 平台使用。另外，该平台初始化任何已注册 portlet 视图插件 202。直至用户显示包含 portlet 的视图的页面或者安装新 portlet，对于门户服务器 204a、204b 没有发生任何特殊的事情。

当用户选择要显示的、包含 portlet 视图的透视图时，相关插件必须获得该 portlet 的标记内容，并且其进行调用以检索该内容。首先，其调用该 portlet 类型的实例对象，以察看相关联的 portlet 服务器 204a、204b 是否已启动。如果 portlet 服务器还未启动，则 portlet 接口 203a、203b（通过 API 205）告诉应用服务器 201 启动相关 portlet 服务器 204a、204b。当 portlet 服务器启动时，其所有已安装的 portlet 206 将被初始化。与动作所针对的 portlet 相关联的 portlet 视图插件 202 包含识别其与之相关联的 portlet 实例的参数。portlet 视图工具创建视图窗口，其中运行网络浏览器，并且通过传递 portlet 实例的 id 与所需模式借助服务器 API 对 portlet 实例进行查询。portlet 服务器返回 URL，并且视图将其传递给嵌入式浏览器 207。浏览器 207 通过 HTTP 将 URL 请求发送给应用服务器 201，并且接收包含由该 portlet 呈现的标记的响应。然后，浏览器 207 根据所接收的标记显示 portlet 内容。portlet 可以包含至其他页面、文档等等的链接，并且如果用户选择其中的任何一个，则该浏览器利用 HTTP 检索所链接的标记。

当用户与 portlet 206 交互时，浏览器 207 与 portlet 服务器 204a 或 204b 之间的交互继续进行。portlet 模式或状态改变请求，例如：编辑、配置、或帮助模式，通过用户点击 portlet 视图的皮肤或窗口框中的控件图标

之一来启动。然后，portlet 视图工具执行 UI 动作处理器 212，UI 动作处理器 212 通知动作所针对的视图的 portlet 接口 214 选择了哪个控件按钮。portlet 接口 214 向网络浏览器传递 URL，该视图应该从该 URL 请求新内容。从该 URL 接收的网络内容使浏览器能够以所请求的模式/状态显示 portlet 206。

现在参照图 3a、3b、3c，从视觉角度描述 portlet 的安装。当平台 200 启动时，将显示欢迎透视图，一般具有控件条/按钮，从该控件条/按钮可以选择诸如“打开空白门户模板”等选项，如图 3a 所示。空白门户模板为一工具，（实现为 Eclipse UI 的“透视图”扩展点的扩展 213），其允许用户创建新的透视图以及利用安装 portlet 向导安装 portlet（参看图 3b）。当选择了安装 portlet 向导 209 时，向用户显示（参看图 3c）客户端系统 100 特定文件夹 302 中 portlet 网络应用存档文件（.war）文件的文件列表 300。如果用户从该列表中选择一或多个 portlet，则向导将每个选定 portlet 的.war 文件安装到相关 portlet 服务器上。（单个.war 文件可以包含多个 portlet，并且用户可能能够选择单个.war 文件内的特定 portlet 进行安装）。当安装了每个选定 portlet 时，向导向用户显示成功安装报告，然后根据所使用的特定平台，用户可能必须重新启动客户端平台使利用新近安装的 portlet。向导还为每个所安装的 portlet 提供客户端平台视图，以后将更详细地解释这一机制。由此，这些视图被添加到平台的可用视图列表，该列表显示了所有视图扩展，并且开发人员或用户可以使用该列表来设计新的显示页面，例如新透视图。

参照图 4a，为了设计新的透视图，当启动 400 时，向用户显示菜单页面（实现为 Eclipse UI 扩展点“ActionSets”的扩展），用户可以从该页面选择并打开 402 空白门户模板。用户从模板下拉菜单中选择“显示视图”选项，并且被显示 404 可用 portlet 视图和/或客户端平台组件视图的列表。选择所列视图之一 406 将使该视图实例化，并且在当前透视图中打开。用户可以添加更多视图至正在创建的页面，并且通过诸如重新调整大小、重新定位（例如拖拉放置等等）等功能，可以在该页面上组织 410 这些视图，并且可以设计透视图，该透视图与 portlet 视图一道显示非 portlet 客户端平台组件，如图 1 所示。以与切换到门户中的新页面类似的方式，可以将所创建的透视图存储到在需要时可以被记起的预定透视图的列表中 412。当关闭透视图时，

该透视图上的试图也被关闭。

参照图 4b，当用户要求显示所存储的透视图时，从“打开透视图”选项下的下拉菜单中选择所需透视图 414。然后，平台 200 实例化为选定透视图的部分的所有插件 416，并且读取其清单文件。根据清单文件的信息，平台利用 portlet 视图工具来创建 418 每个视图窗口的框以及任何必要工具条与菜单，并且将网络浏览器嵌入每个实例化的 portlet 视图中。对于由 portlet 视图插件提供的视图，启动每个相关插件，并且每个相关插件借助嵌入式网络浏览器发送 420 对于其相关 portlet 内容的请求。该请求去向应用服务器，应用服务器将其指向相关 portlet 服务器。（在做这些时，应用服务器通过检查请求 URL 并且查找至其容器（例如 portlet 服务器）的映射表，确定处理该请求的正确容器。）然后，向用户显示透视图 422，其具有显示通过嵌入式网络浏览器接收的内容的任一 portlet 视图。

然后，用户与任一所显示视图交互。如果用户（例如）通过点击在视图中显示的 URL 链接与 portlet 视图交互，则嵌入式网络浏览器借助 HTTP 从 portlet 服务器请求具有该 URL 的内容。URL 指向单个 portlet。portlet 服务器只返回来自所指向的 portlet 的 html 内容，然后该内容由嵌入式网络浏览器显示。透视图上其余视图不更新。

参照图 5，现在描述在将 portlet 安装到应用服务器中所涉及的功能步骤。该平台配备有 portlet 安装向导，组件 209，当用户选择 502 该组件时，该组件提供用户界面，该界面使用户能够从 war 文件列表中选择特定 portlet war 文件。war（网络应用存档）文件为 portlet 的一般包裹。响应于用户选择 portlet 504，安装向导 209 根据 war 文件确定 portlet 类型，并且通过初始化相关 portlet 服务器 204a、204b（如果其还未被初始化的话），创建 portlet 接口 203 的实例。借助 portlet 实例 203a、203b，安装向导 209 指令 portlet 服务器 204a、204b 安装 portlet .war 文件 211，这导致了 portlet 服务器上 portlet 206 的新实例 506。在该过程中，portlet 服务器安装（多个）portlet 文件，并且将 portlet 细节添加到管理其 portlet 的 portlet 容器的注册表。

当成功安装了 portlet .war 文件 211 时，安装向导 209 生成 508 与该 portlet 相关联的、并且包含其自身清单文件的新 portlet 视图插件 202。清单文件列出显示标签（例如插件视图的标题），视图插件所需的控件以及扩展，

并且声明两个扩展：Eclipse UI 视图扩展点的视图扩展；以及用于扩展由项目插件 110 提供的 JSR-168 或 WebSphere portlet 扩展点 232 的 JSR-168 或 WebSphere portlet 接口扩展 214。portlet 接口扩展使 portlet 能够声明与其相关联的是哪种类型的 portlet，以及在相关 portlet 服务器组件 204a、204b 中包含的引用代码，以提供相关 portlet 服务器接口 203a、203b。视图扩展引用标准 portlet 视图代码，当向用户显示视图时，portlet 视图插件运行该标准 portlet 视图代码。通过使用这些，portlet 视图工具创建所需菜单与工具条，并且将该工具所提供的图标应用到 portlet 视图的框中。如上所述，在 Eclipse 平台中注册 510 portlet 视图插件后，用户可以将新 portlet 的视图添加到 Eclipse 透视图。

该结构与服务器端门户中 portlet 实例的不同之处在于：本系统中门户服务器 204 不将来自多个 portlet 的内容聚集到门户页面中。相反，每个 portlet 具有其自身的、可以在用户的 Eclipse 平台屏幕上显示的 portlet 视图窗口。另外，门户服务器不是通过向每个窗口添加框（包含标题与控件按钮），将每个 portlet 片断打包到 portlet 窗口中。相反，portlet 视图工具根据 portlet 视图清单文件，提供其 portlet 视图窗口的框，包含标题条与控件按钮。

对于每个已安装的 portlet，安装向导在 Eclipse 平台上安装新的 portlet 视图插件。Eclipse 平台在独立的窗口或者视图（每个都具有其自身的框、控件按钮与嵌入式网络浏览器）中显示每个 portlet 视图插件。

现在描述在 portlet 和/或客户端组件之间的数据共享机制。为了实现本发明，创建 portlet 和客户端组件，具有接收并与零或更多个其他组件共享数据的能力。在创建组件时，开发人员定义该组件内每个数据字段的数据类型（即字符、串、实数、整数等等）。开发人员还定义数据字段为输入字段、输出字段、内部字段、或输入/输出字段。指定为输入字段或输入/输出字段的数据字段可以接收来自另一组件或内容提供者的数据。类似地，指定为输出字段或输入/输出字段的数据字段可以与另一组件共享数据。指定为内部字段的数据字段不能接收或与另一组件或内容提供者共享数据。

该数据共享机制使用允许将一个组件的源字段（即输出或输入/输出字段）映射/链接到另一个组件的目的字段（即输入或输入/输出字段）的映射系统。当映射两个字段时，源字段中的数据将自动与目的字段共享。该映射

系统一般包含允许开发人员或者用户将所希望的字段链接（“连线（wire）”）在一起的用户界面。一旦映射了源与目的字段，（多个）源字段中的任何数据就将自动与目的字段共享。在提供正确适当的映射时，该映射系统首先可以确保数据可以被适当共享（即数据类型兼容，并且相应字段被适当指定为输入和/或输出字段）。如果允许共享，则可以呈现显示该新映射的图形表示。

在优选实施方式中，该映射系统被实现为上述 Eclipse 平台中的连线工具 215。如图 6 所示，连线工具 215 包含其提供给 Eclipse 平台视图扩展点的视图扩展，以及属性中介方（property broker）。

每个 portlet 与客户端组件向属性中介方注册，通过显露其数据字段表示其是否能够发布数据和/或消耗数据。属性中介方维护其可以用来映射一个组件的数据源字段（即输出或输入/输出字段）到另一个组件的目的字段（即输入或输入/输出字段）的注册表。当映射两个字段时，源字段中的数据将自动与目的字段共享。映射字段列表或“连线”可以在 XML 文档中提供，或者由用户以图形方式编辑/创建。

如上所述，在处理请求时，portlet 一般具有两个主要阶段：动作阶段与呈现阶段。动作阶段由门户服务器调用，针对被点击的、“动作所针对”的特定 portlet，处理该动作，然后对于页面上所有 portlet，调用呈现阶段。只有当 portlet 上某物被点击时，才调用动作阶段，而不是首次显示页面时初次调用。

关于属性中介方通信的开始与结束的通知由属性中介方直接转发给合作方组件，属性中介方不知道当前组件的确切性质。通过这种方式，所有属性提供者都以一致的方式发挥作用，并且所有属性提供者都以相同的语义接收相同的通知。在不修改 portlet 容器的前提下之所以可能如此的原因在于客户端门户的性质，其中客户端平台完全控制了每个 portlet 的用户交互与显示，因此在必要时能够发出正确的事件。

参加属性中介的组件（portlet 或插件）都不允许开始呈现内容，直至完成所有的属性中介。这是为了确保组件不会呈现陈旧的数据。开始不能呈现数据，因为其可能被属性中介方改变。

为了解决该问题，可以将称为 BeginEventPhase 与 EndEventPhase 的新生命周期方法引入 WebSphere portlet API。在开始呈现 portlet 之前，保证完成这些生命周期阶段，由此给出了完成属性中介的时机。然而，因为

JSR-168 portlet API 不包含 BeginEventPhase 与 EndEventPhase 生命周期方法，所以需要一新机制来协调用于客户端属性共享的中介与呈现的时序。

为了达到这一点，引入称为 communicaitonPhaseBegun 与 communicaitonPhaseEnded 的两个方法。属性中介方在 portlet 插件以及在该页面上具有视图的每个其他插件上调用这两个方法。当调用 communicaitonPhaseBegun 方法时，插件设置标志，例如信号量，以通知自身不要开始呈现，因为正在进行属性中介。当调用 communicaitonPhaseEnded 时，插件清除该标志，以通知自身现在可以进行内容呈现。然后，该插件能够通过向其传递 URL 来更新嵌入式浏览器。

注意：当进行属性中介时，嵌入式浏览器中的所有 URL 点击都被 portlet 插件截获，而不仅仅是改变 portle 模式的点击。因为任何交互都可能激活属性中介周期，所以这是必要的。

为了截获这些用户动作，portlet 插件在嵌入式浏览器中注册事件监听器。诸如 Mircrosoft Internet Explorer 等嵌入式浏览器提供该事件监听功能。

参照图 9，该图描述当 portlet 为检测到客户端交互的组件时所涉及的方法步骤。当用户在 portlet 上执行动作 901 时，嵌入式浏览器检测用户点击，并且抛出事件通知，该事件通知由与其相关联的 portlet 插件拾取。然后，该 portlet 插件通知属性中介方要处理新请求。然后，属性中介方通知 902 页面上的每个组件 communicaitonPhaseBegun。动作所针对的 portlet 组件现在允许其浏览器对 URL 进行动作 903，并且浏览器将 URL 发送给动作所针对的 portlet，由此触发其动作处理与呈现阶段两者。即使在插件中设置了标志，也允许动作所针对的 portlet 呈现；这之所以可以是因为在动作阶段已完成其属性共享。在其动作阶段过程中，动作所针对的 portlet 进行调用以通知属性中介方所改变的属性。进而属性中介方跟随其连线，并且通知 904 属性改变的目标 portlet，由此允许这些 portlet 更新其数据。这些 portlet 可以处理其接收的数据属性的新值，并且产生另一数据属性的新值。如果该另一数据属性为共享属性，则 portlet 将通知属性中介方该属性的新值。属性中介方将再次利用连线通知所有链接的 portlet（或者其他链接的组件）这些属性改变。

借助事件监听器，使与动作所针对的 portlet 相关联的 portlet 视图插

件获知已经通过嵌入式浏览器完成了动作所针对的 portlet 的呈现。然后，该 portlet 视图插件通知属性中介方请求已经完成，并且属性中介方通过调用方法 communicaitonPhaseEnded，通知 905 每个在显示页面上具有视图的其他组件通信阶段已经结束。然后，每个其他组件刷新其内容 906，以反映更新后的数据值，或者在修改方式下，只有那些改变了属性的组件才刷新其内容。对于每个 portlet 视图插件，刷新涉及其嵌入式浏览器借助 HTTP 与其相关 portlet 通信，并且 portlet 以新近呈现的内容响应，该新近呈现的内容反映属性中介所产生的所有改变。

在优选实施方式中，每个组件都能在其自身私有存储空间中读取并写入数据。当修改源字段的数据值时，“源”组件将消息发送给属性中介方。属性中介方访问为源字段定义的映射，并且将消息发送给与源字段共享的每个“目的”组件。所述消息包含目的字段的标识，以及目的字段的更新后数据值。

人们可能希望在具有不同数据类型的字段之间共享数据。为了解决此类问题，可以提供转换系统以将共享数据从源字段数据类型/格式转换为目的字段数据类型/格式。由此，该转换系统总是允许容纳任意数目的数据类型。为了提供此类有效的数据转换，一般提供转换数据（例如，存储在数据库中），以由转换系统访问。

当必要时，属性中介方可以使用该转换系统以在源字段与目的字段数据类型之间转换数据。然后，“目的”组件将新数据值写入其自身私有存储空间，并且执行所需的所有功能。

可替换地，属性中介方可以为每个 portlet/组件提供每个共享字段的字段以及“目的”组件列表。然后，组件能够将消息直接发送给每个“目的”组件。

可替换地，可以利用共享存储器实现组件之间的数据共享。在这种情况下，组件可以将所有源字段写入共享存储器位置。随后，另一组件为映射的目的字段从该共享存储器位置读取数据。所述读取与写入可以由属性中介方和/或转换系统执行，从而在不要求 portlet 或客户段组件中附加功能的情况下，进行所需的通信与数据类型转换。

图 7 显示网络搜索 Eclipse 透视图 700，其包含五个 portlet 的视图，即搜索条 portlet 701、搜索历史 portlet 702、以及三个搜索引擎 portlet 704、705、706，这三个搜索引擎分别（例如）称为“Youwho”、“Ask Bertie”、

以及“Goggles”。该网络搜索透视图还包含由连线工具的视图扩展 602 提供的连线工具视图 703。搜索条 portlet 701 包含：搜索条 707，用户可以向其输入搜索词；以及搜索按钮 708，用来表示用户希望搜索所输入的词。搜索历史 portlet 包含用来显示搜索历史的显示区域（在图 7 中为空），并且包含“清除历史”按钮，用户通过该按钮可以清除显示区域。图 7 所示透视图为透视图的基本版本，当用户首次选择透视图时，在用户与页面上任何视图交互之前，显示该基本版本。当没有进行搜索时，每个搜索引擎 portlet 视图都显示消息“没有结果要显示”。

连线工具视图 703 提供允许开发人员或用户将所希望的字段链接在一起的用户界面。在图 8A 中，作为连线工具的 Eclipse 视图，显示了以图形方式映射/链接一或多个源字段到一或多个目的字段的示范性图形用户界面。利用该用户界面以及图 7A 的示例透视图，首先从视觉角度描述利用连线工具 600 将搜索引擎 portlet 与搜索历史 portlet “连线”到搜索条 portlet 的方法。

如图 8A 所示，显示开发来共享属性的页面上的每个组件，其能够共享的属性（字段）在该组件名称下缩进显示。由此，列出了每个搜索引擎 portlet，并且每个搜索引擎 portlet 都具有称为“searchcommand”的属性。搜索历史 portlet 列为“网络搜索历史”，并且具有两个数据字段：即“repeatsearchterm”与“lastsearchterm”。搜索条 portlet 列为“网络搜索条”，并且具有一个数据字段“searchtarget”。

虽然未显示，但是根据属性的 I/O 规格，可以不同格式在连线工具视图中显示每个属性。例如，输入字段、输出字段、以及输入/输出字段可以特有颜色显示，包含特有形状或标记等等。

当用户选择所列属性时，例如 searchtarget 数据字段，可以显示下拉菜单 800。该菜单给予用户以下选项：添加至选定属性的新连线，即指定选定属性为新映射中的目的字段，或者添加来自选定属性的新连线，即指定选定属性为源字段。为了创建将搜索条连接到 Youwho 搜索 portlet 的连线，用户选择创建来自 searchtarget 属性的新连线的选项。然后，显示称为“创建新连线”的新窗口 810（参看图 8B）。该窗口提供其他 portlet、插件及其属性的列表，从这些其他 portlet、插件及其属性用户可以选择 Youwho 搜索 portlet 的 searchcommand 属性。这就导致将来自搜索条的 searchtarget 串的连接到 Youwho 搜索 portlet 的 searchcommand 串，其中 searchtarget 串

为源字段，searchcommand 串为新连线的目的字段。图 8c 显示连线工具视图的显示，其通过利用所连线的属性下的箭头显示连接。

图 7B 显示通过在网络搜索条中输入词“IBM”并且选择搜索按钮会显示的结果。如图所示，Youwho 搜索 portlet 显示利用 Youwho 搜索引擎对词 IBM 的搜索结果。其他两个搜索引擎 705、706 保持其原来的“没有结果要显示”状态。

现在相对于上面参照图 7A 至 7C 与图 8A 至 8C 所述的实施方式，详细描述属性中介方的功能。如上所述，属性中介方维护注册表，属性中介方使用该注册表来将待映射/链接的一个组件的数据源字段映射/链接到另一组件的目的字段。当用户利用连线工具设置两个组件之间的新连线时，该连线的源组件注册其源属性，目的组件注册其目的属性。在图 8A 至 8C 的例子中，属性中介方存储从网络搜索条 portlet 的 searchtarget 属性至 Youwho 搜索 portlet 的 searchcommand 属性的映射。当用户在搜索条中输入搜索词后，点击搜索按钮表示网络搜索条 portlet 中的动作。与搜索条 portlet 相关联的 portlet 插件（未显示）截获该用户动作，并且获得 URL，然而还不允许网络搜索条 portlet 允许执行动作。

网络搜索条 portlet 告诉属性中介方 604 通信开始。然后，属性中介方告诉在透视图上显示视图的所有 portlet 与客户端组件正在进行通信。然后，网络条 portlet 的相关插件将该 URL 传递给其嵌入式网络浏览器，该嵌入式网络浏览器利用 HTTP 联系相关 portlet 服务器，然后，该相关 portlet 服务器处理所请求的动作。在处理动作的过程中，网络搜索条 portlet 通知属性中介方其 searchtarget 属性已经改变，并且提供 searchtarget 属性新值给属性中介方。中介方利用其注册表，并且通知链接到 searchtarget 属性的所有 portlet 或客户端组件该属性的新值。进而，所链接的 portlet 或所链接的客户端组件可以通过中介方触发结果的属性改变。在图 8A 至 8C 的例子中，属性中介方通知 Youwho 搜索 portlet searchtarget 属性新值，然后 Youwho 搜索 portlet 相应的修改其自身的 searchcommand 属性。当网络搜索条 portlet 完成动作时，该 portlet 发送新的刷新后的内容，以在其视图窗口中由嵌入式网络浏览器显示；然后，其相关 portlet 插件通知中介方通信阶段完成。然后，中介方通知页面上每个注册组件通信阶段已经完成。然后，刷新在页面上出现的每个其他 portlet/组件视图。对于每个 portlet，这通

过其相关 portlet 插件告诉其嵌入式网络浏览器刷新来完成。然后，在 portlet 视图窗口中呈现从相关 portlet 服务器接收回来的、更新后的内容。

在通信阶段过程中，Youwho portlet 通过属性中介方获得其 searchtarget 属性新值，即搜索词“IBM”。在其呈现阶段过程中，Youwho portlet 在网络上查询其相关搜索引擎，以获得对于词“IBM”的搜索结果。然后，从该搜索引擎接收的结果由 portlet 呈现，并且发送给嵌入相关 Youwho portlet 视图中的网络浏览器，以在屏幕上显示。

在如上所述地创建了搜索条 portlet 的 searchtarget 属性至其他两个搜索 portlet 中每一个的 searchcommand 属性的连线之后，所有这三个搜索 portlet 通过搜索用户在搜索条 707 中输入的字符串，将对用户在搜索条 portlet 中选择搜索按钮进行反应，如图 7C 所示。

最后，可以连线搜索历史 portlet 702，以记录通过搜索条 portlet 进行的搜索。为了达到这一点，将网络搜索条的 searchtarget 属性连线到历史 portlet 的 lastsearchterm 属性，lastsearchterm 属性出现在其显示的先前搜索词列表的顶部。由此，当刷新搜索历史 portlet 时，在输入词或词组至网络搜索条之后，新的 lastsearchterm 值将被添加到其搜索词列表的顶部，先前的搜索词相应地向列表下方顺序移动。在图 7D 的例子中，搜索引擎在先前搜索词列表中列出两个词 IBM 与 Microsoft。

通过利用搜索历史 portlet 的 repeatsearchterm 属性，可以实现搜索历史 portlet 的进一步特征。该 repeatsearchterm 属性连线到网络搜索条 portlet 中的 searchtarget 属性，如上所述，该 searchtarget 属性连线到每个搜索引擎 portlet 的 searchcommand 属性。通过用户点击历史 portlet 所显示的搜索历史中的项目，例如在在图 7D 的例子中，在词“IBM”或“Microsoft”上点击，重置 repeatsearchterm 属性。这表示请求由所连线的搜索引擎重复执行对于选定的（多个）搜索词的搜索。

替换图 8A、8B、8C 所示的用户界面，将一或多个源字段映射到一或多个目的字段的用户界面可以（例如）包含具有源侧与目的侧的表。可以使用下拉菜单以确保只选择有效的连接。例如，在选择源 portlet/组件之后，定义为输出或输入/输出字段的字段可以在源字段下拉菜单中列出。可以类似方式选择（多个）目的字段。当组件只包含对于选择有效的单个字段时（即只包含一个输入字段的天气 portlet，即城市列表），则在指定该组件之后，可以

替用户自动选择该字段。类似地，当在门户页面上只包含两个 portlet 时，一旦将一个 portlet 选定为源，则可以将另一 portlet 选定为目的。不管怎样，如上所述，一旦将源字段映射到目的字段，源字段中的任何数据将自动与目的字段共享。

虽然连线所述组件的方法指连线 portlet，但是也可以利用上述属性中介方机制，将客户端组件的属性连线到 portlet 的属性，以及连线到其他客户端组件的属性。

只要本发明的实施方式是可实现的，至少部分地利用软件控制的可编程处理设备，例如微处理器、数字信号处理器、或者其他处理设备、数据处理装置或系统，就可以理解用来配置可编程设备、装置、或系统以实现上述方法的计算机程序被认为是本发明的一个方面。该计算机程序可以实现为源代码，或者经过编译，以在处理设备、装置、或系统上实现，或者实现为（例如）目标代码。另外，包含这样的计算机程序的制造物也被认为是本发明的一个方面。

例如，可以在诸如一或多个 DVD/CD-ROM 和/或软盘等介质上提供用来实现各种功能或者传送信息的计算机程序，然后将其存储在硬盘上。也可以在远程通信介质上提供可以由数据处理系统实现的程序，例如通过远程通信网络和/或因特网，以及实现为电子信号。对于作为无线终端在射频电话网络上运行的数据处理系统，该远程通信介质可以是承载表示所述计算机程序与数据的适当编码信号的射频载波。可选地，该载波可以是光纤链路的光载波，或者远程通信系统的任何其他适当的载波介质。

本领域技术人员可以理解：虽然针对上述实例实施方式描述了本发明，但是本发明不限于此，并且存在落入本发明范围内的许多可能的修改与变动。

本发明的范围包含任何此处所述的任何新颖特征或者特征组合。在此本申请人提请注意在本申请或者任何从其导出的进一步申请的处理的过程中，可能会形成对这些特征或者特征组合的新权利要求。具体地，参照附图，来自从属权利要求的特征可能与独立权利要求的特征组合，并且来自各个独立权利要求的特征可能以任意适当的方式组合，而不是只有权利要求所列出的特定组合。

为避免疑问，本说明书与权利要求中所用的名词“包含”不应理解为表示“只包含”。

附件 1

Eclipse Platform
Plug-in Manifest

Version 0.90 - Last revised March 15, 2001

The manifest markup definitions below make use of various naming tokens and identifiers. To eliminate ambiguity, here are some production rules for these [are referenced in text below]. In general, all identifiers are case-sensitive.

```
SimpleToken := sequence of characters from ('a-z','A-Z','0-9')
ComposedToken := SimpleToken | (SimpleToken '.' ComposedToken)
JavaClassName := ComposedToken
PlugInId := ComposedToken
PlugInPrereq := PlugInId | 'export' PlugInId
ExtensionId := SimpleToken
ExtensionPointId := SimpleToken
ExtensionPointReference := ExtensionPointID | (PlugInId '.' ExtensionPointId)
```

The remainder of this section describes the plugin.xml file structure as a series of DTD fragments. File plugin.dtd presents the DTD definition it its entirety.

```
<?xml encoding="US-ASCII"?>
<!ELEMENT plugin (requires?, runtime?, extension-point*, extension*)>
<!ATTLIST plugin
  name                CDATA #REQUIRED
  id                  CDATA #REQUIRED
  version             CDATA #REQUIRED
  vendor-name         CDATA #IMPLIED
  class               CDATA #IMPLIED
>
```

The <plugin> element defines the body of the manifest. It optionally contains definitions for the plug-in runtime, declarations of any new extension points being introduced by the plug-in, as well as configuration of functional extensions (configured into extension points defined by other plug-ins, or introduced by this plug-in). <plugin> attributes are as follows:

- **name** - user displayable (translatable) name for the plug-in
- **id** - unique identifier for the plug-in.
    - To minimize potential for naming collisions, the identifier should be derived from the internet domain id of the supplying vendor (reversing the domain name tokens and appending additional name

tokens separated by dot [.]). For example, vendor ibm.com could define plug-in identifier com.ibm.db2

- [production rule: PlugInId]

- **version** - plug-in version number. See org.eclipse.core.runtime.PluginVersionIdentifier for details. Plug-in version format is **major.minor.service**. Change in the major component is interpreted as an impompatible version change. Change in the minor component is intepreted as a compatible version change. Change in the service component is intepreted as *cumulative* service applied to the minor version.
- **vendor-name** - user-displayable name of the vendor supplying the plug-in.
- **class** - name of the plug-in class for this plug-in. The class must be a subclass of org.eclipse.core.runtime.Plugin.

The XML DTD construction rule *element** means zero or more occurrences of the element; *element*? means zero or one occurrence of the element; and *element*+ (used below) means one or more occurrences of the element. Based on the <plugin> definition above, this means, for example, that a plug-in containing only a run-time definition and no extension point declarations or extension configurations is valid (for example, common libraries that other plug-ins depend on). Similarly, a plug-in containing only extension configurations and no runtime or extension points of its own is also valid (for example, configuring classes delivered in other plug-ins into extension points declared in other plug-ins).

The <requires> section of the manifest declares any dependencies on other plug-ins.

```
<!ELEMENT requires (import+)>
<!ELEMENT import EMPTY>
<!ATTLIST import
 plugin              CDATA #REQUIRED
 version             CDATA #IMPLIED
 match               (exact | compatible) "compatible"
 export              (true | false) "false"
>
```

Each dependency is specified using an <import> element. It contains the following attributes:

- **plugin** - identifier of the required plug-in
- **version** - optional version specification

- **match** - version matching rule. Ignored if version attribute is not specified. Determines whether the dependency is satisfied only with a plug-in of the specified version (possibly with additional service applied), or the dependency can be satisfied with any compatible version (including a more recent minor version of the plug-in)
- **export** - specifies whether the dependent plug-in classes are made visible (are (re)exported) to users of this plug-in. By default, dependent classes are not exported (are not made visible)

The <runtime> section of the manifest contains a definition of one or more libraries that make up the plug-in runtime. The referenced libraries are used by the platform execution mechanisms (the plug-in class loader) to load and execute the correct code required by the plug-in.

```
<!ELEMENT runtime (library+)>
<!ELEMENT library (export*)>
<!ATTLIST library
  name              CDATA #REQUIRED
>
<!ELEMENT export EMPTY>
<!ATTLIST export
  name              CDATA #REQUIRED
>
```

The <runtime> element has no attributes.

The <library> elements collectively define the plug-in runtime. At least one <library> must be specified. Each <library> element has the following attributes:

- **name** - string reference to a library file or directory containing classes (relative to the plug-in install directory). Directory references must contain trailing file separator.

Each <library> element can specify which portion of the library should be exported. The export rules are specified as a set of export masks. By default (no export rules specified), the library is considered to be private. The <export> elements have the following attributes:

- **name** - specifies the export mask. Valid values are:
  - * - indicates all contents of library are exported (public)
- **package-name.*** - indicates all classes in the specified package are exported. The matching rules are same as in Java import statement.
- **class-name** - fully qualified java class name

The platform's architecture is based on the notion of configurable extension points. The platform itself predefines a set of extension points that cover the task of extending the platform and desktop (for example, adding menu actions, contributing embedded editor). In addition to the predefined extension points, each supplied plug-in can declare additional extension points. By declaring an extension point the plug-in is essentially advertising the ability to configure the plug-in function with externally supplied extensions. For example, the Page Builder plug-in may declare an extension point for adding new Design Time Controls (DTCs) into its builder palette. This means that the Page Builder has defined an architecture for what it means to be a DTC and has implemented the code that looks for DTC extensions that have been configured into the extension points.

```
<!ELEMENT extension-point EMPTY>
<!ATTLIST extension-point
  name              CDATA #REQUIRED
  id                CDATA #REQUIRED
  schema            CDATA #IMPLIED
>
```

The <extension-point> element has the following attributes:

- **name** - user-displayable (translatable) name for the extension point
- **id** - simple id token, unique within this plug-in. The token cannot contain dot (.) or whitespace.
    - [production rule: ExtensionPointId]
- **schema** - schema specification for this extension point. The exact details are being defined as part of the Plug-In Development Environment (PDE). The schema is currently not used at runtime. The reference is a file name relative to the plug-in installation location.

Actual extensions are configured into extension points (predefined, or newly declared in this plug-in) in the <extension> section. The configuration information is specified as well-formed XML contained between the <extension> and </extension> tags. The platform does not specify the actual form of the configuration markup (other than requiring it to be well-formed XML). The markup is defined by the supplier of the plug-in that declared the extension point.

The platform does not actually interpret the configuration markup. It simply passes the configuration information to the plug-in as part of the extension point processing (at the time the extension point logic queries all of its configured extensions).

```
<!ELEMENT extension ANY>
<!ATTLIST extension
  point             CDATA #REQUIRED
  id                CDATA #IMPLIED
  name              CDATA #IMPLIED
>
```

The <extension> element has the following attributes:

- **point** - reference to an extension point being configured. The extension point can be one defined in this plug-in or another plug-in
    - [production rule: ExtensionPointReference]
- **id** - optional identifier for this extension point configuration instance. This is used by extension points that need to uniquely identify (rather than just enumerate) the specific configured extensions. The identifier is specified as a simple token unique within the definition of the declaring plug-in. When used globally, the extension identifier is qualified by the plug-in identifier
    - [production rule: ExtensionId]
- **name** - user-displayable (translatable) name for the extension

Important: The content of the <extension> element is declared using the ANY rule. This means that any well-formed XML can be specified within the extension configuration section (between <extension> and </extension> tags).

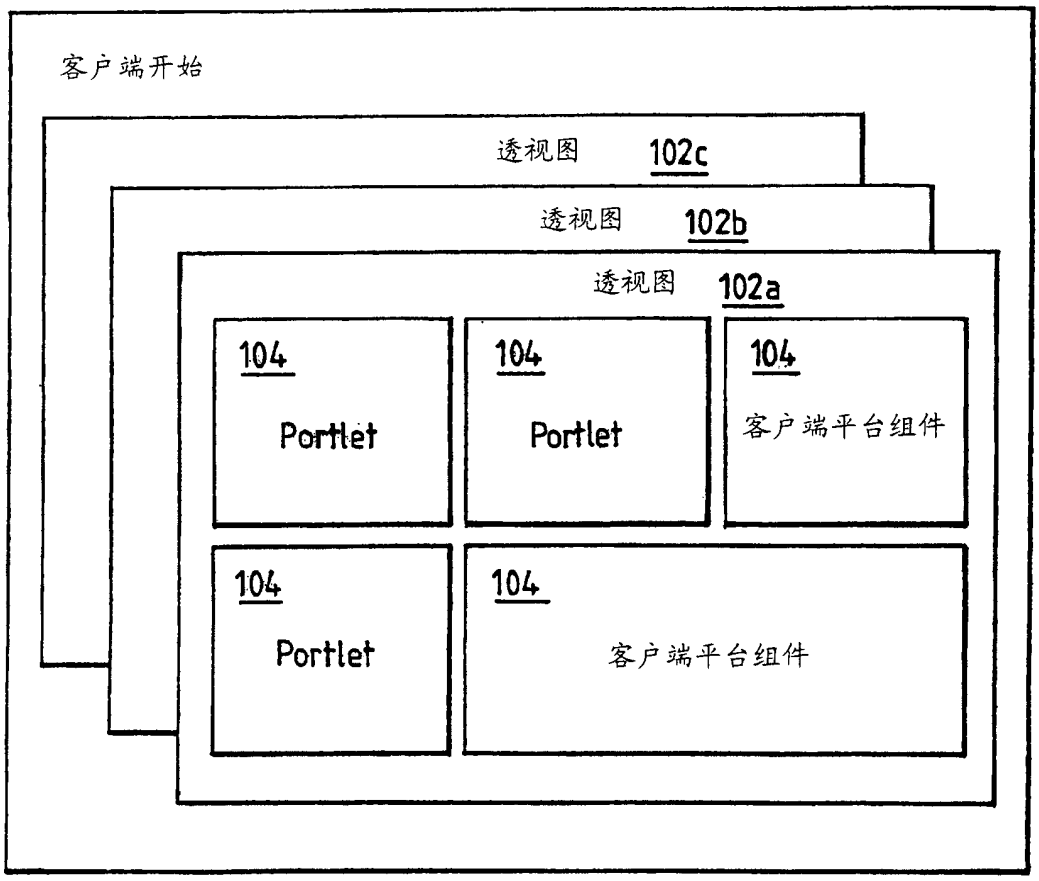

客户端开始
透视图 102c
透视图 102b
透视图 102a
104
Portlet
104
Portlet
104
客户端平台组件
104
Portlet
104
客户端平台组件

图 1

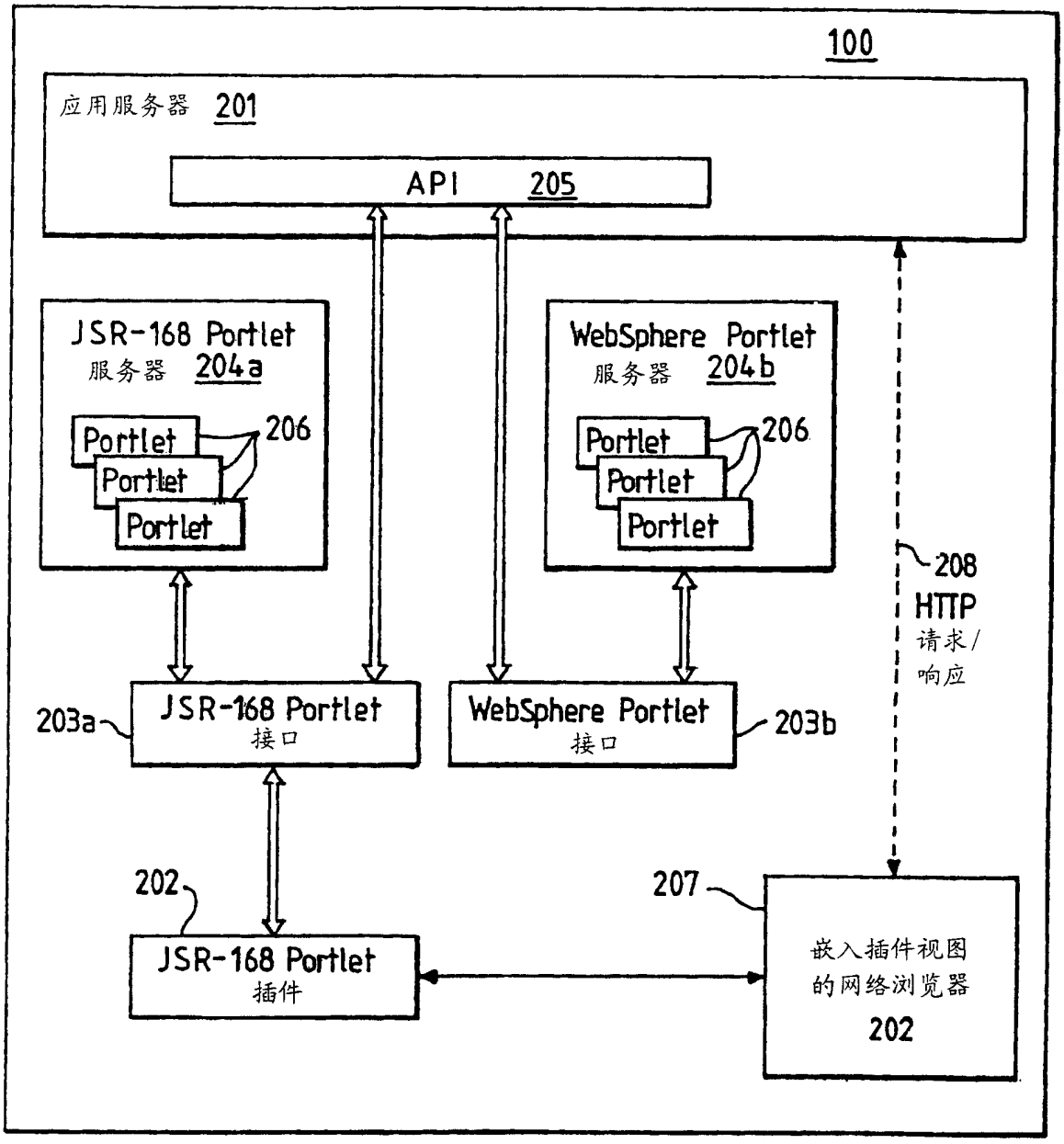

100
应用服务器 201
API 205
JSR-168 Portlet
服务器 204a
Portlet
Portlet
Portlet
206
WebSphere Portlet
服务器 204b
Portlet
Portlet
Portlet
206
208
HTTP
请求/
响应
203a
JSR-168 Portlet
接口
WebSphere Portlet
接口
203b
202
JSR-168 Portlet
插件
207
嵌入插件视图
的网络浏览器
202

图 2A

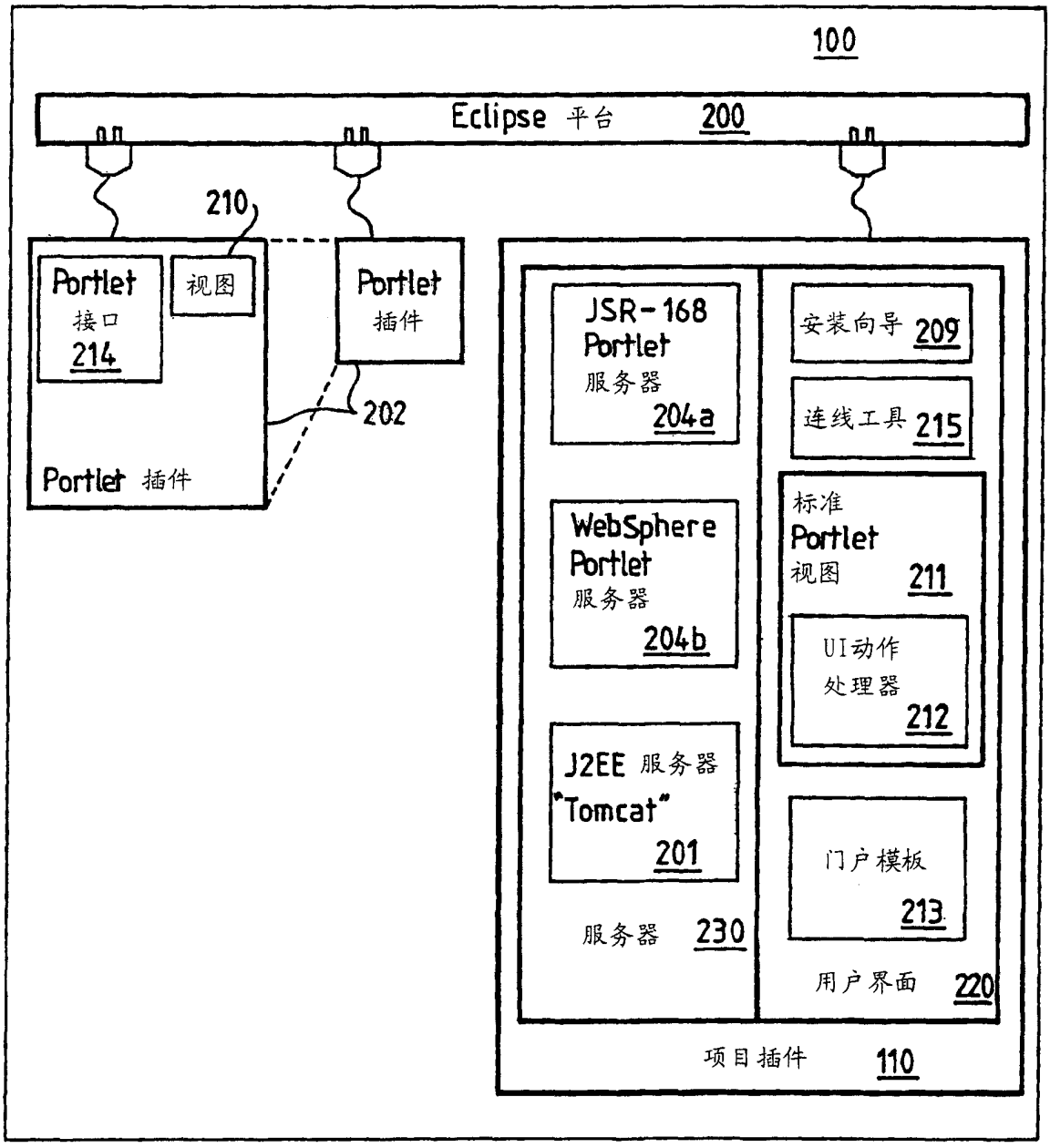

100
Eclipse 平台 200
210
Portlet 接口 214
视图
Portlet 插件
202
Portlet 插件
JSR-168 Portlet 服务器 204a
WebSphere Portlet 服务器 204b
J2EE 服务器 "Tomcat" 201
服务器 230
安装向导 209
连线工具 215
标准 Portlet 视图 211
UI动作处理器 212
门户模板 213
用户界面 220
项目插件 110

图 2B

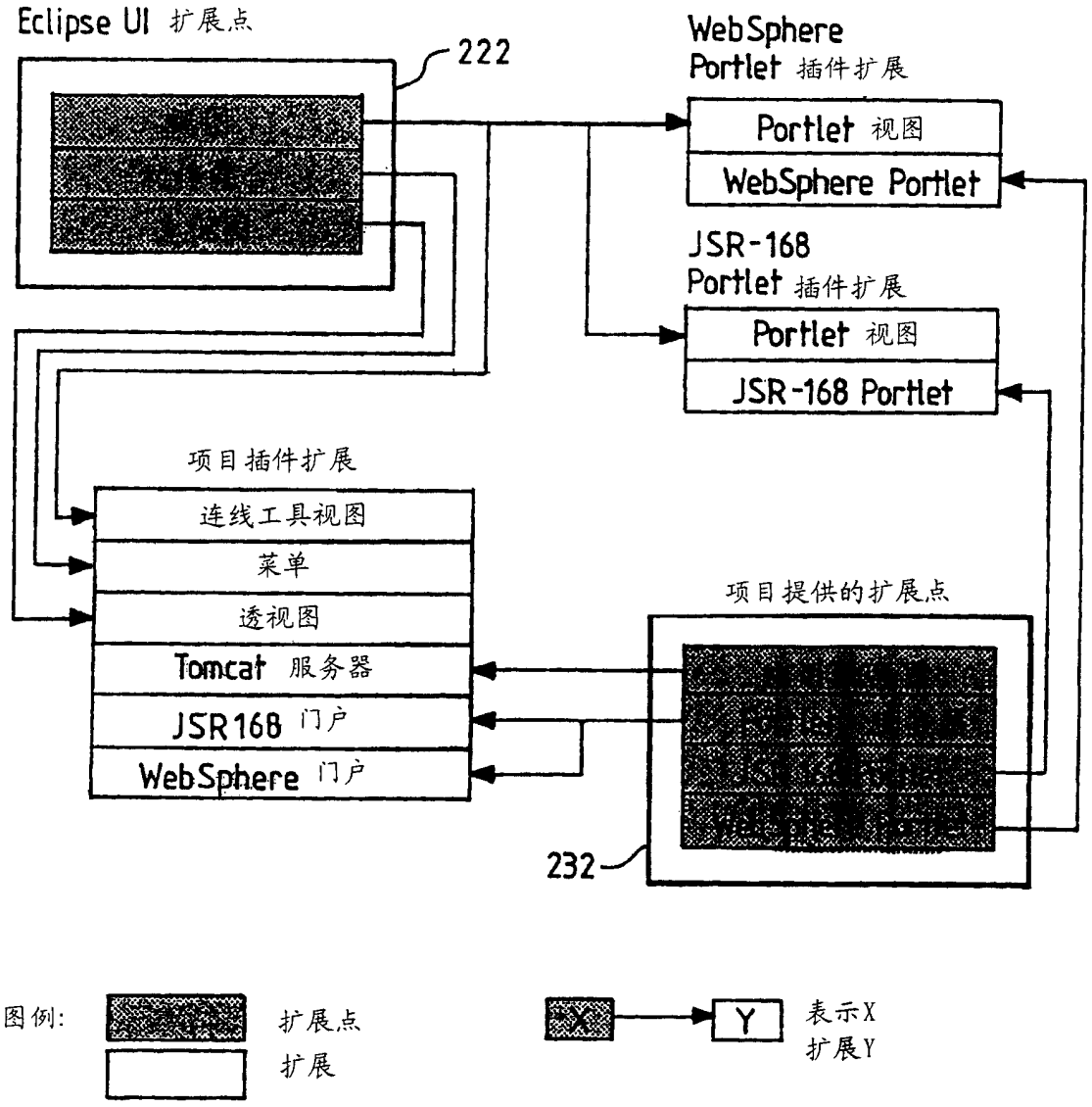

Eclipse UI 扩展点
222
WebSphere
Portlet 插件扩展
Portlet 视图
WebSphere Portlet
JSR-168
Portlet 插件扩展
Portlet 视图
JSR-168 Portlet
项目插件扩展
连线工具视图
菜单
透视图
Tomcat 服务器
JSR168 门户
WebSphere 门户
项目提供的扩展点
232
图例:
扩展点
扩展
X
Y
表示X
扩展Y

图 2C

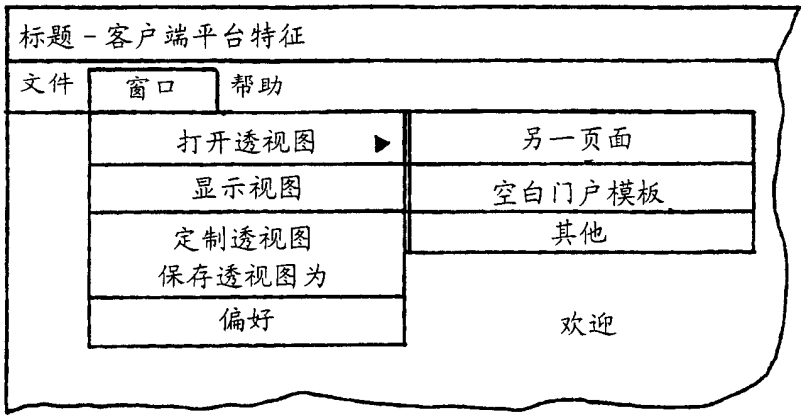

标题－客户端平台特征
文件
窗口
帮助
打开透视图
显示视图
定制透视图
保存透视图为
偏好
另一页面
空白门户模板
其他
欢迎

图 3A

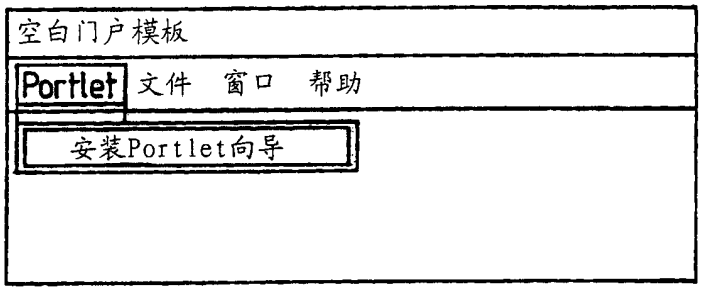

空白门户模板
Portlet
文件
窗口
帮助
安装Portlet向导

图 3B

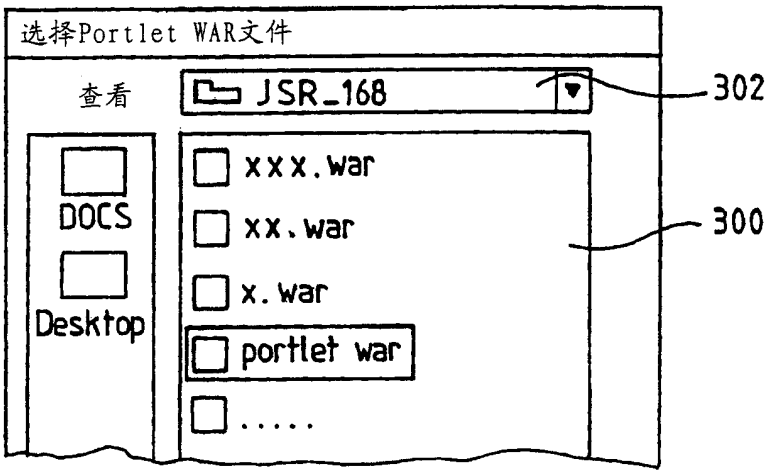

选择Portlet WAR文件
查看
JSR_168
302
DOCS
Desktop
xxx.war
xx.war
x.war
portlet war
.....
300

图 3C

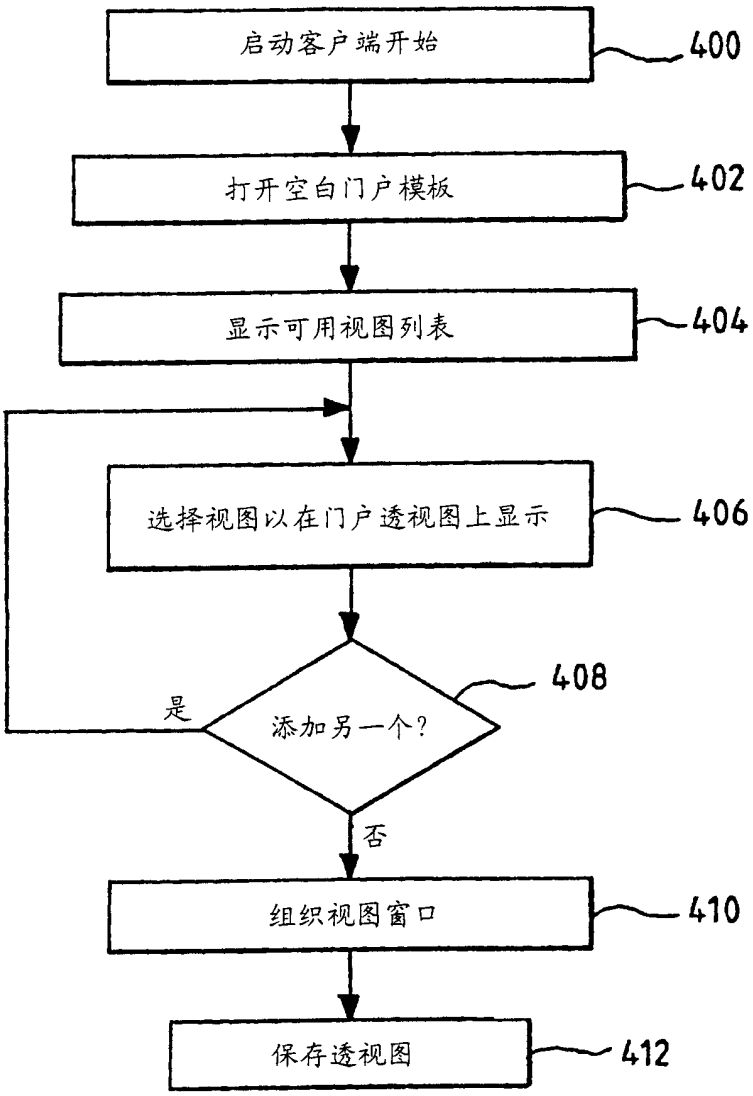

启动客户端开始
400
打开空白门户模板
402
显示可用视图列表
404
选择视图以在门户透视图上显示
406
添加另一个?
408
是
否
组织视图窗口
410
保存透视图
412

图 4A

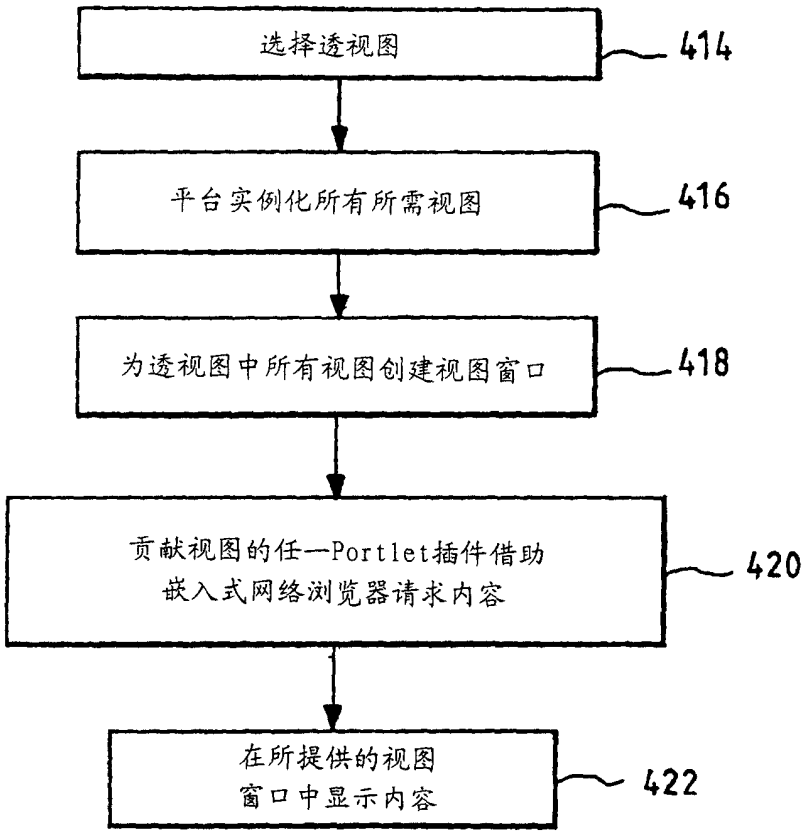

选择透视图
414
平台实例化所有所需视图
416
为透视图中所有视图创建视图窗口
418
贡献视图的任一Portlet插件借助
嵌入式网络浏览器请求内容
420
在所提供的视图
窗口中显示内容
422

图 4B

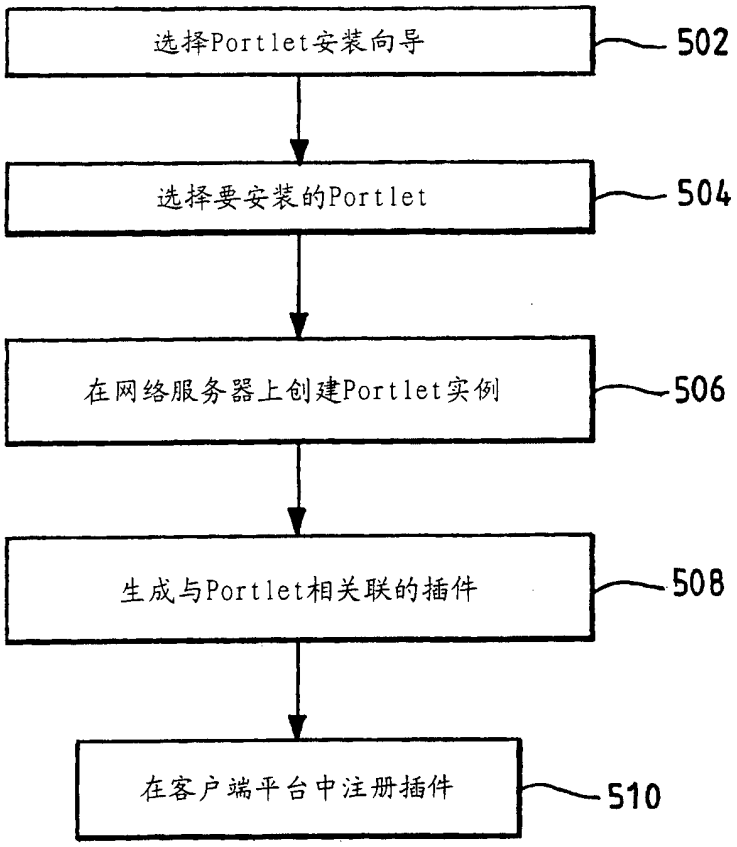

选择Portlet安装向导
502
选择要安装的Portlet
504
在网络服务器上创建Portlet实例
506
生成与Portlet相关联的插件
508
在客户端平台中注册插件
510

图 5

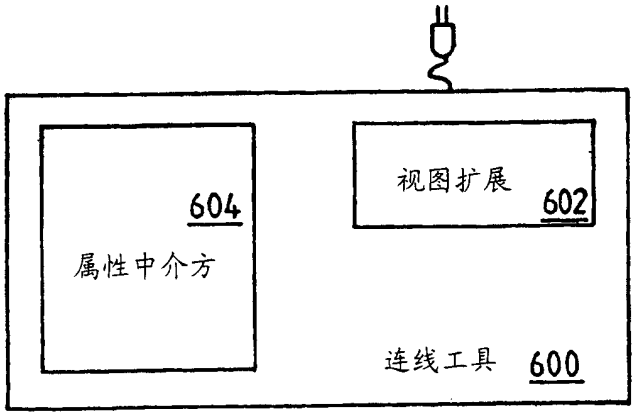

604
属性中介方
视图扩展
602
连线工具
600

图 6

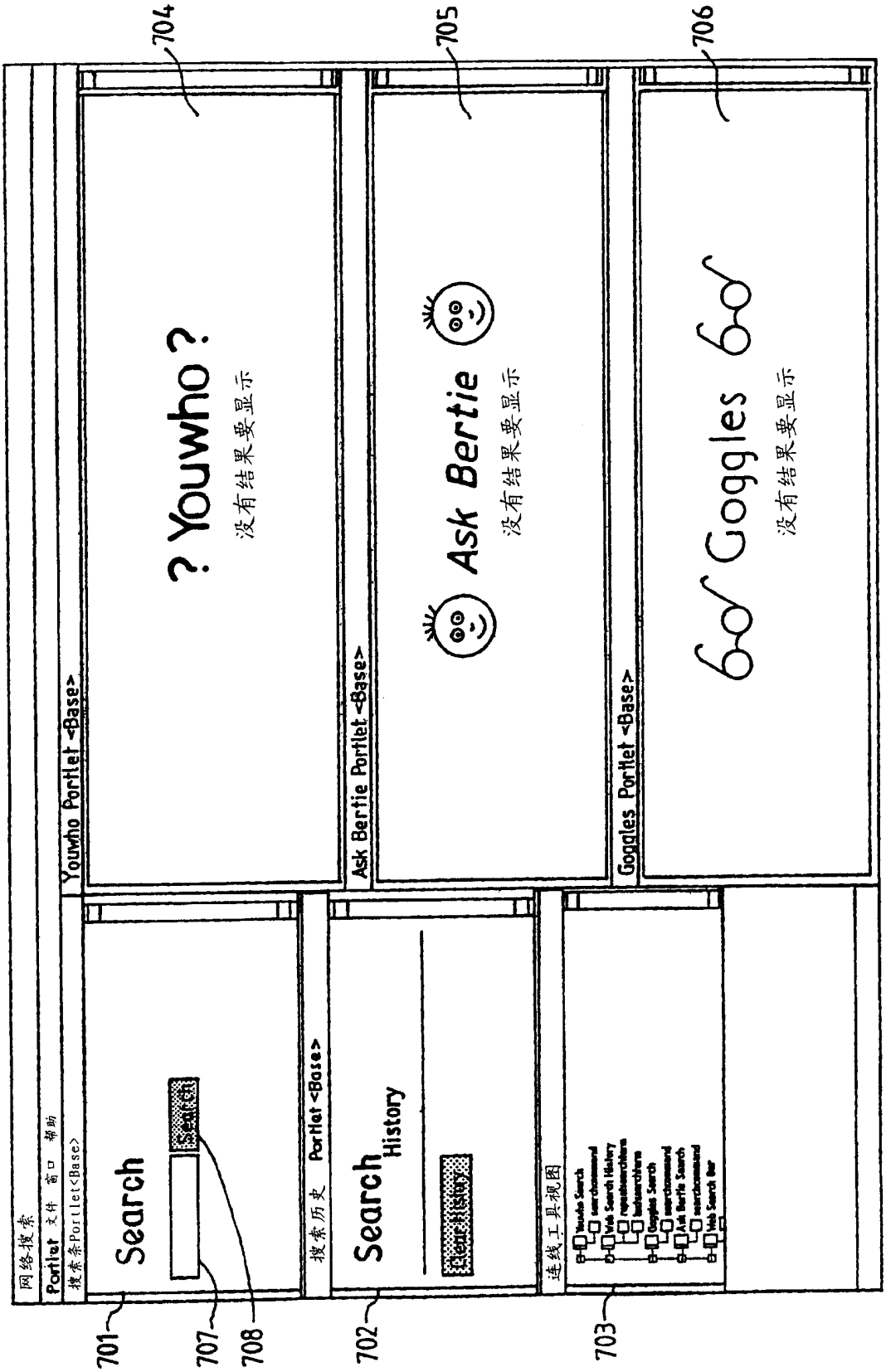

网络搜索
Portlet 文件 窗口 帮助
搜索Portlet<Base>
Search
Search
Youwho Portlet <Base>
? Youwho?
没有结果要显示
Ask Bertie Portlet <Base>
Ask Bertie
没有结果要显示
Goggles Portlet <Base>
Goggles
没有结果要显示
搜索历史 Portlet <Base>
Search History
Clear History
连线工具视图
701
702
703
704
705
706
707
708

图 7A

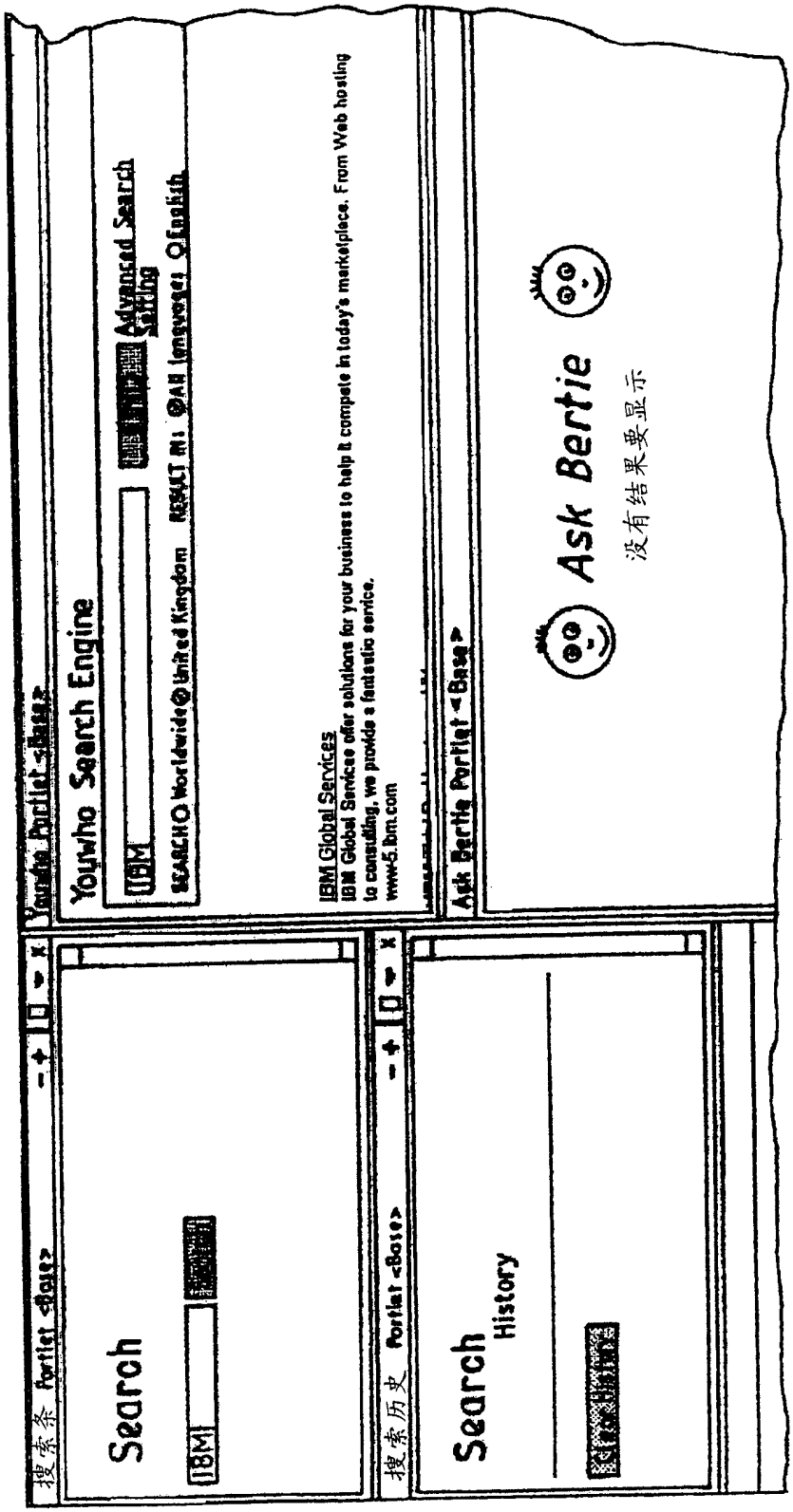

搜索条 Portlet <Base>
Search
IBM
Youwho Portlet <Base>
Youwho Search Engine
IBM
Advanced Search
Setting
SEARCH! Worldwide United Kingdom RESULT IN: All languages English
IBM Global Services
IBM Global Services offer solutions for your business to help it compete in today's marketplace. From Web hosting to consulting, we provide a fantastic service.
www5.ibm.com
搜索历史 Portlet <Base>
Search
History
Ask Bertie Portlet <Base>
Ask Bertie
没有结果要显示

图 7B

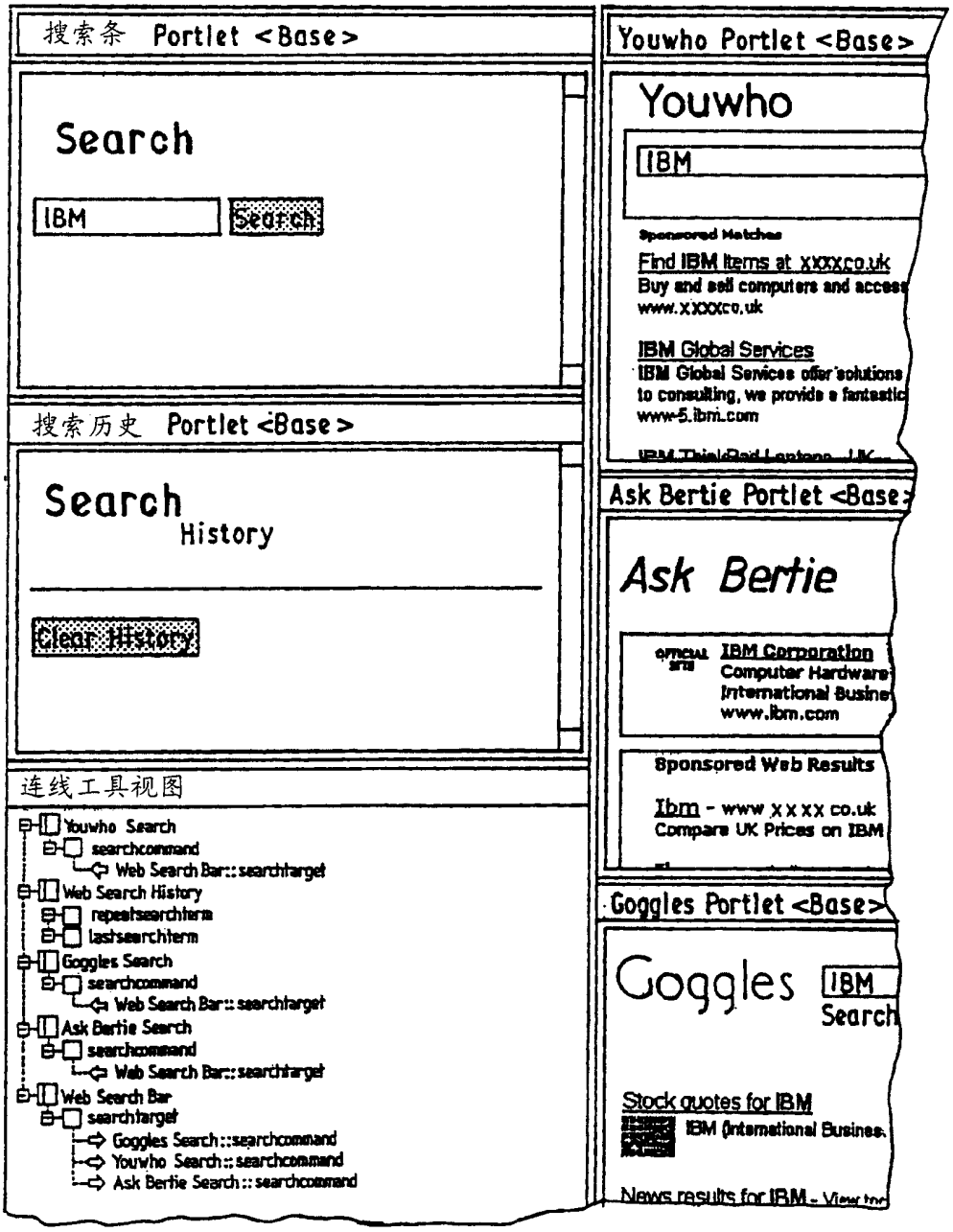

搜索条 Portlet <Base>
Search
IBM
Search
搜索历史 Portlet <Base>
Search
History
Clear History
连线工具视图
Youwho Search
searchcommand
Web Search Bar::searchtarget
Web Search History
repeatsearchterm
lastsearchterm
Goggles Search
searchcommand
Web Search Bar::searchtarget
Ask Bertie Search
searchcommand
Web Search Bar::searchtarget
Web Search Bar
searchtarget
Goggles Search::searchcommand
Youwho Search::searchcommand
Ask Bertie Search::searchcommand
Youwho Portlet <Base>
Youwho
IBM
Sponsored Matches
Find IBM Items at XXXXco.uk
Buy and sell computers and access
www.XXXXco.uk
IBM Global Services
IBM Global Services offer solutions
to consulting, we provide a fantastic
www-5.ibm.com
Ask Bertie Portlet <Base>
Ask Bertie
IBM Corporation
Computer Hardware
International Busine
www.ibm.com
Sponsored Web Results
Ibm - www XXXX co.uk
Compare UK Prices on IBM
Goggles Portlet <Base>
Goggles
IBM
Search
Stock quotes for IBM
IBM (International Busines
News results for IBM

图 7C

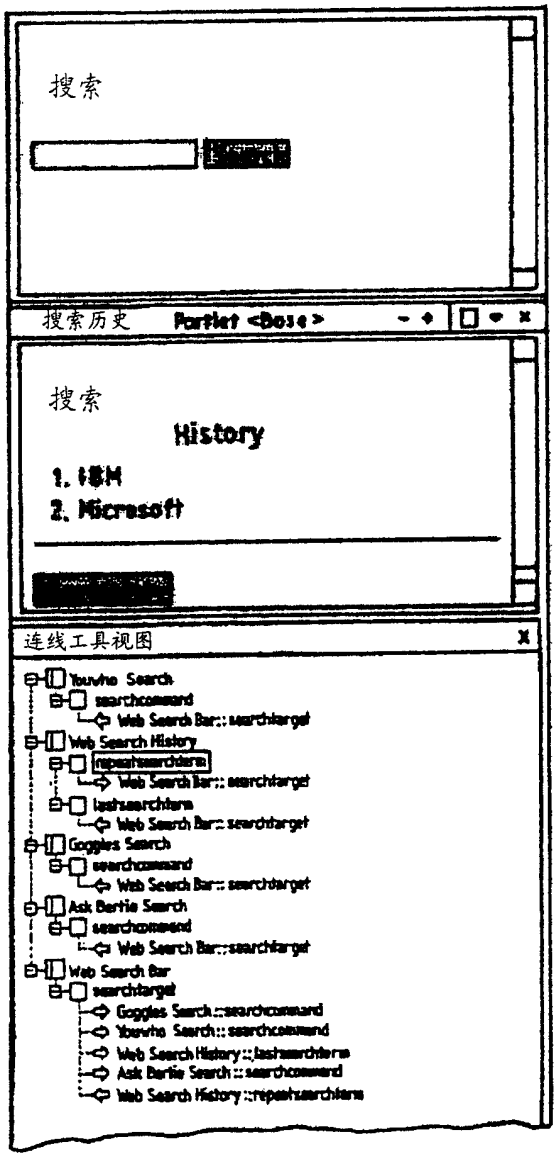

搜索
搜索历史
Portlet <Base>
搜索
History
1. IBM
2. Microsoft
连线工具视图
Youvho Search
searchcommand
Web Search Bar::searchtarget
Web Search History
repeatsearchitems
Web Search Bar::searchtarget
lastsearchitems
Web Search Bar::searchtarget
Goggles Search
searchcommand
Web Search Bar::searchtarget
Ask Barbie Search
searchcommand
Web Search Bar::searchtarget
Web Search Bar
searchtarget
Goggles Search::searchcommand
Youvho Search::searchcommand
Web Search History::lastsearchitems
Ask Barbie Search::searchcommand
Web Search History::repeatsearchitems

图 7D

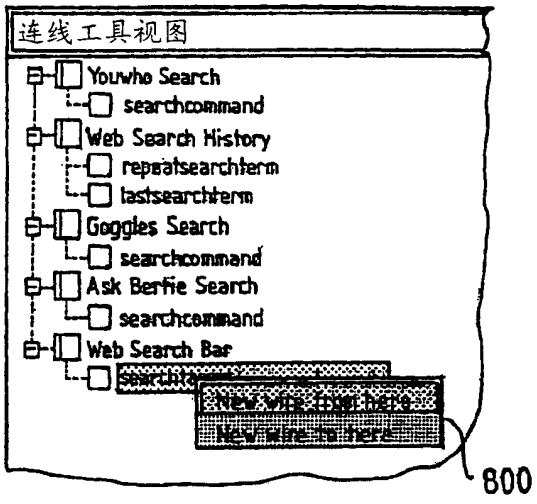

连线工具视图
Youwho Search
searchcommand
Web Search History
repeatsearchterm
lastsearchterm
Goggles Search
searchcommand
Ask Berfie Search
searchcommand
Web Search Bar
New wire from here
New wire to here
800

图 8A

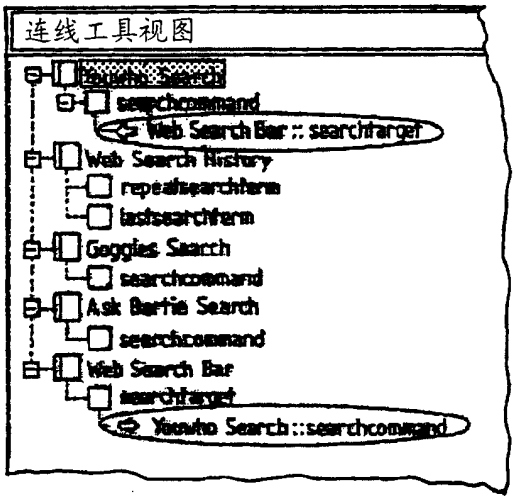

连线工具视图
searchcommand
Web Search Bar :: searchtarget
Web Search History
repeatsearchterm
lastsearchterm
Goggles Search
searchcommand
Ask Berfie Search
searchcommand
Web Search Bar
searchtarget
Youwho Search::searchcommand

图 8B

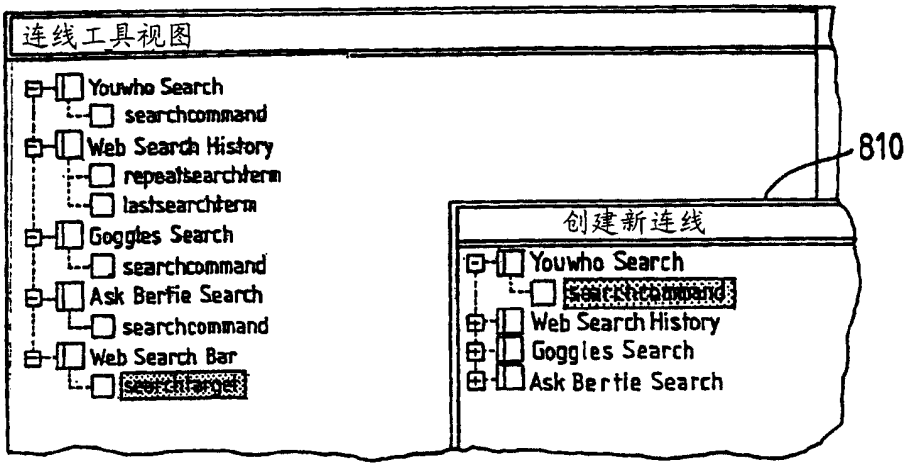

连线工具视图
Youwho Search
searchcommand
Web Search History
repeatsearchterm
lastsearchterm
Goggles Search
searchcommand
Ask Berfie Search
searchcommand
Web Search Bar
searchtarget
810
创建新连线
Youwho Search
searchcommand
Web Search History
Goggles Search
Ask Berfie Search

图 8C

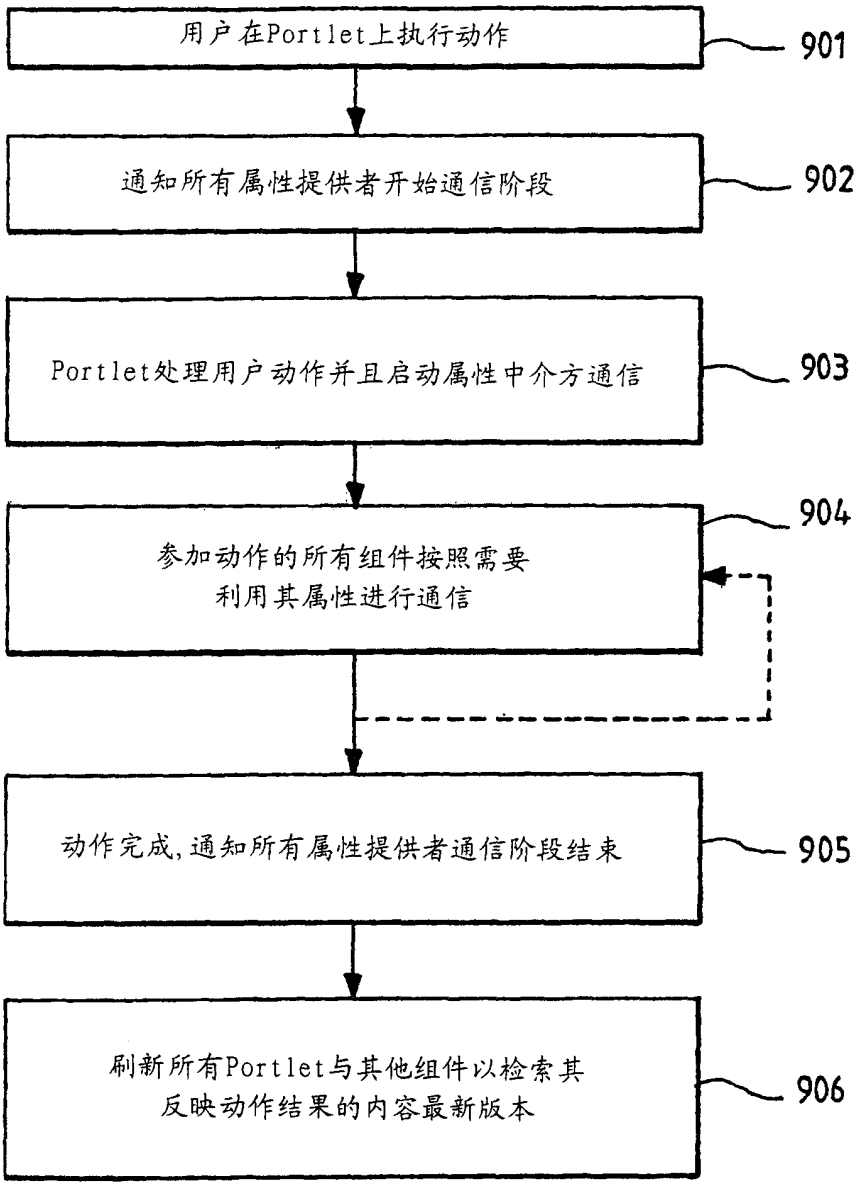

用户在Portlet上执行动作
901
通知所有属性提供者开始通信阶段
902
Portlet处理用户动作并且启动属性中介方通信
903
参加动作的所有组件按照需要
利用其属性进行通信
904
动作完成,通知所有属性提供者通信阶段结束
905
刷新所有Portlet与其他组件以检索其
反映动作结果的内容最新版本
906

图 9