[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/46

[12] 发明专利申请公开说明书

[21] 申请号 200410057037.4

[43] 公开日 2005 年 3 月 9 日

[11] 公开号 CN 1591339A

[22] 申请日 2004.8.25
[21] 申请号 200410057037.4
[30] 优先权
[32] 2003. 8.28 [33] US [31] 10/650,894
[71] 申请人 国际商业机器公司
地址 美国纽约
[72] 发明人 J·R·麦吉 M·J·莫顿
B·A·彼得斯

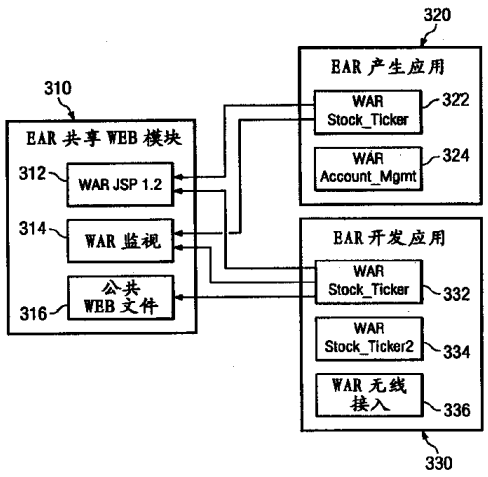
[74] 专利代理机构 北京市中咨律师事务所
代理人 于 静 杨晓光

权利要求书 3 页 说明书 16 页 附图 4 页

[54] 发明名称 提供共享 Web 模块的系统和方法

[57] 摘要

本发明提供了用于在多个其它 Web 应用中间共享 Web 模块的系统和方法。 通过该系统和方法，共享 Web 模块被存储在档案数据结构中，并且可被 Web 应用访问。 Web 应用和/或 Web 应用的 Web 模块可以包含共享 Web 模块指定文件，其标识将被引入到 Web 应用的 Web 模块中的共享 Web 模块。假定运行时部件在加载 Web 应用的 Web 模块时确定 Web 应用和/或 Web 模块是否具有用来标识将被引入到 Web 应用的 Web 模块中的共享 Web 模块的共享 Web 模块指定文件。 如果有，运行时部件找到这些共享 Web 模块并且将其与 Web 应用的 Web 模块逻辑合并，其考虑共享 Web 模块指定文件指定的共享 Web 模块的优先级，从而产生包含共享 Web 模块指定文件中引用的共享 Web 模块的逻辑合并的 Web 模块。



ISSN1008-4274

1. 一种用于为 Web 应用产生逻辑合并的 Web 模块的方法，包括:

确定 Web 应用是否包含针对可以引入到多个 Web 应用中的至少一个共享 Web 模块的引用;

识别至少一个共享 Web 模块的位置; 以及

如果存在引用，则逻辑合并至少一个共享 Web 模块和 Web 应用的 Web 模块以产生逻辑合并的 Web 应用。

2. 如权利要求 1 所述的方法，还包括:

将逻辑合并的 Web 应用加载到 Web 容器。

3. 如权利要求 1 所述的方法，其中确定 Web 应用是否包含针对至少一个共享 Web 模块的引用的步骤包括确定 Web 应用是否包括共享 Web 模块指定文件。

4. 如权利要求 1 所述的方法，其中 Web 应用是企业档案 (EAR)，并且逻辑合并的 Web 应用是逻辑合并的 EAR。

5. 如权利要求 1 所述的方法，其中所述至少一个共享 Web 模块包含 Web 档案 (WAR)文件、企业 Java Bean (EJB)档案文件和资源档案 (RAR)文件中的至少一个。

6. 如权利要求 1 所述的方法，其中逻辑合并至少一个共享 Web 模块与 Web 应用的 Web 模块的步骤包括:

确定与至少一个共享 Web 模块相关的优先级; 以及

如果存在至少一个共享 Web 模块中的共享 Web 模块之间的任何冲突，或至少一个共享 Web 模块与 Web 应用的 Web 模块之间的冲突，则消除所述冲突。

7. 如权利要求 1 所述的方法，其中在用于初始化将在服务器上运行的 Web 应用的运行时环境的初始化过程期间，执行确定、识别和逻辑合并步骤。

8. 如权利要求 1 所述的方法，其中逻辑合并至少一个共享 Web 模块

与 Web 应用的 Web 模块的步骤包括使用服务提供商接口 (SPI)，其提供用于合并不同模块类型的合并逻辑。

9. 如权利要求 2 所述的方法，其中当加载逻辑合并的 Web 应用时，容器使用一或多个应用程序接口(API)以识别到达至少一个共享 Web 模块的路径，并且加载该至少一个共享 Web 模块。

10. 如权利要求 1 所述的方法，其中逻辑合并至少一个共享 Web 模块与 Web 应用的 Web 模块的步骤包括以下操作中的至少一个：在 Web 应用的 Web 模块中重新链接针对至少一个共享 Web 模块的引用，根据与 Web 应用相关的策略文件推断至少一个共享 Web 模块的策略信息，和修改 Web 应用的类路径以包含到达至少一个共享 Web 模块中的每个的路径。

11. 一种计算机可读介质中用于为Web应用产生逻辑合并的Web模块的计算机程序产品，包括：

第一指令，用于确定 Web 应用是否包含针对可以引入到多个 Web 应用中的至少一个共享 Web 模块的引用；

第二指令，用于识别至少一个共享 Web 模块的位置；以及

第三指令，用于在存在引用的情况下，逻辑合并至少一个共享 Web 模块和 Web 应用的 Web 模块以产生逻辑合并的 Web 应用。

12. 如权利要求 11 所述的计算机程序产品，还包括：

第四指令，用于将逻辑合并的 Web 应用加载到 Web 容器。

13. 如权利要求 11 所述的计算机程序产品，其中用于确定 Web 应用是否包含针对至少一个共享 Web 模块的引用的第一指令包括用于确定 Web 应用是否包含共享 Web 模块指定文件的指令。

14. 如权利要求 11 所述的计算机程序产品，其中用于逻辑合并至少一个共享 Web 模块与 Web 应用的 Web 模块的第三指令包括：

用于确定与至少一个共享 Web 模块相关的优先级的指令；以及

用于在存在至少一个共享Web模块中的共享Web模块之间的任何冲突，或至少一个共享 Web 模块与 Web 应用的 Web 模块之间的冲突的情况下，消除所述冲突的指令。

15. 如权利要求 11 所述的计算机程序产品，其中在用于初始化将在服务器上运行的 Web 应用的运行时环境的初始化过程期间，执行第一、第二和第三指令。

16. 如权利要求 11 所述的计算机程序产品，其中用于逻辑合并至少一个共享 Web 模块与 Web 应用的 Web 模块的第三指令包括用于使用提供合并不同模块资源的合并逻辑的服务提供商接口(SPI)的指令。

17. 如权利要求 12 所述的计算机程序产品，其中容器使用一或多个应用程序接口（API）以识别到达至少一个共享 Web 模块的路径。

18. 如权利要求 11 所述的计算机程序产品，其中用于逻辑合并至少一个共享 Web 模块与 Web 应用的 Web 模块的第三指令包括以下指令中的至少一个：用于在 Web 应用的 Web 模块中重新链接针对至少一个共享 Web 模块的引用的指令，用于根据与 Web 应用相关的策略文件推断至少一个共享 Web 模块的策略信息的指令，和用于修改 Web 应用的类路径以包含对至少一个共享 Web 模块中的每个的路径的指令。

19. 一种用于为 Web 应用产生逻辑合并的 Web 模块的设备，包括：

用于确定 Web 应用是否包含针对可以引入到多个 Web 应用中的至少一个共享 Web 模块的引用的装置；

用于识别至少一个共享 Web 模块的位置的装置；以及

用于在存在引用的情况下，逻辑合并至少一个共享 Web 模块和 Web 应用的 Web 模块以产生逻辑合并的 Web 应用的装置。

20. 如权利要求 19 所述的设备，还包括：

用于将逻辑合并的 Web 应用加载到 Web 容器的装置。

提供共享Web模块的系统和方法

技术领域

本发明涉及提供共享Web模块的系统和方法。更具体地，本发明涉及逻辑合并共享Web模块和主Web模块以产生逻辑合并的Web模块的系统和方法。

背景技术

与 Java2 企业版 (J2EE)相关的编程模型要求由档案，例如企业档案(EAR)，构成的应用封装结构包含一个或多个子档案，例如Web档案 (WAR)，企业 Java Bean (EJB)档案 (JAR)，资源档案 (RAR)等等。 EAR文件封装了用于在 J2EE 环境中执行的应用所需的所有这些档案。WAR 文件包含HTTP客户机，例如浏览器通常需要的与"Web"相关的文件 (Java， HTML，GIF等等)。

经常出现这样的情况，其中应用开发人员具有在多个 WAR 中相同的、又称作Web模块的代码和结构。由于这种代码和结构在多个Web模块中是相同的，而不是将代码和结构的这些公共部分改写或复制到每个Web模块，期望能够在多个Web模块中间共享具有这种代码/结构的Web模块。然而，当前没有允许共享Web模块的机制。

相反，当前的机制受限于：公共代码/结构必须由应用开发者明确插入到Web模块中。例如，假定存在一个包含 100 个 JSP文件的Web模块。现在假定要开发两个或更多个也需要包含这些相同 JSP文件的Web模块。通过当前的机制，应用开发者必须将所有 JSP文件复制到每个新Web模块中。这对存储器的使用来说显然是低效的，另外还带来维护问题，因为现在必须在所有Web模块中兼顾任何未来对一或多个 JSP文件的改变。

作为另一个例子，假定存在 100 个全部具有相同安全要求的 Web 模块。例如，所有 Web 模块必须被构造成具有相同的用户角色。这需要应用开发者在每个 Web 模块的 Web 模块装配阶段执行相同的安全配置步骤，从而需要更多的人工和时间。另外，类似的维护问题在于，当改变用户角色时，这种改变必须被传播到所有 100 个 Web 模块。

于是，期望具有这样的机制，其允许其它 Web 模块以存储效率最大的方式共享 Web 模块，并且减少共享 Web 模块中与未来改变相关的维护问题。

发明内容

本发明提供了用于在多个其它 Web 应用中间共享 Web 模块的系统和方法。通过本发明的系统和方法，共享 Web 模块被存储在档案数据结构中，并且可被 Web 应用访问。Web 应用和/或 Web 应用的 Web 模块可以包含共享 Web 模块指定文件，其标识将被引入到 Web 应用的 Web 模块中的共享 Web 模块。

提供一个运行时部件，其确定 Web 应用和/或 Web 模块是否具有用来标识将被引入到 Web 应用的 Web 模块中的共享 Web 模块指定文件。如果有，运行时部件找到这些共享 Web 模块并且在考虑共享 Web 模块指定文件指定的共享 Web 模块的优先级的情况下将它们与 Web 应用的 Web 模块逻辑合并，从而产生包含共享 Web 模块指定文件中引用的共享 Web 模块的逻辑合并 Web 模块。

于是，通过本发明，不是将共享 Web 模块复制到每个利用共享 Web 模块的 Web 应用的 Web 模块中，本发明而是允许在引用特定存储位置的共享 Web 模块的 Web 应用的 Web 模块中使用针对共享 Web 模块的引用。当运行时环境加载 Web 应用的 Web 模块时，本发明的机制自动合并共享 Web 模块和 Web 应用的 Web 模块。

此外，由于多个 Web 应用仅仅引用一个副本，或在某些情况下仅仅引用数量相对较少的副本，明显减少了维护问题。也就是说，不必将针对共享 Web 模块的改变传播到利用共享 Web 模块中的代码的每个 Web 应用的 Web

模块，本发明通过在加载时逻辑合并共享 Web 模块和 Web 应用的 Web 模块，从而自动执行这种传播。于是，只需改变共享 Web 模块库中存储的共享 Web 模块的副本，而不需显式改变利用共享 Web 模块的每个 Web 应用。

本领域普通技术人员通过以下对优选实施例的详细描述可理解本发明的这些和其它特征及优点。

附图说明

在所附权利要求书中提出了被认为是本发明的特征的特性。然而参照下列结合附图对一个示例性实施例进行的详细描述可以更好地理解本发明自身、最优使用模式、其它目标和优点，其中：

图 1 是可以实现本发明的分布式数据处理系统的示例性模块图；

图 2 是可以实现本发明的服务器计算设备的示例性模块图；

图 3 是说明了根据本发明的 Web 模块共享的示例性模块图；

图 4 是说明了根据本发明一个示例性实施例的逻辑合并 Web 模块的生成的示例性模块图；

图 5 是说明了根据本发明一个示例性实施例的逻辑合并 Web 模块生成运行时部件的示例性模块图；

图 6 是说明了根据本发明一个示例性实施例的共享 Web 模块指定文件的示例图；以及

图 7 是概述了本发明在产生逻辑合并 Web 模块时的示例性操作的流程图。

具体实施方式

本发明提供了用于在多个 Web 应用的 Web 模块中间共享 Web 模块的机制。同样地，本发明尤其适于在分布式数据处理环境中使用，虽然本发明也可以用于独立或客户端计算设备。于是，为了提供有助于理解本发明优选实施例的说明的环境，以下提供了有关可以实现本发明的示例性分布式数据处理环境的简要描述。

现在参照附图，具体是参照图 1，其中图示了可以实现本发明的数据处理系统的网络。网络数据处理系统 100 是其中可以实现本发明的计算机的网络。网络数据处理系统 100 包含网络 102，网络 102 是被用来在一起连接到网络数据处理系统 100 内的各种设备和计算机之间提供通信链路的介质。网络 102 可以包含诸如电缆，无线通信链路或纤维光缆等连接。

在描述的例子中，服务器 104 和存储单元 106 一起被连接到网络 102。另外，客户机 108、110 和 112 也连接到网络 102。这些客户机 108、110 和 112 可以是例如个人计算机或网络计算机。在描述的例子中，服务器 104 向客户机 108 - 112 提供诸如引导文件、操作系统映像和应用等数据。客户机 108、110 和 112 是服务器 104 的客户机。网络数据处理系统 100 可以包含附加的服务器、客户机和其它设备(未示出)。在描述的例子中，网络数据处理系统 100 是具有网络 102 的因特网，其中网络 102 表示全世界使用传输控制协议/网际协议(TCP/IP)协议族彼此通信的网络和网关的集合。因特网的核心是主要节点或主计算机之间的高速数据通信线路组成的干线，主要节点或主计算机包括数千个路由数据和消息的商业、政府、教育和其它计算机系统。当然，网络数据处理系统 100 也可以被实现成若干不同类型的网络，例如内部网、局域网 (LAN)或广域网 (WAN)。图 1 仅用于举例，而不对本发明产生结构限制。

参照图 2，其中根据本发明的优选实施例描述了数据处理系统的模块图，该数据处理系统可以被实现成诸如图 1 的服务器 104 的服务器。数据处理系统 200 可以是对称多处理器 (SMP)系统，包含多个被连接到系统总线 206 的处理器 202 和 204。可选地，可以使用单处理器系统。存储器控制器/高速缓存 208 也被连接到系统总线 206，存储器控制器/高速缓存 208 提供针对本地存储器 209 的接口。I/O 总线桥 210 被连接到系统总线 206，并且提供针对 I/O 总线 212 的接口。如这里描述的，可以集成存储器控制器/高速缓存 208 和 I/O 总线桥 210。

连接到 I/O 总线 212 的外围部件互连 (PCI)总线桥 214 提供针对 PCI 局部总线 216 的接口。若干调制解调器可以被连接到 PCI 局部总线 216。

典型的 PCI 总线实现会支持 4 个 PCI 扩充槽或内插式连接器。利用调制解调器 218 和通过内插板被连接到 PCI 局部总线 216 的网络适配器 220，可以提供针对图 1 中客户机 108-112 的通信链路。

附加 PCI 总线桥 222 和 224 为附加 PCI 局部总线 226 和 228 提供接口，其中从附加 PCI 局部总线 226 和 228 可以支持附加的调制解调器或网络适配器。通过这种方式，数据处理系统 200 允许针对多个网络计算机的连接。存储器映射图形适配器 230 和硬盘 232 也可以直接或间接地连接到所述 I/O 总线 212。

图 2 中描述的数据处理系统可以是例如 IBM eServer pSeries 系统，这是位于纽约阿蒙克的国际商业机器公司的产品，其运行先进交互执行(AIX)操作系统或 Linux 操作系统。然而，本领域的普通技术人员会理解，图 2 中描述的硬件是可以改变的。例如，也可以使用其它诸如光盘驱动器和类似设备的外部设备补充或取代上述硬件。上述例子并不意味着对本发明有结构性限制。

本发明提供了一种用于根据 Web 应用的 Web 模块中的对共享 Web 模块的引用，将指定为在多个 Web 应用中间共享的 Web 模块的 Web 模块与 Web 应用的 Web 模块进行逻辑合并的系统和方法。本发明的机制根据共享 Web 模块的优先级和先后次序执行共享 Web 模块的合并。本发明还根据这些优先级和先后次序消除共享 Web 模块之间的任何冲突。

本发明的主要部件包含核心运行时环境部件和 Web 容器部件。核心运行时环境部件负责在 Web 应用的初始化期间识别具有继承共享 Web 模块的 Web 模块的 Web 应用。核心运行时环境部件接着合并这些共享 Web 模块和 Web 应用的现有 Web 模块，从而产生逻辑合并的 Web 模块。接着核心运行时环境部件通知 Web 容器部件加载逻辑合并的 Web 模块。

Web 容器是用于操作 Web 应用的执行基础设施。在本说明书的上下文中，Web 应用表示服务器端上使用 Web 容器提供的各种功能执行某些处理操作，例如分析客户机的输入数据和产生动态 HTML 页面，的应用。这种 Web 应用的例子包含 Java 服务器页面 (JSP)和 小服务程序（servlet）。

现在参照图 3，示出的示例性模块图说明了根据本发明的 Web 模块共享。如图 3 所示，两个 Web 应用企业档案 (EAR) 320 和 330 包含各种 Web 档案 (WAR) 文件，例如 Stock-Ticker 322， Account-Mgmt 324，Stock-Ticker 332， Stock-Ticker2 334 和 WirelessAccess 336。另外，在 EAR 310 中提供共享 Web 模块库。如此后更详细地描述的，通过将 EAR 310 的 Web 模块 312 - 316 包含在 EAR 310 中， 其被指定为共享 Web 模块，于是可以根据本发明的机制将其与 Web 应用的 Web 模块逻辑合并。

Production-Application EAR 320 包含按照相应顺序继承共享 Web 模块 JSP 1.2 312 和 Monitoring 314 的 WAR Stock-Ticker 322。于是，在应用初始化时，即当 Production-Application 320 被载入 Web 容器时，本发明的运行时部件执行所有这些 Web 模块的逻辑合并，并且产生一个接着由 Web 容器加载的虚拟 Web 模块。

对于 Development-Application EAR 330 的 Web 模块 Stock-Ticker 332 也是如此。Web 模块 Stock-Ticker 332 继承共享 Web 模块 JSP 1.2 312，Monitoring 314 和公共 Web 文件 316。 通过本发明的操作，会产生作为 Web 应用的 Web 模块和共享 Web 模块的逻辑合并版本的虚拟 Web 应用，并且由 Web 容器加载。于是，本发明提供了被用来加载使用共享 Web 模块的 Web 应用的逻辑合并 Web 应用的动态生成。

本发明的合并操作是多个 Web 模块的逻辑排序合并。合并操作与顺序相关的原因是为了能够确定先后次序以提供覆盖 Web 资源的能力。通过与共享 Web 模块相关的指定的优先级或先后次序来确定顺序。于是例如，如果 JSP 1.2Web 模块 312 比监视 Web 模块 314 具有更高优先级，则通过优先级或先后次序为 JSP 1.2Web 模块 312 消除这两个 Web 模块的 Web 资源之间的任何冲突。此后更详细地讨论 Web 模块的合并。

术语“逻辑合并” Web 应用是指，定义 Web 应用的数据结构是 Web 应用中现有的 Web 模块代码与共享 Web 模块代码的副本的组合。例如，假定第一 Web 模块 A 包含文件 Web.xml， servletA.class 和 pageA.html，第二 Web 模块 B 包含文件 Web.xml， servletB.class 和 pageB.html。 源

于第一 Web 模块和第二 Web 模块的逻辑合并 Web 应用包括文件 Web.xml、servletA.class、servletB.class、 pageA.html 和 pageB.html。

在上述例子中，根据所得到的合并 Web 应用在其被写入的盘上的物理表现来对其进行描述。然而 Web 模块的“逻辑”合并涉及识别文件物理上所在的路径，以及将这些路径添加到整个 Web 应用的类路径上。在 Java 中，这意味着当服务器加载应用时，服务器需要知道合并 Web 应用所引用的文件的物理存储单元。于是，在使用上述例子的情况下，到达存储 Web 模块 A 和 Web 模块 B 的文件的存储单元的所有路径都会被添加到合并 Web 应用的类路径中。

另外，每个 Web 模块需要包含基于 Servlet 规范的 Web.xml 文件。本发明的逻辑合并操作将所有这些单独的 Web.xml 文件合并成用于合并 Web 应用的单个 Web.xml 文件。Web.xml 文件的这种合并可以通过多个不同方式中的任何一种来执行。一个合并 Web.xml 文件的示例性技术是简单地将一个 Web.xml 文件的一部分复制到另一个 Web.xml 文件中。当然，需要根据针对 Web 模块确定的优先级或先后次序消除 Web.xml 文件之间的任何冲突。

例如，假定 Web 模块 A 具有 Web.xml 文件，该文件具有以下内容:

```
<Web-App id="WebApp_ID">
<display-name>Michaels Application</display-name>
<description> This is MichaelsWeb Application. </description>
<servlet id="Servlet 1">
<servlet-name>Michael Servlet</servlet-name>
<display-name>Michael Servlet</display-name>
<description> This servlet returns information about Michael's
favorite beers. </description>
<servlet-class>servletA</servlet-class>
<load-on-startup></load-on-startup>
</servlet>
```

```
<servlet-mapping id="ServletMapping_1">
<servlet-name>Michael Servlet</servlet-name>
<url-pattern>/Michael/*</url-pattern>
</servlet-mapping>
<welcome-file-list id="WelcomeFileList_1">
<welcome-file>pageA.html</welcome-file>
</welcome-file-list>
</Web-App>
```

现在假定 Web 模块 B 具有 Web.xml 文件，该文件具有以下内容:

```
<Web-App id="WebApp_ID">
<display-name>Steves Application</display-name>
<description> This is Steves Web Application. </description>
<servlet id="Servlet_1">
<servlet-name>Steve Servlet</servlet-name>
<display-name>Steve Servlet</display-name>
<description>This servlet returns information about Steve's
favorite foods.</description>
<servlet-class>servletB</servlet-class>
<load-on-startup></load-on-startup>
</servlet>
<servlet-mapping id="ServletMapping_1">
 <servlet-name>Steve Servlet</servlet-name>
<url-pattern>/steve/*</url-pattern>
</servlet-mapping>
<welcome-file-list id="WelcomeFileList_1">
<welcome-file>pageB.html</welcome-file>
</welcome-file-list>
</Web-App>
```

现在假定通过合并上述 Web.xml 文件而获得的合并 Web.xml 文件具有以下内容:

```
<Web-App id="WebApp_ID">
<display-name>Steves Application</display-name>
<description>This is StevesWeb Application.</description>
<servlet id="Servlet 1">
<servlet-name>Steve Servlet</servlet-name>
<display-name>Steve Servlet</display-name>
<description>This servlet returns information about Steve's
favorite foods.</description>
<servlet-class>servletB</servlet-class>
<load-on-startup></load-on-startup>
</servlet>
<servlet id="Servlet_2">
 <servlet-name>Michael Servlet</servlet-name>
 <display-name>Michael Servlet</display-name>
<description>This servlet returns information about Michael's
favorite beers.</description>
<servlet-class>servletA</servlet-class>
<load-on-startup></load-on-startup>
</servlet>
<servlet-mapping id="ServletMapping_1">
<servlet-name>Steve Servlet</servlet-name>
<url-pattern>/steve/*</url-pattern>
</servlet-mapping>
<servlet-mapping id="ServletMapping_2">
<servlet-name>Michael Servlet</servlet-name>
<url-pattern>/Michael/*</url-pattern>
```

</servlet-mapping>

<welcome-file-list id="WelcomeFileList_1">

<welcome-file>pageB.html</welcome-file>

</welcome-file-list>

</Web-App>

通过上述内容可以发现，Web模块A和Web模块B的两个Web.xml文件的合并会产生文件中在以下行之间的冲突:

<welcome-file-list id="WelcomeFileList_1">

<welcome-file>pageA.html</welcome-file>

</welcome-file-list>

和

<welcome-file-list id="WelcomeFileList_1">

<welcome-file>pageB.html</welcome-file>

</welcome-file-list>

然而，根据本发明确定的Web模块的优先级或先后次序规定了如何消除这种冲突。由于在这个特定例子中Web模块B具有更高优先级或先后次序，消除这种冲突的结果是在所得到的合并Web.xml文件中包含来自Web模块B的Web.xml文件的行。可以针对Web.xml文件之外的其他文件在合并Web模块中执行类似的文件之间的冲突的消除。

于是，所得到的逻辑合并Web应用包含初始Web应用的各种Web模块和共享Web模块，以及标识到达合并Web应用中每个Web模块的每个文件的路径的单独的类路径，并且包含单独的合并Web.xml文件。

图4的示例性模块图说明了根据本发明一个示例性实施例的逻辑合并Web模块的生成。如图4所示，Web应用410包含档案文件412和414以及共享Web模块指定文件416。在描述的示例性实施例中，档案文件412和414是Web档案(WAR)文件。然而，可以包含其它类型的档案文件以补充或替换WAR文件。这些其它类型的档案文件可以包含例如企业Java Bean(EJB)档案文件、资源档案(RAR)文件等等。

另外，虽然图 4 描述的例子说明被存储在 Web 应用 410 中的共享 Web 模块指定文件 416，例如 Web 应用的 EAR 文件，然而本发明不局限于此。共享 Web 模块指定文件 416 可以被存储为：作为父 Web 模块文件一部分的单独的文件，父 Web 模块文件中的绑定或扩展，父 Web 模块 Web.xml 文件的一部分，父 Web 应用文件中的单独的文件(如所描述的)，父 Web 应用文件中的绑定或扩展，或存储在父 Web 应用文件 Application.xml 中，等等。

在一个最优实施例中，共享 Web 模块指定文件 416 是作为 Web 应用 EAR 文件的一部分的单独的文件。在这种实施例中，共享 Web 模块指定文件 416 包含引用 Web 应用 410 中所有 Web 模块 412-414 的共享 Web 模块 422-426 的所有描述符。通过这种方式，提前提供 Web 应用的每个 Web 模块 412-414 所需的描述符信息，这与针对 Web 应用 410 中的每个父 Web 模块 412-414 一次一个地寻找相关性的方式相反。这些允许在处理之前预取/隐藏共享 Web 模块信息，所述处理在 Web 应用 410 的初始化期间针对每个单独 Web 模块而进行。于是，为了说明示例性优选实施例，下面会根据在 Web 应用 410 或 EAR 等层次存储的共享 Web 模块指定文件 416 来描述本发明。

如上所述，共享 Web 模块指定文件 416 存储将与 Web 应用 410 的 Web 模块 412-414 逻辑合并的共享 Web 模块 422-426 的描述符。这些描述符可以包含例如共享 Web 模块的文件名 (并且可能包含路径)，和共享 Web 模块的合并优先级。路径和文件名被用来获取合并共享 Web 模块，例如 Web 模块 424 与 Web 应用 410 中的现有 Web 模块 412-414 所需的共享 Web 模块信息。如下面更详细地描述的，合并优先级被用来消除合并操作期间 Web 模块之间的冲突。

当在服务器计算设备，例如图 2 说明的服务器计算设备的运行时环境中部署 Web 应用 410 时，Web 应用 410 经过初始化过程。在这个初始化过程期间，运行时环境处理 Web 应用 410，例如 EAR 文件的所有 Web 模块 412-414，并且通知 Web 容器 450 加载，即启动运行时环境部件传递到 Web 容器的 Web 模块。通过本发明，为运行时环境提供用于识别 Web 应用是否利用共享 Web 模块的附加部件 430，并且如果利用了共享 Web 模块，将那

些 Web 模块合并到逻辑合并 Web 模块中。

本发明的运行时环境部件 430 考察 Web 应用和/或 Web 应用的 Web 模块，以确定是否存在共享 Web 模块指定文件 416。如果是，根据共享 Web 模块指定文件 416 中设置的每个将被合并的共享 Web 模块的优先级或先后次序，运行时环境部件 430 合并指定的共享 Web 模块和 Web 应用的现有 Web 模块。

共享 Web 模块 422-426 与现有 Web 模块 412-414 的合并涉及根据优先级或先后次序消除共享 Web 模块 422-426 和现有 Web 模块 412-414 的 Web 资源之间的冲突。另外，执行合并操作可能需要重新链接合并的共享 Web 模块 422-426 与现有的 Web 模块 412-414，由 Web 应用 410 中的策略文件推断出针对所合并的共享 Web 模块 422-426 的策略信息，等等。可以在提供用于合并不同文件类型的合并逻辑的服务提供商接口(SPI)中指定在 Web 模块合并中执行的所有各种功能。这些合并操作的结果是逻辑合并的 Web 应用 440，其可以被 Web 容器 450 加载以便在服务器计算设备上使用。

一旦使用本发明的运行时环境部件 430 产生了逻辑合并的 Web 应用 440，就将逻辑合并的 Web 应用 440 提供给 Web 容器 450 以便加载。小服务程序规范中规定的 Web 容器 450 提供各种应用程序接口（API），其为 Web 模块资源提供路径信息。因此，对于逻辑合并的 Web 模块，Web 容器 450 识别到达合并的共享 Web 模块 422-426 的路径，使得用户应用可以利用 getRealPath() API 和 getPathTranslated() API，并且获得所请求的 Web 模块资源在本地文件系统上的位置。getRealPath() API 返回本地文件系统上的文件位置。getPathTranslated() API 返回任何在小服务程序名之后、任何查询字符串之前的额外路径信息，并且将其转换成真实路径。

Web 容器 450 以本领域已知的方式加载逻辑合并 Web 应用 440 的每个 Web 模块，包含逻辑合并的共享 Web 模块 422-426。一旦 Web 模块被加载，Web 应用可被服务器计算设备使用。

图 5 的示例性模块图说明了根据本发明一个示例性实施例的逻辑合并 Web 模块生成运行时部件，此后被称作运行时部件。如图 5 所示，运行时

部件包含彼此通信的多个子部件，包含运行时合并控制模块 510、Web 容器接口模块 520、共享文件识别模块 530 和合并模块 540。合并模块 540 还包含合并模块控制器 550、冲突检查模块 560、重新链接模块 570、策略推断模块 580 和父 Web 模块类路径修改模块。

运行时合并控制模块 510 控制运行时部件的总体操作，并且协调其它部件 520-540 的操作。运行时合并控制模块 510 考察被初始化的 Web 应用，以确定是否存在任何需要在 Web 容器加载 Web 应用之前进行合并的共享 Web 模块。如果存在，运行时合并控制模块 510 指示共享文件识别模块 530 读取 Web 应用的共享 Web 模块指定文件，并且识别必须合并的共享 Web 模块和共享 Web 模块指定文件中设定的所述共享 Web 模块的优先级或先后次序。这种信息接着被传递给合并模块 540，以用于合并共享 Web 模块和 Web 应用的现有 Web 模块，从而产生逻辑合并的 Web 应用。

合并模块 540 的合并模块控制器 550 合并共享文件识别模块 530 识别的共享 Web 模块。合并模块控制器 550 寻求冲突检查模块 560 的帮助以确定将被合并的 Web 模块之间的 Web 资源冲突。冲突检查模块 560 使用所识别的共享 Web 模块的先后次序或优先级确定如何消除冲突(如果存在)。

另外，合并模块控制器 550 使用重新链接模块 570 在 Web 应用的现有 Web 模块中重新链接针对共享 Web 模块的引用。策略推断模块 580 可以被用来根据与 Web 应用相关的策略文件推断合并的共享 Web 模块的策略信息。父 Web 模块类路径修改模块 590 可以被合并模块控制器 550 用来将合并处理中涉及的每个模块的存储单元的位置/路径添加到父模块的 类路径中。

共享文件识别模块 530 识别的共享 Web 模块上的这些部件 550-590 的操作产生通过 Web 容器接口模块 520 输出到 Web 容器的逻辑合并 Web 应用。Web 容器接着加载逻辑合并 Web 应用的 Web 模块，并且 Web 应用可被服务器使用。

图 6 的示例图说明了根据本发明一个示例性实施例的共享 Web 模块指定文件。如图 6 所示，共享 Web 模块指定文件包含共享 Web 模块的列表，包含共享 Web 模块的名称 610、到共享 Web 模块 620 的可选路径以及合并

优先级 630。本发明的机制使用名称 610 和路径 620 查找共享 Web 模块，以产生和加载逻辑合并的 Web 应用。如前面描述的，合并优先级被用来消除将被合并的共享 Web 模块的 Web 资源之间的冲突。

图 7 的流程图概述了本发明在产生逻辑合并Web模块时的示例性操作。应当理解，流程图说明的每个块和流程图说明中的块的组合可以通过计算机程序指令来实现。这些计算机程序指令可以提供给处理器或其它可编程数据处理装置以产生这样一种机器，使得在处理器或其它可编程数据处理装置上执行的指令创建用于实现流程图块中规定的功能的装置。这些计算机程序指令也可以存储在能够指示处理器或其它可编程数据处理装置以特定方式操作的计算机可读存储器或存储介质中，使得计算机可读存储器或存储介质中存储的指令产生一种制造产品，其包含实现流程图块中规定的功能的指令单元。

因此，流程图说明的块支持用于执行规定功能的装置的组合，用于执行规定功能的步骤的组合，和用于执行规定功能的程序指令单元的组合。还能够理解，流程图说明的每个块，和流程图说明中块的组合可以通过基于专用硬件的计算机系统(执行规定功能或步骤)，或专用硬件与计算机指令的组合来实现。

如图 7 所示，本发明的这个示例性实施例的操作从 Web 应用的部署和初始化开始 (步骤 710)。在初始化过程期间，确定 Web 应用是否包含需要与现有 Web 模块逻辑合并的共享 Web 模块 (步骤 720)。如果没有，操作继续执行到步骤 780，其中 Web 应用的 Web 模块被 Web 容器加载，并且使得 Web 应用可被服务器使用。

如果 Web 应用包含需要逻辑合并的共享 Web 模块，操作读取 Web 应用的共享 Web 文件指定文件，并且识别将被合并的共享 Web 模块，以及其优先级 (步骤 730)。确定到共享 Web 模块的路径 (步骤 740)，并且使用识别的优先级消除共享 Web 模块的 Web 资源之间的任何冲突 (步骤 750)。接着，共享 Web 模块与 Web 应用的现有 Web 模块合并以产生逻辑合并的 Web 应用 (步骤 760)。接着，逻辑合并的 Web 应用被提供给 Web 容器 (步骤

770)，Web 容器加载逻辑合并的 Web 应用的 Web 模块并且使得它们可被服务器使用 (步骤 780)。操作接着结束。

于是，本发明提供了可以使多个 Web 应用共享 Web 模块的机制。这允许 Web 应用引用共享 Web 模块而无需将 Web 模块的内容复制到该 Web 应用。此外，由于共享 Web 模块被集中起来，可以容易地修改这些共享 Web 模块，并且发布到利用共享 Web 模块的每个 Web 应用。

应当注意，虽然根据共享 Web 模块与 EAR 或其它类型的 Web 应用的现有 Web 模块进行组合描述了本发明，然而本发明不局限于此。EAR 或 Web 应用不必具有现有 Web 模块，并且可以包含共享 Web 模块指定文件，而无需任何其他在 EAR 或 Web 应用中存在的 Web 模块。于是，在这种情况下，本发明执行的逻辑合并会产生仅仅包含共享 Web 模块指定文件指定的共享 Web 模块的逻辑合并的 Web 应用。

另外，虽然就运行时环境在应用加载时执行 Web 应用的 Web 模块与共享 Web 模块的逻辑合并描述了本发明，然而本发明不局限于此。本发明可以“脱机”，即在加载应用之前使用，以产生可以接着被运行时环境加载的逻辑合并的 Web 应用。这有助于通过减少应用加载期间所需的计算量来提高运行时环境所在的服务器或计算设备的性能。

必须注意，虽然前面针对一个全功能数据处理系统的环境描述了本发明，但本领域的普通技术人员会理解到，可以通过计算机可读指令介质的形式和其它各种形式分布本发明的过程，并且无论实际被用来完成分布的信号承载介质的具体类型如何，本发明均同样适用。计算机可读介质的例子包含可记录类型的介质(例如软盘，硬盘驱动器， RAM， CD-ROM， DVD-ROM)，和传输类型的介质(例如数字和模拟通信链路，使用例如射频和光波传输等传输形式的有线或无线通信链路)。计算机可读介质可以具有编码格式的形式，其中在实际用于具体数据处理系统时被加以解码。

前面对本发明进行的描述只是为了说明的目的，并不打算对具有公开形式的本发明进行详细定义和限制。本领域的普通技术人员显然可以进行许多修改和改变。选择和描述实施例是为了提供对本发明原理及其实际应

用的最优说明，并且也是为了使本领域的普通技术人员可以根据所考虑的具体使用情况针对具有各种修改的各种实施例来理解本发明。

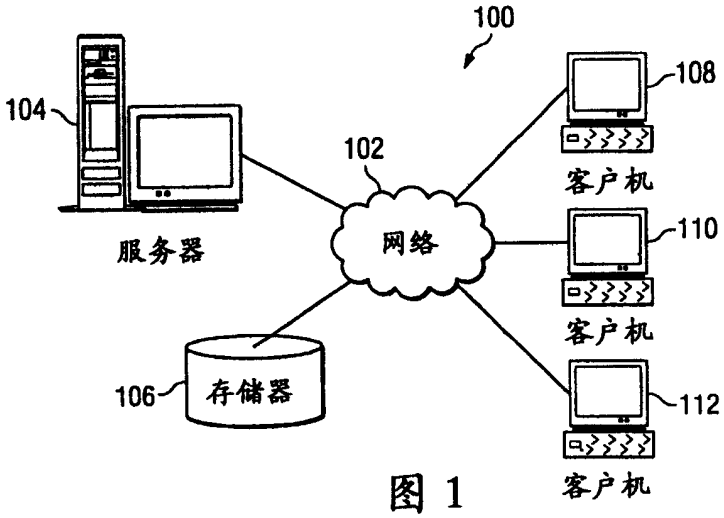

100
104
服务器
102
网络
108
客户机
110
客户机
112
客户机
106
存储器

图 1

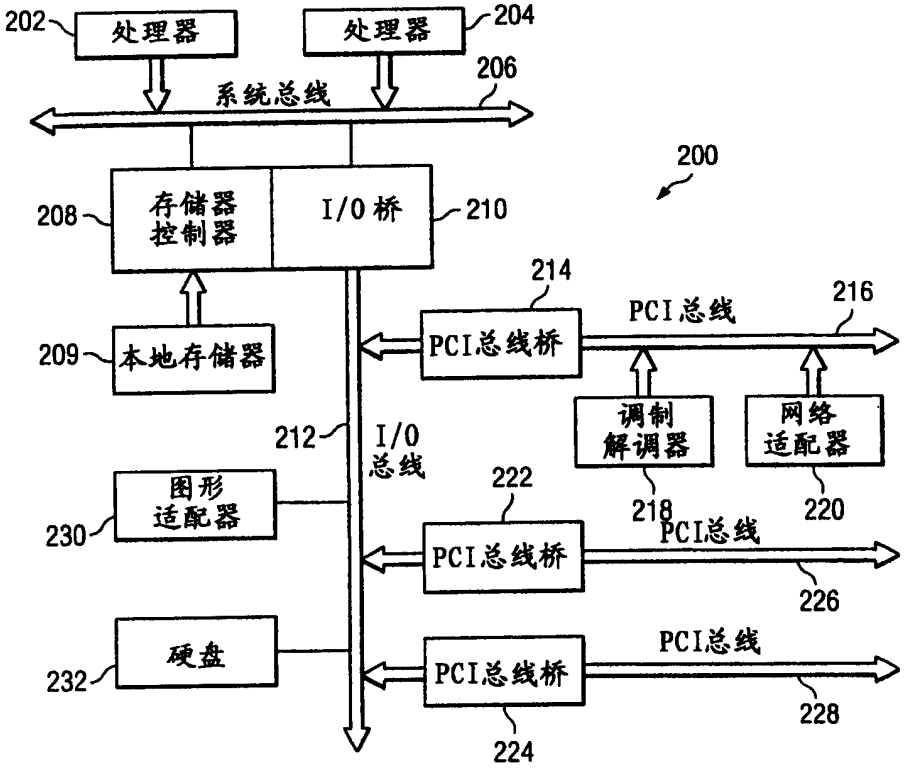

202
处理器
处理器
204
206
系统总线
200
208
存储器
控制器
I/O 桥
210
214
PCI总线桥
PCI总线
216
209
本地存储器
调制
解调器
网络
适配器
218
220
212
I/O
总线
222
图形
适配器
230
PCI总线桥
PCI总线
226
硬盘
232
PCI总线桥
PCI总线
228
224

图 2

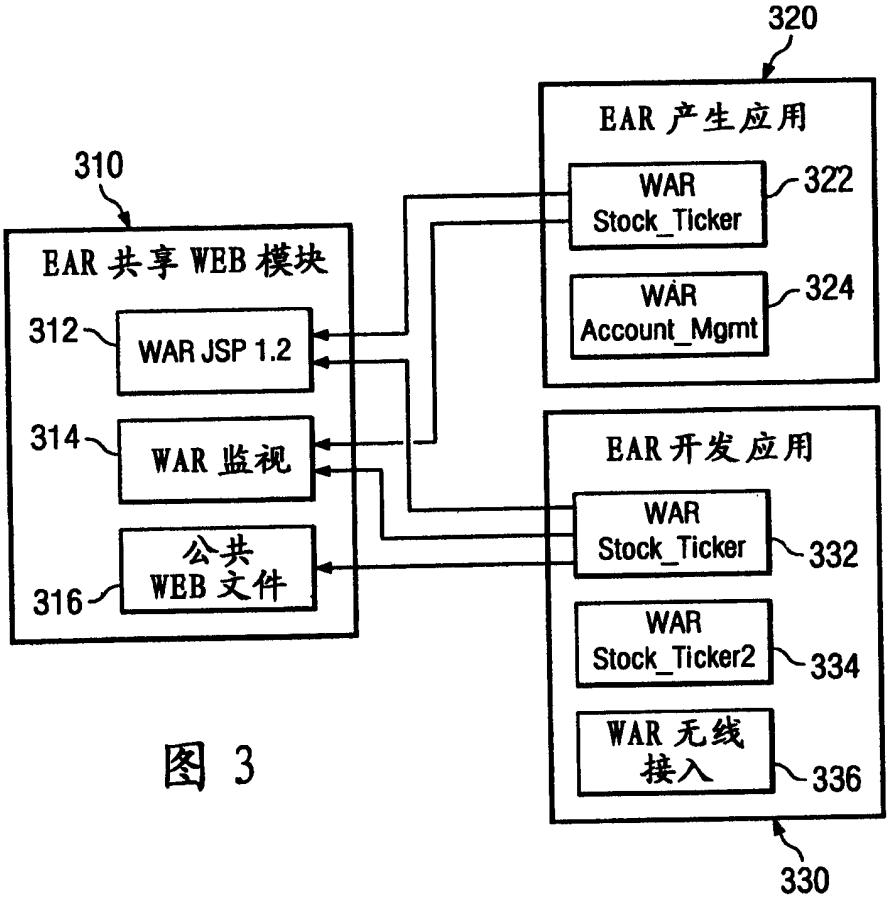


图 3

图 6

```
<shared-web-module-config>                                  610        630
    <web-module name="DefaultWebApplication" mergePriority="2">
                                                                       620
        <shared-web-module uri="${APP_INSTALL_ROOT}/
        SharedWebModulesApp/jsp.war" mergePriority="1"/>

        <shared-web-module uri="${APP_INSTALL_ROOT}/
        SharedWebModulesApp/common.war" mergePriority="3"/>

    </web-module>
</shared-web-module-config>
```

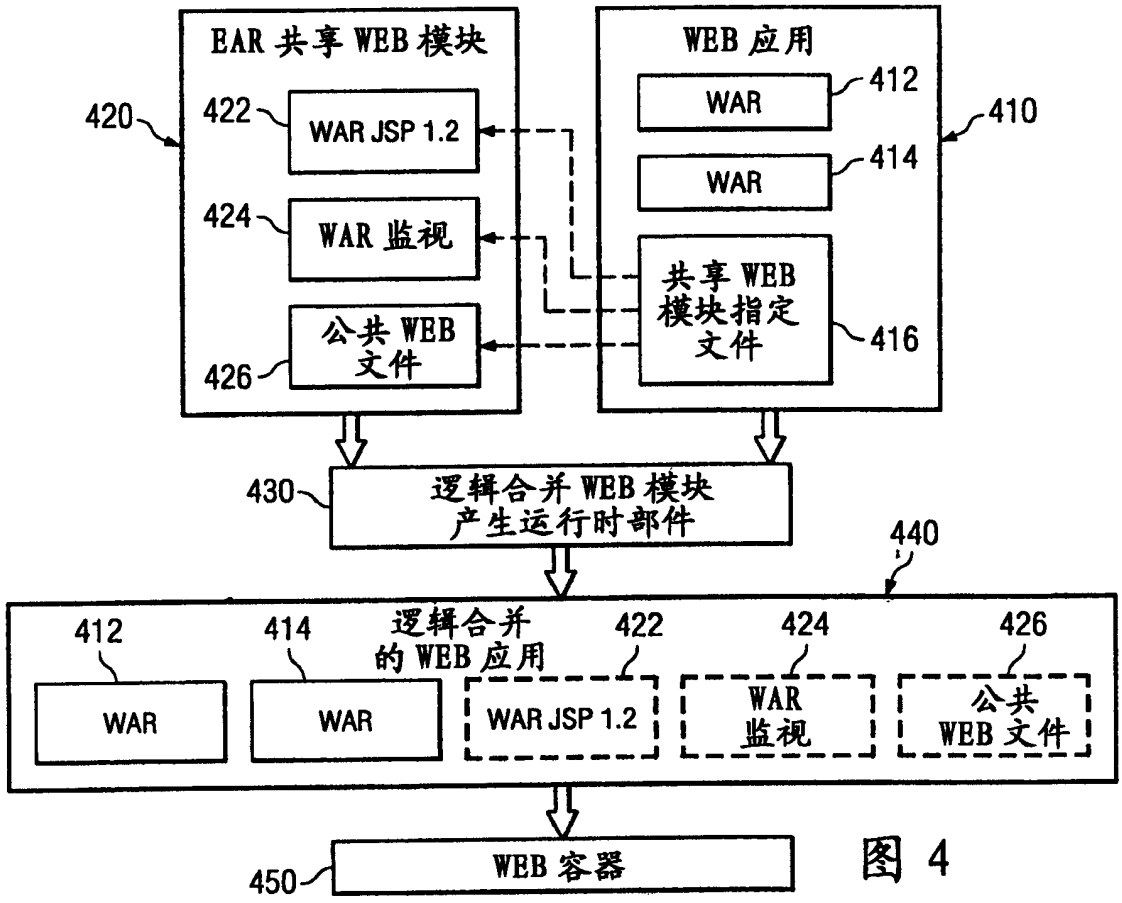

EAR 共享 WEB 模块
420
422
WAR JSP 1.2
424
WAR 监视
426
公共 WEB 文件
WEB 应用
412
WAR
414
WAR
410
共享 WEB 模块指定文件
416
430
逻辑合并 WEB 模块产生运行时部件
440
412
414
逻辑合并的 WEB 应用
422
424
426
WAR
WAR
WAR JSP 1.2
WAR 监视
公共 WEB 文件
450
WEB 容器

图 4

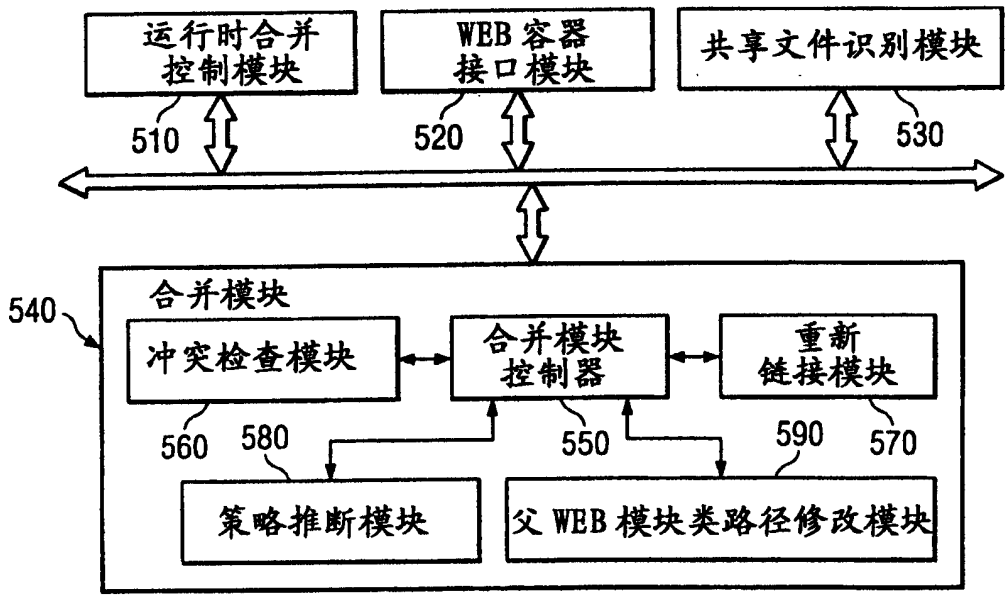

运行时合并控制模块
510
WEB 容器接口模块
520
共享文件识别模块
530
合并模块
540
冲突检查模块
合并模块控制器
重新链接模块
560
580
550
590
570
策略推断模块
父 WEB 模块类路径修改模块

图 5

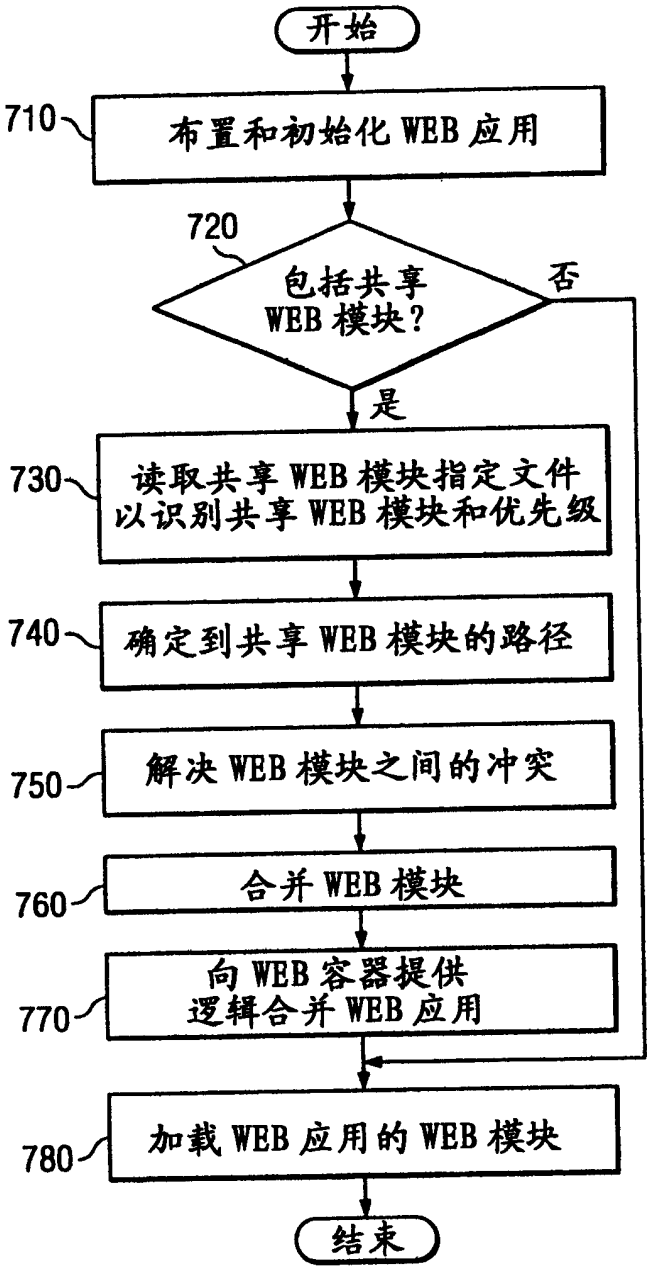

开始
710
布置和初始化 WEB 应用
720
包括共享 WEB 模块?
否
是
730
读取共享 WEB 模块指定文件以识别共享 WEB 模块和优先级
740
确定到共享 WEB 模块的路径
750
解决 WEB 模块之间的冲突
760
合并 WEB 模块
770
向 WEB 容器提供逻辑合并 WEB 应用
780
加载 WEB 应用的 WEB 模块
结束

图 7