

Java小应用程序下插件功能的实现方法

申请号：[200410001279.1](#)

申请日：2004-01-06

申请(专利权)人 [英业达股份有限公司](#)

地址 [台湾省台北市](#)

发明(设计)人 [刘文涵](#) [陈玄同](#) [李海军](#)

主分类号 [G06F9/44](#)

分类号 [G06F9/44](#)

公开(公告)号 [1641569A](#)

公开(公告)日 [2005-07-20](#)

专利代理机构 [隆天国际知识产权代理有限公司](#)

代理人 [陈晨](#) [郭凤麟](#)

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/44 (2006.01)



[12] 发明专利说明书

专利号 ZL 200410001279.1

[45] 授权公告日 2008 年 1 月 30 日

[11] 授权公告号 CN 100365568C

[22] 申请日 2004.1.6

[21] 申请号 200410001279.1

[73] 专利权人 英业达股份有限公司

地址 台湾省台北市

[72] 发明人 刘文涵 陈玄同 李海军

[56] 参考文献

CN1265489A 2000.9.6

WO03027840A1 2003.4.3

WO02097610A1 2002.12.5

审查员 李楠

[74] 专利代理机构 隆天国际知识产权代理有限公司

代理人 陈晨 郭凤麟

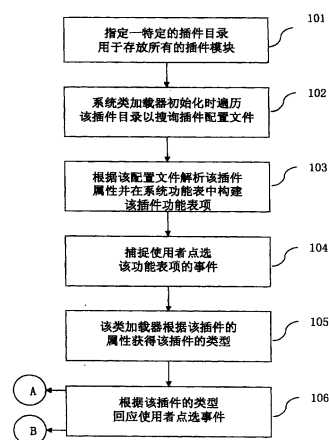
权利要求书 1 页 说明书 6 页 附图 6 页

[54] 发明名称

Java 小应用程序下插件功能的实现方法

[57] 摘要

一种 Java 小应用程序下插件功能的实现方法，首先指定一插件目录存放插件模块，然后系统类加载器在初始化时搜寻插件配置文件，解析插件属性并在系统菜单中构建该插件菜单项，当捕捉到使用者点选该菜单项的事件时，类加载器根据插件的类型响应使用者点选事件，本发明利用 Java 的动态加载类的功能特性，通过类加载器动态的加载了插件模块，为系统设计带来很大的可扩展性。



1、一种 Java 小应用程序下插件功能的实现方法，用以在 Java 小程序下动态加载插件模块来扩展系统功能，该方法包括如下步骤：

指定一特定的插件目录用于存放所有的插件模块；

系统类加载器初始化时遍历该插件目录以搜寻插件配置文件；

根据该配置文件解析该插件属性并在系统菜单中构建该插件菜单项；

捕捉使用者点选该菜单项的事件；

该类加载器根据该插件的属性获得该插件的类型；及

根据该插件的类型响应使用者点选事件，其中该插件的类型为应用程序或页面。

2、如权利要求 1 所述的 Java 小应用程序下插件功能的实现方法，其中该插件属性包括在系统菜单中的位置、状态以及名称。

3、如权利要求 1 所述的 Java 小应用程序下插件功能的实现方法，其中该在系统菜单中构建插件菜单项的步骤，是根据该插件的属性在系统菜单中加入该菜单项。

4、如权利要求 1 所述的 Java 小应用程序下插件功能的实现方法，其中该插件类型为应用程序时，该响应步骤更包括如下步骤：

在插件目录中查找相对应的类；

根据该类构造该类的一个实例；及

呼叫该实例的初始化方法启动该应用程序。

5、如权利要求 1 所述的 Java 小应用程序下插件功能的实现方法，其中该指定的插件目录位于服务器端时，更包括一将该配置文件中的参数传递到客户端的步骤。

Java 小应用程序下插件功能的实现方法

技术领域

本发明涉及一种插件功能的实现方法，特别是关于一种在 Java 小应用程序下实现插件功能的方法。

背景技术

插件 (Plug-in)，英文的原意是指电源插头，在软件领域中，特别是在浏览器中，是指一些扩展小程序。严格的插件 (Plug-In) 大家都可以看得到，比如和浏览器相关的各种各样的插件，用来显示不同格式的文件和播放不同的多媒体。又如 PhotoShop 中的各种各样的插件，显然也是用来实现一些辅助的功能。插件 (Plug-In) 目前最流行的作用就是能使应用比较灵活，可以自由地为应用增加新的插件或者是卸载现有的插件，而应用程序不需要重新编译和连接。

Java 语言可以编写两种类型的程序：应用程序 (Application) 和小应用程序 (Applet)。应用程序是可以独立运行的程序，而 Applet 不能独立运行，需要嵌入 HTML (Hypertext Markup Language, 超文字标记语言) 文件，遵循一套约定，在支持 Java 的浏览器 (如：Netscape Navigator 2.02 版本以上，HotJava, Microsoft Internet Explorer 3.0 版本以上) 运行，是 Java 一个重要的应用分支，也是当时 Java 最令人感兴趣的地方 (它一改网页呆板的接口)，就是在 WWW 网页 (Home Page / Pages) 设计中加入动画、影像、音乐等，而要达到这些效果，使用最多的是 Java Applet 和 Java Script。

JavaScript 是一种基于对象 (Object) 和事件驱动 (Event Driven) 并具有安全性能的脚本语言。使用它的目的是与 HTML 超文字标记语言、与 Web 客户交互作用，从而可以开发客户端的应用程序等。它是通过嵌入或文件引用在标准的 HTML 语言中实现的。它的出现弥补了 HTML 语言的缺陷，它是 Java 与 HTML 折衷的选择，具有基于对象、简单、安全、动态、跨平台性等特性。

如同 Applet 执行于浏览器而延伸了浏览器的功能一样，HTTP Servlet 也

执行于具有 Java 功能的 Web 服务器，而延伸了 Web 服务器的功能。Servlet 是利用 Java Servlet 应用程序设计接口（API）的 Java 程序。它们以 `ServletName.class` 的方式存在，也可能并入 JAR（Java 保存文件）文件中。

Java 是一种极具有动态性的语言，类似 Windows 的动态链接库（DLL），Java 应用程序总是被编译成若干个单独的类（Class）文件，程序执行时根据需要由 Java 虚拟机动态加载相应的类。术语“类加载”指的就是找出一个给定类名的字节所在的位置并且将这些字节转换成 Java 类实例的过程。Java 虚拟机（JVM）中所有的 `java.lang.Class` 实例都作为一个数组开始，该数组被组织成由 JVM 规范定义类文件格式。

类加载是由 JVM 在启动过程中执行的，紧接着由 `java.lang.ClassLoader` 类中的子类执行。这些类加载器提供一种抽象概念，使 JVM 不需要知道类字节的具体位置就可以将其加载，能够进行本地和远程存储，以及实现动态类生成。

简而言之，`classloader` 是 `java.lang.ClassLoader` 类的子类，负责类的加载。在 Java 应用程序中，有很多使用不同机制加载类的不同类加载器。每一个类加载器，都是为了与一个或者多个代码源（`Codesource`）协同工作而设计的。代码源是一个根位置，从这个位置，类加载器去寻找类。代码源被定义用来表示二进制类文件、必须先被编译的 Java 源代码或者实时生成的类的物理存储地址。

正是由于 Java 语言的强安全性、平台无关性、硬件结构无关性、语言简洁同时面向对象的特点，其已在网络编程语言中占据了无可比拟的优势。

而另一方面，随着经济和高科技的发展，企业开始大量应用网络程序来管理生产、经营等工作。由于企业在不断的发展与进步，按企业早期经营模式设计的系统有可能已经不能满足现在或将来的应用，需要将系统升级，系统升级最常用的办法是增加功能模块或是重新改写。但是重新改写成本太高，除非系统结构需要改变，比较经济的办法是根据企业的新需求增加一个新模块，这样对企业 and 开发者来说都有利。这就要求在开发阶段需要合理设计系统，以便在将来能增加模块。

一种可行的办法是加入插件（Plug-in）功能，系统设计完成投入使用后，用户可根据需要将一些额外的功能以插件（Plug-in）的形式加入系统，系统将把插件（Plug-in）模块当作自己的一个功能模块进行调度。而在网络应用

系统中，Java 的 Applet 被大量应用，如能在 Applet 中实现插件（Plug-in）功能，将会给设计带来很大的方便，因此，如何能在 Java Applet 下实现插件（Plug-in）功能，已经成为急待解决的问题。

发明内容

本发明为解决上述问题而提供一种 Java 小应用程序下插件功能的实现方法，用以在 Java Applet 下呼叫插件模块。

本发明提供一种 Java 小应用程序下插件功能的实现方法，用以在 Java 小程序下动态加载插件模块来扩展系统功能，该方法首先是指定一特定的插件目录用于存放所有的插件模块；然后系统类加载器在初始化时遍历该插件目录以搜寻插件配置文件；根据该配置文件解析该插件属性并在系统菜单中构建该插件菜单项；捕捉使用者点选该菜单项的事件；该类加载器根据该插件的属性获得该插件的类型；根据该插件的类型响应使用者点选事件，其中该插件的类型为应用程序或页面。

本发明利用 Java 的动态加载类的功能特性，通过类加载器动态加载插件模块，使应用一网络系统人员可以按照格式编写自己的插件模块，并整合到 Java 小程序中，而无需因为功能上的某些局部改进而重新升级系统，可以为系统设计带来很大的可扩展性。

附图说明

第 1 图为本发明在 Java 小应用程序中插件功能的实现方法流程图；

第 2 图为本发明所提供的在插件为应用程序时响应使用者点选事件的实施例流程图；

第 3 图为本发明所提供的在插件为页面时响应使用者点选事件的实施例流程图；

第 4 图为本发明的应用环境整体流程图；

第 5 图为本发明所提供的构建菜单项的流程图；及

第 6 图为本发明所提供的动态加载类的实施例流程图。

其中，附图标记说明如下：

步骤 101 指定一特定的插件目录用于存放所有的插件模块

步骤 102 系统类加载器初始化时遍历该插件目录以搜寻插件配置文件

步骤 103 根据该配置文件解析该插件属性并在系统菜单中构建该插件菜单项

步骤 104 捕捉使用者点选该菜单项的事件

步骤 105 该类加载器根据该插件的属性获得该插件的类型

步骤 106 根据该插件的类型响应使用者点选事件

步骤 201 插件目录中查找相对应的类

步骤 202 根据该类构造该类的一个实例

步骤 203 呼叫该实例的初始化方法启动该应用程序

步骤 301 呼叫 Applet 的 Jsp 文件中加入一个呼叫另一个页面的 JavaScript 函数

步骤 302 将插件作为参数传递给该 JavaScript 函数

步骤 303 在 Applet 呼叫该 JavaScript 函数时, 由该方法呼叫该插件模块

步骤 401 查找存于插件目录下的 Plug-in 配置文件 Config.xml

步骤 402、403 将配置文件中的参数传递到客户端

步骤 405、406 以页面形式实现功能的动态插入

步骤 404 直接插入程序

步骤 501 系统初始化

步骤 502 搜索 Plug-in 目录

步骤 503 是否有 Plug-in 配置文件

步骤 504 获得该 Plug-in 在系统菜单中的多项属性

步骤 505 在系统菜单上加入该项菜单

步骤 601 若 Plug-in 类型为应用程序

步骤 602 搜索对应的类

步骤 603、604 找到该类后构造该类的一个实例

步骤 605 呼叫该实例的初始化方法

步骤 606 若 Plug-in 为一个 Jsp 页面

步骤 607 加载该页面

具体实施方式

请参阅图 1，为本发明 Java 小应用程序下插件功能的实现方法流程图。该方法首先指定一特定的插件目录用于存放所有的插件模块（步骤 101）；然后系统类加载器初始化时遍历该插件目录以搜寻插件配置文件（步骤 102）；根据该配置文件解析该插件属性并在系统菜单中构建该插件菜单项（步骤 103）；捕捉使用者点选该菜单项的事件（步骤 104）；该类加载器根据该插件的属性获得该插件的类型（步骤 105）；根据该插件的类型响应使用者点选事件（步骤 106）。

其中，该插件类型为应用程序时，该响应步骤（步骤 106）请参阅图 2，其首先在插件目录中查找相对应的类（步骤 201）；然后根据该类构造该类的一个实例（步骤 202）；最后呼叫该实例的初始化方法启动该应用程序（步骤 203）。

而当该插件类型为页面时，该响应步骤（步骤 106）请参阅图 3，其首先在呼叫 Applet 的 Jsp（Java Server Pages）文件中加入一个呼叫另一个页面的 JavaScript 函数（步骤 301）；然后将插件作为参数传递给该 JavaScript 函数（步骤 302）；最后在 Applet 呼叫该 JavaScript 函数时，由该方法呼叫该插件模块（步骤 303）。

本发明主要是利用了 Java 的动态加载类的功能特性，在一个网络环境中，如果上述的插件目录位于服务器端时，还包括一将该配置文件中的参数传递到客户端的步骤，其整体结构流程图如图 4 所示，首先查找保存于插件目录下的 Plug-in 配置文件 Config.xml（步骤 401）；如果这个插件目录位于服务器端，则需要将配置文件中的参数传递到客户端（步骤 402、403）；由于 Plug-in 对系统来说是一个单独的模块，可以是一个应用程序，也可以是一个页面，因此可以对这两种类型分别处理，如果 Plug-in 是一个页面，我们可以在调用 Applet 的 Jsp 文件中加入一个调用另外一个页面的 JavaScript 函数，将 Plug-in 作为参数传给这个 JavaScript 函数，当 Applet 调用这个方法时，由这个方法调用 Plug-in 模块，以页面形式实现功能的动态插入（步骤 405、406）；另一种情况是 Plug-in 是一个独立的应用程序，则应用 Java 的动态加载机制直接插入程序（步骤 404）。

如图 5 所示，为本发明所提供的构建菜单步骤的实施例流程图。首先系统可以提供一个类装载模块，该模块在初始化（步骤 501）时会搜索 Plug-in 目

录(步骤 502), 寻找 Plug-in 目录下是否有 Plug-in 配置文件(步骤 503), 如果有配置文件, 则说明有一个 Plug-in 模块需要加载系统, 系统解析该配置文件, 取得该 Plug-in 在系统菜单中的位置、状态、名称等属性(步骤 504), 根据这些属性在系统菜单上加入该项菜单(步骤 505), 以使用户在选中此菜单时启动 Plug-in 模块。

请参阅图 6, 为本发明所提供的响应使用者点选事件, 动态加载类的实施例流程图。Plug-in 菜单构造完成后, 可以响应用户点击事件, 用户点击该项菜单时, 类装载模块根据 Plug-in 的 class type 属性判断出该 plug-in 为何种类型, 若 Plug-in 类型为应用程序(步骤 601), 则在 Plug-in 目录中寻找对应的类(步骤 602), 找到该类后构造该类的一个实例(步骤 603、604), 然后呼叫该实例的初始化方法启动应用程序(步骤 605); 若 Plug-in 为一个 Jsp 页面(步骤 606), 则利用动态加载模块加载该页面(步骤 607)。

以上所述, 仅为本发明其中的较佳实施例, 并非用来限定本发明的实施范围; 凡依本发明申请专利范围在不脱离本发明的精神和范围内所作的均等变化与修饰, 均应属于本发明专利权利要求书所要求保护的范围内。

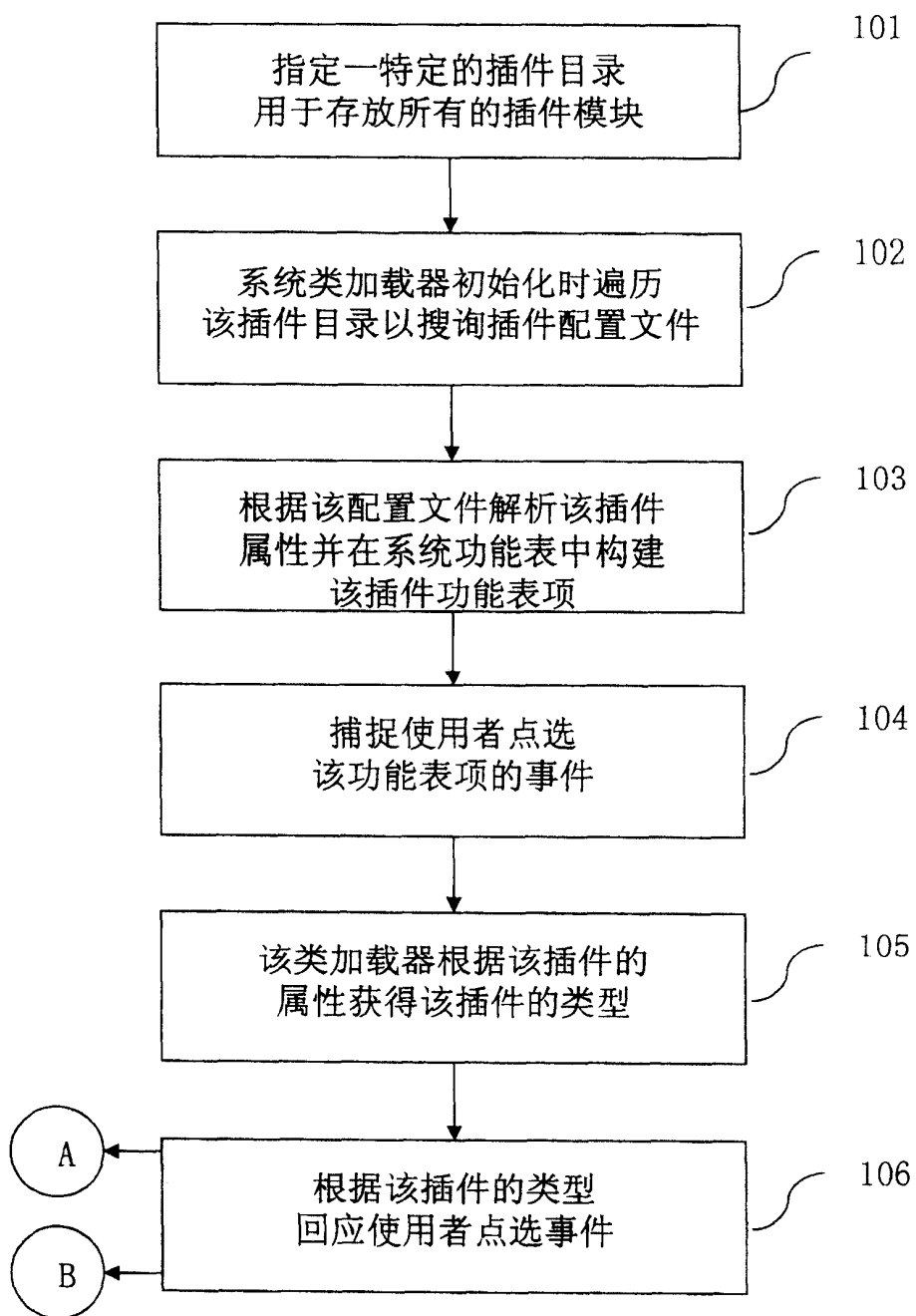


图1

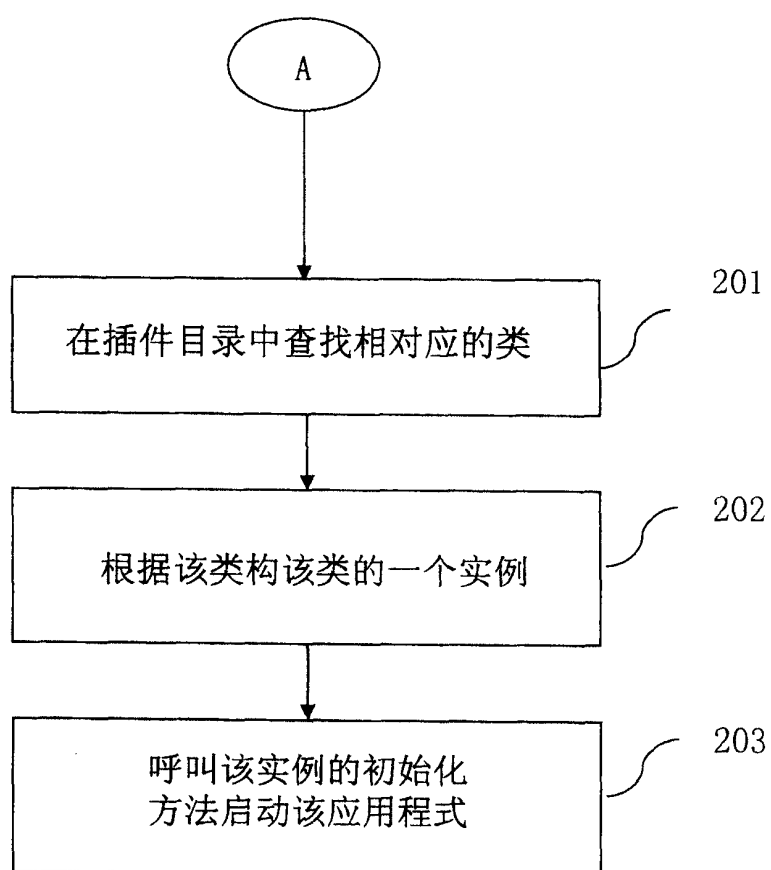


图2

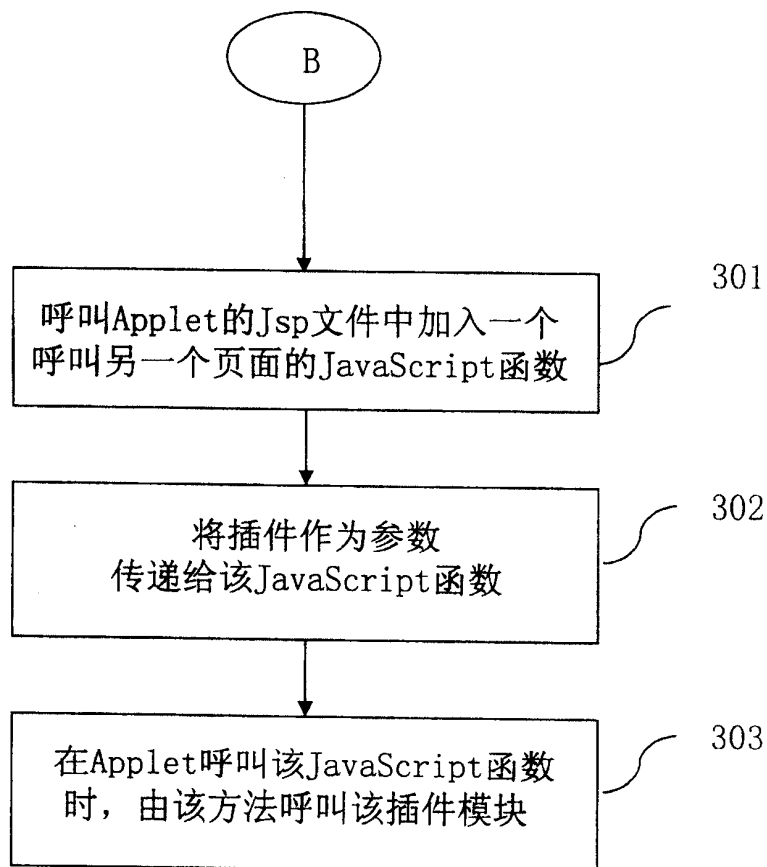


图3

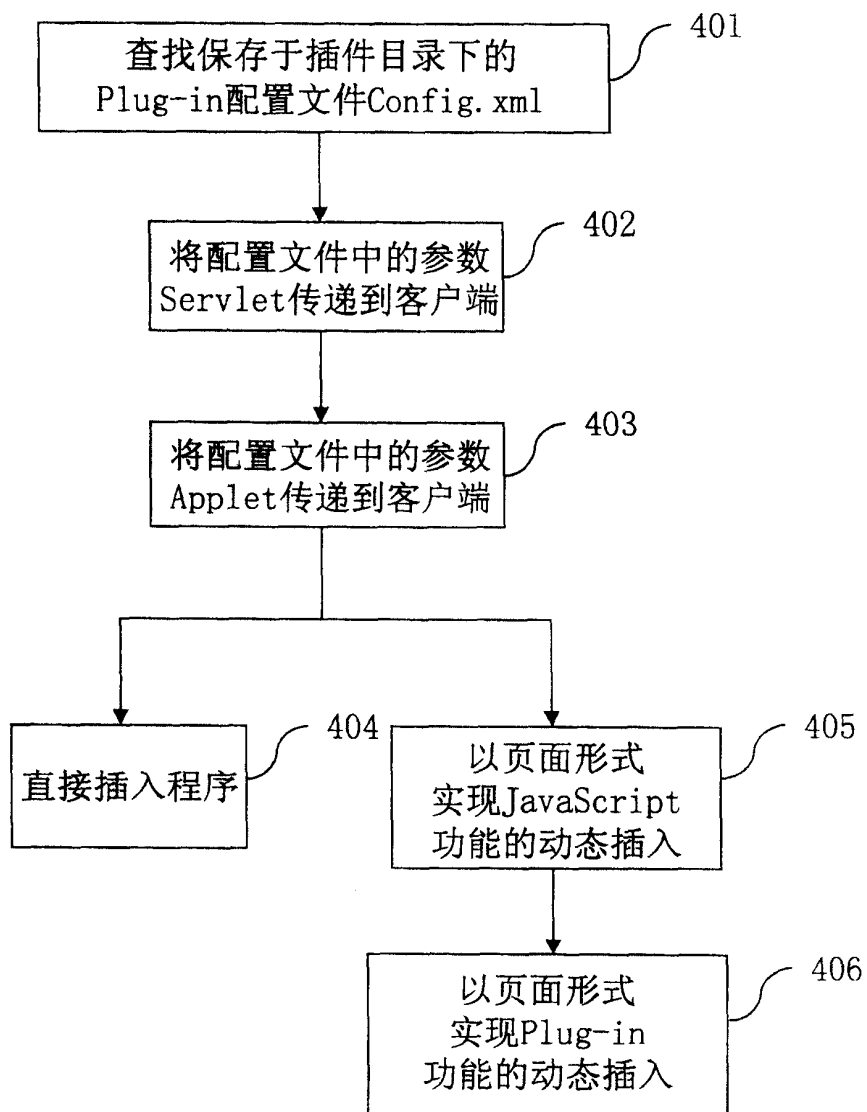


图4

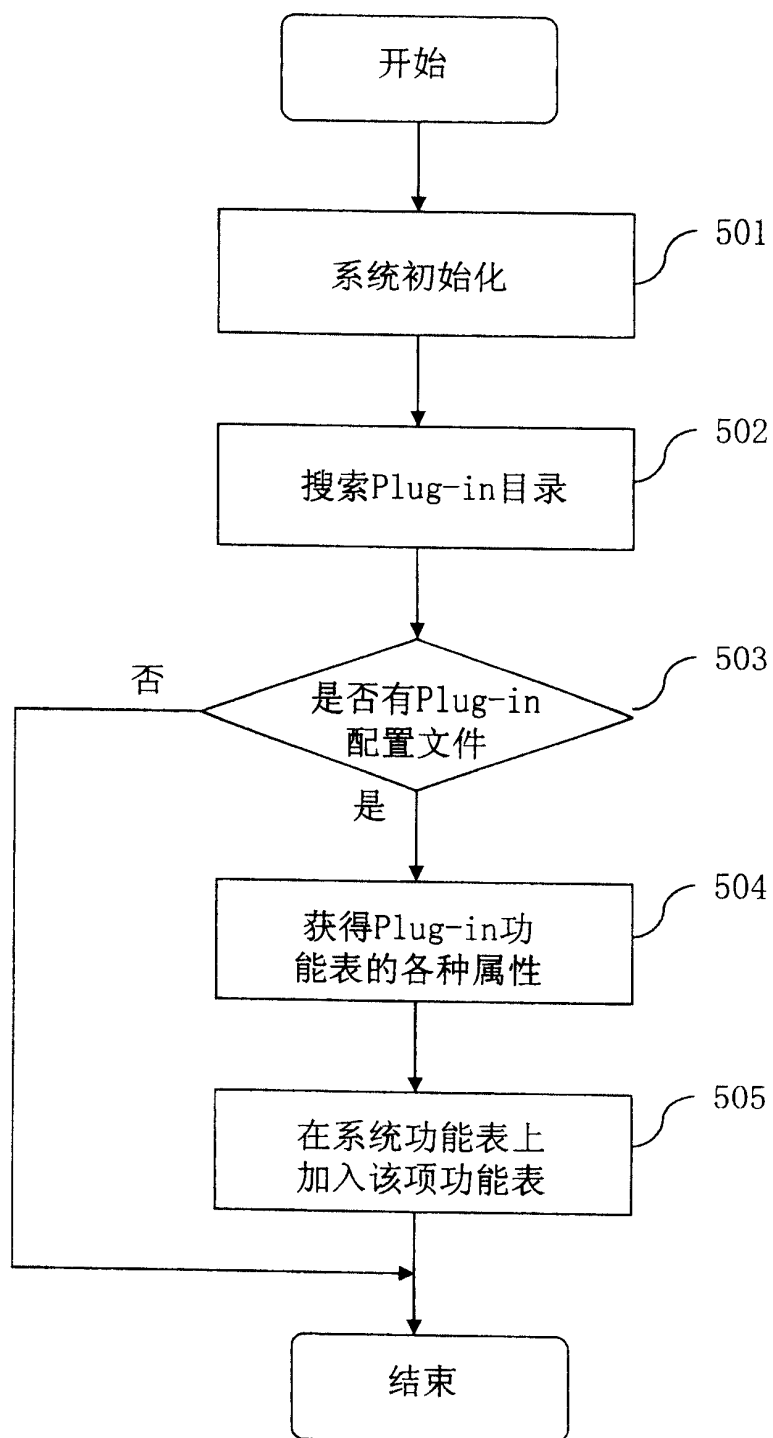


图5

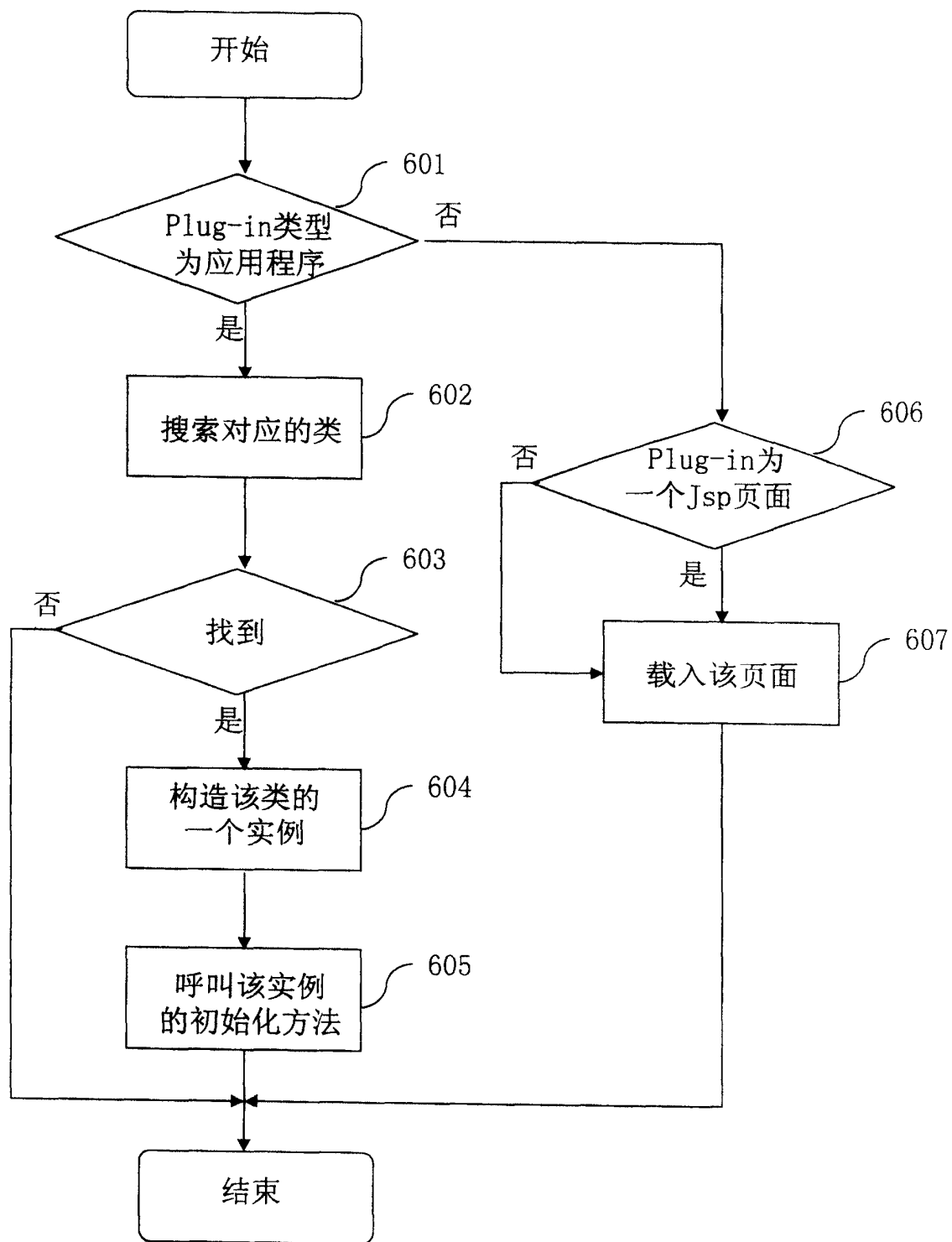


图6