(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102339219 B
(45)授权公告日 2016.08.24

(21)申请号 201010236014.5

(22)申请日 2010.07.20

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 李海军 徐新 孙鹏
V·埃格罗夫 杜宏伟

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 袁玥

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

(56)对比文件
US 2010138448 A1,2010.06.03,说明书第[0002]－[0066]段.
CN 1945527 A,2007.04.11,全文.
CN 101339500 A,2009.01.07,全文.
Douglas Bullard.Object-Oriented Ant Scripts for the Enterprise.《JavaOne conference 2009》.2009,

审查员 杨蕊

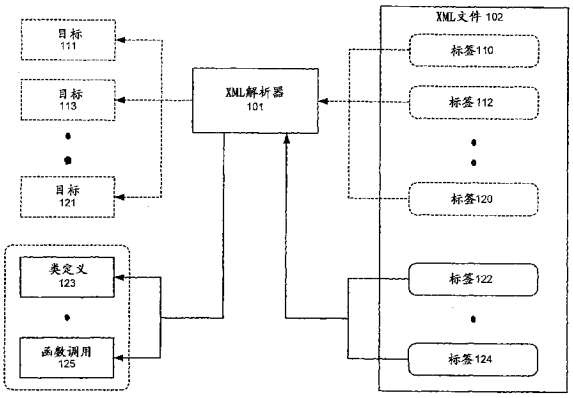
权利要求书3页 说明书5页 附图4页

(54)发明名称

用于支持面向对象脚本工具的系统和方法

(57)摘要

公开了用于支持使用XML文件的面向对象脚本工具的系统和方法。一种面向对象脚本工具使用XML文件进行软件开发和域管理。该XML文件包括以面向对象脚本语言定义脚本类的至少一个第一标签。该脚本类包括以该XML文件中的第二标签定义的至少一个方法。通用的软件开发和域管理脚本可以封装到通用脚本类中,该通用脚本类可以扩展到用于特定软件开发和域管理任务的个体脚本类。



CN 102339219 B

1.一种支持使用XML文件的面向对象脚本工具的方法,包含:

使用该XML文件中的第一标签以脚本语言定义软件类,其中该软件类包括至少一个方法;

在该XML文件中的第二标签中参考所述至少一个方法,其中所述第二标签与用于调用该至少一个方法的语法相关联,所述语法被存储在类型定义表中;

使用解析器来解析该XML文件;

确定所述第二标签是该XML文件中的未知元素;

使用所述第二标签作为关键字来从所述类型定义表中获得所述用于调用该至少一个方法的语法;以及

使用所获得的语法以该XML文件中的所述第二标签调用该至少一个方法。

2.根据权利要求1所述的方法,还包含:

允许该脚本语言是Ant。

3.根据权利要求1所述的方法,还包含:

使用XML解析器来解析该XML文件。

4.根据权利要求1所述的方法,还包含:

使用该XML文件中的标签定义该至少一个方法。

5.根据权利要求1所述的方法,还包含:

使用该XML文件中的标签检查一个软件对象是否是该软件类的实例。

6.根据权利要求1所述的方法,还包含:

支持以下至少之一:

类扩展和继承,

方法覆盖,

实例多态性,和

特定“this”和“super”实例。

7.根据权利要求1所述的方法,还包含:

支持以该XML文件的所述第二标签进行用于实例方法调用的语法,其中该语法的格式为“[类参考标号]·[方法名称]”。

8.根据权利要求7所述的方法,还包含:

使用所述类型定义表来定义所述用于实例方法调用的语法。

9.根据权利要求1所述的方法,还包含:

将通用应用构建脚本封装到所述软件类中,以及

执行另一软件类中的特定应用构建脚本,其中该另一软件类扩展或继承自所述软件类。

10.根据权利要求1所述的方法,还包含:

将通用域管理脚本封装到该软件类中,以及

使用另一软件类中的域管理脚本启动特定域,其中该另一软件类扩展或继承自该软件类。

11.根据权利要求1所述的方法,还包含:

允许该软件类是工具类,该工具类能够接受另一软件类的实例作为自变量。

12.根据权利要求1所述的方法，还包含：

使用该面向对象脚本工具以链接到数据库。

13.根据权利要求7所述的方法，还包含：

使用该面向对象脚本工具将应用部署到不同的服务器。

14.一种支持使用XML文件的面向对象脚本工具的系统，包含：

用于使用该XML文件中的第一标签以脚本语言定义软件类的装置，其中该软件类包括至少一个方法；

用于在该XML文件中的第二标签中参考所述至少一个方法的装置，其中所述第二标签与用于调用所述至少一个方法的语法相关联，所述语法被存储在类型定义表中；

用于使用解析器来解析该XML文件的装置；

用于确定所述第二标签是所述XML文件中的未知元素的装置；

用于使用所述第二标签作为关键字来从所述类型定义表中获得所述用于调用所述至少一个方法的语法的装置；以及

用于使用所获得的语法以该XML文件中的所述第二标签调用该至少一个方法的装置。

15.根据权利要求14所述的系统，还包含：

用于允许该脚本语言是Ant的装置。

16.根据权利要求14所述的系统，还包含：

用于使用XML解析器来解析该XML文件的装置。

17.根据权利要求14所述的系统，还包含：

用于使用该XML文件中的标签定义该至少一个方法的装置。

18.根据权利要求14所述的系统，还包含：

用于使用该XML文件中的标签检查一个软件对象是否是该软件类的实例的装置。

19.根据权利要求14所述的系统，还包含：

用于支持以下至少之一的装置：

类扩展和继承，

方法覆盖，

实例多态性，和

特定“this”和“super”实例。

20.根据权利要求14所述的系统，还包含：

用于支持以该XML文件的所述第二标签进行用于实例方法调用的语法的装置，其中该语法的格式为“[类参考标号]·[方法名称]”。

21.根据权利要求20所述的系统，还包含：

用于使用所述类型定义表来定义所述用于实例方法调用的语法的装置。

22.根据权利要求14所述的系统，还包含：

用于将通用应用构建脚本封装到所述软件类中的装置，以及

用于执行另一软件类中的特定应用构建脚本的装置，其中该另一软件类扩展或继承自所述软件类。

23.根据权利要求14所述的系统，还包含：

用于将通用域管理脚本封装到该软件类中的装置，以及

用于使用另一软件类中的域管理脚本启动特定域的装置，其中该另一软件类扩展或继承自该软件类。

24.根据权利要求14所述的系统，还包含：

用于允许该软件类是工具类的装置，该工具类能够接受另一软件类的实例作为自变量。

25.根据权利要求14所述的系统，还包含：

用于使用该面向对象脚本工具以链接到数据库的装置。

26.根据权利要求20所述的系统，还包含：

用于使用该面向对象脚本工具将应用部署到不同的服务器的装置。

用于支持面向对象脚本工具的系统和方法

[0001] 著作权声明

[0002] 本专利文件的公开的一部分包含受到著作权保护的内容。当本公开的专利文件出现在专利商标事务所专利文件或记录中时，著作权人不反对任何人对其进行复制，但是在其他情况，保留所有的著作权权利。

技术领域

[0003] 本发明一般涉及用于软件开发和域管理的脚本工具，且尤其涉及基于XML的脚本工具。

背景技术

[0004] 脚本工具是一种使用高级脚本语言实现的软件工具。典型地，以脚本工具编写的脚本可以在运行时由执行环境解释以执行特定编程任务。

[0005] 这种脚本工具的一个示例是Ant(或者“Another Neat tool”，另一种整洁的工具)。Ant是一种最初为自动化软件构建过程而开发的软件工具。Ant使用JAVA编程语言实现。Ant可用于在JAVA平台中构建JAVA项目。典型的Ant脚本文件具有XML文件格式。

发明内容

[0006] 根据一实施例，一种面向对象的脚本工具使用XML文件来进行软件开发和域管理的。该XML文件包括以面向对象脚本语言定义脚本类的至少一个第一标签。该脚本类包括以该XML文件中的第二标签定义的至少一个方法。通用软件开发和域管理脚本可以封装到通用脚本类中，该通用脚本类可以扩展为用于特定软件开发和域管理任务的个体脚本类。

附图说明

[0007] 图1示出面向对象Ant脚本工具环境的示例性视图。

[0008] 图2是示出根据一实施例在面向对象Ant脚本语言和JAVA编程语言中语法之间的样例映射关系的图。

[0009] 图3是示出根据一实施例用于构建软件应用的面向对象Ant类层级的示例的图。

[0010] 图4是示出根据一实施例用于构建应用的面向对象Ant脚本类的样例用法的图。

[0011] 图5是示出根据一实施例用于管理应用服务器域的类层级的示例的图。

[0012] 图6是示出根据一实施例用于管理应用服务器域的面向对象Ant脚本类的样例用法的图。

[0013] 图7是示出根据一实施例的面向对象Ant工具类的样例用法的图。

[0014] 图8是示出根据一实施例用于在分布式环境中测试软件应用的示例性步骤的图。

具体实施方式

[0015] 在附图中仅以示例方式而绝非限制性方式说明本发明，其中，相似的附图标记表

示相似的元件。应当注意，在本公开中，对“一”或“一个”或“一些”实施例的引用并不一定表示引用相同的实施例，且这种引用表示至少一个。

[0016] 下面的本发明的实施例的描述使用JAVA平台作为面向对象编程语言平台的示例。本领域技术人员应当意识到，可以无限制地使用其他类型的面向对象编程语言平台。

[0017] 根据一实施例，诸如Ant的基于XML的脚本工具可以扩展为包括面向对象的特征。在一个实施例中，面向对象脚本工具可以实现为基于XML的脚本工具的扩展。例如，面向对象Ant脚本工具或面向对象Ant可以使用JAVA编程语言实现为Ant的扩展。在一个实施例中，实现面向对象Ant脚本工具的JAVA代码可以被编译且部署在由Ant脚本环境指定的库目录中，从而存储该Ant脚本工具的扩展任务。

[0018] 图1示出面向对象Ant脚本工具环境的示例性视图。

[0019] 如图1所示，Ant脚本环境可以使用XML解析器101来解析包含在具有不同标签110、112和120的XML文件102中的脚本。XML文件102中的每个标签110、112或120与一类脚本任务或目标111、113或121相关联。

[0020] 根据一实施例，诸如面向对象Ant的面向对象脚本工具可以利用包括面向对象脚本语法的附加标签122和124来支持面向对象特征。如图1所示，标签122可用于支持面向对象Ant脚本环境中的类定义任务123；且另一标签124可用于支持面向对象Ant脚本环境中的函数调用任务125。

[0021] 根据一实施例，面向对象脚本语法支持使用可以在类型定义表中保存的不同的面向对象脚本类型。该类型定义表可以使用哈希(harsh)表实现，其中标签名称用作哈希表的关键字。另外，面向对象Ant脚本工具的每个任务实例可以存储在实例池中。该实例池也可以以使用每个实例的id或参考标号作为关键字的哈希表实现。

[0022] 在另一实施例中，这种类型定义机制可以被解释为低优先级的，从而防止与软件开发者用于其他目的的相同标签的其他用法存在名称冲突。在另一示例中，可以定义唯一标签以启动面向对象Ant脚本环境，从而防止名称冲突。

[0023] 根据一实施例，面向对象脚本工具可以具有面向对象编程语言提供的类似特征。而且，熟悉面向对象编程语言的软件开发者可以容易地理解面向对象脚本语言。例如，基于Ant脚本语言的面向对象Ant可以具有与JAVA类似的特征。这允许熟练的JAVA开发者快速学会使用面向对象Ant脚本工具。

[0024] 图2示出在面向对象Ant脚本语言的语法和JAVA编程语言的语法之间的样例映射关系。

[0025] 根据一个实施例，面向对象脚本工具可以使用类标签来定义脚本类，就像JAVA编程语言可以定义JAVA类一样。例如，如图2中第1行所示，在面向对象Ant脚本语言中定义样例“Hello”类，其可以映射到JAVA编程语言的Hello类。

[0026] 根据一个实施例，面向对象脚本工具中的类标签可以包含封装脚本类的行为的任意脚本。在一个实施例中，该类标签可以存在两个属性。一个属性是定义底层类的名称的名称属性。另一属性是定义基类的类名称的基本属性。基本属性的缺省值是“对象”，其表示底层类是从根类直接扩展的类。

[0027] 根据一实施例，类似于在JAVA类中定义方法，面向对象脚本工具可以使用方法标签来为类定义方法。在一个实施例中，方法标签可以具有名称属性。例如，如图2中第3行所

示，该方法的名称属性被赋予值“sayHi”。另外，方法标签还允许在方法标签的实体中定义一个或多个属性标签。这种属性标签定义当方法被调用时可以由该方法使用的一个或多个参数。在如图2中第4行所示的实例中，方法“sayHi”采用指定消息的内容的参数。

[0028] 根据一实施例，面向对象编程工具可以为类分配参考值。例如，在图2中第7行，标签可用于类的启动，这类似于在JAVA中创建新类。使用图2中第1行的标签创建的“Hello”类的实例被赋予了参考值“h”以唯一地识别该实例。

[0029] 根据一实施例，面向对象脚本工具允许用户以“[类参考标号]-[方法名称]”的格式使用类调用标签来调用脚本类中的方法。例如，如图2中第9行所示，可以使用“h.sayHi”的语法调用在图2中第3-5行定义的“sayHi”方法，这重新汇编了JAVA中的类调用语法。该“sayHi”方法设置有包括消息的内容的参数，符合图2中第4行消息属性的定义。

[0030] 根据一实施例，面向对象脚本工具提供可以写出消息内容的回响(echo)标签。例如，如图2的第11行所示，回响标签可以以类似于JAVA中的System.out的方式操作。

[0031] 根据一实施例，面向对象脚本工具提供可用于检查某面向对象Ant对象是否是某特定Ant类类型的<instance of>标签。在一个实施例中，<instance of>标签提供描述Ant对象id的“id”属性；以及描述要验证的Ant类的名称的“classname”属性。

[0032] 根据一实施例，就像面向对象编程语言一样，面向对象脚本工具可以提供其他有用的特征。就像JAVA提供的那样，这些有用的特征包括允许方法覆盖以及特定“this”和“super”实例。而且，类似于JAVA，面向对象Ant可以支持类扩展和继承；实例方法调用语法的反射特征以及获得实例属性的语法。

[0033] 根据一实施例，面向对象脚本工具可用于构建软件应用。通用应用构建脚本可封装到类中。软件开发者然后可以创建他或她自己版本的通用应用构建脚本，以包括他或她希望的某些特定特征。

[0034] 图3示出用于构建软件应用的类层级的示例。如图3所示，App.class.xml301是封装了通用应用构建脚本的基类。MyAppBase.class.xml 302是从基类App.class.xml 301继承的类。软件开发者不需要重写或复制和粘贴App.class.xml 301中已经存在的实现细节。软件开发者仅需要在MyAppBase.class.xml 302中添加新方法或覆盖已有方法。

[0035] 另外，基于MyAppBase.class.xml 302，软件开发者可以针对不同项目创建不同的面向对象Ant脚本类MyApp1.class.xml 303和MyApp2.class.xml 304。不同面向对象Ant脚本类MyApp 1.class.xml303和MyApp2.class.xml 304包括所有必要信息以成功编译和构建项目，诸如项目目录和到有用库的链接。

[0036] 图4示出用于构建应用的面向对象Ant脚本类的样例用法。如图4所示，两个单独的面向对象Ant脚本类MyApp1和MyApp2可用于构建两个单独的应用：app1和app2。

[0037] 根据一实施例，面向对象脚本工具还可用于管理不同网络或应用服务器域。

[0038] 图5示出用于管理应用服务器域的类层级的示例。如图5所示，通用域相关脚本可以封装成类Domain.class.xml 501。网络管理员可以在另一个类MyDomainBase.class.xml 502中指定他或她自己版本的通用域相关脚本。MyDomainBase.class.xml 502包括符合网络管理员的定制特征。此处，MyDomainBase.class.xml 502是继承了基类Domain.class.cml 501的类。在这种场景中，网络管理员不需要重写或复制和粘贴在通用域管理教本中已经存在的实现细节。软件开发者仅需要根据他或她自己的需要添加新方法

或覆盖现有方法。

[0039] 另外，基于MyDomainBase.class.xml 502，网络管理员可以针对特定域创建单独的面向对象Ant脚本类MyDomain1.class.xml 503。此处，面向对象Ant脚本类包括所有必要信息以成功准备和启动该特定域。

[0040] 图6示出用于管理应用服务器域的面向对象Ant脚本类的样例用法。如图6所示，可以使用该面向对象Ant脚本类准备和启动应用服务器域MyDomain1。

[0041] 根据一实施例，面向对象脚本工具可用于生成可接受收其他类型脚本类的对象作为自变量的工具脚本类。

[0042] 图7示出面向对象Ant工具类的样例用法。如图7所示，工具Ant类“MyApp1Util”类具有参考标号或id“applutil”。“MyApp1Util”类具有可通过参考标号或id“app1”获得“MyApp1”的所有实例属性的处理方法。可以基于获得的“MyApp1”类的实例属性执行其他操作。这些操作包括但不限于管理部署描述符文件等。

[0043] 根据一实施例，面向对象脚本工具可用于分布式环境中的软件应用测试，诸如设置应用服务器的域和链接到数据库。

[0044] 图8示出用于在分布式环境中测试软件应用的示例性步骤。如图8所示，该软件应用测试脚本可以包括以下步骤：首先编译该应用801，并且针对不同目的开发和编译该应用的不同测试案例802。然后，需要准备803且启动804目标域，以准备好在该域中的服务器上部署应用805。在完成测试案例806之后，可以关闭807且准备清理808域。

[0045] 由于软件测试的复杂性，可以开发不同测试案例。对于每个测试方案可能存在大量的脚本文件。从一个测试案例到另一个测试案例迁移脚本文件原先涉及通过复制和粘贴途径使用大量类似测试脚本。复制和粘贴方法是低效的，因为测试脚本可能非常大且涉及大量细节。很多时候，软件开发者甚至愿意重写测试脚本而不重新使用旧测试脚本以避免错误。通过使用面向对象Ant，软件开发者可以容易地从一个测试案例延伸到另一测试案例而无需复制和粘贴海量代码。

[0046] 根据一实施例，在Ant中引入面向对象语法使得更加容易地重复使用现有脚本。例如，Ant宏“A”可以内部地调用另一宏“B”。在一个实例中，软件开发者只希望改变“B”的行为且重复使用“A”。使用复制和粘贴方法，软件开发者需要创建两个新的宏“A1”和“B1”。软件开发者需要将“A”的内容复制到“A1”中且将调用“B”改为调用“B1”。使用面向对象Ant，软件开发者可以简单地使用新类“B1”扩展基类“B ”，该新类“B1”使用新实现方式覆盖“B”中的一个或多个方法。

[0047] 使用本公开的教导编程的常规通用或专用数字计算机或微处理器可以方便地实现本发明。软件领域相关人员应当意识到，基于本公开的教导，熟练的程序员容易编写适当的软件代码。

[0048] 在一些实施例中，本发明包括计算机程序产品，该计算机程序产品是其上存储有指令且该指令可用于对计算机进行编程以执行本发明的任意处理的存储介质。存储介质可以包括但不限于任意类型的盘，包括软盘、光盘、DVD、CD-ROM、微驱动器以及磁光盘、ROM、RAM、EPROM、EEPROM、DRAM、VRAM、闪存装置，磁或光卡、纳米系统(包括分子存储器IC)或适于存储指令和/或数据的任意类型的介质或装置。

[0049] 提供本发明的上述描述是用于说明和描述目的。其并不是穷举的或旨在将本发明

限制为所公开的精确形式。对本领域技术人员而言显见很多修改和变型。给出的代码实例是出于说明的目的。很明显，可以使用其他代码语言且使用不同代码应用此处描述的技术。

[0050] 选择和描述实施例是为了最好地解释本发明的原理及其实践应用，由此使得本领域技术人员理解本发明的各种实施例以及可以预期的适于特定用途的各种变型。旨在表明，本发明的范围受到下面的权利要求及其等价物的限定。

[0051] 根据本发明的一个实施例，提供了一种计算机可读介质，其上存储有指令，在执行该指令时该指令使系统：使用XML文件中的第一标签以脚本语言定义软件类，其中该软件类包括至少一个方法；以及以该XML文件中的第二标签调用该至少一个方法。

[0052] 根据本发明的另一个实施例，提供了一种支持使用XML文件的面向对象脚本工具的系统，包含：该XML文件中的第一标签，其以脚本语言定义软件类，其中该软件类包括至少一个方法；以及该XML文件中的第二标签，其调用该软件类中的该至少一个方法。

[0053] 根据本发明的另一个实施例，提供了一种支持使用XML文件的面向对象脚本工具的系统，包含：用于使用该XML文件中的第一标签以脚本语言定义软件类的装置，其中该软件类包括至少一个方法；以及用于以该XML文件中的第二标签调用该至少一个方法的装置。

[0054] 优选地，该系统还包含：用于允许该脚本语言是Ant的装置。

[0055] 优选地，该系统还包含：用于使用XML解析器来解析该XML文件的装置。

[0056] 优选地，该系统还包含：用于使用该XML文件中的标签定义该至少一个方法的装置。

[0057] 优选地，该系统还包含：用于使用该XML文件中的标签检查一个软件对象是否是该软件类的实例的装置。

[0058] 优选地，该系统还包含：用于支持以下至少之一的装置：类扩展和继承，方法覆盖，实例多态性，和特定“this”和“super”实例。

[0059] 优选地，该系统还包含：用于支持以该XML文件的标签进行实例方法调用的语法的装置，其中该语法的格式为“[类参考标号].[方法名称]”。

[0060] 优选地，该系统还包含：用于当XML文件中的一个标签被确定为是未知元素时，将该标签与用于实例方法调用的语法相关联的装置。

[0061] 优选地，该系统还包含：用于使用类型定义表来定义所述用于实例方法调用的语法的装置。

[0062] 优选地，该系统还包含：用于将通用应用构建脚本封装到所述软件类中的装置，以及用于执行另一软件类中的特定应用构建脚本的装置，其中该另一软件类扩展或继承自所述软件类。

[0063] 优选地，该系统还包含：用于将通用域管理脚本封装到该软件类中的装置，以及用于使用另一软件类中的域管理脚本启动特定域的装置，其中该另一软件类扩展或继承自该软件类。

[0064] 优选地，该系统还包含：用于允许该软件类是工具类的装置，该工具类能够接受另一软件类的实例作为自变量。

[0065] 优选地，该系统还包含：用于使用该面向对象脚本工具以链接到数据库的装置。

[0066] 优选地，该系统还包含：用于使用该面向对象脚本工具将应用部署到不同的服务器的装置。

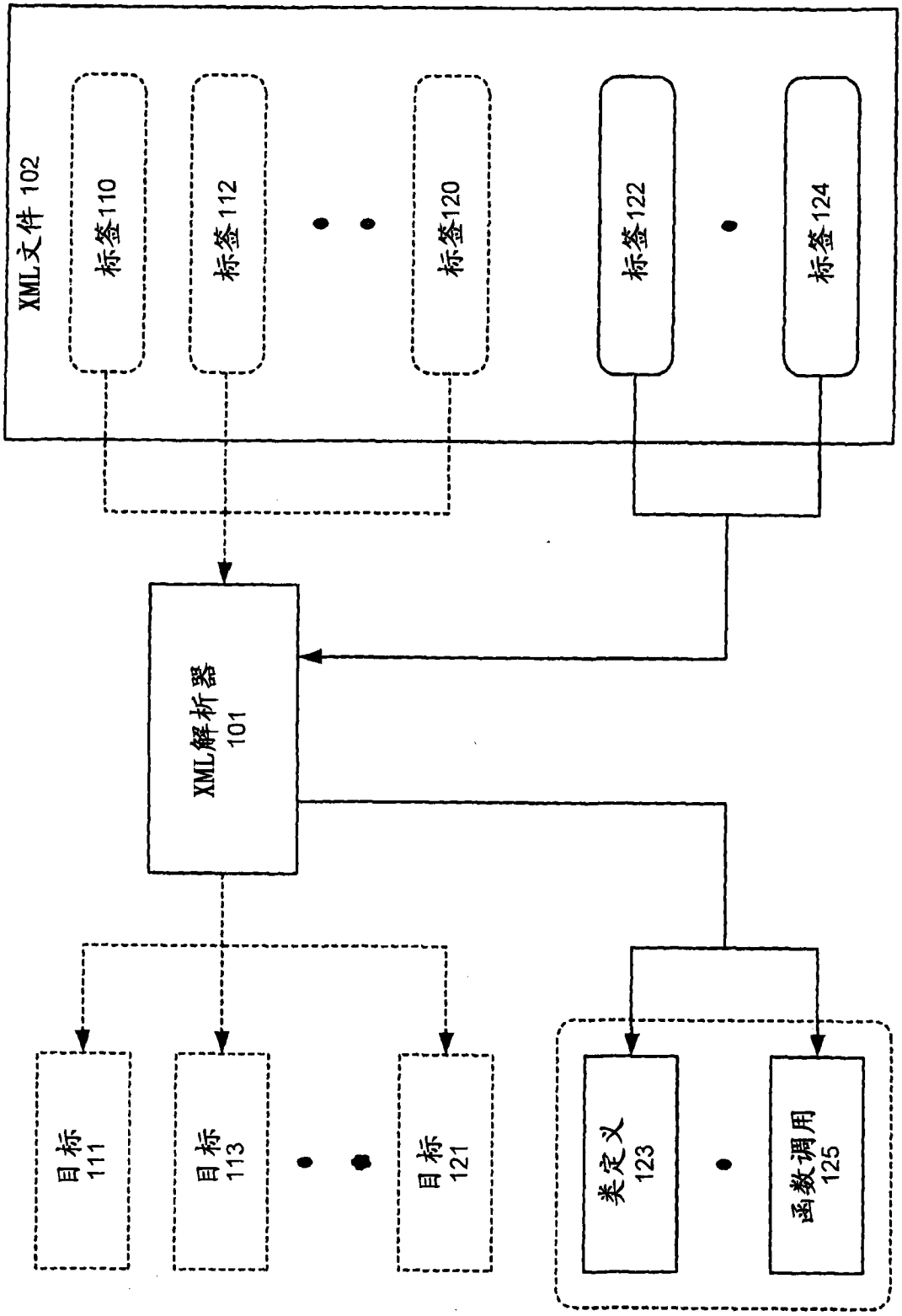

XML文件 102
标签110
标签112
标签120
标签122
标签124
XML解析器
101
目标
111
目标
113
目标
121
类定义
123
函数调用
125

图1

面向对象Ant

```
1  <class name="Hello"> ... </Class>
2
3  <Method name="sayHi">
4      <attribute name="message"/>
5  <Method/>
6
7  <Hello.init id="h"/>
8
9  <h.sayHi message="Hi"/>
10
11 <echo message="${h.message}"/>
12
13 <super.sayHi message="Hi"/>
14
15 <this.sayHi message="Hi"/>
16
```

JAVA

```
public class Hello  {...}

public void sayHi(String message)

Hellow h = new Hello();

h.sayHi("Hi");

System.out.println(h.getMessage());

super.sayHi(Hi");

this.sayHi(Hi");
```

图2

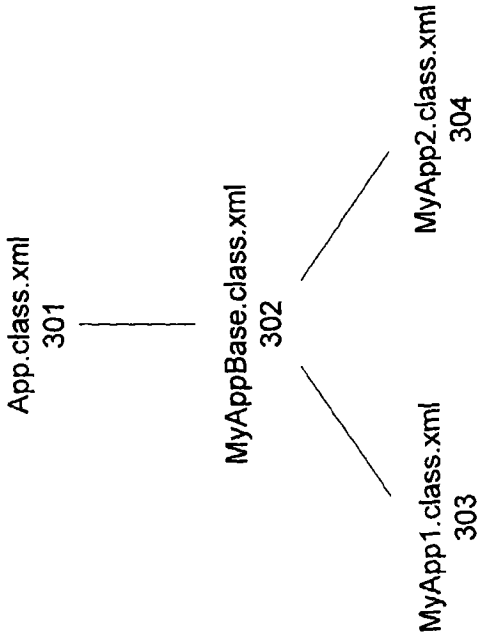


图3

```
<MyApp1.init id="app1" dir=""/>
<MyApp2.init id="app2" dir=""/>

<app1.build/>
<app2.build/>
```

图4

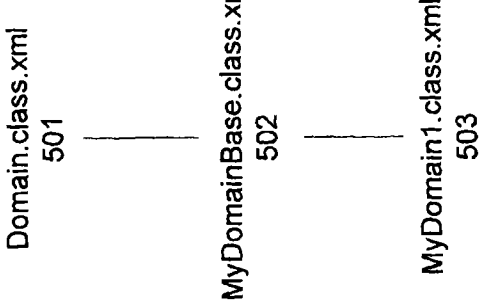


图5

```
<MyDomain1.init id="domain1" domainNmae="xxxDomain"/>

<domain1.prepare/>
<domain1.startup/>
```

图6

```
<MyApp1.init id="app1" dir=""/>
<MyApp1Util.init id="applutil"/>
<applutil.process app="app1"/>
```

图7

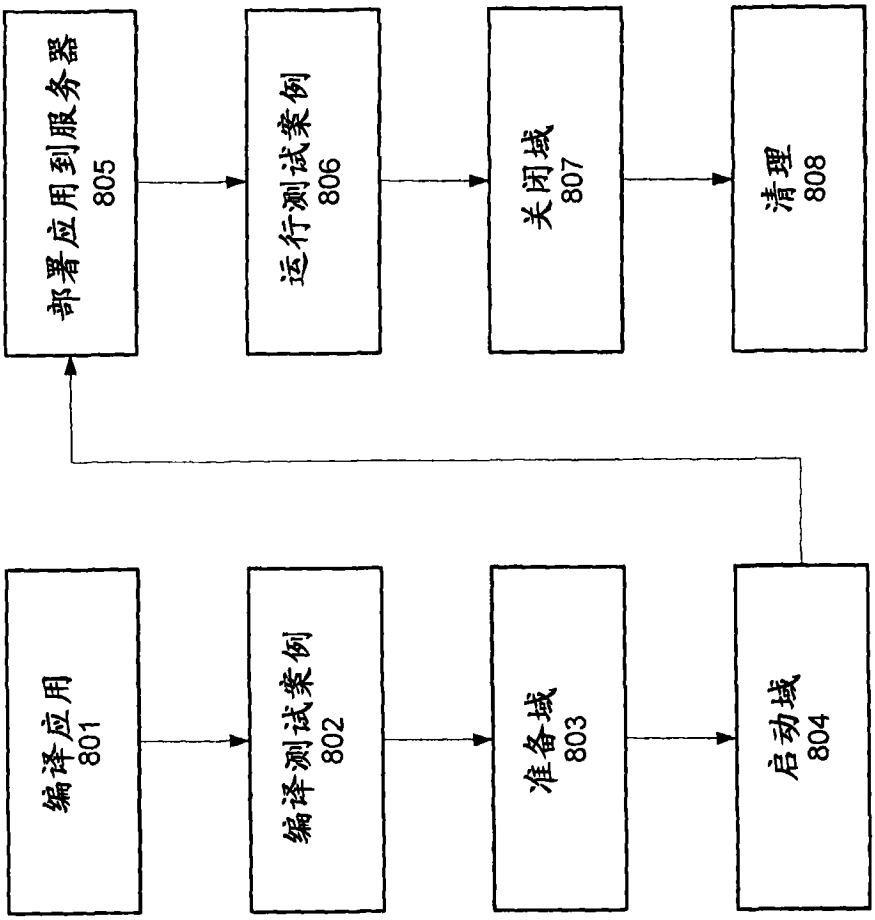

编译应用
801
编译测试案例
802
准备域
803
启动域
804
部署应用到服务器
805
运行测试案例
806
关闭域
807
清理
808

图8