基于对象视图模型 WebView 的 Web 应用框架*

张 波, 冯玉琳, 黄 涛

(中国科学院 软件研究所 计算机科学重点实验室,北京 100080);
(中国科学院 软件研究所 软件工程技术研究开发中心,北京 100080)
E-mail: {zb,tao}@otcaix.iscas.ac.cn; feng@ios.ac.cn
http://www.iscas.ac.cn

摘要: 作为 Web 应用的基础,资源模型的抽象能力明显不足,使得成熟的软件开发方法无法被应用到 Web 应用的开发过程中.提出了 Web 应用的对象视图模型 WebView,并在现有 Web 技术的基础上实现了基于 WebView 的 Web 应用框架.对象视图模型以对象视图作为统一的概念对 Web 实体进行建模,有效地增强了可复用性,并提高了开发效率.基于 WebView 的 Web 应用框架具有将不同类型的对象视图映射为相应的 HTML 实现的能力.

关 键 词: Web 应用;分布式计算;应用框架
中图法分类号: TP393 文献标识码: A

Web 在分布式计算中扮演着越来越重要的角色.许多分布式应用,从小规模、临时性的服务到部署在多个服务器上的大规模的企业业务系统,都以 Web 作为它们的计算平台.基于 HTML 的前端和无处不在的 Web 浏览器使得用户可以随时对这些应用进行跨平台的访问;此外,瘦客户的概念和集中的管理使得系统的部署能够即时完成,并且将系统维护的代价降到了最低限度.这些特点使得 Web 应用越来越普遍.

然而,从 Web 的发展过程和核心实现技术来看,它只是一种信息平台而不是计算平台,传统的 Web 应用开发包括信息资源的组织以及如何利用超链接实现信息资源之间的导航.作为 Web 应用的基础,资源模型本质上是一种实现级的模型,其抽象能力明显不足[1].因此,成熟的软件开发方法无法被应用到 Web 应用的开发过程中.这使得目前 Web 应用的开发和维护在很大程度上依赖于开发者个人的知识和经验.

针对上述问题,本文提出了 Web 应用的对象视图模型 WebView,在该模型中,我们将一个 HTML 页面视为多个对象视图的集合,以对象视图作为统一的概念对 Web 实体进行建模.与传统的资源模型相比,对象视图以一种更加自然的方式,提供不同粒度的 Web 实体抽象,因而提高了抽象能力.作为对象视图模型在现有 Web 技术基础上的一种实现,基于 WebView 的 Web 应用框架具有将不同类型的对象视图映射为相应的 HTML 实现的能力.

本文第 1 节在对 Web 平台进行分析的基础上,介绍了 Web 应用的对象视图模型 WebView.第 2 节介绍了基于 WebView 的 Web 应用框架,包括对象视图描述语言 OVL 以及应用框架的总体结构和实现技术.第 3 节将本文的工作和其他相关工作进行了比较.最后给出结论.

1 Web 应用的对象视图模型

1.1 资源模型及扩充

Web 提供了针对各种类型信息的表示方法,并支持它们之间的相互关联,使得最终用户很容易进行信息的

* 收稿日期: 2000-12-20; 修改日期: 2001-02-21
基金项目: 国家自然科学基金资助项目(69833030);国家 863 高科技发展计划项目(863-306-ZD02-01-1)
作者简介: 张波(1972－),男,山东兖州人,博士,助理研究员,主要研究领域为分布对象计算,软件体系结构;冯玉琳(1942－),男,江苏泰县人,博士,研究员,博士生导师,主要研究领域为软件工程,形式化方法,分布对象计算;黄涛(1965－),男,江苏淮安人,博士,研究员,博士生导师,主要研究领域为分布对象计算,对象语义理论,软件工程方法学.

访问.在 Web 的实现模型中,各种自含的信息片断构成了 Web 应用的资源,一方面资源之间相对独立,另一方面还可以通过关联组合成一个完整的文档或文档集合.在这种资源模型下,Web 应用的开发主要包含如何将信息组织为相互关联的信息片断并利用超链接实现这种关联.然而,资源模型本质上是一种实现级的模型,缺乏必要的抽象能力,因而使得 Web 应用开发更类似于作品出版而不是软件开发.

近年来,Web 应用远远超出了传统方式下的以文档为中心的范畴,Web 的实现技术与实现机制不断地被扩充以使其能够支持功能性的计算任务.有关 Web 的扩充可以有如下的分类[2]:

(1) 服务器端的扩充

通过 CGI 和 ISAPI 等 Web 服务器应用程序接口,用户可以访问服务器上其他的应用,例如数据库的查询等.在此基础上产生了 ASP 和 Servlet 等服务器端的网页动态生成技术,使得网页的内容可以根据用户的要求被动地生成.本质上讲,这些扩充只是为传统应用提供了 Web 方式的访问接口,所有的计算都集中在服务器端,浏览器仅仅起到一种易于使用的图形化终端的作用.

(2) 浏览器端的扩充

在页面中嵌入 Java applet 或 ActiveX 控件使客户端具备了一定的计算能力[3].然而,由于 Web 应用自身的特点,在客户端计算能力提高的同时也有可能给用户带来意想不到的灾难性后果.出于对安全性问题的考虑,Java applet 和 ActiveX 控件的使用往往会受到很多的限制.此外,一些浏览器支持某些插件(plug-in),通过其他协议与相应的服务器连接,这种与浏览器相关的扩充实际上已经超出了 Web 技术的范围.

(3) 与中间件相结合

采用 CORBA 或 JavaBeans 等中间件作为服务器端和客户端之间计算的互操作平台,使计算任务的合理分配成为可能[4,5].Web 应用的各部分之间通过 IIOP 等协议实现远程方法调用或通过消息队列等机制实现消息传递.然而在这种架构下,由于中间件的实现模块在 Java applet 或 ActiveX 控件运行时需要被下载到浏览器,实际上 Web 更多地起到了分布式应用的动态部署工具的作用.

从以上分析我们可以看出,虽然 Web 的实现技术与机制不断被扩充,但是这些扩充并没有改变 Web 技术的本质,Web 应用的基础仍然是实现级的资源模型.资源模型较低的抽象能力不利于 Web 应用的开发.如何提高 Web 应用的基础模型的抽象能力,成为 Web 应用开发和维护的关键问题.

1.2 对象视图模型WebView

为了提高抽象能力,我们采用对象视图模型作为 Web 应用的基础模型.在对象视图模型中,Web 页面由多个对象视图组成,对象视图是某个对象的内部状态的可视化表示.此时,我们称该对象为该对象视图的源对象.一个对象视图可以是一个具有独立含义的 HTML 文档片段(如一个 Form),也可以由一个 XML 文档和相对应的 XSL 文档构成,其中 XML 表示对象视图的数据内容,XSL 表示数据的显示方式.

为了支持 Web 平台的计算能力,在对象视图模型中,对象视图分为三种类型:常量视图、静态视图和动态视图.下面分别介绍这 3 种视图.

- 常量视图是一种不变的视图,无论何时访问包含该视图的页面,该视图总是一样的.例如,我们可以定义某个公司的网页中用于填写客户信息的表单为一个常量视图.通常情况下,常量视图没有源对象.
- 静态视图是某个对象在某一时刻的内部状态的可视化表示,当包含某静态视图的页面被访问时,将获取其源对象的某些内部状态,并对这些内部状态进行可视化表示.因此,与常量视图不同,页面的刷新将可能导致静态视图的改变.
- 动态视图具有一定的计算能力,通常利用 Java applet 或 ActiveX 控件来实现.动态视图不仅具有可视化的表示,而且可以向源对象发出服务请求并接收响应.因此,即使页面不刷新,动态视图也会发生变化.由于具有处理用户输入的能力,因此动态视图可以作为用户与应用系统之间进行交互的可视化界面.

与资源模型相比,对象视图模型能够以更加自然的粒度对 Web 实体进行抽象.同时,对象视图还能将 Web 页面和提供服务的对象系统有机地结合起来.作为一组对象视图的集合,Web 页面的设计不再局限于信息的组织,而是作为对象系统设计的一种自然延伸.在开发过程中,开发者能够以独立于应用的方式对某个对象视图进行

设计与实现,并把该对象视图运用到其他应用中,因而能够有效地增强可复用性并提高开发效率.对象视图与源对象相分离,保证了两者在访问接口不变的情况下,可以进行独立的修改,因此体现了良好的封装性.同时,静态视图和动态视图保持了 Web 现有的支持功能性计算的能力,另一方面也保证了对象视图模型在实现方面的可行性.

2 基于 WebView 的 Web 应用框架

从本质上来说,对象视图模型是一种设计级的模型,与实现级的资源模型相比,它具有更好的抽象能力.然而,对象视图模型的实现必须依赖于现有的 Web 技术.基于 WebView 的 Web 应用框架是对象视图模型在现有的 Web 技术基础上的一种实现.

2.1 对象视图描述语言OVL

对象视图描述语言是一种基于 XML 的语言,用于定义对象视图.在 OVL 文档中,可以包含多个对象视图的定义.每个对象视图都具有 name 属性和 type 属性,分别表示其名字和类型.对象视图的名字在其所在的文档中是惟一的,视图的类型表明该视图是常量视图,还是静态视图或动态视图.

对于常量视图,需用 *Format* 标签来指明该常量视图是一个 HTML 文档片段还是由 XML 文档和 XSL 文档构成的.若是后者,其属性 meta 指明了 XML 文档所满足的 DTD.*Content* 标签的 ref 属性的值为 HTML 文档或 XML 文档的名字;若是 XML 文档,则用 *Display* 标签的 ref 属性表示 XSL 文档的名字.关于这两种定义方式,请参见例 1 中两个对象视图的定义.对 *Content* 和 *Display* 标签来说,若其 ref 属性值为空,则可把相应的内容嵌在该标签内部.

例 1.常量视图的定义(customer.ovl)

```
〈View name='VCustomers1' type='Constant'〉
    〈Format meta='c.dtd'〉XML〈/Format〉
    〈Content ref='c.xml'/〉
    〈Display ref='c.xsl'/〉
〈/View〉
〈View name='VCustomers2' type='Constant' 〉
    〈Format〉HTML〈/Format〉
    〈Content ref='c.html'/〉
    〈Display ref=''/〉
〈/View〉
```

在静态视图中,*Format* 标签和 *Display* 标签的含义与常量视图中的相同,而 *Content* 标签用来表示数据来源.其中,*Object* 标签的 name 属性指明了源对象的名字,*Method* 标签的 name 属性的值为该对象的一个方法,在该对象的定义中,要求该方法的返回值是字符串类型,并且其格式满足 XML 或 HTML 的语法,而 *Format* 标签应该与该格式一致.*Param* 标签可以有多个,其属性 name 和 value 分别表示该方法的某个参数的名字和值.对象视图描述语言中定义了一些缺省的函数和运算以支持对参数值的计算,限于篇幅,这里不再介绍.静态视图的定义请参见例 2.

例2.静态视图的定义(price.ovl)

```
〈View name='VPrice' type='Static'〉
    〈Format meta='p.dtd'〉XML〈/Format〉
    〈Content〉
        〈Object name='VStock'/〉
        〈Method name='PriceQuery'/〉
        〈Param name='StockID' value=Request.Form('StockID')/〉
```

〈/*Content*〉

〈*Display ref=‘p.xsl’*/〉

〈/*View*〉

动态视图由 Java applet 或 ActiveX 控件来实现,因此其定义相对简单.我们以 Applet 为例,如例 3 所示,*Object* 标签的 name 属性指明了源对象的名字,而 *Applet* 标签用于指明该动态视图所对应的 Applet,其格式和含义与 HTML 中的定义相同.关于动态视图中的 Applet 的实现以及它与源对象之间的交互,将在第 2.2 节中详细地加以介绍.

例 3.动态视图的定义(auction.ovl)

〈*View name=‘VAuction’ type=‘Dynamic’*〉

〈*Object name=‘AuctionController’*/〉

〈*Applet Code=auction.class Width*=30 *Height*=50〉

〈*Param Name=‘ID’ Value=‘’*/〉

〈/*Applet*〉

〈/*View*〉

我们约定对象视图的定义文件的后缀为 ovl,一个 ovl 文件中可以包含一个或多个视图的定义.对象视图作为 Web 应用的构建模块,可以在应用中或其他对象视图的定义中被引用.如例 4 所示,标签 *Refview* 被用来实现这类引用,其属性 name 和 from 分别指明了视图的名字和定义该视图的文件名.由于我们在 HTML 的基础上扩充了 *Refview* 标签,所以我们以后缀 wapp 来命名扩充后的 HTML 文档.

例 4.对象视图的引用(price.wapp)

〈*HTML*〉

〈*BODY*〉

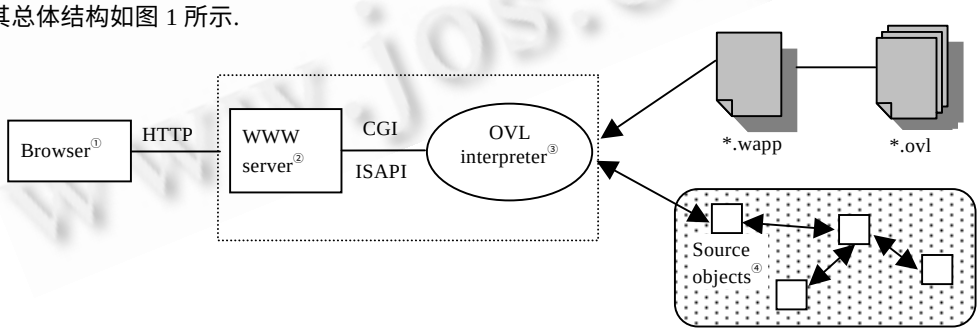
〈*Refview name=‘VPrice’ from=‘p.ovl’*/〉

〈/*BODY*〉

〈/*HTML*〉

2.2 Web应用框架的实现

OVL 提供了描述对象视图的能力,然而 Web 应用的最终实现还必须基于现有的 Web 技术,因此我们需要将对象视图的定义映射为具体的 HTML 实现.在基于 WebView 的应用框架中,OVL 解释器负责在 wapp 页面被请求时将其转换为相应的 HTML 页面,并返回给浏览器.在转换过程中,如果需要,还将请求相应源对象所提供的服务.其总体结构如图 1 所示.



①浏览器,②WWW 服务器,③OVL 解释器,④源对象.

Fig.1 The structure of the Web application framework

图 1 Web 应用框架的总体结构

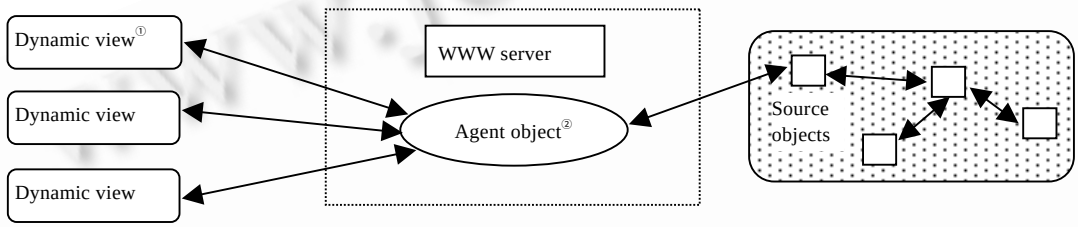
OVL 解释器将根据对象视图的类型做不同的处理.对于常量视图来说,只需直接生成相应的 HTML 片段就可以了.而对于静态视图,首先要根据 *Content* 标签中的定义向源对象发出服务请求,然后根据服务请求的结果

生成相应的 HTML 片段.源对象与 OVL 解释器不在一个进程中,要么是本地其他进程中的对象,要么是远程对象,因此解释器对源对象的方法调用是一种远程方法调用.我们将服务请求和返回结果打包成消息,并采用消息队列来进行消息传递.当然,也可以采用其他机制(例如 RPC 等),只要解释器和源对象的实现满足相应的约定就可以了.这些实现技术目前都已经相当成熟,因此这里不再介绍.

我们以 Java applet 为例讨论动态视图的实现.考虑到安全方面的限制,在应用框架的实现中禁止 Applet 与源对象之间的直接通信.Applet 与源对象的通信通过代理对象来完成,如图 2 所示.代理对象一方面负责接收 Applet 发出的对源对象的服务请求,并将请求转发给源对象,另一方面接收由源对象发出的返回结果,并将结果转发给相应的 Applet.因此对代理对象来说,每个 Applet 都有一个惟一的标识,OVL 解释器负责生成该标识,并在将动态视图映射为 HTML 片段时通过参数指派给相应的 Applet.

代理对象具有如下的生命周期:它在相应的动态视图第 1 次被请求时由 OVL 解释器创建并将计数器设为 0,对应的 Applet 在 init 操作中向代理对象申请注册,代理对象接受注册申请并将计数器加 1;Applet 在 destroy 操作中向代理对象申请注销,代理对象接受注销申请并将计数器减 1,当计数器为 0 时,代理对象被删除.

动态视图的上述实现方式具有以下的优点:(1) 限制了 Applet 只能和代理对象进行通信,因而具有良好的安全性;(2) 源对象只负责处理由代理对象发出的服务请求,避免了由处理动态视图的生命周期引起的额外开销,从而屏蔽了 Web 应用系统用户的不确定性和访问的随机性对源对象的设计和实现所造成的影响.



①动态视图,②代理对象.

Fig.2 The implementation of dynamic view

图 2 动态视图的实现

3 相关工作

本文的工作基于对 Web 应用的开发方法和体系结构的研究.为了将成熟的软件开发方法运用到 Web 应用的开发过程中,近年来有许多研究致力于寻找一条 Web 技术与对象技术相结合的途径[6].文献[1]中介绍了一种面向对象的 Web 应用开发方法,采用对象作为统一的概念对 Web 实体进行建模,提高了 Web 应用的可复用性和可维护性.但是,由于该方法没有提供对 Web 计算能力的支持,因而更适合静态文档之间的复用.考虑到 Web 本身的特点,本文采用对象视图而不是对象本身对 Web 实体进行建模.作为对象内部状态的可视化表示,对象视图除了具有可复用性和可维护性等优点外,还能够被动态生成或与源对象进行交互,因而保持了 Web 现有的支持功能性计算的能力.

随着技术的发展,Web 从传统的信息平台逐渐发展成为一种分布式计算平台.由于具有用户的不确定性和访问的随机性等特点,Web 应用的体系结构逐渐成为研究和开发的关键问题[2].作为对象视图模型的实现,基于 WebView 的应用框架同时也对 Web 应用的体系结构进行了规范,例如对象视图与源对象相分离的原则、基于 XML 消息的交互机制、基于代理对象的对用户不确定性的屏蔽等.这些特点使得应用系统不仅具有良好的可复用性和可扩展性,还具有很好的安全性.

4 结 论

作为 Web 应用的基础,资源模型本质上是一种实现级的模型,其明显不足的抽象能力导致了成熟的软件开发方法无法运用到 Web 应用的开发过程中.本文提出了 Web 应用的对象视图模型 WebView,采用对象视图对

Web 实体进行抽象并将对象视图划分为常量视图、静态视图和动态视图等 3 种类型,不仅提高了设计和实现的可复用性,而且保持了 Web 平台的计算能力.作为对象视图模型在现有 Web 技术基础上的一种实现,基于 WebView 的 Web 应用框架实现了对象视图与源对象之间的分离,并采用代理对象的方式屏蔽了用户的不确定性和访问的随机性,使得基于该框架的 Web 应用系统具有很高的可复用性、可扩展性和良好的安全性.

References:

[1] Gellersen, Hans-W., Gaedke, M. Object-Oriented Web application development. IEEE Internet Computing, 1999,3(1):60~68.

[2] Ciancarini, D., Tolksdorf, R., Vitali, F., *et al*. Coordinating multiagent applications on the WWW: a reference architecture. IEEE Transactions on Software Engineering, 1998,24(5):362~375.

[3] Hoque, R., Sharma, T. Programming Web Component. New York: McGraw-Hill, 1998.

[4] Ben-Shaul, I., Kaiser, G. Coordinating distributed components over the Internet. IEEE Internet Computing, 1998,2(2):83~86.

[5] Evans, E., Rogers, D. Using Java applets and CORBA for multi-user distributed applications. IEEE Internet Computing, 1997,1(3): 43~55.

[6] Manola, F. Technologies for a Web object model. IEEE Internet Computing, 1999,3(1):38~47.

A Web Application Framework Based on Object View Model WebView*

ZHANG Bo, FENG Yu-lin, HUANG Tao

(*Key Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*);

(*Software Engineering Technology Center, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: {zb,tao}@otcaix.iscas.ac.cn; feng@ios.ac.cn

http://www.iscas.ac.cn

Abstract: By lack of suitable abstractions of the resource model, it is difficult to apply well-established software development methods to Web applications. An object view model WebView is outlined and the model is implemented in this paper as a Web application framework. WebView uses object view as a uniform concept to model Web entities. By using the model in designing, it can enhance reusability and productivity in the Web application development. Based on the available Web technologies, the Web application framework can map object views into the HTML format.

Key words: Web application; distributed computing; application framework

* Received December 20, 2000; accepted February 21, 2001

Supported by the National Natural Science Foundation of China under Grant No.69833030; the National High Technology Development 863 Program of China under Grant No.863-306-ZD02-01-1