云南大学学报（自然科学版），2002，24（6）：417~420　　CN 53-1045/N　ISSN 0258-7971
Journal of Yunnan University

基于永久对象实现的 Web 页面的数据存储*

李永尧，李　彤，胡　盛
（云南大学 计算机科学系，云南 昆明　650091）

摘要：把 Web 页面作为永久对象进行数据存储，利用永久对象的机制存储分解所得的 Web 页面对象及其成分对象潜在的好处是使用它们的 Web 页面不用改变源代码就可以把对象从一个数据存储向另一个数据存储转移. 同时，这种机制也给 Web 页面对象的存储和再存储提供了灵活的框架.

关键词：Web 页面；永久对象；数据存储
中图分类号：TP 393.09　　文献标识码：A　　文章编号：0258-7971(2002)06-0417-04

随着 Internet 的发展，人们越来越依赖于 Internet 所提供的资源，Internet 上的资源也越来越多. 然而静态的 Web 页面越来越不能满足商业用户对信息服务的动态性、实时性和交互性的要求. 因此，将 Web 技术和数据库技术相结合，从而实现两者的交互式应用，已成为必然，同时也为 Internet/Intranet 以及 e-commerce 的大众化提供了前提条件[1].

目前，很多的 Web 数据库解决方案都是从 Web 页面的内容作为出发点，数据库仅仅是作为一个网页内容的提供者，这为网页内容的更新带来了很大的方便，但是对于网页结构的变化还是无能为力. 也就是说，这些解决方案只是通过 Web 服务器发布数据库中的信息，并不支持 Web 页面的维护工作，这与数据库强大的数据管理和数据处理能力不相称，也使得用户更新 Internet 网站的工作还是十分的繁琐. 如果能将 Web 页面的结构信息也存储到数据库中（不只是 Web 页面的内容），由发布引擎实时通过 Web 服务器发布，那么就可以减少很多 Internet 网站的维护工作. 如何将 Web 页面的结构信息存储到数据库是问题的关键. 笔者试图在这方面做一些工作.

1　Web 页面

目前，通用的 Web 页面标记语言是 HTML（hyper text markup language），而 XML（extensible markup language）是将来流行的 Web 页面标记语言，两者都是 SGML 的一个子集，有很多共同点. 因此，我们主要考虑 HTML，同时也考虑到将来向 XML 拓展的问题. 所以，下面我们提到的 Web 页面主要是 HTML 页面.

由于 HTML 语言的风格和 Web 页面源码的组成结构，我们很容易想到采用面向对象的方法来对 Web 页面源码进行分解. 由于动态 Web 设计的需要，Microsoft 公司提出了 HTML 的对象模型体系，图 1 所示的就是 Microsoft 的文档对象模型（DOM）的基本结构[2].

我们可以借鉴 Microsoft DOM 模型，采用平行的对象概念来对 Web 页面进行分解，每个对象都有唯一的标识，我们称之为对象标识（OID）. 同时，Web 页面的对象中，有很多永久的成分，意味着 Web 对象的生命期可以超过 Web 应用程序执行的生命期. 所以，我们可以利用永久对象的机制存储和管理 Web 页面. 利用永久对象的机制存储分解所得的各页面对象及其成分对象潜在的好处是使用它们的 Web 页面不用改变源代码就可以把对象从一个数据存储向另一个数据存储转移，这不仅大大地减少了网站设计人员的工作量，同时，这种机制还为 Web 页面对象的存储和再存储提供了灵活的框架.

* 收稿日期：2002-03-04
基金项目：云南省自然科学基金资助项目（2001F0006M）；云南省中青年技术带头人培养基金资助项目（998-37）.
作者简介：李永尧（1975-　），男，福建人，硕士生，主要从事 Internet 及 Web 方面的研究.

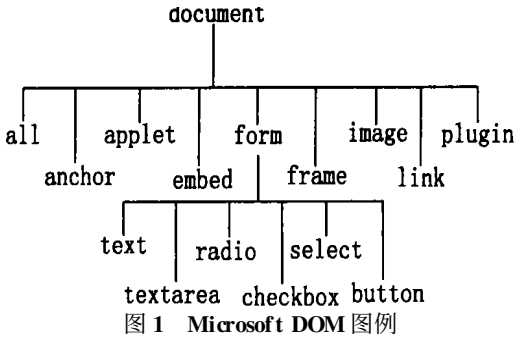

图 1 Microsoft DOM 图例

Fig. 1 Microsoft DOM

2 Web 页面永久对象实现的体系结构

Web 永久对象实现的体系结构有多个组成部分. 其中主要的组成部分如图 2 所示. 它们是: 永久对象(PO) ——用来表示 Web 页面及其成分对象(标记) 的对象; 永久对象标识(POID) ——用来描述永久对象在数据存储中的位置并为永久对象生成一个字符串标识; 永久对象接口(POI) ——为永久对象的数据存储及其基本操作提供一个统一的输入输出接口; 数据存储 ——提供一种独立地存储永久对象数据的方法.

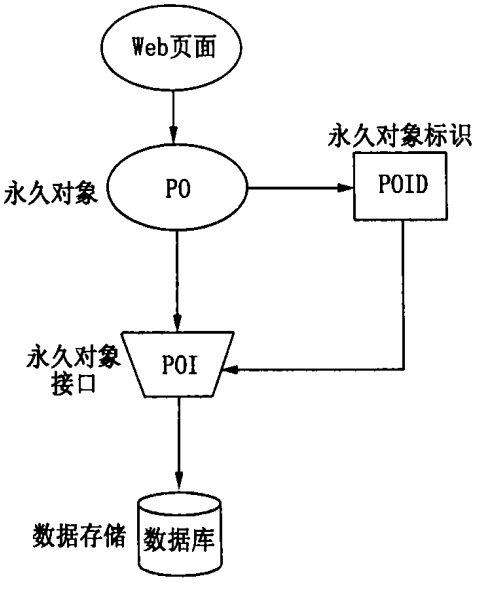

图 2 永久对象的体系结构

Fig. 2 Persistent object architecture

实现 Web 页面永久对象涉及的核心问题是永久对象接口的设计以及保存与装载永久对象的数据存储的设计. POID 是从数据存储装载任意永久对象的键值.

3 数据存储

数据存储是用来存储 Web 页面的永久对象, 而我们的实现方案中数据存储是数据库. 当 Web 页面各自的永久对象被保存和装载时对数据存储进行读写操作, 这是由永久对象接口实现的. 如何将 Web 页面的永久对象存储到数据库中是这里要解决的问题.

Web 页面的结构很象一棵树, 而树的结点就是 Web 页面的成分对象. 所以我们把 Web 页面看作一棵树, 有唯一的根结点, 根结点不同于根元素, 对于 HTML 而言, 根元素通常是〈html〉, 〈/ html〉, 而根结点是在页面上附加的一个元素, 是一个虚拟的结点, 对于 Web 页面没有实际意义, 可以把它看作是 Web 页面本身, 用 root 表示. 根结点可能包含多个子结点, 根元素只是根结点的一个子结点. 由于 HTML 是一种 SGML 语言, 它包含了文档类型定义(DTD), 所以根结点可能还包括文档类型定义(DTD) 的子结点, 在 Web 页面中, 文档类型定义通常先与根元素. 这和 XML 中的一些定义是一致的, 这样 HTML 可以方便地和 XML 相兼容, 便于从 HTML 向 XML 扩充. 既然 Web 页面是一棵树, 那么就可以用树的表示方法来表示一个 Web 页面, 树的表示方法通常有 3 种: 父亲数组表示法, 儿子链表表示法, 左儿子右兄弟表示法. 为了能更好地讨论各种表示法对于用数据库存储 Web 页面的利弊, 我们引入一个简单的 Web 页面(如图 3 所示), 并且把它用一棵树来表示(如图 4 所示), 然后, 我们把它分别用树的 3 种表示方法存储到数据库中(如表 1 所示). 下面分别讨论树的 3 种表示方法的利弊.

3.1 父亲数组表示法 父亲数组表示法就是用一个一维数组存储每个结点各自的父亲[3]. 这使得要获得一个结点的父亲结点是十分方便(只要进行一次数据库操作就能得到它的父亲结点), 同时存储在数据库中也十分方便, 也很少出现数据冗余(如表 1 所示), 当我们用 0 表示根结点以后, 就不再出现数据冗余. 虽然这种表示法在用数组表示时, 要获得一个结点的儿子就必须遍历整个数组, 效率比较低, 但是当我们用数据库来存储时, 不管是要获得一个结点的儿子还是要获得它的父亲, 对于数据库来说操作都是一样的, 效率也差不多(只要进行一次数据库操作就能得到它的所有的儿子

结点). 因此, 用这种表示法对于 Web 页面的维护比较方便, 效率也比较高.

3.2 儿子链表表示法　儿子链表表示法是对树的每个结点建立一个儿子结点表[3]. 由于各结点的儿子结点数目不一, 所以用关系数据库来存储 Web 页面结构就比较麻烦, 数据库的数据完整性很难保证, 并且增加了数据冗余(如表 1 所示). 因此儿子链表表示法对于用关系数据库存储 Web 页面并不是很方便.

```
<html>
<head>
<title>书目</title>
</head>
<body>
<hr>
<p><a>作者甲</a>，发表年份，
出版物标题，出版商。</p>
<hr>
<h5>修订时间</h5>
</body>
</html>
```

图 **3**　一个 **Web** 页面

Fig. 3 An example of Web homepage

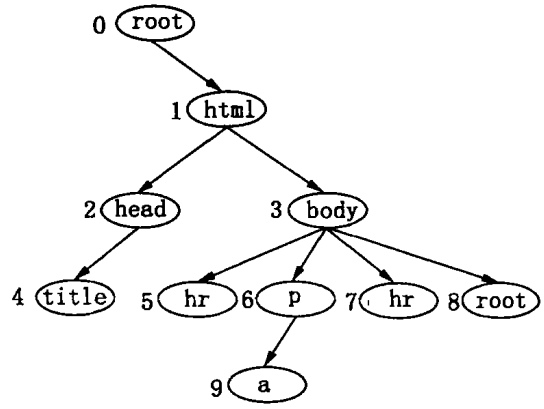

图 **4**　**Web** 页在的树形结构

Fig. 4 Tree of the Web homeqage

表 **1**　**3** 种表示法的数据存储

Tab. 1 The datastores of three notation for the Web homepage

父亲数组表示法		儿子链表表示法		左儿子右兄弟表示法		
ID	父亲	ID	儿子	ID	左儿子	右兄弟
1	0	1	2	1	2	
2	1	1	3	2	4	3
3	1	2	4	3	5	
4	2	3	5	4		
5	3	3	6	5		6
6	3	3	7	6	9	7
7	3	3	8	7	8	
8	3	4		8		
9	6	5		9		
		6	9			
		7				
		8				
		9				

3.3 左儿子右兄弟表示法　左儿子右兄弟表示法, 也叫二叉链表表示法, 也就是以二叉链表作为树的存储结构. 链表中结点的 2 个域分别指向该结点的最左儿子和右邻兄弟[3]. 虽然这种表示法用链接表表示时比较方便, 但是当用关系数据库来实现这种表示法时, 要实现这种表示法就不是很容易的事, 并且也增加了数据冗余(如表 1 所示), 在 3 种表示法中是最大的. 同时这种表示法也增加了 Web 页面的维护的难度, 这是因为当要获得一个结点的所有儿子结点时, 必须进行很多次数据库操作的才能获得需要的(一次数据库操作只能得到一个儿子结点), 显然效率是比较低下的; 另外, 当要获得一个结点的父亲结点时, 就更加的麻烦(必须遍历整棵树, 既整个 Web 页面).

从以上的比较中, 可以看出, 用父亲数组表示法来表示 Web 页面是最合适的, 不仅数据冗余最小, 而且有利于对 Web 页面的操作和管理, 同时从整体上减少了操作的延迟, 提高了整体的性能. 因

此,选用这样表示法是比较合理的.

4 永久对象接口

我们在永久对象接口中定义了 2 个基本操作: LoadObject 和 SaveObject.

```
// 永久对象接口
interface poi{
    object LoadObject(in string poid);
     void SaveObject(in object obj, in string poid);
};
```

LoadObject 使用 Web 永久对象标识(POID)从数据存储中装载永久对象及其成分对象. 我们把永久对象标识定义为 POID= WebPageName: ObjectName: Order, 比如, 在 index 页面中有一个 image 对象, 则它的永久对象标识是: index: image: 1. 这个过程在数据存储中搜索指定的永久对象标识, 如果找到一个与之匹配的 POID, 那么再用树的算法搜索这个对象的所有成分对象, 并把它们和这个对象一起返回. 否则, 返回一个空值.

SaveObject 从 Web 永久对象中获得 POID 和这个永久对象, 并且把 POID 和永久对象作为一个整体存储到数据存储中. 在这个过程中, 首先用指定的 POID 搜索数据存储, 如果找到与之匹配的 POID, 那么更新这个永久对象及其成分对象. 否则, 将这个 POID 和永久对象作为一个新的实体一起保存到数据存储中. 同时, 如果这个对象有成分对象, 那么再次调用这个过程对成分对象进行处理, 否则结束这个过程.

5 实 现

根据以上的构思, 我们实现了一个全新的 Web 发布平台——WebBuilder. 运用这个发布平台, 可以实现 Web 页面的完全数据库驱动, 不仅仅只是把 Web 页面的内容信息存储到数据库中, 而且可以把 Web 页面的结构信息存储到数据库中. 同时, 我们在 WebBuilder 中提供了 Web 页面的维护工具, 这样为 Web 页面的维护带来了很大的方便. 而传统的 Web 数据库实现技术, 比如 CGI, ISAPI, Java 等等, 通常都只是把 Web 页面的内容信息存储到数据库, 而没有涉及到 Web 页面的结构信息, 也就是说仅仅只是通过 Web 服务器发布数据库中的信息. 和传统的 Web 数据库实现技术相比, WebBuilder 不仅仅只是通过 Web 服务器发布数据库中的信息, 而是使 Web 页面和数据库紧密结合, 从而发挥了数据库强大的数据管理和数据处理能力, 实现了 Web 页面的完全数据库驱动.

6 结 论

我们运用的 Web 页面永久对象实现的体系结构, 给 Web 页面对象的存储和再存储提供了灵活的框架. 在此基础上, 比较容易运用 PHP 语言设计出发布数据存储中的 Web 页面的工具. 这样, 就充分发挥数据库的功能, 扩大了 Web 数据库的应用范围和深度, 尤其适用于商业网站的建设, 也给 Web 网站的维护和更新提供了方便.

参考文献:

[1] 梁 彬, 李 彤. 一个完全数据驱动的 Web 发布平台[J]. 云南大学学报(自然科学版), 2000, 22(5): 333—338.

[2] 於志渊. 动态 Web 网页技术大全[M]. 北京: 清华大学出版社, 1999.

[3] 傅清祥, 王晓东. 算法与数据结构[M]. 北京: 电子工业出版社, 1999.

Databases implementation of Persistent Object for Web page

LI Yong-yao, LI Tong, HU Sheng

(Department of Computer Science, Yunnan University, Kunming 650091, China)

Abstract: Web pages are stored as persistent object in the datastore. The main potential advantage of our implantation of Persistent Objects for Web pages is that it allows Persistent Objects to migrate from one datastroe to another without changing the code of Web pages that use them. It also defines a flexible framework that stores and restores Persistent Objects for Web pages.

Key words: Web page; Persistent Object; datastore