文章编号：1005-8451(2007)11-0028-04

Web应用开发中高效安全的数据持久层研究

吕 珅
(首都师范大学 信息工程学院计算机科学与技术系，北京 100037)

摘 要：阐述企业数据的重要性，分析当前流行的持久层解决方案的优缺点，分析XML在持久层设计中的重要性，以数据访问对象包装数据操纵，以改进的XVO(XML Value Object)作为数据对象，包装在应用程序中流动的数据，利用数据锁模式提供安全性的保证。给出基于XML和设计模式建立一个高性能，高安全性的，可拓展性的数据持久层的方法。同时，还用一个信息查询的完整业务流程解释持久性框架的主要设计思想。

关键词：数据持久层；XML；设计模式；安全性
中图分类号：TP311 文献标识码：B

Research on efficient and secure data persistent layer in Web development

LV Shen
(Department of Computer Science & Technology Captain Normal University, Beijing 100037, China)

Abstract: It was demonstrated how important the enterprise data was, as well as analysed the benefits with drawbacks of some popular persistent-layer solutions and the crucial role of XML in persistent-layer design. Data Access Object was used to encapsulate data operating. Packing data flowing in the application, the modified XVO was used as Data Object. The guarantee to security was provided by Data Locker design pattern. It was evaluated the way to construct a high performance as well as strict security data persistent layer with strong expansibility based on XML and design patterns. What was more, the essay also accounted for the main idea of the framework in design with an entire instance of a search process.

Key words: data persistent layer; XML; design pattern; security

数据是企业非常有价值的财产，没有数据，公司无法正常运作。因此，每一家公司都有义务来保证这些重要数据资源的完整性和安全性。在设计企业级Web应用程序的时候，开发人员应该重视对数据访问模块的设计，即数据持久层的设计。建立数据持久层就是创建一个集成数据库存取逻辑的门户，隐藏数据访问代码细节，该层位于业务逻辑对象和分离的数据存储之间。

本文提出了一种高效率，安全性高的数据持久层实现策略，帮助开发人员快速开发企业Web应用程序中数据持久层的解决方案。

1 数据持久层架构

本文提出了一种架构（如图1），以数据访问对象（Data Access Object）包装数据操纵，以改进的XML数值对象（XVO）作为数据对象，包装在应用程序中流动的数据，并提供了处理器，使开发人员有更多的操纵数据方式的选择和对当前系统更好的适应性，另外安全上下文（Security Context）的存在使得对数据恶意侵犯的可能性降低，同时该以XML为核心的架构具有很高的拓展性。

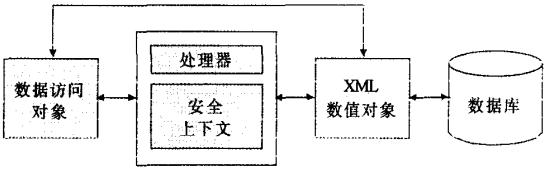

图1 数据访问层架构

由于XML技术本身的许多技术特性，使得它非常适用于在N层系统中担当重要的角色。使用XML可以便于组件、系统与企业之间的数据交换，并且基于XML的数据可以遵循一定规则被统一地创建和维护。它实现了高级组件的重用和分布式系统的交互性，让系统的接口更加完整和易被使用者理解和拓展。更重要的是，XML是一种工业标准，它几

收稿日期：2007-03-13
作者简介：吕 珅，在读硕士研究生。

乎可以运行在任何Web服务器上而不必担心是哪个制造商，它从系统集成的角度出发，实现了数据级别的集成，和用户应用程序与其他系统的松耦合，并且增强了框架体系的伸缩性。

XML在本持久性框架中的作用很大，除了作为XVO组件的基础外，它还构成了框架的配置体系，如应用程序的信息配置，安全角色和级别配置，SQL查询字符串配置。在程序需要用到的地方用相关Java API来访问。XML也可以用来表示复杂的数据结构，比如图结构或表结构，不同的程序需要不同的数据表示结构，目前在本框架当中用来进行系统配置的XML文档结构为DOM树结构。框架对XML的依赖性比较强，许多服务都要基于XML来完成。

2　传统数值对象（VO）与XML数值对象（XVO）的分析与比较

VO模式是传统数据层的核心模式，一个可序列化的数值对象表现了一组复杂的数据集，并用以在应用程序各层之间传送。它在结构上有点像C语言当中的struct，但是可以包含行为，如验证（Validate）。举例来说，在信息查询的业务流程当中，可以定义一个数值对象StuffVO，它包含了关于一个员工的许多重要信息（不排除其中的敏感信息），并且用来向用户显示信息，如图2所示。

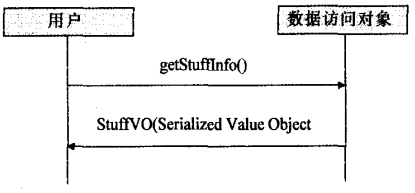

图2　传统的数据层核心模式

这样做的好处是如果需要的话可以在其中加入验证逻辑。但是它的缺陷是：（1）它没有可重用性，因为它只代表了雇员实体，并不能表示系统中其它任何实体，比如值班记录，因此，必须创建尽可能多的数值对象来将系统中的实体一一描述；（2）在StuffVO当中，涉及到员工的地址信息时包含了一个AddressVO的引用，这样系统开销明显要比接下来要介绍的XVO所使用的机制大；（3）用户需要维护大量的getter和setter方法，如果数据表列变动频繁的话，将是个很大的工作量；（4）最严重的问题是接受StuffVO的用户必须知道它是什么和如何访问其中的数据，而且必须在接口级别（Interface Level）明白它和其它的如AddressVO有何不同并且要不同地处理它们。

而XVO，由于采用了文件对象模式（DOM）对象，不用管它所包含的内容，客户只需要通过XML API来访问数据和控制数据类型。而且，数据完整性验证工作也可以通过XML验证解析器和DTD来自动完成。图3展示了包含stuff信息的XVO。

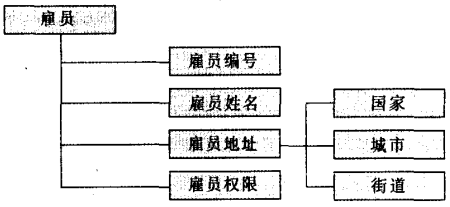

图3　用XML树结构表示的雇员信息

这种DOM树的结构非常便于使用JDOM API进行各种数据操纵。在提高可重用性这个角度上，在组件中使用XML有很大益处。如图4。

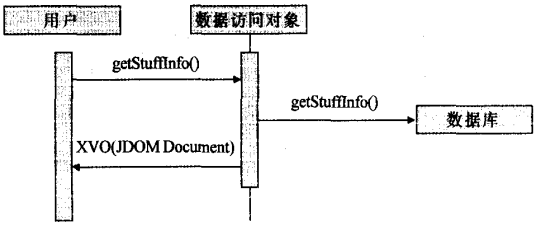

图4　基于XVO的数据层访问模式

以下是数据访问组件中的主要代码：

```
…
…

public class CustDAO {
        /**
         *     构造方法和数据变量的定义
         */
        public Document getStuffInfo(String id) {
                /**
                 *  连接数据库并查找，获取
ResultSet rs
                 */
                Element root = new Element
(“stuff”);
                doc = new Document(root);
```

```
                Element nameElem = root.
addContent(new Element("name"));
                nameElem.addContent(rs.getString
("NAME"));
                /**
                 * 省略其他字段的添加
                 */
                return doc;
            }
        }
```

当获取了 Document 对象的实例之后，可以将它转换为 DOM 结构：

```
DOMoutputter out = new DOMoutputter();
try{
  return out.output(doc);
 }
...
```

可以直接对内存中的 DOM 树进行操作，也可以将它传送到处理器进行处理。处理器在框架中存在的目的主要是尽可能地利用 XVO 对用户提供服务接口。处理器可以提供 XVO 向改进型 VO（经过数据完整性验证）的转换。处理器的另外一个作用就是基于 XML 的跨平台特性解决不同平台共享信息和系统移植的问题。

在实际开发中，一般需要一个可以容纳更加普遍的数据的 XVO 来实现真正持久的透明性。当获得了结果集对象实例 rs 之后，可以通过它取得 ResultMetaData 对象：

```
ResultMetaData rsmd = rs.getMetaData();
```

3 持久层的安全性

网络传输的安全性由互联网设备和协议决定，服务器的安全性则需要人为的维护，而数据访问的安全性则由框架解决。基本模式都是来源于对角色的控制，它将用户与权限分离，把用户指定为几个角色，在分别定义每个角色的一个或多个权限。在持久性框架中加入保证安全性的机制，主要是当表现层逻辑的验证机制失效或者有人企图绕过表现层直接对数据层进行操纵的情况发生时，能够有效地保护企业数据，并强制访问者进行一次验证。持久层安全性的实现与表现层关系紧密，首先表现层使用单入口点（Single Access Point）模式和检查点（Check Point）模式，前者描述进入系统只有唯一的一个通道，后者则是实现了角色权限的定义，即检查角色是否有权限来操纵相应数据。用户通过表现层的验证逻辑之后其角色信息至少应该被放在 session 中，而当用户将要对数据库进行操作时，持久性框架中的处理器会从 session 中取出角色信息，与持久性框架的角色配置文件中的信息进行比对，看该角色有权访问那个表，并把可访问的表缓存起来，在当数据库返回 XVO 时，处理器便会进行检查（要求返回的 XVO 中有包含所访问数据表表名的节点），如果在其中发现了之前缓存里没有的表，则终止服务，通知表现层向用户显示警告信息。这种方式称为数据锁（Data Locker）模式，如图 5。

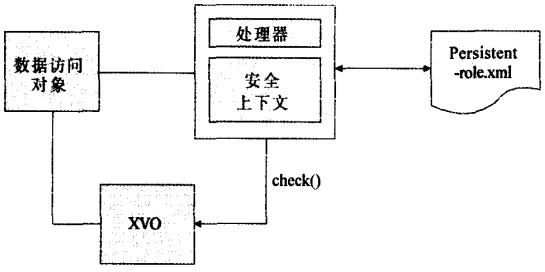

图 5 数据锁模式

4 结束语

本持久层架构通过使用改进了的 XML 数值对象，增强了系统的拓展性和灵活性，并且 JDOM 的使用使得对数据对象的操作更加简单和高效。实验表明，相比当前的持久层技术，该架构的速度优势和易操作易维护性在一定程度上是很明显的。

目前，敏捷 Web 应用程序开发比较流行，针对企业数据层的复杂性，如何在短时间内开发一个健壮的数据访问层成为了一个重要问题，良好的持久层设计应该包括如下几点：

（1）封装数据操纵细节，使数据访问过程和数据源对用户是透明的；

（2）用户 SQL 语句的零输入；

（3）将数据持久逻辑和业务逻辑分开；

（4）良好地解决对象—关系的阻抗不匹配问题；

（5）实现安全访问控制，降低不良操作对企业数据的损害；

文章编号：1005-8451（2007）11-0031-02

P2P协议及其应用

许 进
（中国铁通集团有限公司 郑州分公司，郑州 450052）

摘 要：描述P2P点对点网络技术协议，分析该协议的原理及其在网络中的应用。

关键词：网络技术；协议；工作原理；应用
中图分类号：TP393.07 文献标识码：A

近年来互联网上对等连接P2P应用发展迅速，MP3和视频文件共享下载的P2P流已经成为宽带用户流量的主体。基于P2P的即时通信和互联网电话（如Skype）发展迅速，对等广播P2P流媒体等正在兴起。P2P协同计算和网格方兴未艾。P2P应用支持网络通信的对象从人—人，人—机发展到机—机，其应用从家庭网络和传感器/执行器网络到军事上网络中心战/全球信息网格GIG。

P2P（Peer-to-Peer）指的是对等网络，也就是网络两个节点之间是点与点的关系，是完全对等的，双方或多方是相互依赖和互相支持的，不存在单向的依赖关系。简单地说，P2P就是一种用于不同PC用户之间、不经过中继设备直接交换数据或服务的技术，它允许网络用户直接使用对方的文件，每个人可以直接连接到其他用户的计算机，并进行文件的交换，而不需要先连接到服务器上再进行浏览与下载。

收稿日期：2007-02-25
作者简介：许 进，工程师。

1 P2P协议的工作原理

在P2P协议出现之前，网络上的很多应用都是基于C/S模式，如HTTP、FTP和PUB等下载方式，一般都是先将文件放到服务器上，然后再由服务器传送到每位用户的机器上，因此，如果同一时刻下

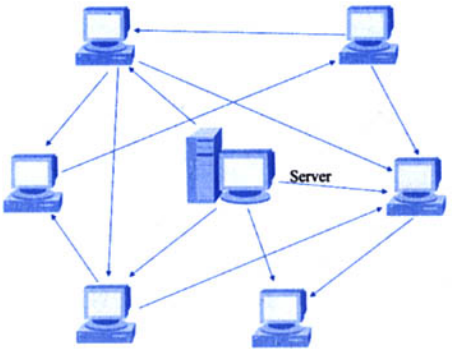

图1 实现文件共享的工作原理

（6）完善日志系统，记录持久层的服务情况，也可以用第三方工具实现；

（7）有效的对象缓存，提高系统性能；

（8）实现同步访问控制，更好地为多用户的同时请求服务；

（9）具有可移植性，与新老系统集成方便；

（10）可拓展性，对于不断变化的用户需求能够快速作出反映。

未来在进行敏捷开发的时候，开发人员只需要将元数据，设计模式驱动的模型和少量模板输入框架，就能产生一个健壮完整的数据持久层，甚至直接生成表现层（XVO为此提供了可能）。在这之中，设计模式做为指导工具将发挥巨大的作用。

参考文献：

[1] Alan Monnox. Rapid J2EE Development[M]. (1st ed) Pearson Education, Inc. 2006.

[2] 徐长盛. J2EE数据持久化技术的研究[J]. 计算机应用与软件，2003，23（4）.

[3] J2EE Design Patterns Applied[J]. (1st ed) Wrox Press, 2003.

[4] XML and Java[J]. (4th ed) Addison Wesley Longman, Inc. 2000.

[5] Scot W. Ambler. Mapping Objects to Relational Databases[EB/OL]. http://www.agiledata.org. 2006.

[6] Chris Richardson. POJOs In Action[M]. (1st ed) Manning, Inc. 2005.

[7] Khawar Zaman Ahmed. Developing Enterprise Java Applications With J2EE and UML[M]. (1st ed) Pearson Education, Inc. 2003.