嵌入式系统基于RMS对象存储机制的设计与实现

郭 宁

(首都经贸大学信息学院 北京 朝阳区 100026)

【摘要】为了使开发者不用关心RMS存取的细节问题，直接对数据对象进行存取，并避免重复编写代码，该文设计了一种对象存储解决方案。利用数据访问对象模式使业务层和数据存取层分开，解决了如何将数据访问从应用中分离出来的问题，实现基于对象的查询和排序等功能，达到了简化应用程序设计，增强源代码可维护性，以及建立灵活、可重用的持久性存储机制的目的。

关 键 词 DAO模式； 嵌入式系统； 对象存储； 记录存储系统

中图分类号 TP311.5 **文献标识码** A

Design and Realization of Object Storage System on Embedded System

GUO Ning

(Information College, Capital University of Economics and Business Chaoyang Beijing 100026)

Abstract The paper describes a method for object storage system on embedded applications. It is utilize data accessor objec (DAO) pattern to divide the business layer and data storage layer to realize the function about object quering and sequencing. Deviser not need to know the detail about record management system (RMS), and to direct accessing object.

Key words DAO pattern; embedded system; object storage; record management system

嵌入式系统是集软件、硬件于一体的高可靠性系统。嵌入式系统中的软件除操作系统外，还需有完成嵌入式系统功能的应用软件[1]。J2ME[2]是Sun公司为资源有限的设备上使用Java编程语言而设计的，可以使用在如智能卡、手机、PDA、电视机顶盒等各种消费电子产品上。J2ME采用典型的Java虚拟机技术，这种全功能的虚拟机包含了基于台式机系统上的虚拟机的所有功能，适用于拥有至少几兆字节内存的高档消费类电子产品和嵌入式设备。在J2ME程序开发中，经常会遇到数据持久化的需求，如手机游戏程序中的比分记录、应用程序中用户信息的本地化存储等。在J2ME的相关规范中，解决问题的机制是采用记录存储系统(record management system，RMS)[3]。

1 移动设备的记录管理系统的局限性

移动信息设备框架(obile information device profile，MIDP)[4]为应用程序提供一种跨多个调用持久存储数据的机制。这种持久存储机制可以被视为一种简单的面向记录的管理系统。因为设备中没有数据文件的概念，所以一般需要保存的数据只能以记录的形式保存。每条记录是一个字节数组，多条记录组成一个记录集。MIDP应用程序称为[5] MIDLet(IDP小应用程序)[5]，它不能单独地运行，必须运行在特定的环境之中，或者说运行在一个容器中。可以把这个容器看作是一个大的应用程序，它运行在Java虚拟机之上，但不能完成任何特定的任务，因此需要程序开发者编写代码以完成该项工作，这些编写的程序就称为小应用程序。

RMS由多个记录存储构成，RMS可以为MIDLet程序提供存储空间，RMS和MIDLet接口的连接如图1所示。

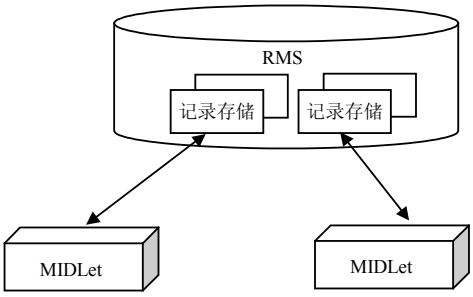

图1 J2ME RMS和MIDLet接口连接的概貌

收稿日期：2006 – 12 – 25；修回日期：2007 – 06 – 12

作者简介：郭 宁(1958 –)，女，副教授，主要从事软件工程、网络技术、软件建模等方面的研究.

记录集相当于数据库的表，每个“表”由若干记录构成，一条记录就是一个用int表示的记录号RecordID和以字节流形式(byte[])表示的内容。记录号可以看作是“主键”，byte[]数组存储内容。每个MIDLet的存储空间中可以存放多个记录集。RMS没有类似关系数据库中索引和关系的实现。因此，RMS是一个比较简单的数据存储机制。

javax.microedition.rms.RecordStore类代表RMS记录存储，它提供了几个方法来管理以及插入、更新和删除记录存储中的记录。如在应用程序中，用户输入的日期和时间被连接成一个字符串，转换成字节，然后被存储。当然该数据库的存储和获取信息的性能受到设备的限制。

RMS提供的记录操作可以实现根据ID直接获得记录，或者枚举出一个表中的所有记录。枚举记录是非常低效的，因为只能比较byte[]数据来确定该记录是否是所需的记录。通过ID获得记录是高效而方便的，然而，通常应用程序很难知道某条记录的ID号，而RMS记录的“主键”又仅限于int类型，无法使用其他类型如String作为主键来查找。因此，对于需要存取不同类型对象的应用程序而言，就需要一个灵活的RMS操作框架。

当使用J2ME平台进行编程时，采用的是基于面向对象的编程方法，数据通常封装于具体的对象当中。而RMS中每个记录都是字节数组，各种操作针对的是记录而不是对象。用户的数据要存入其中，就需要将对象中的数据转化为字节数组，进行编码和解码的编程。同时，对于不同的数据对象，进行数据的CRUD操作都需要写类似的代码，导致大量地重复编程；另外很难实现基于对象的查询和排序等功能。所有这些问题与J2EE平台中没有ORM机制时的情况十分类似。

2 对象存储机制的设计

要解决上述问题，需要建立类似ORM的机制，把所有的数据访问集中到一个独立的层，管理所有的数据访问复杂性，减少业务对象中代码的复杂度，使开发者不必关心RMS存取的细节问题，直接对数据对象进行存取。

2.1 设计目标

在进行对象存储机制的设计时，本文采用数据访问对象(DAO)设计模式[6]，完成对象的存储功能，实现了以下目标：

(1) 数据存储逻辑的分离。通过对数据访问逻辑进行抽象，为上层结构提供抽象化的数据访问接口。

(2) 数据访问底层实现的分离。通过将数据访问划分为抽象层和实现层，从而分离数据使用和数据访问的底层实现细节。

(3) 资源和调度的分离。将数据访问逻辑从业务逻辑中脱离开来，使得在数据访问层实现统一的资源调度，提升系统性能。

(4) 数据抽象。通过对底层数据的封装，为业务层提供一个面向对象的接口，使业务逻辑开发人员可以面向业务中的实体进行编码。业务层屏蔽了数据库访问的底层实现，业务层仅包含与本领域相关的逻辑对象和算法。所有与实现有关的代码(如SQL语句)都被包含在DAO中，而不是包含在业务对象中，提高了代码的可读性，以及涉及对象存储机制代码的生产效率。

2.2 对象存储机制整体设计

只要与对象持久性[7]存储有关的应用，都会建立灵活、可重用的持久性存储机制，主要是解决如何将数据访问从应用层中分离出来，使用DAO抽象和封装所有对数据源的访问。DAO模式将数据访问划分为抽象层和实现层，分离了数据使用和数据访问的底层实现细节。DAO模式实际上是两个模式的组合，即Data Accessor和ActiveDomain Object模式，其中Data Accessor模式实现数据访问和业务逻辑的分离，即对RMS的访问细节，只给用户显示一个对象存取接口。而ActiveDomain Object模式实现业务数据的对象化封装，即将用户数据封装成Java Bean对象[8](包含相关属性的getter和setter方法)，使应用程序代码避免了与数据存储机制的任何直接交互。基于DAO模式的系统的整体结构如图2所示。

图2 系统结构

DAO管理与数据源的连接，便于检索和存储数据，DAO实现了用于操作数据源的访问机制。依赖于DAO的业务组件为其客户端使用DAO提供了更简单的接口，DAO完全向客户端隐藏了数据源实现细节，业务对象不了解低层数据实现。当应用程序要在不同的持久性存储间迁移时，该迁移只涉及对DAO层的变化，DAO向客户端提供的接口不会变化，这些访问特定持久存储层的代码不必重写。DAO

充当组件和数据源之间的适配器。DAO是把对数据库的操作(如最基本的CRUD操作)全部封装在里面，例如，当要插入一个新的用户时，只需要提供一个insertUser(User)就可以了，具体的操作是在DAO中实现的。那么当要调用DAO时，只要知道insertUser(User)是用来插入一个新的用户，而不需要知道是如何实现的。这种集中化使应用程序更容易维护和管理。

2.3 RMSDAO包设计[9]

基于DAO模式，本文设计了用于J2ME平台的ORM库，即RMSDAO包，其类图如图3所示。针对J2ME的特点，本文在设计时对体积和性能等方面进行了优化，尽量使用轻量级的结构和算法，精简了一些不太核心的功能，使编译后的RMSDAO包只有4 KB大小，对应用程序的体积不会有太多的影响。RMSDAO包提供了一个J2ME环境下的DAO模式的实现，数据访问层使用的是RMS。RMSDAO包含有一个RMSDAO类，实现Data Accessor模式；一个RMSDomain接口，实现ActiveDomain Object模式；RMSFilter和RMSComparator两个辅助接口，分别协助RMSDAO对象实现查询和排序功能。

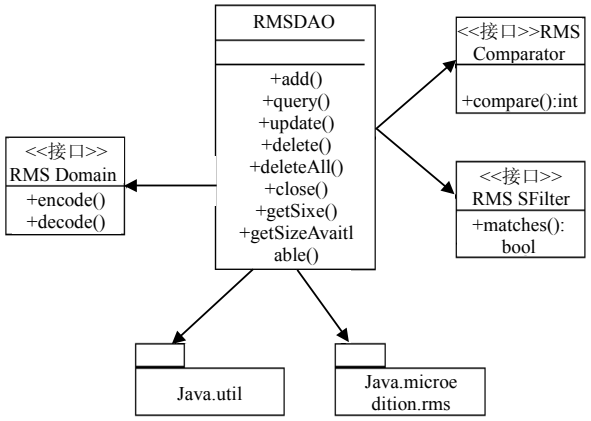

图3　RMSDAO类图

(1) RMSDomain接口[10]：实现ActiveDomain Object模式，用户的领域对象必须实现该接口以适配RMS存储要求，包含encode()和decode()两个方法。encode方法将领域对象编码成byte数组；decode方法将byte数组解码成领域对象。因为RMS中记录使用字节数组，所以用户必须实现对象到字节数组的encode和decode操作。J2ME不提供反射的能力，因此无法根据用户的Java Bean对象的属性自动生成这两个操作。

(2) RMSDAO类：实现Data Accessor模式，提供给用户对领域对象进行各种操作的方法，形成一个领域对象访问层，屏蔽对RMS的直接访问。构造方法RMSDAO(String RMSDomainClassName)是根据用户的RMSDomain对象的名字生成对应的DAO类。对象存取方法有add()、delete()、deleteAll()、update()和query()等，实现RMSDomain对象的存取、查询、排序等操作。其他辅助方法包括close()、getter()、getSizeAvaliable()等，实现关闭连接、获得存储空间大小等操作。

(3) RMSFilter接口：包含matches()方法，检查对象是否符合过滤条件，协助RMSDAO对象实现查询功能。

(4) RMSComparator接口：包括compare()方法，比较两个对象的顺序，协助RMSDAO对象实现排序功能。考虑到程序的大小和运行空间都受到严格的限制，为了使整个包的体积尽量小，本文采用辅助接口的简洁方式，让用户自己来实现具体的查询。

3 应用示例

下述例子说明如何使用RMSDAO包在J2ME程序中实现对象存储。假设使用J2ME编写一个电话簿程序，在手机上记录姓名和电话号码信息，可以按照姓名进行查询和排序，还可以对记录进行修改和删除等编辑操作。程序有一个输入窗口，可以输入姓名和电话号码；一个查询窗口，可以输入查询条件；一个列表窗口，显示符合条件的记录并且按照姓名排序；一组命令菜单，可以进行新建、修改、删除等编辑操作。

本文不介绍界面代码，只关注数据存储的部分。首先定义数据对象，即领域对象Phone，它包含两个String类型的属性，name和phoneNumber，有相应的getter和setter方法，并实现RMSDomain接口。其Java代码如下：

```
import java.io.*;
import com.gamil.test.j2me.rmsado.*;
public class Phone implements RMSDomain{
   private Sring_name;
private Sring_phoneNumber;
public Phone() {
}
public phone(String name,String pheneNumber) {
  _name=name;
  _phoneNumber=phoneNumber;
  }
  public String getName() {
```

```
        return_name;
    }

    public void setName(String name)
    _name=name;
    }
    public String getPhoneNumber(){
      return_phoneNumber;
    }
    public btye[]encode() {
      btye[]result=null;
    try{
    BtyeArryOutputSream                    bos=new
ByteArrayOutputStream();
    DataOutputStream                       dos=new
DataOutputStream(bos);
    dos.writeUTF(_name);
    dos.writeUTF(_phonenumber);
    result=bos.toByteArry();
    dos.close();
    bos.close();
    } catch(Exception e) {
    }
    pubic void decode(btye[]data){
    try {
    BtyeArryInputSream                     bis=new
ByteArrayInputStream(data);
    DataInputStream                        dis=new
DataInputStream(bis);
    _name=dis.readUTF();
    _phoneNumber=dis.readUTF();
    dis.close();
    bis.close();
    } catch(Exception e) {
    }}}
```

如果要对Phone对象进行存取，需要先创建一个RMSDAO对象：

```
RMSDAO                                 phoneDAO=new
RMSDAO(“phone”);
```

本文直接使用领域对象的类名字作为传入的参数。利用RMSDAO对象可以进行增加、删除、修改、查询等各种对象的存取操作。

4 结 束 语

采用本文设计，可使需要读取或保存数据的类完全不必涉及底层的RMS操作，大大简化了应用程序的设计，增强了源代码的可复用性与可维护性。RMSDAO包提供了对领域对象的各种查询操作，也封装了一些RMS的工具方法，用户一般不需要再直接使用RMS。目前，在J2ME平台上，RMS是主要的底层机制，如果考虑适应多种底层存储机制，如JDBC、XML等，还需要结合工厂模式的导入，但这会增加包的体积。

参 考 文 献

[1] KAMAL R A J. Embedded systems: architecture, programming and design[M]. 北京: 清华大学出版社, 2005.

[2] 詹建飞. J2ME开发精解[M]. 北京: 电子工业出版社, 2006.

[3] 陈 洋. J2ME编程实践之灵活的RMS应用[EB/OL]. [2006-03-10]. http://www.zydg.net/computer/book/read/j2ee/w712769801. html.

[4] 关文柏. J2ME应用程序架构模型[EB/OL]. [2006-05-20]. http://blog.csdn. net/k7sem.html.

[5] 陈跃峰. J2ME程序打包发布小技巧[EB/OL]. [2006-03-10]. http://www. zydg.net/computer/book/read/j2ee/w722769801. html.

[6] GAMMA E. HELM R. Design patterns elements of reusable object-oriented software[M]. [S.l.]: Addison-Wesley Longman Inc, 2000.

[7] 邵维忠, 麻志毅, 马浩海, 等译. UML用户指南[M]. 第2版. 北京: 人民邮电出版社, 2006.

[8] 陈立伟, 张克非, 黎秀红. 精通JAVA手机游戏与应用程序设计[M]. 北京: 中国青年出版社, 2005.

[9] 池雅庆, 王 耀. J2ME手机应用项目开发实践[M]. 北京: 中国铁道出版社, 2006.

[10] YUAN M. 手机游戏趋势和相关J2ME APIs[EB/OL]. [2006-08-10]. http://www.j2medev.com/blog/user2/66722/4637.html.

编 辑 熊思亮