移动终端信息提取方法的研究

Study on Method to Extract Information of Movable Terminals

董济农　徐 进 （北京理工大学信息科学技术学院　100081）

摘　要：本文在对移动信息体系及相关技术研究的基础上，通过对CLDC/MIDP、MIDlet、MIDP的UI模型、抽象Command命令、持续存储与RMS等的分析，讨论了移动终端信息提取的方法和实现途径。

关键词：移动终端 J2ME CLDC MIDP MIDlet

1 引言

随着移动通信技术的发展，移动通讯设备已经不仅仅是民用生活中的通信或咨询工具，它可以作为商业、工业、交通、安全等更广阔应用中的数据采集和实时控制设备。然而，移动设备基于它有限的资源，进行有限的网络连接和有限图形功能，成为它展现更广泛潜力的一大障碍。

目前Java技术已经在部分移动终端上得到应用，使移动设备能够通过无线互联网及时地、互动地得到无限网络资源。其中的J2ME技术是专为移动设备应用设立的开发平台，在此基础之上开发移动通信的应用，可为更广泛的移动应用带来新的生机。通过对J2ME的学习和应用研究，在移动终端的信息提取方法基础上寻找具体高性能的实现路径。

2 J2ME移动信息平台体系结构

为了支持用户的需求，以及嵌入式性能的灵活性和可定制性，J2ME设计更强调模块化和可缩放性。它的核心在虚拟机。用于连接虚拟机的是一系列配置和简表，其中提供用于特定J2ME环境的应用类接口(见图1)。每个配置和简表对应通用的或者特定的应用。配置和简表规范是由不同的设备厂商和用户共同开发并建立的。一组配置实际上是用于一组通用设备的最小Java平台。通常它对应共享相同的内存、通信带宽和能量的横向设备分组。

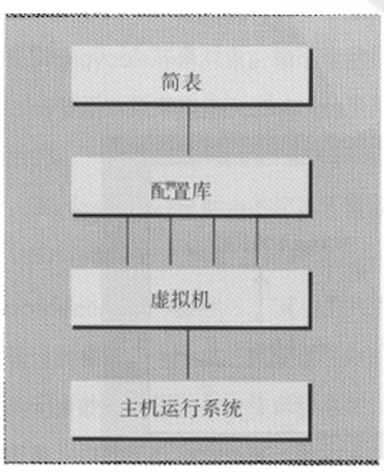

图1　J2ME层次结构图

J2ME定义了两种规范类型：配置（Configurations）和简表（Profile's）：

配置定义了针对一系列设备的Java平台，它与Java虚拟机（JVM）紧密联系。实际上，配置定义的是Java语言特性和VM核心类库。不同的配置应用分类主要是基于不同的设备内存、显示、网络连接和处理能力来考虑的。两种典型的配置包括：连接设备配置(CDC Connected Device Configuration)和连结有限设备配置(CLDC Connected Limited Device Configuration)。

J2ME简表基于配置之上，根据衍生类库及支持设备的不同，又分为基础简表（Foundation profile）和个人简表（Personal profile）等。因为CLDC没有提供与用户、存储设备和网络直接交互的工具，它只是一个基础层，在上层可以架设一系列的简表层来对应每一种简表适应的特定类型的设备形式。移动信息设备简表（Mobile Information Device Profile，MIDP）就是这些简表中的一种。

3 移动终端信息提取

3.1 J2ME的CLDC/MIDP软件体系架构

目前J2ME平台主要分为两大体系，一是基于CDC的CVM\CDC\Foundation profile体系，该架构与CLDC相比，特别针对于内存需求较大、处理器能力较强的设备。另外一种是基于CLDC的KVM\CLDC\MIDP体系，该架构针对于小内存、资源受限的小型设备如手机，PDA等。

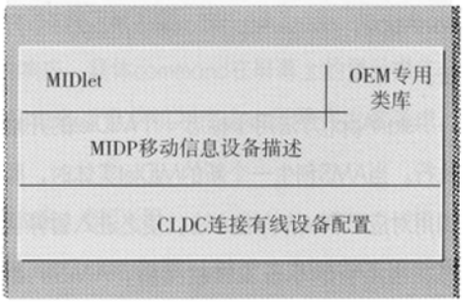

图2　J2ME的CLDC/MIDP软件体系架构

图2展示了CLDC/MIDP的软件体系架构。MIDP在软件体系中的逻辑位置，其上层

有被称为MIDlet的应用程序，其下层有CLDC的支持，MIDP则提供可获得更多功能的包，其中也包括网络持续性存储等。上层的MIDlet可以直接使用MIDP提供的类库，也可以直接使用MIDP从CLDC中继承的API。

3.2 MIDlet的生命周期、套件及运行环境

（1）MIDlet及其运行机制

在MIDP设备上运行的Java程序被称为MIDlet，与通常的Java程序相比，MIDlet的不同在于它更类似于Applet。与J2SE程序相比，MIDlet没有main()初始程序入口点。并且MIDlet也不能调用Runtime.exit()和System.exit()方法来中止虚拟机运行。如果调用则系统会显示SecurityException异常。由于MIDlet是在MID设备上运行的Java程序，所以每个MIDlet必须由抽象类javax.microedition.midlet.MIDlet派生。

（2）MIDlet State（状态）和lifecycle（生命周期）

MIDlet State 包括暂停（Paused）、激活（Active）、消亡（Destroyed）三种状态。它们标识了MIDlet的生命周期。MIDlet的生命周期是由AMS（Application Management software）来控制的。AMS基于操作系统级上，管理着MIDlet运行的底层机制，MIDlet state正是AMS用来控制管理的方法。它可以确保AMS随时消灭一个MIDlet，使某个MIDlet进入暂停态，或者激活一个MIDlet。

在源程序中可以看到，MIDlet程序有startApp()，pauseApp()和destroyApp()3个方法。

startApp()方法用于标志一个MIDlet的开始执行。当AMS创生一个新的MIDlet实体时，即调用对应MIDlet的构造方法，使之进入暂停状态，当所有的准备工作就绪后，AMS判断MIDlet可以运行了，于是调MIDlet.startApp()方法。使MIDlet进入激活态。startApp()方法与HelloWorld程序的构造方法不同，它不仅是在初始化一个MIDlet后执行，而且只要该MIDlet从暂停态被激活，变为激活态，startApp()就会被调用。

pauseApp()方法标志着MIDlet进入暂停态。当AMS决定要把MIDlet转入暂停态，就会调用MIDlet.pauseApp()方法，MIDlet就会暂停MIDlet的执行，并在该状态下释放所占的资源。

destroyApp()方法标志着MIDlet进入消亡态。当AMS判断MIDlet不再需要，就会调用MIDlet.destroyApp()，将MIDlet消毁。

图3展示了在AMS控制下，利用三个状态方法的MIDlet生命周期。

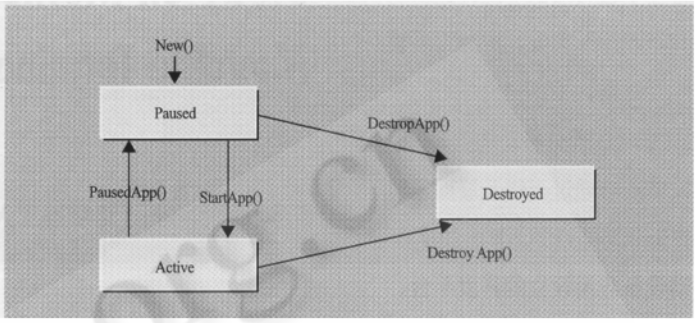

图 3　MIDlet 的生命周期

（3）MIDlet 套件（suite）

通常一组相关MIDlet操作控制集合,被称为MIDlet 套件。MIDlet套件是安装发布到目标设备，或是从设备上卸载移除的一个基本单位。MIDlet套件中的所有MIDlet都共享同一名字空间。同一MIDlet套件中的MIDlet之间可以相互访问记录存储。MIDlet套件被封装到一个JAR中，这在MIDP规范中被称为MIDlet 套件包。在JAR文件中包括继承MIDlet的类和实现该类的文件，还有资源文件和描述JAR文档的清单文件。

（4）MIDlet的执行环境

MIDP规范定义了MIDlet的执行环境，在同一MIDlet套件中的所有MIDlet共享相同的执行环境，MIDlet套件中的任一MIDlet都可与同一MIDlet套件中的其他MIDlet交互在MIDlet的执行环境中，MIDlet可以访问的内容包括：

.实现CLDC和MIDP的类以及它们的本地代码；

.MIDlet Suite（即JAR文件）中的类；

.MIDlet Suite中的资源文件（JAR中的非类文件，如：图片、文本等）；

.描述文件，即JAD文件；

.通过RMS的Record store存储在永续存储介质中的内容。

图4展示了文件之间的关系。

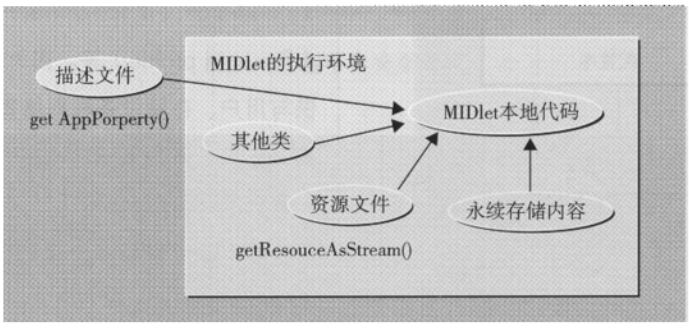

图 4　MIDlet 各文件的关系

3.3 MIDP的UI编程

对于移动设备来说，其用户界面UI (User interface)与我们日常熟悉的PC 机UI有很大不同。移动设备的显示范围相比要小的多，而且输出设备没有鼠标和键盘。在CLDC中没有包含任何UI包，并且在MIDP中虽然有针对移动设备的特定的UI包，但是它们并不支持AWT (Abstract Window Toolkit)。因为MIDP建立的是一个与AWT完全不同的、功能紧缩的GUI包lcdui。

MIDP的lcdui包，包含两种类型的API，针对这两者分别定义了单独的事件处理模式：

高级别API类型，面向移动设备之间的可移植性（portability）的应用。为了达到高可移植性，采用了更高的抽象。因此，对特定GUI可做的视感控制很有限，开发者不能自定义高级别可视化外观的组件。UI组件的交互全部都由具体的实现来执行，开发者并不需要关心这些交互的过程。底层的具体实现会根据设备的硬件特性与UI风格来做出必要的配接工作，这样可使得应用具有较高的可移植性。

低级别API类型，针对的是那些需要精确定位并控制图像元素的应用，并且这种应用往往需要获得较低级别的事件输入。低级API能使开发者全面控制显示的元素。它由javax.microedition.lcdui.Canvas 与javax.microedition.lcdui.Graphics这两个类来实现。

在MIDP设备上进行屏幕显示，有两个概念是重要的，一个是Display显示对象，另一个是Displayable屏幕对象。输出显示是通过Display显示对象来操作Displayable屏幕对象显示的。

Display显示对象相当于设备管理器。每个MIDlet有唯一对应的显示对象（javax.microedition.lcdui.Display），其中的方法是用来管理屏幕的。当向该对象发送消息时，屏幕的元素就会运作。一个应用程序可以有多个屏幕，但在一个显示对象中，同一时刻只能有一个屏幕。

Displayable屏幕对象是MIDP规范中定义的一个抽象，在MIDP规范中屏幕被抽象为一个封装了的设备，它包括特有的图形显示元素和用户输入的对象。目前允许两种类型的屏幕方式：一是Screen封装的UI对象类型，它是指高级抽象的UI组件，如AlertListTextBoxForm；另一个是开发者自定义的图像与输入对象，它的抽象级别较低，如Canvas。

实现显示的步骤如下程序示例：

通过给Display对象的静态方法getDisplay()提供一个MIDlet的自引用，使MIDlet获得了Display对象实例。我们在MIDlet的startApp()方法中来完成这一过程。

```
...
public viod startApp ( ) {
display = Display.getDisplay ( this )
}
...
```

接着，通过把Displayable对象传递给Display的setCurrent()方法，来显示Dispalyable屏幕对象。

```
public void setCurrent ( Displayable d ) ;
public void setCurrent ( Alert alent, Displayable nextDisplayable ) ;
```

我们可以用Display对象提供的getCurrent()，来返回当前被显示的Displayable对象的引用。

```
public void setCommandListener ( CommandListener l ) ;
```

3.4 抽象Command命令与Command侦听

由于MIDP UI的适应广泛性，对于不同的设备来说，MIDP需要高度抽象的输入机制。即在理想情况下，开发者不需考虑键的数量、键的位置，键的绑定等设备特性问题。这样，MIDP定义了抽象的Command机制。

Command对象类似于普通GUI编程的button，可以设定其标题，如：ok、cancel、goback等；当用户调用command时，应用程序做出适当的响应。具体command在屏幕上的显示操作，以及具体设备的与按键建立联系等等，是根据不同设备适合的机制而定，无需开发者考虑。Command对象有3个参数，分别为：

?Label command的标题：Command所显示的内容，如OK等；

?CommandType：用来表示command的类型值，如CommnadType被设定为BACK，可映射到设备的goback键，以及标准的回退操作；

?Priority：command的优先级设置。

举例一个显示“OK”的command：

```
public Displayable getCurrent ( ) ;
```

Command的事件处理模式采取的是标准的Listener侦听模式，需要使用Displayable对象的setCommandListener()方法来注册Command的Listener侦听：

```
Command c = new Command ( "OK" , Command. OK, 0 ) ;
```

这样Command事件就传送到MIDlet的CommandListener()接口应用通过实现commandAction()来处理相应的Command事件：

```
public viod commandAction ( Command c, Displayable d ) ;
```

3.5 可持续存储与RMS

在某些情况下，程序中生成的对象在程序结束后并不立即被消灭，而需要把它存储起来，以供后用,就是所谓的可持续存储（persistent Storage）机制。而RMS(Record Management System)就是MIDP中管理MIDlet的可持续存储介质的系统。

在RMS中，基本的存储单位是记录存储器（Record Store），其中的每一项被称为一个记录。记录存储器是一个由记录组成的二进制文件，它的存储以及完整性读写保证是由相应设备平台完成的。

MIDP通过RecordStore类来操纵记录存储。通过RecordStore的openRecordStore()方法来创建新的记录存储器并打开，或者打开一个已经存在的记录存储器。

记录的操作方法还包括：addRecord()、deleteRecore()、getRecordSize()、setRecord()和getRecord()等。

建立或打开了一个记录存储器后，对其中的记录进行操作，例如可以给记录存储器中添加一个记录，如下代码段所示：

```
String meg1=" hi" ,friend! How are you?"
//数据被存储到 record 之前必须先转为一个 byte数组
byte[] data= meg1.getBytes ( ) ;
try {
rs.addRecord ( data, 0 data.length ) ;
System.out.println ( "Writing record: " + meg1 )
} catch ( Exception e ) {
System.out.printi1n ( " Error adding record:" +e.toString ( ) ) ;
}
```

RMS还提供了接口，方便程序员对记录进行过滤、比较、侦听和枚举。

4 结束语

我们讨论了基于CLDC有限设备连接的移动设备应用实现问题，对在J2ME平台上的MIDlet进行了分析，从它的状态和生命周期、提供的套件功能到运行环境，在理解的基础上给出实现实例。同时讨论了移动信息设备描述MIDP框架下的界面实现技术，包括应用接口类型、屏幕支持类型以及控制移动设备屏幕操作等技术途径。最后我们讨论了移动设备交互操作的高度抽象命令技术和在MIDP上的简单数据库管理系统RMS。通过讨论我们可以清楚地看到，J2ME体现了Java技术特有的优势，它带来的高可移植性，代码小，安全性等，极好地支持移动技术的发展。同时它提供的方便、周到的编程平台，必将成为移动信息终端应用开发的首选方案。

参考文献

1 Roger Riggs Programming Wireless Decices with the Java 2 Platform[M],Micro Edition Addison Wesley 2001。

2 Qusat Mahmound Learning Wireless Java O'Reilly[M], 2001。

3 Yu Feng,Dr.Jun Zhu Wireless Java Programming with Java 2 Micro Edition[M] Smas 2001。

4 John W.Muchow Core J2ME Technology and MIDP[M] Sun Microsystems Press 2001。

5 王森 Java手机程式设计入门[M]，知城数位，台湾，2001。

6 焦祝军、张威，J2ME无线通讯技术应用开发[M]，北京电子希望出版社，2002。