文章编号:1006-2475(2005)08-0005-03

CLDC/MIDP 客户端编程方法综述

李　翀[1],罗家融[1],王华忠[1],刘　隽[2]

(1.中国科学院等离子体物理研究所计算机应用研究室,安徽 合肥　230031;
2.安徽省六安市环保局,安徽 六安　237006)

摘要:对 CLDC/MIDP 客户端编程进行了深入的研究。以 CLDC/MIDP 为基础,结合实例,详细介绍了使用 MIDP 2.0 进行图形界面设计和网络通讯。最后针对 CLDC 本身的限制,提出了进行 MIDP 2.0 设计的一些建议。

关键词:联接设备配置(CDC);有限联接设备配置(CLDC);移动信息设备框架(MIDP)

中图分类号:TP311　　**文献标识码**:A

Program Composition Method Summary on CLDC/MIDP Client

LI Chong[1], LUO Jia-rong[1], WANG Hua-zhong[1], LIU Jun[2]

(1. Computer Application Lab, Institute of Plasma Physics, The Chinese Academy of Sciences, Hefei　230031, China;
2. Anhui Province Liuan City Environmental Protection Bureau, Liuan　237006, China)

Abstract: The paper introduces MIDP 2.0 architecture and it's program composition special for CLDC/MIDP on J2ME Wireless Toolkit and proposes some advice for MIDP 2.0 programmer to solve CLDC's limits.

Key words: connected device configuration (CDC); connected limited device configuration (CLDC); mobile information device profile (MIDP)

0 引　言

随着无线应用与嵌入式系统愈来愈受到人们的重视,SUN 推出了 J2ME (Java 2 Platform, Micro Edition),J2ME 是一种新的、非常小的 Java 应用程序运行环境,主要用于嵌入式系统的 Java,它所定义的构架主要用在手持式设备上。而 MIDP 是 J2ME 的一个关键组件,用于支持小型信息设备(手机、PDA 等)的编程,目前的最新版本是 MIDP 2.0。它具有高级 API 接口,编程简单、可移植性好等特点,深受广大移动通讯厂商的喜爱。

1 CLDC/MIDP 体系结构

J2ME 把设备分为两类,一种是联接设备,另一种是有限联接设备。前一种对应于那些有电源的、电力充裕、较大的设备,例如:电视机、冰箱等;后一种对应于主要使用电池、小型的设备,例如:手机、PDA 等。本文只涉及后者。

J2ME 的实现分为两层:configuration 和 profile。其中 configuration 包括虚拟机(virtual machine)、核心的类库与 API,它提供了一个最基础、最核心的 Java 平台。对应于两种设备的 configuration,分别为 CDC 和 CLDC。CDC 一般使用 JavaVM,而 CLDC 使用的是 KVM(The K Virtual Machine)。KVM 是 SUN 专门为使用 16/32 位 RISC/CISC 微处理器或控制器,其可用内存为 160kB ~ 512kB 的设备而开发的。Profile 层也包含一组 API,主要针对于特定的某一族系的设备而定义。对于手机、PDA 等 CLDC,它们的 profile 层称为 MIDP,它是一个 Java API

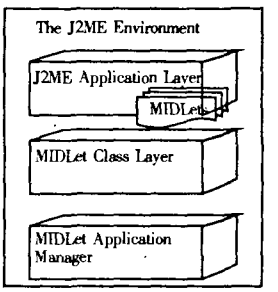

图 1　MIDP 的体系结构

收稿日期:2004-10-14

作者简介:李翀(1978-),男,安徽霍邱人,中国科学院等离子体物理研究所计算机应用研究室硕士研究生,研究方向:软件开发;罗家融(1948-),男,上海人,研究员,博士生导师,硕士,研究方向:数据采集及相关控制;王华忠(1972-),男,安徽桐城人,副研究员,博士,国家自然科学基金项目负责人(10475749),研究方向:计算机控制;刘隽,男,安徽六安人,安徽省六安市环保局工程师,研究方向:环保信息开发。

集合,其体系结构如图 1 所示。它可以实现诸如用户界面、持久存储和联网的功能。MIDP 的核心是一个 MIDlet 应用程序,它继承了 MIDlet 类,允许应用成管理软件对 MIDlet 进行控制、从应用程序描述符检索属性以及对状态变化进行通知和请求。MIDlet 类提供了用于调用、暂停、重新启动和终止 MIDlet 应用程序的 API。

MIDP API 类的完整集合可以分为两个类别。

(1)用于用户界面的 MIDP API。设计这些 API 是为了能以一系列屏幕显示为基础与用户进行交互操作,每一屏幕显示把适量的数据显示给用户。命令以每屏幕为基础提供给用户。这些 API 允许应用程序决定下一屏显示什么、执行什么计算和使用网络服务的何种请求。

(2)用于处理数据库的 MIDP API。这些 API 负责组织和操作设备数据库,这个数据库由在 MIDlet 的多个调用之间跨越时保持持久的信息组成。

下面简单地介绍一下 MIDP API 的 UI 主要元素:

①Alert 用于在屏幕上向用户显示关于异常情况或错误的信息;

②Choice 用于实现从既定数量的选项中进行选择;

③ChoiceGroup 提供一组相关选项;

④Form 作为其它 UI 元素的容器;

⑤List 提供一个选项列表,支持单选和多选功能;

⑥StringItem 当作只显(display-only)字符串使用;

⑦TextBox 允许用户输入和编辑文本的屏幕显示,支持多行显示;

⑧DateField 是一个可编辑的组件,用于表示日期和时间信息。DateField 可以放到 Form 中;

⑨Ticker 用于文本的可滚动显示。

MIDP 还提供了一组用于组织和操作设备数据库的类和接口:RecordStore、RecordComparator 和 RecordFilter。RecordStore 由大量的记录组成,这些记录在 MIDlet 的多个调用之间跨越时保持持久。对 RecordStore 中的记录进行比较,或者从 RecordStore 中抽取若干组记录,都是 RecordComparator 和 RecordFilter 接口提供的功能。下面主要介绍 MIDP 的图形和网络编程。

2 编程环境

操作系统用 Windows 2000p,Java 平台为 j2sdk1.4.2_01,开发工具为 J2ME Wireless Toolkit 1.0.4,它是 MIDP 2.0 编程仿真器,可以对 Java 程序进行仿真处理。编码工具可用 SUN 提供的 IDE Forte for Java。

3 CLDC/MIDP 编程概述

3.1 J2ME Wireless Toolkit 1.0.4 的使用

J2ME Wireless Toolkit 1.0.4 是进行 MIDP 开发的工具之一。先举例介绍一下它的使用,打开 KtoolBar,新建一个工程,输入工程名 Test 和类名 Test,单击生成工程,然后在 setting 里设置所需参数,生成一个工程目录。它包括:bin, classes, lib, res, src, tmpclasses, tmplib 七个目录。把编好的 Test.java 程序拷到 src 目录里。然后在 KtoolBar 中单击编译成功后,再在 device 里选择一个模板,运行后结果如图 2 右边所示。

```
import Javax.microedition.midlet.*,import Javax.microedition.lcdui.*,
public class Sample extends Midlet implements CommandListener{
private Command exitCommand;private TextBox tb;
public Sample(){exitCommand = new Command("Exit", Command.Exit, 1);
tb = new TextBox("Sample MIDlet","Hollo,World!",15,0);
tb.addCommand(exitCommand);tb.setCommandListener(this);}
protected void startApp(){
Display.getDisplay(this).setCurrent(tb);}
protected void pauseApp(){}
protected void destroyApp(boolean u){}
public void commandAction(Command c,Displayable d){
if(c==exitCommand){destroyApp(false);notifyDestroyed();}}}
```

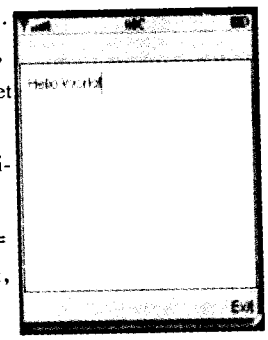

图 2 用 MIDP 2.0 编写的 Test.java 源码及用 KtoolBar 仿真结果

3.2 MIDP 2.0 图形编程

由于受到 CLDC 的限制,MIDP 2.0 只能进行比较简单的图形编程。它提供图形编程用到的类,图 3 显示了这些类及其继承关系。

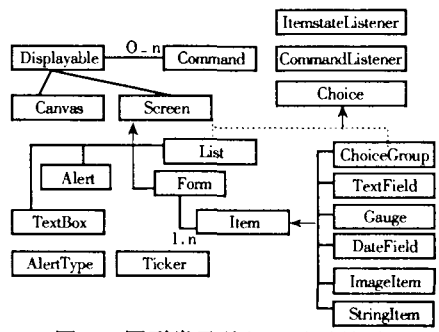

图 3 图形类及其相互继承关系

由图 3 可知,Displayable 类有 Canvas 和 Screen 两

个子类,其中 Screen 属于高级图形界面控件,而 Canvas 属于低级图形界面控件,同一时刻只能显示其中之一。Screen 类有 Alert、List、TextBox、Form 四个子类,其中前三者是事先定义好结构的,只能拿来用,对其内部结构无法修改。而 Form 属于界面容器类,它可以容纳 Item 类别的实现类,当前三种不能满足要求的时候才用它。具体功能在前面已经介绍。

3.3 MIDP 中的事件处理

根据 MIDP 规范,有高级事件处理和低级事件处理以及绘图事件处理。高级事件处理比较简单,由按钮(abstract command)事件产生。低级事件处理比较复杂,由设备上的按钮或屏幕被触摸产生。不同的是采用高级事件处理的 MIDlet 是可移植的,而采用低级事件处理的 MIDlet 可能受某些设备的限制。绘图事件由 Canvas 类的 paint 方法被调用时产生,会传入一个 Graphics 对象作参数,来完成操作。当调用 Display 的 callSerially 会引起继承 Runnable 接口的对象的 Run 方法的调用。

有三种高级事件处理:CommandListener、ItemCommandListener、ItemStateListener,由他们处理不同的事件。CommandListener 用于捕获执行中的高级事件,应用程序需要提供实现类,可通过 AddCommand 方法来实现,然后再注册事件源。另两种处理方法相似,只是方法名不同,处理对象不同而已。

3.4 MIDP 的网络通讯

移动设备的网络编程显然非常重要。通讯所有的类、接口、违例都包含在 javax.microedtion.io 包里面,共有 13 种通讯类型,有 HttpConnection、SocketConnection、CommConnection(串口通信)等。下面介绍并使用 HTTP 来实现 MIDlet 与 Servlet 端对端的无线通讯。Servlet 的开发平台用 SUN 提供的 Tomcat,它支持 Servlet 和 JSP。

首先利用 Tomcat 和 J2SE 来进行 Java Servlet 开发,由它来充当 HTTP 服务器,来返回 HTTP 的请求细节。然后进行客户端编程,包括创建 MIDP 界面和进行通讯,通讯主要有两个基本步骤:发送 HTTP 请求和接收 HTTP 请求。

(1)发送 HTTP GET 请求。

先使用 Connector 类打开一个到服务器的连接,把这个连接强制转换为需要的类型,hcon = (HttpConnection) Connector.open(url);再使用 dis = new DataInputStream(hcon.openInputStream());得到 HttpConnection 上的一个 DataInputStream,允许逐字符地读取服务器的响应数据。使用 DataInputStream 的 read() 方法,服务器响应的每个字符都被集中起来放入 StringBuffer 对象。最后清空连接对象以保存资源。

(2)发送 HTTP POST 请求。

第一步也使用 Connector 类打开一个到服务器的连接,但需加入一个参数来允许客户端通过连接实现在服务器端读和写,同时设置 HttpConnection 对象使用的请求方法为 POST(默认的方法是 GET),得到一个用于现有的 HTTP 连接的 DataOutputStream 对象。用法如下:

```
hcon = (HttpConnection) Connector.open(url, Connector.
READ_WRITE);
hcon.setRequestMethod(HttpConnection.POST);
dos = hc.openDataOutputStream();
```

接下来声明一个字节数组,并通过检索一个来自 requestBody 字符串的字节数组初始化。然后把 DataOutputStream 的缓冲写入字节数组内。

这样客户端(MIDP)和服务器端(Java Servlet)就可以通过超文本传输协议进行通讯了。

4 CLDC/MIDP 编程挑战与展望

CLDC 平台和 PC 平台差别很大,目前 CPU 速度大概 20MHz、内存小于 100kB。必须仔细地评估需要的特性,彻底地优化代码,这样才能达到最佳效果。设计时应该如下:

(1)用轻量级库。在可以达到性能要求的情况下尽量用轻量级库,但可能程序可移植性降低;

(2)减少应用程序占用的空间。鉴于其空间有限,可采用优化程序打包(packaging)过程和分割应用程序的方法来缓解;

(3)最小化无用单元收集过程。虽然 Java 可自动垃圾回收,但必须自行加大回收频率,以最大限度利用空间;

(4)使用移动入口。允许移动客户端利用多种通讯和消息传递协议,使用聚合后端服务并激活捆绑(bundled)服务等;

(5)合理的设计模式。没有适合任何环境的通用设计模式,必须因地制宜。

由此可以看出,MIDP 的发展空间很大,不仅在软件优化上,硬件的发展也非常重要。如果目前的硬件上升一个层次,那么上述的问题也许将不再是一个问题。同时,在软件上进一步优化也是非常有可能的,相信未来的 MIDP 规范能解决上述问题。

5 结束语

通过对 CLDC/MIDP 的研究,可以更好地开发出支持 MIDP 的 CLDC 产品,为广大用户服务。同时也针对目前 CLDC/MIDP 2.0 的局限,给出了一些折衷的办法。不仅对于初学者,而且对于 (下转第 10 页)

一个 final 方法被调用,编译器可以通过调用方法的编译代码直接内嵌来备份子程序的字节码,这样减小了与方法调用有关的系统开销。对 final 方法的调用是在编译时解决的,称为早期绑定(early binding),这依赖于编译器的实现。

5 IO

影响 Java IO 性能最主要的原因之一在于大量地使用单字符 IO 操作,即用 InputStream.read()和 Reader.read()方法每次读取一个字符。Java 的单字符 IO 操作继承自 C 语言,在 C 语言中,单字符 IO 操作是一种常见的操作,比如重复地调用 getc()读取一个文件。C 语言单字符 IO 操作的效率很高,因为 getc()和 putc()函数以宏的形式实现,且支持带缓冲的文件访问,因此这两个函数只需要几个时钟周期就可以执行完毕。在 Java 中,情况完全不同:对于每一个字符,不仅要有一次或者多次方法调用,而且更重要的是,如果不使用任何类型的缓冲,要获得一个字符就要有一次系统调用。对于需要大量读写的操作,应该尽可能地多使用缓存,但如果要经常对缓存进行刷新(flush),则建议不要使用缓存,以下面例子说明。

```
static String readFixedString(int size, DataInput in) throws IOException
{StringBuffer strb = new StringBuffer(size);
Int i = 0;
Boolean more = true;
While(more&&I < size)
{Char ch = in.readChar();
i ++ ;
if(ch = = 0) more = false;
else strb.append(ch);}
in.skipBytes(2 * (size - i));
return strb.toString();}
```

readFixedString从输入流读取字符,直到已读取 size 个字符或者遇到一个 Unicode 值为 0 的字符为止。接着,它跳过剩余的 0 字符。这里使用一个辅助类 StringBuffers,它允许分配一个指定长度的内存块。在这个例子中,字符串最多为 size 个字节长。先创建一个 size 字节的字符串缓存,然后往其中追加读取的字符,每次把字符追加到字符串时,字符串对象需要寻找新内存以存放更大的字符串,这是很费时间的操作。追加很多字符意味着字符串需要一次又一次地重新定位。使用 StringBuffer 则避免了该问题,Java 提供了丰富的缓冲流类可根据实际的情况选择使用。

6 结束语

通过以上的一些方法优化后,Java 程序运行的效率可以有明显的提高。但是在获得效率的同时,也可能会带来一些负面影响,如源代码可读性降低导致可维护性差,文件需占用更多空间等。对于不同的应用程序,对性能的要求也不同。例如,大部分的应用程序在启动时需要较长的时间,从而对启动时间的要求有所降低;服务器端的应用程序通常都分配有较大的内存空间,所以对内存的要求也有所降低。这需要在使用中权衡利弊,针对实际的需要选择使用。

参考文献:

[1] 黄伟峰.Java 性能的优化(上,下)[EB/OL].**http**://www.fanqiang.com/a4/b5/20011113/0908011562.html, 2001-01-29.

[2] [美]Cay S Horstmann, Gary Cornell. Java2 核心技术(第 5 版)卷Ⅰ,卷Ⅱ[M].北京:机械工业出版社,2003.

[3] 俞良松.提高 Java IO 操作的性能[DB/OL].**http**://www.ccw.com.cn/htm/center/prog/02_3_25_2.asp,2002-03-25.

(上接第 7 页)从事 CLDC 产品开发人员也有一定的指导意义。

参考文献:

[1] Laura Lemay, Rogers Cadenhead. Java 2 编程 21 自学通(第 2 版)[M].北京:清华大学出版社,2002.

[2] Michael JunTao Yuan. High-availability Mobile Applications [DB/OL]. **http**://www.javaworld.com/javaworld/jw-06-2003/jw-0606-wireless-p1.html,2003-06-06.

[3] Jonathan Knudsen, Dana Nourie. Wireless Development Tutorial [DB/OL]. **http**://wireless.java.sun.com/midp/articles/wtoolkit. 2002-12-01.

[4] 肖菁. MIDP 2.0 编程[DB/OL]. **http**://www.csdn.net/sun/,2003-12-01.

[5] Jonathan Knudsen. What's New in the J2ME Wireless Toolkit 2.0[DB/OL]. **http**://developers.sun.com/techtopics/mobility/midp/articles/wtk20/,2003-12-21.

[6] 王森.Java 手机/PDA 程序设计入门[M].北京:电子工业出版社,2004.

[7] 黄聪明.Java 移动通信程序设计:J2ME MIDP[M].北京:清华大学出版社,2003.