基本定制指南

J2ME Wireless Toolkit

2.2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, California 95054
U.S.A.
1-800-555-9SUN 或 1-650-960-1300

2004 年 10 月

目录

前言

《*J2ME Wireless Toolkit 基本定制指南*》介绍如何创建自己的设备样机、如何创建混淆器插件，以及如何在 J2ME Wireless Toolkit 上进行其他定制。

本书的适用读者

本指南供需要配置 J2ME Wireless Toolkit 以适应新的设备仿真器样机的开发者使用。本文假定您熟悉 Java 编程、 Mobile Information Device Profile (MIDP) 和 Connected Limited Device Configuration (CLDC) 规范。

本书的组织结构

本指南包含以下章节和附录：

第 1 章概要介绍工具箱定制的可能性。

第 2 章是有关如何创建设备属性文件的教程。该教程说明如何获取和输入图像文件、屏幕属性、按钮属性、软按钮标签区域和图标属性。该教程还介绍了如何设置颜色属性以及如何运行新设备的仿真器。

第 3 章介绍如何创建混淆器插件。

印刷惯例

字体	含义	示例
AaBbCc123	命令、文件和目录的名称，计算机屏幕输出	编辑 .login 文件。 使用 ls -a 列出所有文件。 % You have mail.
AaBbCc123	输入的内容 （相对于计算机屏幕输出信息）	% **su** Password:
AaBbCc123	书名、新词或新术语以及要强调的词	请阅读 《*用户指南*》中的第 6 章。 这些称为*类*选项。 您*必须*是超级用户才能执行此操作。
	命令行变量；请使用实际名称或值替换	要删除文件，请键入 rm *filename*。
{*AaBbCc.dir*}	可变的文件名和目录。	这些文件位于 {*toolkit*}\apps\{*demo_name*}\bin\ 目录下，其中 {*toolkit*} 是 J2ME Wireless Toolkit 的安装目录；{*demo_name*} 是某一个演示程序的名称。

相关文档

应用程序	书名
J2ME Wireless Toolkit	*J2ME Wireless Toolkit 用户指南*
J2ME Wireless Toolkit	*J2ME Wireless Toolkit Toolkit 发行说明*

访问联机文档

下列站点提供与 Java 技术有关的技术文档。

http://developer.sun.com/

http://java.sun.com/docs/

欢迎您提出意见

我们愿意改进自己的文档，欢迎您提出意见和建议。您可以将您的意见通过电子邮件发送给我们：

wtk-comments@sun.com

第 1 章

简介

J2ME Wireless Toolkit 提供了开发 MIDP 应用程序的模拟环境。本文档提供相关说明，以两种有用方式对该工具箱进行定制：

- 创建新的仿真器样机
- 创建混淆器插件

本章其余部分简要介绍每种定制方法。

1.1 创建新的仿真器样机

在 J2ME Wireless Toolkit 中有三种方法对仿真器进行定制：

1. 下载第三方仿真器，并将其安装到 J2ME Wireless Toolkit 中。有关详细信息，请参见《*J2ME Wireless Toolkit 用户指南*》中的第 4.7 节 “使用第三方仿真器”。
2. 基于 J2ME Wireless Toolkit 的缺省仿真器创建新的仿真器样机。在第 2 章 “设置仿真器样机”中对此过程进行了介绍。
3. 定制缺省仿真器的实现。要定制仿真器的实现，您需要获准使用 J2ME Wireless Toolkit 源代码。

1.2 创建混淆器插件

*混淆器*是用于减少可执行 MIDlet 套件大小的工具。 MIDlet 套件越小，就意味着下载时间越短，在目前带宽有限的无线应用中也就意味着更少的等待时间，因此节省了用户的网络费用。

J2ME Wireless Toolkit 包括对 ProGuard 混淆器的支持 (`http://proguard.sourceforge.net/`)，同时还包括灵活的体系结构，能支持各种类型的混淆器。

第 3 章 “创建混淆器插件”中提供了有关的技术细节。

第 2 章

设置仿真器样机

本章介绍如何定义仿真器样机。您可以修改仿真器的现有样机或为其创建新的样机。此过程即为 *设置仿真器样机*。

2.1 样机属性文件

仿真器样机由一个属性文件定义。每个样机属性文件均包含在其 *{toolkit}*\wtklib\devices 子目录中，其中 *{toolkit}* 是 J2ME Wireless Toolkit 的安装目录。属性文件的名称与目录名称相同。

例如， DefaultColorPhone 样机由 *{toolkit}*\wtklib\devices\DefaultColorPhone 目录中的 DefaultColorPhone.properties 定义。

样机属性文件定义仿真器样机的外观和性能。包括指向图像和声音的指针，这些图像和声音文件可以在相同目录中，也可以不在相同目录中。例如，DefaultColorPhone 目录本身包含电话图像，而 DefaultColorPhone 的图标和声音则在 wtklib\devices\Share 中定义。

本章其余部分介绍样机属性文件的内容。该属性文件是纯文本文件。您可以使用任何文本编辑器对其进行修改。通常，属性文件中的条目都有一个属性名，其后有一个属性值。由冒号或等号分隔属性名及其值。以散列标记 (#) 开始的行是注释行。

创建新样机的最简单方法是复制现有样机并对其进行修改。例如：

1. 复制 DefaultColorPhone 目录。
2. 将该目录命名为新样机的名称。
3. 重新命名属性文件使其与目录名相同。如果目录命名为 NewSkin，则将该目录中包含的属性文件重命名为 NewSkin.properties。

2.2 样机外观

仿真器的整体样机外观取决于各种因素，本节将逐一进行介绍：

- 样机图像
- 屏幕边界和可绘画区域
- 屏幕特征
- 图标
- 字体
- 命令
- 声音

2.2.1 样机图像

样机外观主要由三种图像决定：

1. *缺省*图像表示设备处于自然状态。
2. *高亮显示图像*表示设备中的所有按钮都处于高亮显示状态，就像用户将鼠标置于按钮上时显示的那样。
3. *按下*图像表示设备中的所有按钮都已按下。

每个图像都显示完整的设备。 J2ME Wireless Toolkit 使用这些图像的各部分来显示按钮高亮显示和按钮按下。

例如， `DefaultColorPhone` 中的这三种图像显示在图 1 中。

图 1 DefaultColorPhone 的图像：自然、高亮显示和按下

此处显示小键盘的放大图像，您可以看到三种图像的不同之处。

图 2 仿真器样机图像特写：自然、高亮显示和按下

在样机属性文件中，这三种图像文件使用以下属性指定：

```
default_image=< 图像文件名 >
highlighted_image=< 图像文件名 >
pressed_buttons_image=< 图像文件名 >
```

图像文件可以是 PNG、 GIF 或 JPEG 格式。这些图像应该具有相同的尺寸。

例如， DefaultColorPhone.properties 具有以下条目：

```
default_image=neutral.png
highlighted_image=hilight.png
pressed_buttons_image=pressed.png
```

2.2.2 屏幕边界和可绘画区域

屏幕反映真实设备的显示内容。由整个屏幕边界、*可绘画*边界和决定诸如颜色数目因素的其他参数定义。

整个屏幕边界是指全部显示区域。它们相对于左上角的图像文件的原点，以像素为测量单位来定义。

图 3 屏幕边界

在属性文件中按如下方法指定屏幕边界：

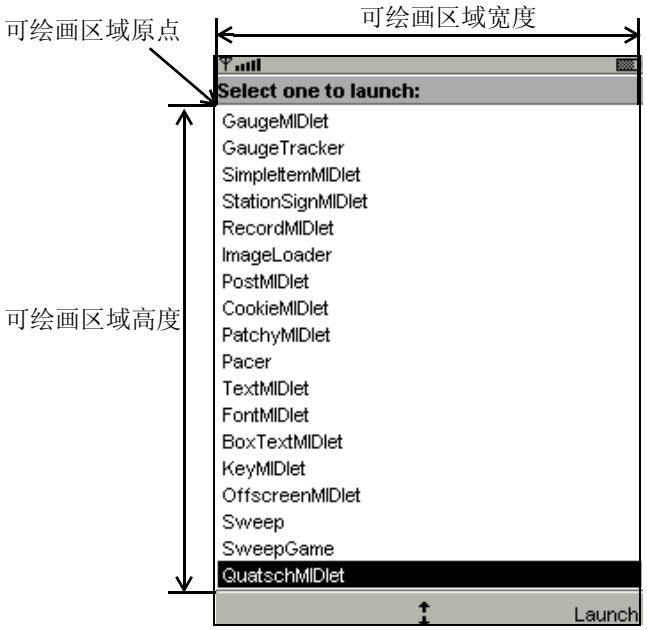
```
screen.x=<x 坐标>
screen.y=<y 坐标>
screen.width=< 宽度>
screen.height=< 高度>
```

例如：

```
screen.x=60
screen.y=76
screen.width=240
screen.height=320
```

大多数设备无法使 MIDP 应用程序使用全部显示区域。屏幕的其余部分通常用于显示图标和各种指示器。同样， J2ME Wireless Toolkit 仿真器允许您定义整个屏幕的子集，即*可绘画*区域，该区域为 MIDP 应用程序所用。可绘画区域用坐标系表示，以屏幕的左上角为原点。例如， `DefaultColorPhone` 仿真器样机使用顶栏显示图标，而使用底栏显示软标签和其他图标，如图 4 所示。

图 4 DefaultColorPhone 中的可绘画屏幕区域



在仿真器样机属性文件中，可绘画区域按如下方法进行定义：

```
screenPaintableRegion.x=<x 坐标>
screenPaintableRegion.y=<y 坐标>
screenPaintableRegion.width=< 宽度>
screenPaintableRegion.height=< 高度>
```

例如：

```
screenPaintableRegion.x=0
screenPaintableRegion.y=10
screenPaintableRegion.width=240
screenPaintableRegion.height=290
```

注—对于全屏模式 （在 MIDP 2.0 中），仿真器可以使用从可绘画区域原点到屏幕右下角之间的区域。在 DefaultColorPhone 中，是指顶栏除外的整个屏幕区域。

2.2.3 屏幕特征

仿真器样机属性文件决定屏幕支持的颜色数目，以及像素的高宽比。首先，下面的属性可以指定仿真器样机使用彩色还是灰度级。

`isColor=<`*true 为彩色 false 为灰度级*`>`

另一个属性是 `colorCount`，用于指定可用颜色的数目。对于灰度级设备，该属性用于指定灰度级别数。

`colorCount=<` *数值* `>`

例如， `DefaultColorPhone` 的彩色屏幕为 4096 色：

```
isColor=true
colorCount=0x1000
```

仿真器对 Alpha （透明）的处理由以下属性决定：

`enableAlphaChannel=<`*true 或 false*`>`

也可以使用以下属性来启用伽玛修正：

`gamma=<` *数值，其中 1 表示没有任何错误修正* `>`

使用以下属性可以启用或禁用双缓冲：

`screenDoubleBuffer = <`*true 或 false*`>`

屏幕非可绘画区域的背景颜色可按下面方法进行定义：

`screenBorderColor=<` *颜色* `>`

例如， `DefaultColorPhone` 使用以下颜色：

```
screenBorderColor=0xb6b6aa
```

在灰度级设备上，可以使用以下属性来设置屏幕的背景颜色：

`screenBGColor=<` *颜色* `>`

2.2.4 图标

J2ME Wireless Toolkit 仿真器支持使用图标，这些图标是能向用户传递信息的小图像。通常情况下，图标位于屏幕上的可绘画区域之外。仿真器实现了一个固定的图标集，如表 1 所示。

表 1 仿真器图标

名称	描述
battery	显示电池状态
domain	表明正在运行的 MIDlet 的保护域。
down	表明可以滚动

表 1 仿真器图标 （续）

名称	描述
inmode	表明输入模式：小写、大写、数字
internet	显示 Internet 活动
left	表明可以滚动
reception	显示无线信号强度
right	表明可以滚动
up	表明可以滚动

可以定义图标的位置 （相对于屏幕原点进行测量）、缺省状态以及与可能的状态相对应的图像列表。例如，这里是对 DefaultColorPhone 中 down 图标的定义。该图标是一个下指箭头，当显示的列表或表单高度超过可用屏幕空间时，就会出现该图标。

```
icon.down:113, 314, off
icon.down.off:
icon.down.on:../Share/down.gif
```

第一行指定图标要显示的位置，对于 DefaultColorPhone 来说，位于底栏中央位置，可绘画屏幕区域以外。缺省状态为 off。

没有与 off 状态相对应的图像文件，而 on 状态使用 wtklib\devices\Share 目录中的 down.gif 图像。

另外一个有趣的示例是 inmode 图标，它有 7 种状态和 6 个对应的图像文件：

```
icon.inmode:113, 2, off
icon.inmode.off:
icon.inmode.ABC:../Share/ABC.gif
icon.inmode.abc:../Share/abc_lower.gif
icon.inmode.123:../Share/123.gif
icon.inmode.kana:../Share/kana.gif
icon.inmode.hira:../Share/hira.gif
icon.inmode.sym:../Share/sym.gif
```

仿真器的另外一个特点是网络指示器，与图标类似。网络指示器不在屏幕中显示，而是显示在仿真器样机上。在 DefaultColorPhone 中，网络指示器是显示在仿真器样机左上角的一个小绿灯。网络指示器由两个属性进行定义：

```
netindicator.image:< 图像 >
netindicator.bounds:<x>, <y>, < 宽度 >, < 高度 >
```

例如，在 DefaultColorPhone 中，网络指示器如下所示：

```
netindicator.image:net_indicator.png
netindicator.bounds: 53, 27, 30, 30
```

其中，宽度和高度应与网络指示器图像的宽度和高度一致。

2.2.5 字体

仿真器使用的字体在样机属性文件中定义。实际上，您可以定义 MIDP `Font` 类中可用的每个字体、字形和字号。格式显示如下：

`font.<` *字体* `>.<` *字形* `>.<` *字号* `>:<` *字体说明符* `>`

您可以推测 MIDP `Font` API 中的字体、字形和字号参数，除非标识符在仿真器样机属性文件中是以小写字母表示。字体为 `system`、 `monospace` 或 `proportional`，字形为 `plain`、 `bold` 或 `italic`，字号为 `small`、 `medium` 或 `large`。

字体说明符遵照 J2SE `java.awt.Font` 类中定义的惯例。 `DefaultColorPhone` 中的以下示例定义了三种字号的成比例斜体字体：

```
font.proportional.italic.small:SansSerif-italic-9
font.proportional.italic.medium:SansSerif-italic-11
font.proportional.italic.large:SansSerif-italic-14
```

您需要指定一个缺省字体，在没有其他定义时使用该字体。在 `DefaultColorPhone` 中，使用的缺省字号为 10 磅，字体为 SansSerif：

```
font.default=SansSerif-plain-10
```

字体还可以添加下划线。缺省情况下， MIDP 实现支持带有下划线的字体，但是，您也可以禁止特定字体使用下划线，方法如下：

`font.<` *字体* `>.<` *字形* `>.<` *字号* `>.underline.enabled=false`

如果您愿意，可以禁止所有字体使用下划线，方法如下：

```
font.all.underline.enabled=false
```

除了使用系统字体之外，您还可以选择使用*位图*字体。位图字体只是一个包含某种字体的字符形状的图像。位图字体图像是一行文本，包含某一种字符形状。要定义位图字体，请使用以下属性：

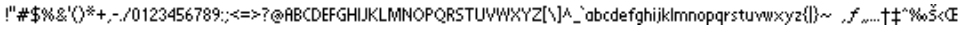
`font.<` *名称* `>=<` *字体属性文件* `>`

字体属性文件包含下列属性定义：

```
font_image = < 图像文件 >
font_height = < 字体高度 >
font_ascent = < 字体上升 >
font_descent = < 字体下降 >
font_leading = < 字体前导 >
```

图像文件应为 PNG、 GIF 或 JPEG 格式。应包含一行字符：

图 5 位图字体图像



高度、上升、下降和前导均以像素为单位指定。如果您不熟悉这些字体术语，请参阅 `java.awt.FontMetrics` 的 J2SE 文档。

字体属性文件还应包含 ASCII 字符代码与像素在图像内的水平偏移量间的映射列表。在以下示例中， ASCII 代码 65 被映射到水平偏移 124：

```
ascii_x-65=124
```

定义位图字体之后，其名称可以用作字体说明符。

2.2.6 软按钮标签

软按钮是没有固定功能的按钮。本章后面将详细论述这些按钮。软按钮的标签显示在屏幕上。仿真器样机属性文件决定软按钮标签的显示位置和显示方式。

软按钮标签的字体使用字体别名进行定义，字体别名是分配给字体的简称。每个软按钮标签由一个属性描述：

```
softbutton.<n>=<x>, <y>, < 宽度 >, < 高度 >, < 字体别名 >, < 对齐方式 >
```

其中，*对齐方式*的有效值为 `left`、 `right` 以及 `center`。

例如，以下属性告诉工具箱对软按钮标签使用 12 磅的 Courier 字体。首先，定义字体别名 `softButton`。第一个标签为左对齐，第二个标签为右对齐。

```
font.softButton=Courier-plain-12
softbutton.0=1,306,78,16, softButton, left
softbutton.1=160,306,78,16, softButton, right
```

2.2.7 声音

MIDP 警报都有相应的声音。在 J2ME Wireless Toolkit 仿真器中，声音使用文件定义，对于 MIDP `AlertType` 类中列举的每种类型都有一个相应的文件。仿真器可以使用由底层 J2SE 实现支持的任何声音文件类型。在 J2SE SDK 1.4 中，支持的声音文件类型包括 AIFF、AU、WAV、MIDI 和 RMF。例如，在 `DefaultColorPhone` 中有如下定义：

```
alert.alarm.sound:../Share/mid_alarm.wav
alert.info.sound:../Share/mid_info.wav
alert.warning.sound:../Share/mid_warn.wav
alert.error.sound:../Share/mid_err.wav
alert.confirmation.sound:../Share/mid_confirm.wav
```

如果没有为特定警报类型定义相应的声音，则将播放*缺省*声音：

```
alert.confirmation.sound:< 声音文件 >
```

此外，还可以定义能模仿电话震动的声音。在 `DefaultColorPhone` 中，应如下所示：

```
vibrator.sound:../Share/vibrate.wav
```

2.3 映射用户输入

通过以下两个部分来说明仿真器样机。第一部分是上文中介绍的仿真器样机外观。第二部分定义用户输入在仿真器中是如何进行映射的。

2.3.1 键盘处理程序

*键盘处理程序*在仿真器中执行按钮按下时的相应操作。例如，如果您使用鼠标按下某个软按钮，则键盘处理程序就在仿真器中执行相应的操作。

键盘处理程序定义了一组标准按钮名称，您将在定义按钮时用到它们。您只需告诉仿真器按钮在样机中的位置，其余的事情则由键盘处理程序来处理。

J2ME Wireless Toolkit 仿真器包括两个键盘处理程序，一个是面向使用 ITU-T 小键盘 (`DefaultKeyboardHandler`) 的电话设备，另外一个是面向使用全标准键盘的设备。例如， `DefaultColorPhone` 中包括该键盘处理程序属性：

```
keyboard.handler = com.sun.kvem.midp.DefaultKeyboardHandler
```

`DefaultKeyboardHandler` 能识别下列标准按钮名称：`0` 到 `9`、 `POUND`、`ASTERISK`、`POWER`、`SEND`、`END`、`LEFT`、`RIGHT`、`UP`、`DOWN`、`SELECT`、`SOFT1`、`SOFT2`、 `SOFT3`、 `SOFT4`、 `USER1` 到 `USER10`。

在 `QwertyDevice` 中，键盘处理程序如下所示：

```
keyboard.handler = com.sun.kvem.midp.QwertyKeyboardHandler
```

`QwertyKeyboardHandler` 支持与 `DefaultKeyboardHandler` 相同的按钮，同时还包括标准键盘上的按钮，如字母键、 Shift 键和 Alt 键。

2.3.2 按钮

按钮使用名称和一组坐标来定义。如果提供两组坐标，则可定义矩形按钮。如果提供两组以上坐标，则使用多边形区域定义按钮。

按钮区域的定义与设备样机图像有关。当用户将鼠标移至某个已定义的按钮区域时，即显示样机高亮图像中的相应区域。如果用户按下某个按钮，即显示样机按下图像中的相应区域。

按钮本身并无特别值得关注之处。它们只是将按钮名称与某个矩形或多边形区域相关联罢了。而将按钮名称映射到仿真器中的某项功能则是键盘处理程序的事情。您随后即会了解如何才能将台式计算机键盘上的键映射到按钮。

以下属性显示如何为按钮 5 定义矩形区域。该区域原点为 140， 553，宽度为 84，高度为 37。

```
button.5 = 140, 553,  84,  37
```

下面是为星号按钮定义多边形的一个示例：

```
button.ASTERISK = 66, 605, 110, 606, 140, 636, 120, 647, 70, 637
```

用直线段连接所列出的各点，来定义该多边形：

```
66, 105
110, 606
140, 636
120, 647
70, 637
```

2.3.3 为按钮分配台式机键盘键

按钮可以与一个或多个台式机键盘键建立关联。也就是说，您可以使用台式机键盘控制仿真器，无需在设备样机上移动鼠标并按下鼠标按钮。

例如， DefaultColorPhone 允许您在台式机键盘上按下 F1 模拟左软按钮。左软按钮在属性文件中定义为 SOFT1：

```
button.SOFT1 = 78, 417, 120, 423, 126, 465, 74, 440
```

并且台式机键盘快捷键这样定义：

```
key.SOFT1 = VK_F1
```

这些键定义实际上是*虚拟键代码*，在 J2SE `java.awt.event.KeyEvent` 类中进行定义。有关详细信息，请参见 J2SE 文档。

如果您愿意，可以为某个按钮分配多个台式机键盘键。在下面 DefaultColorPhone 的示例中，台式机键盘上的 5 键或数字键盘上的 5 键都被定义为仿真器样机上按钮 5 的快捷键：

```
key.5 = VK_5 VK_NUMPAD5
```

2.3.4 映射游戏键

游戏操作已在 DefaultKeyboardHandler 中进行定义，但是您也可以用 QwertyKeyboardHandler 指定自己的游戏操作。使用如下格式：

game.< *功能* > = < *按钮名称* >

可以为 LEFT、 RIGHT、 UP、 DOWN 和 SELECT 功能中的任意一种。标准按钮名称已经在本章前面进行了介绍。

缺省设置为：

```
game.UP = UP
game.DOWN = DOWN
game.LEFT = LEFT
game.RIGHT = RIGHT
game.SELECT = SELECT
```

2.3.5 将键映射到字符

使用 `QwertyKeyboardHandler`，您可以指定按钮按下时生成的字符，包括单独按下按钮和使用 Shift 或 Alt 组合按钮时生成的字符。

使用如下格式：

`keyboard.handler.qwerty.<` *按钮* `> = '<` *基本字符* `>' '<`*Shift 字符*`>' '<`*Alt 字符*`>'`

其中基本字符是该按钮正常情况下生成的字符， Shift 字符是在按住 Shift 键的同时按下该按钮时生成的字符， Alt 字符是在按住 Alt 键的同时按下该按钮时生成的字符。

按住 Shift 或 Alt 键的同时按下按钮有两种方式：

- 如上节所述，将按钮映射到键盘，在按住 Shift 或 Alt 键的同时按下与此按钮关联的键。
- 按住 Shift-Lock 或 Alt-Lock 键，然后按下按钮。再次按下 Shift-Lock 或 Alt-Lock 键，返回到初始状态。

例如：

```
keyboard.handler.qwerty.A = 'a' 'A' '?'
```

2.3.6 将命令映射到软按钮

*命令*是 MIDP 规范的一部分。通过它们可以灵活地指定用户应当可以使用哪些操作，而不用管具体设备是如何使操作变为可用的。

一般来说， MIDP 设备使用软按钮来调用命令。命令文本显示在屏幕上，位于软按钮附近。如果命令数量多于可用的软按钮数量，则实现会将一个软按钮标签显示为菜单。按下菜单软按钮会显示一个包含可用命令的菜单。

根据 `javax.microedition.lcdui.Command` 中指定的命令类型，J2ME Wireless Toolkit 仿真器允许您指定特定类型命令出现的位置。例如，在具有两个软按钮的仿真器样机上，您可能希望 `BACK` 和 `EXIT` 命令出现在左软按钮上，而 `OK` 命令出现在右软按钮上。

您可以在仿真器样机属性文件中使用如下格式指定这些偏好类型：

```
command.keys.< 命令类型 >=< 按钮 >
```

例如， DefaultColorPhone 这样来定义命令偏好：

```
command.keys.BACK = SOFT1
command.keys.EXIT = SOFT1
command.keys.CANCEL = SOFT1
command.keys.STOP = SOFT1

command.keys.OK = SOFT2
command.keys.SCREEN = SOFT2
command.keys.ITEM = SOFT2
command.keys.HELP = SOFT2
```

通过指定其他按钮名称，您可以为特殊的命令类型指定其他首选按钮。例如，下面一行代码告诉仿真器 BACK 命令应该映射到 END （如果可用），或映射到 SOFT1。

```
command.keys.BACK = END SOFT1
```

最后，如果您愿意，可以指定软按钮只用于特定命令类型。下列定义只允许将 BACK、EXIT、 CANCEL 和 STOP 命令类型映射到 SOFT1 键。

```
command.exclusive.SOFT1 = BACK EXIT CANCEL STOP
```

2.3.7 命令菜单

命令数量多于可用的软按钮数量时，会将命令放入一个菜单中。 J2ME Wireless Toolkit 仿真器提供对命令菜单的控制。您可以选择用于显示该菜单的按钮，也可以选择用于遍历菜单项的按钮，还可以选择为菜单显示的文本标签。

DefaultColorPhone 中的以下属性告诉仿真器样机使用第二个软按钮来显示或隐藏菜单。

```
command.menu.activate = SOFT2
```

缺省情况下，UP 和 DOWN 按钮用于遍历菜单，而 SELECT 按钮则用于选择命令。您可以使用下列属性更改这些分配：

```
command.menu.select = < 按钮 >
command.menu.up = < 按钮 >
command.menu.down = < 按钮 >
```

2.3.8 暂停和恢复

MIDP 规范允许随时暂停应用程序 (MIDlet)，可能是对诸如传入呼叫等其他电话事件作出响应。

您可以使用仿真器样机属性文件定义暂停或恢复 MIDlet 的台式机键盘快捷键。例如，DefaultColorPhone 使用 F6 暂停 （挂起），使用 F7 恢复：

```
midlet.SUSPEND_ALL = VK_F6
midlet.RESUME_ALL = VK_F7
```

2.3.9 指针事件

有一个属性可确定仿真器样机是否具有触摸屏。如果具有触摸屏，指针事件将被传送给 Canvas。

```
touch_screen=<true 或 false>
```

2.4 语言环境和字符编码

语言环境是指拥有相同语言、习惯或文化传统的地理位置、行政区域或社区。在软件中，语言环境是文件、数据和代码的集合，它包括使软件适应特定地理位置所必需的信息。

某些操作对语言环境较敏感，需要有指定的语言环境才能使信息与用户相适应，例如：

- 显示给用户的消息
- 文化信息，如日期和货币格式

在 J2ME Wireless Toolkit 仿真器中，缺省语言环境由平台的语言环境决定。

要定义特定的语言环境，请使用以下定义：

```
microedition.locale:< 语言环境名称 >
```

语言环境名称由两部分组成，用破折号 (-) 进行分隔，例如， en-US 是指英语 / 美式英语，而 en-AU 是指英语 / 澳大利亚英语。

第一部分是有效的 ISO 语言代码。这些代码是 ISO-639 定义的两个小写字母代码，可以在很多站点上找到这些代码的完整列表，例如：

http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

第二部分是有效的 ISO 国家 / 地区代码。这些代码是 ISO-3166 定义的两个大写字母代码，可以在很多站点上找到这些代码的完整列表，例如：

http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

CLDC 中的输入和输出 API 使用命名的字符编码，可以在 8 位字符与 16 位 Unicode 字符之间转换。特定的 MIDP 实现可以仅选择一个小编码集供 MIDlet 使用。

在仿真器中，缺省编码是正在平台上运行的缺省编码。您的仿真器可能使用其他编码，如 UTF-8 和 UTF-16，假设可以在 J2SE 平台中使用这些编码。

要定义仿真器样机使用的字符编码，请使用以下定义：

`microedition.encoding:<` *编码* `>`

要定义包含所有可用编码的编码集，请使用以下定义：

`microedition.encoding.supported:<` *编码列表* `>`

例如：

```
microedition.encoding:UTF-8
microedition.encoding.supported:UTF-8, UTF-16, ISO-8859-1,
    ISO-8859-2, Shift_JIS
```

要支持 J2SE 平台支持的所有编码，请保留 `microedition.encoding.supported` 定义为空白，如下所示：

```
microedition.encoding.supported:
```

注一无论 ISO-8859-1 编码是否在 `microedition.encoding.supported` 条目中列出，该编码对运行在仿真设备上的应用程序总是有效的。

第 3 章

创建混淆器插件

通过 J2ME Wireless Toolkit，您可以使用字节码混淆器来减小 MIDlet 套件 JAR 的大小。该工具箱附带对 ProGuard 和 RetroGuard 的支持，在《*J2ME Wireless Toolkit 用户指南*》中进行了相关介绍。

如果您希望使用不同的混淆器，还可以为 J2ME Wireless Toolkit 编写一个插件。

3.1 编写插件

混淆器插件扩展了 `com.sun.kvem.environment.Obfusctor` 接口。该接口本身包含在 *{toolkit}*\wtklib\kenv.zip 中。

`Obfuscator` 接口包含两个您必须实现的方法：

```
public void createScriptFile(File jadFilename, File projectDir);

public void run(File jarFileObfuscated, String wtkBinDir,
    String wtkLibDir, String jarFilename, String projectDir,
    String classPath, String emptyAPI) throws IOException;
```

要对您的混淆器插件进行编译，请确保将 `kenv.zip` 添加到您的 `CLASSPATH` 中。

例如，现在有一个非常简单的插件的源代码。该插件实际上并没有调用某个混淆器，但它说明了实现 Obfuscator 接口的方法。

```
import java.io.*;


public class NullObfuscator
    implements com.sun.kvem.environment.Obfuscator {
  public void createScriptFile(File jadFilename, File projectDir) {
    System.out.println("NullObfuscator:createScriptFile()");
  }

  public void run(File jarFileObfuscated, String wtkBinDir,
      String wtkLibDir, String jarFilename, String projectDir,
      String classPath, String emptyAPI) throws IOException {
    System.out.println("NullObfuscator:run()");
  }
}
```

假设您将这段代码保存为 *{toolkit}*\wtklib\test\NullObfuscator.java。然后您就可以在命令行中对其进行如下编译：

```
set classpath=%classpath%;\WTK22\wtklib\kenv.zip
javac NullObfuscator.java
```

3.2 配置 Toolkit

编写完您的混淆器插件之后，需要向工具箱指明该插件所在的位置。这就需要编辑 *{toolkit}*\wtklib\Windows\ktools.properties。您需要编辑该混淆器插件的类名，并向工具箱指明该类所在的位置。如果沿用该示例，请按如下方法编辑属性：

```
obfuscator.runner.class.name:NullObfuscator
obfuscator.runner.classpath:wtklib\\test
```

重新启动 KToolbar 并打开一个项目。选择 **“项目”** > **“包”** > **“创建混淆包”**。在 KToolbar 控制台中，您将看到 NullObfuscator 的输出内容：

```
Project settings saved
Building "Tiny"
NullObfuscator:createScriptFile()
NullObfuscator:run()
Wrote C:\WTK22\apps\Tiny\bin\Tiny.jar
Wrote C:\WTK22\apps\Tiny\bin\Tiny.jad
Build complete
```

索引

T

Y

Z