如何创建定制的 BlackBerry UI Field

作者：杨江

目录

一 概述

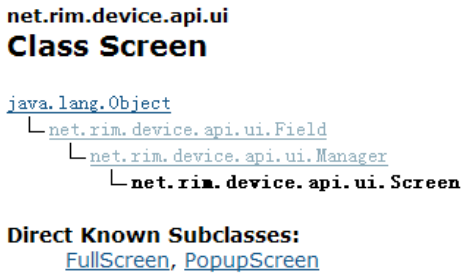
BlackBerry 平台为开发人员提供了很多标准的 UI 组件，为程序员快速开发手机应用奠定了基础。但大量手机软件是消费类、娱乐类软件，这些软件的 UI 界面需要独特、于众不同，能抓住用户的眼球。

标准的 UI 组件经常不能满足开发商和用户的独特要求，很多时候开发人员需要发挥其想象力去创造和定制出新的 UI 组件。所幸 BlackBerry 平台上面创建定制的 UI 组件是一件比较简单的事情。

二 BlackBerry UI Field

从 BlackBerry API Java doc 中我们看到，我们常用的 UI 组件，即 Field，比如文本框 ButtonField、LabelField、TextField 等等都是扩展 net.rim.device.api.ui.Field 而来。

更高级的 BlackBerry UI 组件，比如管理器 Manager 和窗口 Screen 也是继承 Field 类。



三 创建定制的 UI Field 的方法

创建自定义字段，编写一个 Field，通常至少需要实现 layout()和 paint()两个方法，以设置 Field 的宽度和高度，显示 UI 组件的界面。其他方法可选，具体列表如下：

Field 是 UI 的最小单元，这个最小单元里面不能放置其他 Field 。但是 Manager 类可以，Manager 类里面可以嵌套放置 Manager 类，Manager 管理 Field 在屏幕上的摆放位置。	`extends Field`并实现相应的方法。 或者是`extends` 现有的`Field`类，比如`LabelField`、`ListField`
Field 在手机屏幕上显示为一个长方形的区域，有高	实现 `layout()`：调用 `setExtent()`方法设

度，宽度	置 Field 的宽度和高度。
在 Field 长方形的区域中更改背景，显示图形/文字，加边框等	实现 paint()：使用 Graphics 对象的 drawLine，drawRect，drawText 等方法来绘制 Field
Field 可以选择是否要处理和如何处理键盘/轨迹球事件，例如用户按下“黑莓退出键”，你可以选择弹出对话框 Dialog，让用户确认需要退出	实现 keyChar()，trackwheelClick()，invokeAction()等方法。
Field 需要告诉 Manager 它的 prefered 宽度和高度，以便 Manager 控制包含的各个 Field 的显示	override getPreferredWidth 和 getPreferredHeight 方法
Field 被选中，on focus 的时候可以选择重新绘制字段。 注：用户希望 Field 选中/非选中状态显示的界面不一样，比如选中的时候希望高亮度显示，加边框；没有选中就低亮度显示即可。	实现 drawFocus ()：使用 Graphics 对象 setBackgroundColor() 方法改变背景色，drawLine，drawRect，drawText 等方法来绘制 Field
Field 可以选择是否要处理和如何处理 Focus/selected 事件	实现 onFocus ()

下面让我们通过实现一组自定义 Field 来让我们的应用程序用户界面更加丰富多彩。

3.1 HyperlinkButtonField

缺省的 ButtonField 

是一个有边框的长方形的按钮，按钮宽度比按钮文字长度稍宽。

在某些信息处理应用中，如果 UI 界面里面按钮和操作太多，用户会觉得按钮多而信息少。为什么不考虑将文字信息内容和文字相关的操作（按钮） 混合显示呢？

或者是用超文本链接按钮代替传统的 ButtonField 

呢？

我们不必从头做一个这样的控件--extends Field 并实现 Field 接口的很多方法。我们大可以找一个和我们需要的界面相似的一个已经存在的 Field 进行扩展。

下面，我们扩展 BlackBerry 平台提供的 LabelField，来实现我们的 HyperlinkButtonField，这样就不需要管基本的文字的显示功能，而只需要扩展实现如下功能：

1. HyperlinkButtonField 显示的文字要有下划线，并且被选中和没有选择的字体颜色要不同，背景色也不同。
2. 用户按下轨迹球，或者安装键盘 Enter 键，要触发 HyperlinkButtonField 关联的动作。

具体实现计划：

HyperlinkButtonField	extends 现有的LabelField
Field 在手机屏幕上显示为一个长方形的区域，有高度，宽度	*不去实现了，借用 LabelField 的 layout()*
显示带下划线的文字	实现 paint()：修改该背景色，修改字体颜色，然后借用父类 LabelField 的 paint() 方法显示文字
Field 可以选择是否要处理和如何处理键盘/轨迹球事件，例如用户按下“黑莓退出键”，你可以选择弹出对话框 Dialog，让用户确认需要退出	实现 keyChar()，trackwheelClick()，invokeAction()等方法。
Field 需要告诉 Manager 它的 prefered 宽度和高度，以便 Manager 控制包含的各个 Field 的显示	override getPreferredWidth 和 getPreferredHeight 方法
Field 被选中，on focus 的时候可以选择重新绘制字段。 注：用户希望 Field 选中/非选中状态显示的界面不一样，比如选中的时候希望高亮度显示，加边框；没有选中就低亮度显示即可。	实现 drawFocus ()：使用 Graphics 对象 setBackgroundColor() 方法改变背景色，drawLine，drawRect，drawText 等方法来绘制 Field

HyperlinkButtonField.java extends LabelField，通过三个color变量来记录和控制字体的颜色。

```
public class HyperlinkButtonField extends LabelField
{
    private int _textColour;
    private int _textColourFocus;
    private int _highlightColour;

    private XYRect _tmpRect = new XYRect();

    public HyperlinkButtonField( String text, int textColour, int textColourFocus, int
highlightColour, int menuOrdinal, int menuPriority, long style )
    {
        super( text, Field.FOCUSABLE | style ); //这个Field需要FOCUSABLE style以让用户能够选中

        _textColour = textColour;
        _textColourFocus = textColourFocus;
        _highlightColour = highlightColour;
    }
```

实现applyFont()。设置使用带下划线的字体Font.*UNDERLINED*，看上去这个象一个hyper link。

```
    public void applyFont()
    {
        Font underlineFont = getFont().derive( Font.UNDERLINED );
        setFont( underlineFont );
    }
```

实现 paint()。 字段的 Manager 将调用 paint()，以在某个字段区域被标记为无效时重新绘制字段。调用 graphics.setColor () 来设置当 Field 在没有选择(非 Focus)状态时候的字体颜色。在在修改了字体颜色参数后，调用 super.paint()也就是 LabelField.paint()方法来绘制显示这个 Field。

```
protected void paint( Graphics g )
    {
        int oldColour = g.getColor();
        try {
            if(g.isDrawingStyleSet( Graphics.DRAWSTYLE_FOCUS ) ) {
                g.setColor( _textColourFocus );
            } else {
                g.setColor( _textColour );
            }
            super.paint( g );  //修改了字体以后，调用父类LabelField的paint()来重新画指定颜色的文字
        } finally {
            g.setColor( oldColour );  //这里用临时变量来恢复Graphics原来的颜色
        }
    }
```

实现 drawFocus ()。 字段的管理器将调用 drawFocus ()，以在某个字段被选中造成该区域在被标记为无效时重新绘制字段。调用 g.setBackgroundColor( ) 来设置当 Field 在被选中(Focus)状态时的画布的背景色；调用 paint()重新画这个字段，当然 paint()里面会调用 super.paint()也就是 LabelField.paint()方法来绘制显示这个 Field。

```
  protected void drawFocus( Graphics g, boolean on )
    {
        getFocusRect( _tmpRect ); //Retrieves this field's current focus region.

        boolean oldDrawStyleFocus = g.isDrawingStyleSet( Graphics.DRAWSTYLE_FOCUS );
        int oldBackgroundColour = g.getBackgroundColor();

        boolean notEmpty = g.pushContext( _tmpRect.x, _tmpRect.y, _tmpRect.width,
_tmpRect.height, 0, 0 );
        try {
            if( notEmpty ) {
                if( on ) {
                    g.setDrawingStyle( Graphics.DRAWSTYLE_FOCUS, true );
                    g.setBackgroundColor( _highlightColour );
                }
                g.clear();
                paint( g );  //修改了背景色后，调用HyperlinkButtonField.paint()来重新画，以显示背景色，
下划线文字
            }
        } finally {
            g.popContext();
            g.setBackgroundColor( oldBackgroundColour );//这里用临时变量来恢复Graphics原来的背景色

            g.setDrawingStyle( Graphics.DRAWSTYLE_FOCUS, oldDrawStyleFocus );
        }
    }
```

实现 keyChar ()、trackwheelClick()、invokeAction()等方法以处理用户按下按钮的事件。

```
protected boolean keyChar( char character, int status, int time )
    {
        if( character == Characters.ENTER ) {
            fieldChangeNotify( 0 );
            return true;  //return ture是告诉Manager，这个事件我已经处理了，你不需要再交由别人处理
        }
        return super.keyChar( character, status, time );
    }

    protected boolean trackwheelClick( int status, int time ) {
        keyChar(Characters.ENTER, status, time );
        return true;      //return ture是告诉Manager，这个事件我已经处理了，你不需要再交由别人处理

    }

    protected boolean invokeAction( int action )
    {
        switch( action ) {
            case ACTION_INVOKE: {
                fieldChangeNotify( 0 );
                return true; //return ture是告诉Manager，这个事件我已经处理了，你不需要再交由别人处理
            }
        }
        return super.invokeAction( action );
    }
```

HyperlinkButtonField的使用方法，注意要添加FieldChangeListener。

```
HyperlinkButtonField link = new HyperlinkButtonField ("Colour Picker", 0x0000FF, 0xFFFFFF,
0x0000FF, 0, 0);
        link.setChangeListener( new FieldChangeListener( ) {
            public void fieldChanged( Field field, int context ) {
               //do something here
            }
        } );
```

小结：

在 HyperlinkButtonField.java 代码中，我们 extends LabelField 字段， paint()方法重绘了界面；用户选择按钮的时候，我们 drawFocus()修改字体前景和背景色后再次重绘界面，同时用户 click 按钮后将触发 FieldChange 事件。

HyperlinkButtonField 字段的界面重绘还是相对简单的，是借用了 super.paint()方法也就是 LabelField.paint()进行绘图显示文字。

其他某些 Field 的定制要求就没有这么简单了，开发者需要自己调用 Graphics 对象的 drawLine、drawRect、drawText 等方法来绘制 Field 的线条、边框、文字等显示内容。

3.2 BitmapButtonField 和 MediaControlStyleField

让我们看一下稍复杂的媒体播放器界面。

用户可以在播放器四个按钮之间横向滚动，按钮选中的时候，比如第二个播放按钮，按钮会高亮度显示。

在这个 MediaControlStyleField 实现中，该 Field 又包含了 4 个 BitmapButtonField 字段，外面裹上圆的边框。 由于最底层的 Field 是不能包含其他字段的，所以 MediaControlStyleField 字段实际上扩展 Manager，里面包含 4 个 BitmapButtonField 基本字段。

	MediaControlStyleField 扩展 HorizontalFieldManager，在里面放四个紧挨着的可以 focus 的图片按钮，在 paintBackground()方法中使用 drawRoundRect()方法 为自己加一个圆角的长方形的。
	BitmapButtonField 是扩展 Field，可以根据 focus 与否切换显示两张图片，当然这个按钮还要相应用户黑莓键盘 enter 或者轨迹球按下的事件。

让我们先实现图片按钮 BitmapButtonField。

首先，把两张 Bitmap 图片  作为参数给其构造方法。

```
public class BitmapButtonField extends Field {

   public BitmapButtonField( Bitmap normalState, Bitmap focusState, long style )
    {
        super( Field.FOCUSABLE | style ); //必须使用Field.FOCUSABLE style，以让这个按钮能够被选择

        if( (normalState.getWidth() != focusState.getWidth())
            || (normalState.getHeight() != focusState.getHeight()) ){

            throw new IllegalArgumentException( "Image sizes don't match" );
        }

        _bitmaps = new Bitmap[] { normalState, focusState };  //把两张Bitmap图片保存在缓存中
    }
```

在 layout()方法中，调用 setExtent()方法设置自己的高度和宽度。这个字段的高度和宽度实际上取 Bitmap 图片  的高度和宽度。在 getPreferredWidth()和 getPreferredHeight()方法中告诉 Manager 自己希望的的高度和宽度。

```
    protected void layout( int width, int height ) {
        setExtent( _bitmaps[NORMAL].getWidth(), _bitmaps[NORMAL].getHeight() );
    }

    public int getPreferredWidth() {
        return _bitmaps[NORMAL].getWidth();
    }

    public int getPreferredHeight() {
        return _bitmaps[NORMAL].getHeight();
    }
```

当黑莓手机屏幕显示这个按钮的时候，在 paint()方法中根据其是否是 Focus 状态，算出要显示的图片的索引 index 数字，然后使用 Graphic.drawBitmap(bitmaps[index])方法把这张图片显示出来。

当黑莓手机屏幕上这个按钮处于 Focus 状态的时候，在 drawFocus()方法中设置 draw style 并调用 paint()方法重绘屏幕，显示 Focus 时候需要显示的图片。

```
    protected void paint( Graphics g ) {
        int index = g.isDrawingStyleSet( Graphics.DRAWSTYLE_FOCUS ) ? FOCUS : NORMAL;
        g.drawBitmap( 0, 0, _bitmaps[index].getWidth(), _bitmaps[index].getHeight(),
_bitmaps[index], 0, 0 );
    }

    protected void drawFocus( Graphics g, boolean on ) {
        // Paint() handles it all
        g.setDrawingStyle( Graphics.DRAWSTYLE_FOCUS, true );
        paintBackground( g );  //修改过被选中的背景色，重绘背景色
        paint( g );     //调用上面的paint()方法重绘屏幕，显示Focus时候需要显示的图片
    }
```

实现了可以根据Focus状态自动显示不同的图片的按钮组件，剩下的工作就简单了。

MediaControlStyleField扩展HorizontalFieldManager，在HorizontalFieldManager里面放四个可以focus的图片按钮，然后setPadding()设置按钮之间和HorizontalFieldManager边角的间距，setMargin()设置HorizontalFieldManager外面的间距。在paintBackground()方法中画实心的圆角的矩形、换个颜色、画圆角的边框。这样，媒体播放器就算实现了。

```
class MediaControlStyleField extends HorizontalFieldManager
{
    MediaControlStyleField()
    {
        super(Manager.FIELD_HCENTER);
        //横向放四个图片按钮
        add( new BitmapButtonField(Bitmap.getBitmapResource("prev.png"), Bitmap.getBitmapResource("prev_focus.png") ));
        add( new BitmapButtonField(Bitmap.getBitmapResource("play.png"), Bitmap.getBitmapResource("play_focus.png") ));
        add( new BitmapButtonField(Bitmap.getBitmapResource("stop.png"), Bitmap.getBitmapResource("stop_focus.png") ));
        add( new BitmapButtonField(Bitmap.getBitmapResource("next.png"), Bitmap.getBitmapResource("next_focus.png") ));

        setPadding(5,5,5,5);          //OS 5.0未公开的API
        setMargin( 10, 10, 10, 10 );  //OS 5.0未公开的API
    }

    protected void paintBackground( Graphics g )
    {
        int oldColor = g.getColor();
        try {
            g.setColor( 0x222222 );
            g.fillRoundRect( 0, 0, getWidth(), getHeight(), 20, 20 );// 画实心的圆角的矩形
            g.setColor( 0x000000 );
            g.drawRoundRect( 0, 0, getWidth(), getHeight(), 20 ,20 ); //换个颜色，画圆角的边框
        } finally {
            g.setColor( oldColor );
        }
    }

}
```

3.3 ProgressAnimationField

前面我们扩展现有的字段 LabelField，实现了能够互动的 HyperlinkButtonField，然后实现了一个完全自定义的根据字段 Focus 与否来显示不同图片的 BitmapButtonField 字段。

我们能否实现更加动感的 UI Field 呢？比如这个 ProgressAnimationField，可以使用后台 Java Thread 刷新自己的显示内容。

注：在 BlackBerry OS 6.0 中已包含 animation field。但使用 BlackBerry OS 5.0 的开发人员仍然会对这段代码感兴趣。

ProgressAnimationField 的实现其实很简单：将下面的长条的图片分成六段，每过 200 毫秒显示下一段，循环往复即可。当屏幕显示这个 Field 的时候，启动线程循环显示；当屏幕没有在显示这个 Field 的时候，停止显示显示线程。

实现计划：

ProgressAnimationField	extends Field并实现相应的方法。
Field 在手机屏幕上显示为一个长方形的区	实现 layout()：调用 setExtent()方法设置 Field

域，有高度，宽度	的宽度和高度为图片的部分宽度 ，图片的高度
在 Field 长方形的区域中显示图片	实现 paint()：使用 Graphics 对象的 drawBitmap 方法显示图片的部分区域内容
后台线程程序定时刷新 Field 显示的图片	每0.2秒invoke新的线程进行刷新屏幕显示 Application.getApplication().invokeLater(this, 200, true);

在ProgressAnimationField构造方法中计算这个Field的高度和宽度（frameWidth和frameHeight），然后在layout()方法中设置自己的宽度为图片一个Frame的宽度，高度为图片的高度。

```
    public ProgressAnimationField( Bitmap bitmap, int numFrames, long style )
    {
        super( style | Field.NON_FOCUSABLE );
        _bitmap = bitmap;
        _numFrames = numFrames;
        _frameWidth = _bitmap.getWidth() / _numFrames;  //只取图片中一个Frame的宽度
        _frameHeight = _bitmap.getHeight();

        _application = Application.getApplication();
    }

    protected void layout( int width, int height )
    {
        setExtent( _frameWidth, _frameHeight );//设定本Field的宽度为图片一个Frame的宽度，高度为图片的高度。
    }
```

在 paint()方法中，调用 Graphics.drawBitmap 显示图片的当前 frame 的图片。

```
    protected void paint( Graphics g )
    {
        g.drawBitmap( 0, 0, _frameWidth, _frameHeight, _bitmap, _frameWidth * _currentFrame, 0 );
        _currentFrame++;
        if( _currentFrame >= _numFrames ) {
            _currentFrame = 0;
        }
    }
```

在 onDisplay()方法中调用 Application.invokeLater 方法每 0.2 秒刷新一下屏幕。_application.invokeLater(this, 200, true)中的参数 200 是说 0.2 秒（200 毫秒，毫秒即 millisecond 千分之一秒）后执行 this 线程，参数 true 是说每过 0.2 秒都要再次执行 this 线程。

invokeLater()可以保证这个后台线程程序不会阻塞应用程序，造成应用程序僵死。

_application = Application.getApplication()，Field 构造的时候做了全局缓存。

在 onUndisplay()方法中停止取消 invoke 操作，不再刷新屏幕图片显示。

```
    public void run()     {
        if( _visible ) {
            invalidate();  // invalidate方法将force手机调用Field的paint()方法重绘界面。
        }
     }

  protected void onDisplay() //OS 5.0里面Field的未公开API
    {
        super.onDisplay();
        _visible = true;
        if( _timerID == -1 ) {
            _timerID = _application.invokeLater( this, 200, true ); //每0.2秒就后台invoke一次
        }
    }

    protected void onUndisplay()//OS 5.0里面Field的未公开API
    {
        super.onUndisplay();
        _visible = false;
        if( _timerID != -1 ) {
            _application.cancelInvokeLater( _timerID );//取消每0.2秒就后台invoke一次的操作
            _timerID = -1;
        }
    }
```

小结

在例子 HyperlinkButtonField 中，我们扩展 LabelField 获得更多的显示特效，滚轮和键盘操作处理；在 BitmapButtonField 和 MediaControlStyleField 中，我们看到如何使用图片 Bitmap 切换图片实现一个简单的图片按钮，然后使用基于 HorizontalManager 的 MediaControlStyleField 的 UI 组件；在例子 ProgressAnimationField 中我们使用后台线程定时刷新界面显示。

从上面三个例子中，我们看到 HyperlinkButtonField/BitmapButtonField/ProgressAnimationField 是最小用户界面元素，换句话说，这些 UI 组件界面不能包含 child 界面元素。

如果我们要实现象 MediaControlStyleField 媒体播放器界面，或者 Table 标签页，可以滚动的表格，那么就需要去实现更复杂的窗口管理器 - Manager。

参考

本节中使用的所有例子代码都可以在下面的链接中找到：How to - Implement advanced buttons, fields, and managers

http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800332/800505/800608/How_to_-_Implement_advanced_buttons,_fields,_and_managers.html?nodeid=2406256&vernum=0

代码下载 How to - Implement advanced buttons, fields, and managers

http://www.blackberry.com/knowledgecentersupport/kmsupport/developerknowledgebase/zip/How_To_Implement_Advanced_UI.zip

UI and Navigation - 开发指南 - BlackBerry Java Application

说明：RIM 网站上 UI 开发入门资料，中文

http://docs.blackberry.com/en/developers/deliverables/16521/UI_components_508102_11.jsp

创建自定义字段

http://docs.blackberry.com/en/developers/deliverables/16521/Create_a_custom_field_508117_11.jsp

Create a custom field

http://docs.blackberry.com/en/developers/deliverables/11958/Create_a_custom_field_508117_11.jsp