

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



## Contents

- [1 About this widget](#)
- [2 WidSets Scripting Language code: clock.he](#)
- [3 Widget.xml](#)
- [4 See also](#)

## About this widget

This is a simple clock widget.

## WidSets Scripting Language code: clock.he

```
class triangleClock
{
    const int CMD_BACK = 1;

    const int ONE_SECOND = 1000;
    const int ONE_MINUTE = 60*1000;

    const String VIEW_ANALOG = "analogView";
    const String VIEW_DIGITAL = "digitalView";

    MenuItem m_back = new MenuItem(CMD_BACK, "Back");
    Calendar m_time = new Calendar();
    Canvas    m_analog;
    Label     m_digi;
    Timer     m_timer;

    void updateDigital()
    {
        if (m_digi != null) {
            m_digi.setText(format("%02i:%02i",
                                m_time[ HOUR_OF_DAY ],
                                m_time[ MINUTE ],
                                m_time[ SECOND ]));
            m_digi.repaint(false);
        }
    }

    Component createElement(String viewName,
                           String elementName,
                           Style style,
                           Object context)
```

## WidClock

```
{
  if (elementName.equals("digital")) {
    m_digi = new Label(style, "");
    updateDigital();
    return m_digi;

  } else if (elementName.equals("analog")) {
    m_analog = new Canvas(style);
    return m_analog;

  } else {
    return null;
  }
}

void resetTimer(int interval)
{
  if (m_timer != null) {
    m_timer.cancel();
    m_timer = null;
  }
  if (interval > 0) {
    m_timer = schedule(1000, interval);
  }
}

void startWidget()
{
  resetTimer(ONE_MINUTE);
  setMinimizedView(createMinimizedView(VIEW_DIGITAL, null));
}

void stopWidget()
{
  resetTimer(0);
}

Shell openWidget()
{
  resetTimer(ONE_SECOND);
  m_time.setMillis(currentTimeMillis());
  Shell shell = new Shell(createMaximizedView(VIEW_ANALOG, null));
  int w, int h = getScreenSize();
  slideIn(0, w, w, h, shell);
  return null;
  //return new Shell(createMaximizedView(VIEW_ANALOG, null));
}

MenuItem getSoftKey(Shell shell, Component focused, int key)
{
  if (key == SOFTKEY_BACK) {
    return m_back;
  }
  return null;
}
```

## WidClock

```
void actionPerformed(Shell shell, Component source, int action)
{
    switch(action)
    {
        case CMD_BACK:
            {
                m_analog = null;
                int w, int h = getScreenSize();
                slideOut(0, h, w, h, shell);
                resetTimer(ONE_MINUTE);
            }
            break;
    }
}

const int SECOND_HAND = 60;
const int MINUTE_HAND = 45;
const int HOUR_HAND = 30;

void drawHand(Graphics g, int cx, int cy, int angle, int base, int height)
{
    int dx = cx+(height*cos(angle))/MATH_SCALE;
    int dy = cy+(height*sin(angle))/MATH_SCALE;

    int ax = cx+(base*cos(angle-90))/MATH_SCALE;
    int ay = cy+(base*sin(angle-90))/MATH_SCALE;

    int bx = cx+(base*cos(angle+90))/MATH_SCALE;
    int by = cy+(base*sin(angle+90))/MATH_SCALE;

    g.fillTriangle(ax, ay, bx, by, dx, dy);
}

void paint(Component c, Graphics g, Style style, int width, int height)
{
    int h = m_time[ HOUR ];
    int m = m_time[ MINUTE ];
    int s = m_time[ SECOND ];

    int cx = width/2;
    int cy = height/2;

    {
        int angle = (30*h)+(m/2) - 90;
        g.setColor(0xdc4200);
        drawHand(g, cx, cy, angle, 6, HOUR_HAND);
    }

    {
        int angle = (6*m)+(s/10) - 90;
        g.setColor(0xdc4200);
        drawHand(g, cx, cy, angle, 6, MINUTE_HAND);
    }

    {
        int angle = 6*s - 90;
        int dx = cx+(SECOND_HAND*cos(angle))/MATH_SCALE;
        int dy = cy+(SECOND_HAND*sin(angle))/MATH_SCALE;
        g.setColor(0xffffffff);
    }
}
```

## WidClock

```
        g.drawLine(cx, cy, dx, dy);
    }

    g.drawImage(style.image(0), cx, cy, HCENTER|VCENTER);
}

void timerEvent(Timer timer)
{
    m_time.setMillis(currentTimeMillis());
    if (m_analog != null) {
        m_analog.repaint(false);
        flushScreen(false);
    } else if (/*isVisible()*/isDashboardVisible()) {
        updateDigital();
        flushScreen(true);
    }
}
}
```

## Widget.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<widget spec_version="2.0">

    <info>
        <name><![CDATA[Triangle clock]]></name>
        <version>0.1</version>
        <author>Antti</author>
        <shortdescription><![CDATA[This is a clock widget made with widget specification v2.]]></shortdescription>

        <clientversion>0.95</clientversion>
        <longdescription>
            <![CDATA[This is a clock widget made with widget specification v2.]]>
        </longdescription>
        <tags>clock time date</tags>
    </info>

    <parameters>
        <parameter type="string" name="widgetname" description="The name of the widget" editable="true">
        </parameter>
    </parameters>

    <services/>

    <resources>
        
        
        
        
        
        <code src="clock.he"/>
        <stylesheet>
            bkg
            {
                background: grid9 "bkg.png" 5 5 5 5;
            }
        </stylesheet>
    </resources>
</widget>
```

## WidClock

```
digital
{
    color-1: white;
    font-1: large bold prop;
    align: hcenter vcenter;
    background: image "center.png" transparent hcenter vcenter repeat-x;
}

analog
{
    background: image "analog.png" transparent hcenter vcenter;
    color-1: black;
    image-1: "nuppi.png";
}

analogbkg
{
    background: image "centerbig.png" black hcenter vcenter repeat-x;
}
</stylesheet>
</resources>

<layout minimizedheight="10px+40sp">
    <view id="digitalView" class="bkg">
        <script id="digital" class="digital" top="50%-20sp" right="100%" bottom="50%+20sp" left="0%">
        </view>

    <view id="analogView">
        <decorate class="analogbkg" top="0%" right="100%" bottom="100%" left="0%"/>
        <script id="analog" class="analog" top="50%-69px" right="50%+69px" bottom="50%+69px" left="0%">
        </view>
    </layout>

</widget>
```

## See also

- [WidSets SDK](#)
- [WidSets Client](#)
- [WidSets Scripting Language](#)
- [Widget examples](#)
  - ◆ [WidClock](#)
  - ◆ [Memory Game](#)
  - ◆ [Filter test](#)
  - ◆ [Hello World](#)
  - ◆ [UITest](#)