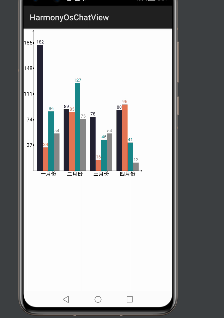
1. **HistogramComponent组件功能介绍**
   1. **功能介绍**

应用开发过程，用鸿蒙提供的 Component 自定义柱状图效果。 HistogramComponent组件可以更快速实现一个简单的柱状图功能。

HistogramComponent对外提供数据源，修改柱状图颜色，间距的接口。

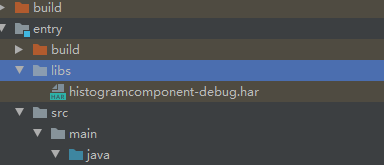
* 1. **phone模拟器上运行效果**

****

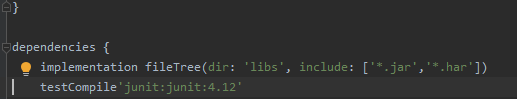
**2 . HistogramComponent使用方法**

**2.1新建工程，增加组件Har包依赖**

在应用模块中调用HAR，常用的添加依赖的方式包括如下两种。

* 方式一：依赖本地HAR,将**histogramcomponent-debug.har**复制到entry\libs目录下即可（由于build.gradle已经依赖的libs目录下的\*.har，因此不需要在做修改）。
* 

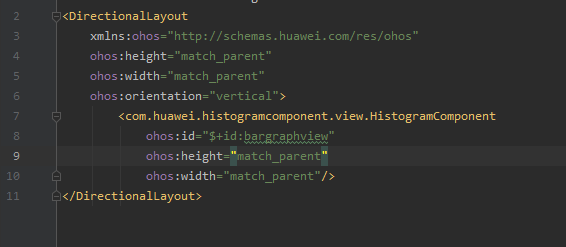
查看工程目录中build.gradle下的\*.har是存在。



以上操作无误之后就可以进行编码了！

**3 . HistogramComponent开发实现**

**3.1 . 主页面的布局文件**



**3.2. 主页面的Ability的代码**

**@Override  
public void onStart(Intent intent) {  
 super.onStart(intent);  
 super.setUIContent(ResourceTable.*Layout\_ability\_main*);  
 bargraphview = (HistogramComponent) findComponentById(ResourceTable.*Id\_bargraphview*);  
 // 设置每组柱状图之间的间距  
 bargraphview.mLineSpaceWidth = 30;  
 final int[][] data = {{182, 89, 78, 88}, {34, 85, 16, 96}, {86, 127, 45, 41},{54, 75, 54, 12}};  
 final int[] colorData = {0xFF222233, 0xFFe67656, 0xFF188888,0xFF888888,0xFF888888};  
 final String[] textData = {"一月份", "二月份", "三月份", "四月份"};**

**// 核心接口，填充数据，柱状图颜色  
 bargraphview.setBarGraphData(data, colorData, textData);**  
}

**3.3. HistogramComponent组件核心代码**

*/\*\**  
 ***\* 定义DrawTask对象的实例*  
 *\* 这里进行具体的绘画工作*  
 *\*/*  
private DrawTask drawTask = new DrawTask() {  
 @Override  
 public void onDraw(Component component, Canvas canvas) {  
 bottomHeight = heightMeasureSpec - mBottomXWidth;  
 if (barGraphDataList == null || barGraphDataList.length <= 0)  
 return;  
 //画柱状图  
 drawBarGraph(canvas);  
 //画XY轴坐标  
 drawXYLine(canvas);  
 //给XY轴坐标写字  
 drawXYText(canvas);  
 }  
};**

**//画柱状图  
private void drawBarGraph(Canvas canvas) {  
 if (barGraphDataList != null && barGraphDataList.length > 0) {  
 for (int i = 0; i < barGraphDataList[0].length; i++) {  
 float startX = mLineSpaceWidth \* (i + 1) + mLineWidth \* barGraphDataList.length \* i + mLeftYWidth + (10) + mLineWidth / 2;  
 int index = 0;  
 while (index < barGraphDataList.length) {  
 if (barGraphColorList != null && barGraphColorList.length > index) {**  
 **mBarGraphPaint.setColor(new Color(barGraphColorList[index]));  
 mBarGraphTextPaint.setColor(new Color(barGraphColorList[index]));  
 } else {  
 mBarGraphPaint.setColor(new Color(barGraphBgColor));  
 mBarGraphTextPaint.setColor(new Color(barGraphBgColor));  
 }  
  
 float stopY = bottomHeight \* 0.9f / maxHeight \* barGraphDataList[index][i];  
  
 canvas.drawLine(new Point(startX, bottomHeight), new Point(startX, bottomHeight - stopY), mBarGraphPaint);  
  
 String text = String.*valueOf*(barGraphDataList[index][i]);  
 float textWidth = mBarGraphTextPaint.measureText(text);  
 canvas.drawText(mBarGraphTextPaint,text, startX - textWidth / 2, bottomHeight - stopY - 10);  
 startX += mLineWidth;  
 index++;  
 }  
 }  
 }  
}**

**//画X轴和Y轴的竖线+箭头  
private void drawXYLine(Canvas canvas) {  
 */\*\**  
 *\* 让Y轴文字与最左有dip2px(10)的边距*  
 *\* \*/*  
//Y轴竖线  
 canvas.drawLine(new Point((10) + mLeftYWidth, bottomHeight), new Point((10) + mLeftYWidth, 10), mXYLinePaint);  
 //X轴竖线  
 canvas.drawLine(new Point((10) + mLeftYWidth, bottomHeight), new Point(widthMeasureSpec - 10, bottomHeight), mXYLinePaint);  
 //画个箭头？？Y轴  
 canvas.drawLine(new Point((10) + mLeftYWidth, 10), new Point((6) + mLeftYWidth, 20), mXYLinePaint);  
 canvas.drawLine(new Point((10) + mLeftYWidth, 10), new Point((14) + mLeftYWidth, 20), mXYLinePaint);  
 //X轴箭头  
 canvas.drawLine(new Point(widthMeasureSpec - 10, bottomHeight), new Point(widthMeasureSpec - 20, bottomHeight - (4)), mXYLinePaint);  
 canvas.drawLine(new Point(widthMeasureSpec - 10, bottomHeight), new Point(widthMeasureSpec - 20, bottomHeight + (4)), mXYLinePaint);  
}  
//给Y轴和X轴写相应的文字  
private void drawXYText(Canvas canvas) {  
 if (isShowYText) {  
 //Y轴写字  
 for (int i = 1; i <= 5; i++) {  
 float startY = bottomHeight - bottomHeight \* 0.9f / maxHeight \* maxHeight / 5 \* i;  
 canvas.drawLine(new Point((10) + mLeftYWidth, startY), new Point((15) + mLeftYWidth, startY), mYTextPaint);  
 float width = mYTextPaint.measureText(maxHeight / 5 \* i + "");  
  
 float dy = 12.0f;  
 canvas.drawText(mYTextPaint,maxHeight / 5 \* i + "", (int) ((10) + mLeftYWidth - width - (5)), startY + dy);  
 }  
 }  
 if (!isShowXText) {  
 return;  
 }  
 //X轴写字  
 if (barGraphTextList != null && barGraphTextList.length > 0) {  
 for (int i = 0; i < barGraphTextList.length; i++) {  
 float startX = mLineSpaceWidth \* (i + 1) + mLineWidth \* barGraphDataList.length \* i + mLeftYWidth + (10);  
 //中间有一个间隔  
 startX = startX + (mLineWidth \* barGraphDataList.length) \* 1.0f / 2;  
 float textWidth = mXTextPaint.measureText(barGraphTextList[i]);  
 canvas.drawText(mXTextPaint,barGraphTextList[i], startX - textWidth / 2, heightMeasureSpec - (5));  
 }  
 }  
}**

**// 对外提供的核心接口**

**public void setBarGraphData(int[][] barGraphDataList, int[] barGraphColorList, String[] barGraphTextList) {  
 this.barGraphDataList = barGraphDataList;  
 this.barGraphColorList = barGraphColorList;  
 this.barGraphTextList = barGraphTextList;  
  
 for(int i = 0; i < barGraphDataList.length; ++i) {  
 for(int j = 0; j < barGraphDataList[i].length; ++j) {  
 if (this.maxHeight < barGraphDataList[i][j]) {  
 this.maxHeight = barGraphDataList[i][j];  
 }  
 }  
 }  
  
 while(this.maxHeight % 5 != 0) {  
 ++this.maxHeight;  
 }  
  
 if (barGraphTextList != null && barGraphTextList.length > 0) {  
 this.isShowXText = true;  
 }  
  
 if (this.isShowYText) {  
 this.mLeftYWidth = this.mYTextPaint.measureText(String.valueOf(this.maxHeight));  
 }  
  
 this.mBottomXWidth = 10.0F;  
 if (this.isShowXText) {  
 FontMetrics fontMetrics = this.mXTextPaint.getFontMetrics();  
 this.mBottomXWidth += ((fontMetrics.bottom - fontMetrics.top) / 2.0F - fontMetrics.bottom) \* 2.0F;  
 }  
  
 this.measureWidth(this.heightMeasureSpec);  
 this.invalidate();  
}**