# RecyclerView ItemDecoration 完全解析

星火燎原16 关注

0.764 2018.12.23 16:24:55 字数 1,398 阅读 3,863

我们都知道，使用 RecyclerView 时，我们不能像 ListView 那样通过 setDivider() 的方式来设置分割线，好在 Android 为我们提供了定制性更强的 ItemDecoration 来为 RecyclerView 设置分割线。

## 什么是 ItemDecoration？

顾名思义 ItemDecoration 就是 Item 的装饰，我们可以在 Item 的上下左右添加自定义的装饰，比如 横线，图案。同时系统已经为我们提供了一个 DividerItemDecoration, 如果这个 DividerItemDecoration 不满足我们的需求，我们就可以通过自定义 ItemDecoration 来实现了。

下面我们看下系统的 DividerItemDecoration：

## 引入 DividerItemDecoration(系统提供)

DividerItemDecoration 的使用非常简单，只需添加下面代码即可：

```
1 DividerItemDecoration decoration = new DividerItemDecoration(this,DividerItemDecoratio
2 recyclerView.addItemDecoration(decoration);
```

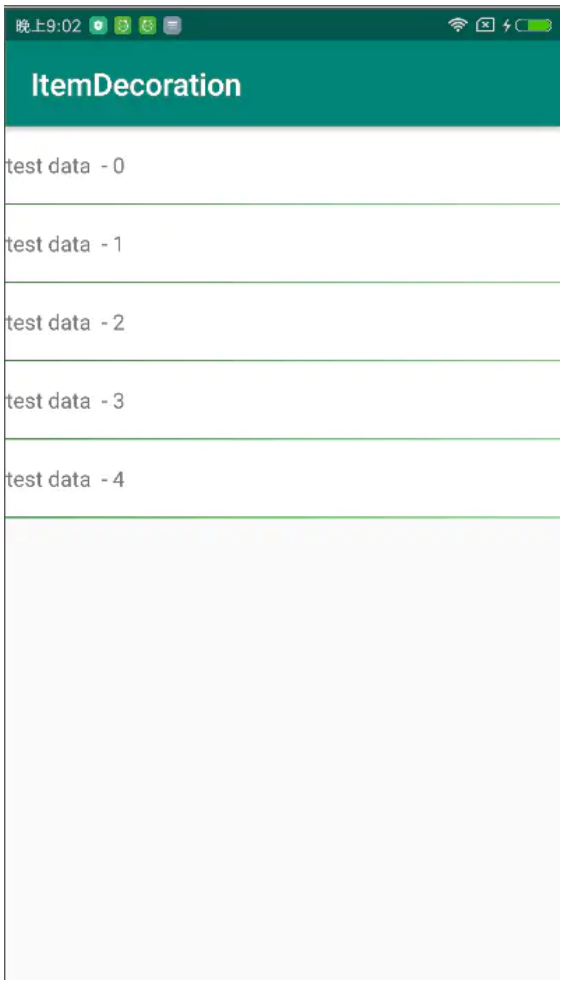
效果：


DividerItemDecoration.png

**如果想要修改 DividerItemDecoration 的颜色和高度，可以调用它的 setDrawable(drawable) 设置一个 Drawable 对象**

```
// MainActivity.java
DividerItemDecoration decoration = new DividerItemDecoration(this, DividerItemDecorati
Drawable dividerDrawable = getResources().getDrawable(R.drawable.drawable_divider);
decoration.setDrawable(dividerDrawable);
recyclerView.addItemDecoration(decoration);

// res/drawable/drawable_divider.xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    android:shape="rectangle">
    <!-- 渐变色 -->
    <gradient
        android:angle="135"
        android:centerColor="#4CAF50"
        android:endColor="#2E7D32"
        android:startColor="#81C784"
        android:type="linear" />

    <size android:height="1dp" />
</shape>
```

效果:




dividerItemDecoration-drawable.png

## 自定义 ItemDecoration

自定义 ItemDecoration ， 主要需要重写以下三个方法:

```
/**
 * 自定义 ItemDecoration
 */
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
    }

    @Override
    public void onDraw(@NonNull Canvas c, @NonNull RecyclerView parent, @NonNull Recyc
        super.onDraw(c, parent, state);
    }

    @Override
    public void onDrawOver(@NonNull Canvas c, @NonNull RecyclerView parent, @NonNull R
        super.onDrawOver(c, parent, state);
    }
```

## 1. getItemOffsets()

getItemOffsets() 主要作用是在 item 的四周留下边距，效果和 margin 类似，item 的四周留下边距后，我们就可以通过 onDraw() 在这个边距上绘制了。

```
public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Recycle
```

(1) 参数 Rect outRect : 表示 item 的上下左右所留下的边距。其中 outRect 的 left, top,right,bottom 即为 item 四周留下的边距的距离，默认都为 0 ； 示意图如下：

(2) 参数 View view : 指当前 item 的 View 对象；

(3) 参数 RecyclerView parent : 指 RecyclerView 本身；

(4) RecyclerView.State state : 指 RecyclerView 当前的状态；

### 1.1 getItemOffsets() 应用例子：

既然 getItemOffsets(Rect outRect) 方法可以设置 item 四周的边距大小，那就可以设置 recyclerview 背景色和 item 四周的边距，使得 item 四周的边距透出 recyclerview 背景色来达到分割线的目的。

当然 item 的背景色需要和 recyclerview 的背景色不一致才有效果；

首先将 recyclerview 的背景色设置为 colorAccent 红色, 将 item 的背景色设置为 白色：

```
<!--- activity_main.xml ----->
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/rv_recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorAccent" />

</android.support.constraint.ConstraintLayout>
```

```
<!--- item_recycler.xml ----->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
```

```
    android:background="#fff"
    android:gravity="center_vertical"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:gravity="center_vertical" />

</LinearLayout>
```

然后继承 ItemDecoration 类，重写 getOffsets() 方法，将 outRect 的**上边距 top** 设置为 10px;

```
/**
 * 自定义 ItemDecoration
 */
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        outRect.top = 10;   // 10px
    }
}
```

最后将这个 LinearItemDecoration 添加到 RecyclerView 中：

```
LinearItemDecoration decoration = new LinearItemDecoration();
recyclerView.addItemDecoration(decoration);
```

效果如下(下方的红色是 因为RecyclerView 高度为 match_parent ,但 item 数据只有 5 条)：


linearItemDecoration.png

从效果图可以看出：每一个 item 的 顶部都有一个红色的背景线，**包括第一个 item 顶部也有(怎么解决呢？见 2.1 节详解)**，

同理，我们可以设置为 底部 10px,左侧 20px ,右侧 40px；

```
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        outRect.bottom = 10;
        outRect.left = 20;
        outRect.right = 40;
    }
}
```

效果：


linearItemDecoration-bottom-left-right.png

可以看到：每个 item 的左测，底部，右侧都有了间距，露出了 RecyclerView 的背景色了。

## 2. onDraw()

```
@Override
public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull Recy
   super.onDraw(canvas, parent, state);
}
```

onDraw() 函数中的 parent , state 参数和 getItemOffsets() 方法中的参数含义是一样的，

canvas 参数是 getItemOffsets() 函数所留下的左右上下的空白区域对应的 Canvas 画布对象。

我们可以在这个区域中利用 Paint 画笔绘制任何图形。

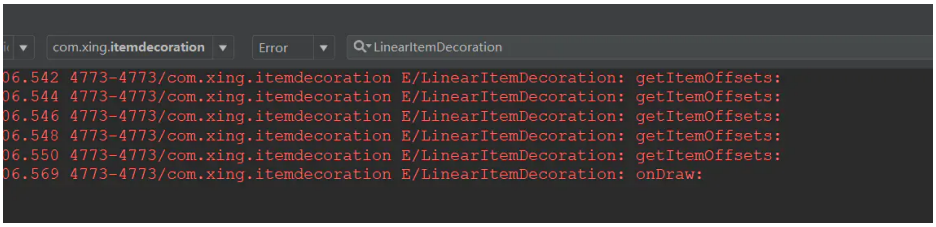
比如在 item 左侧绘制一个 空心圆。

```
/**
 * 自定义 ItemDecoration
 */
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    private static final String TAG = "LinearItemDecoration";
    private Paint paint;

    public LinearItemDecoration() {
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5);
    }

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Recyc
        super.getItemOffsets(outRect, view, parent, state);
        Log.e(TAG, "getItemOffsets: " );
        outRect.bottom = 10;
        outRect.left = 100;
        outRect.right = 40;
    }

    @Override
    public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull Re
        super.onDraw(canvas, parent, state);
        Log.e(TAG, "onDraw: ");
        for (int i = 0; i < parent.getChildCount(); i++) {
            View childView = parent.getChildAt(i);
            canvas.drawCircle(50, childView.getTop() + childView.getHeight() / 2, 20, pa
        }

    }
}
```

效果：

onDraw-circle.png

我们在 getItemOffsets() 和 onDraw() 方法中都添加了日志，日志打印如下：




log.png

**从日志中可以看出： getItemOffsets() 方法执行了 5 遍，和 数据源个数是一样的，但 onDraw() 方法只执行了一遍，由此我们知道， getItemOffsets() 是针对每个 item 都会执行一次，也就是说 每个 item 的 outRect 可以设置为不同值，但是 onDraw(),onDrawOver() 是针对 ItemDecoration 的，不是针对 item 的，只执行一次。所以我们在 onDraw() ,onDrawOver() 中绘制的时候，需要遍历每个 item 进行绘制。**

**优雅获取 outRect 中的值**

在上面例子中，我们在 onDraw 中获取 outRect 中的值都是写成计算好的固定值，显然这种硬编码的方式不利于扩展，其实，我们可以通过 LayoutManager 来获取 getItemOffsets() 中设置的 outRect 的值。

```
@Override
public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull Recy
    super.onDraw(canvas, parent, state);
    Log.e(TAG, "onDraw: ");
    RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
    for (int i = 0; i < parent.getChildCount(); i++) {
        View childView = parent.getChildAt(i);
         int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
         int topDecorationHeight = layoutManager.getTopDecorationHeight(childView);
         int rightDecorationWidth = layoutManager.getRightDecorationWidth(childView);
         int bottomDecorationHeight = layoutManager.getBottomDecorationHeight(childVie
    }
}
```

上面硬编码可以改成：

```
@Override
public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull Recy
    super.onDraw(canvas, parent, state);
    Log.e(TAG, "onDraw: ");
    RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
    for (int i = 0; i < parent.getChildCount(); i++) {
        View childView = parent.getChildAt(i);
        int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
        int left = leftDecorationWidth / 2;
        canvas.drawCircle(left, childView.getTop() + childView.getHeight() / 2, 20, pa
    }
}
```

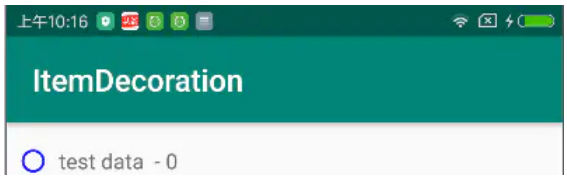
### 2.1 扩展1 -- 减少背景设置，避免过度绘制

为了减少过度绘制，我们将 activity_main 中 RecyclerView , item_recycler 中的背景全部去掉，不设置任何背景，然后在 LinearItemDecoration 中进行纯绘制分割线，代码如下：

```
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    private static final String TAG = "LinearItemDecoration";
    private Paint paint;
    private Paint dividerPaint;

```

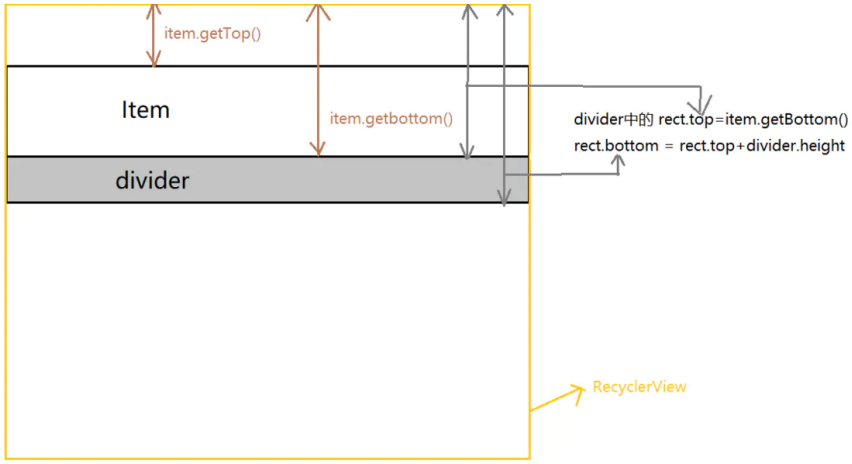
```
    public LinearItemDecoration() {
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5);

        dividerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        dividerPaint.setColor(Color.parseColor("#e6e6e6"));
        dividerPaint.setStyle(Paint.Style.FILL);
    }

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        Log.e("ItemOffsets", "getItemOffsets: ");
        outRect.bottom = 5;
        outRect.left = 100;
    }

    @Override
    public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull
        super.onDraw(canvas, parent, state);
        Log.e(TAG, "onDraw: ");
        RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
        for (int i = 0; i < parent.getChildCount(); i++) {
            View childView = parent.getChildAt(i);
            int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
            int topDecorationHeight = layoutManager.getTopDecorationHeight(childView);
            int rightDecorationWidth = layoutManager.getRightDecorationWidth(childView
            int bottomDecorationHeight = layoutManager.getBottomDecorationHeight(child
            int left = leftDecorationWidth / 2;
            canvas.drawCircle(left, childView.getTop() + childView.getHeight() / 2, 20
            // getItemOffsets()中的设置的是 bottom = 5px;所以在 drawRect 时，top 为 childV
            canvas.drawRect(new Rect(
                    leftDecorationWidth,
                    childView.getBottom(),
                    childView.getWidth() + leftDecorationWidth,
                    childView.getBottom() + bottomDecorationHeight
            ), dividerPaint);
        }
    }

    @Override
    public void onDrawOver(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonN
        super.onDrawOver(canvas, parent, state);
    }
}
```

效果：




divider-draw.png

**注意：** 上面 getItemOffsets() 中设置的是 bottom = 5px; 所以在 onDraw() 方法的 drawRect 时，top值为 childView.getBottom, bottom值为 top+bottomDecorationHeight。




divider-top-bottom.png

同理：getItemOffsets() 中设置是 top = 5px， 那么在 onDraw() 方法 drawRect 时，bottom 值为 childView.getTop() , top 值为 bottom - topDecorationHeight

### 2.2 扩展2 -- 实现竖直进度分割线

在 getItemOffsets() 方法中左侧留下空白区域，然后在 onDraw() 方法中绘制 圆和 竖线。代码如下：

```
<!--- item_recycler.xml ---->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:padding="10dp">

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#332"
        android:textSize="16sp"
        tools:text="我是 title" />

    <TextView
        android:id="@+id/tv_time"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:gravity="center_vertical"
        android:textColor="#666"
        android:textSize="14sp"
        tools:text="woshi" />

</LinearLayout>

<!--- activity_main.xml --->
<?xml version="1.0" encoding="utf-8"?>
```

```xml
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/rv_recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#dfdfdf" />

</android.support.constraint.ConstraintLayout>
```

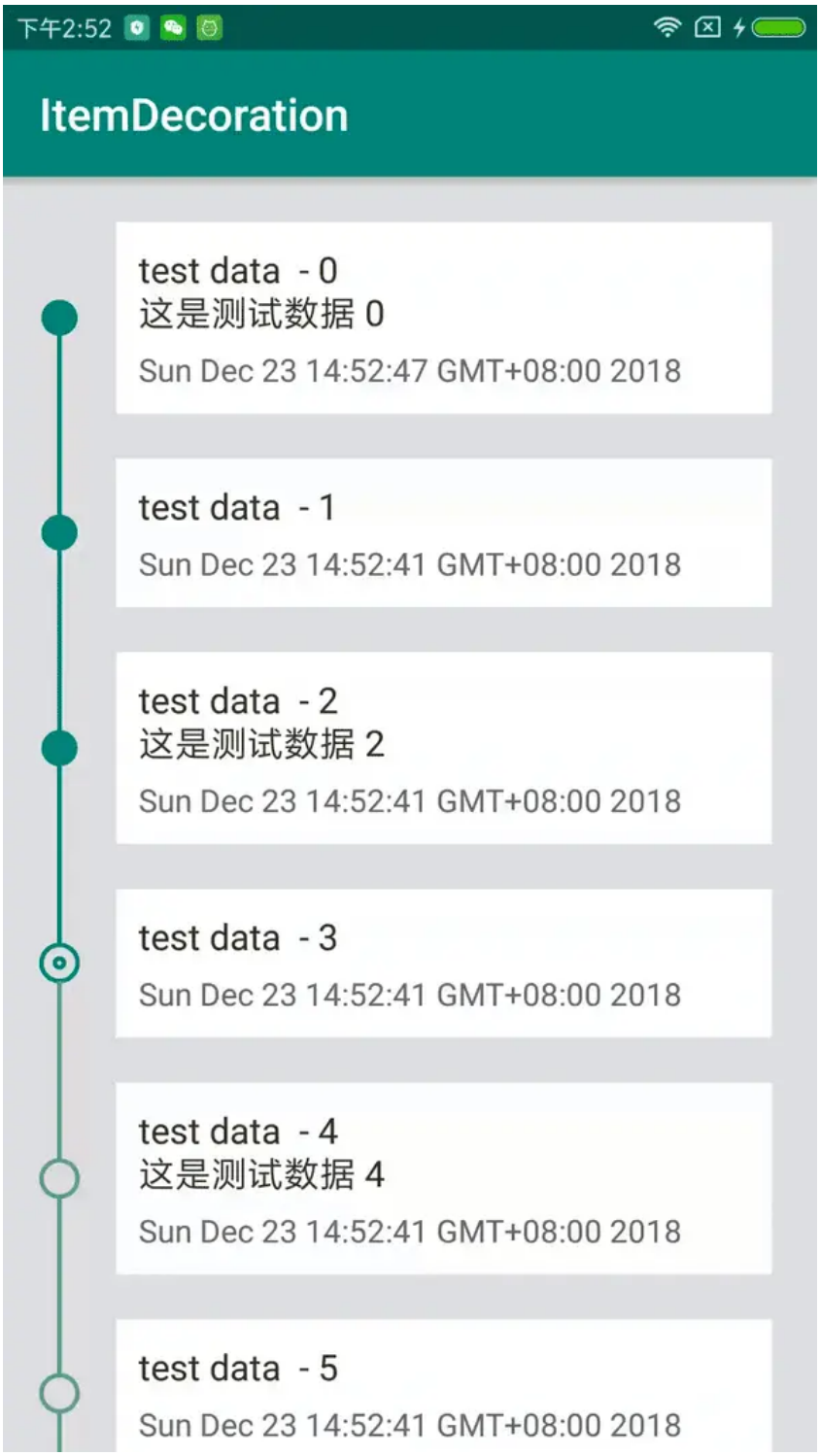
```java
// ProgressItemDecoration.java
public class ProgressItemDecoration extends RecyclerView.ItemDecoration {

    private Context context;
    private Paint circlePaint;
    private Paint linePaint;
    private int radius;
    private int curPosition = 0;   // 当前进行中的位置

    public ProgressItemDecoration(Context context) {
        this.context = context;
        circlePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        circlePaint.setColor(context.getResources().getColor(R.color.colorPrimary));
        circlePaint.setStyle(Paint.Style.FILL);
        radius = dp2Px(8);
        circlePaint.setStrokeWidth(dp2Px(2));


        linePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        linePaint.setColor(context.getResources().getColor(R.color.colorPrimary));
        linePaint.setStrokeWidth(dp2Px(2));
    }

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        outRect.top = dp2Px(20);
        outRect.left = dp2Px(50);
        outRect.right = dp2Px(20);

    }

    @Override
    public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull
        super.onDraw(canvas, parent, state);
        int childCount = parent.getChildCount();
        RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
        for (int i = 0; i < childCount; i++) {
            View childView = parent.getChildAt(i);
            int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
            int topDecorationHeight = layoutManager.getTopDecorationHeight(childView);
            // 获取当前 item 是 recyclerview 的第几个 childview
            int childLayoutPosition = parent.getChildLayoutPosition(childView);
            float startX = leftDecorationWidth / 2;
            float stopX = startX;
            // 圆顶部部分竖线，起点 Y
            float topStartY = childView.getTop() - topDecorationHeight;
            // 圆顶部部分竖线，终点 Y
            float topStopY = childView.getTop() + childView.getHeight() / 2 - radius;

            // 圆底部部分竖线，起点 Y
            float bottomStartY = childView.getTop() + childView.getHeight() / 2 + radi
            // 圆底部部分竖线，终点 Y
            float bottomStopY = childView.getBottom();

            // 位置超过 curPosition 时，竖线颜色设置为浅色
            if (childLayoutPosition > curPosition) {
                linePaint.setColor(context.getResources().getColor(R.color.colorPrimar
                circlePaint.setColor(context.getResources().getColor(R.color.colorPrim
                circlePaint.setStyle(Paint.Style.STROKE);
            } else {
                linePaint.setColor(context.getResources().getColor(R.color.colorPrimar
                circlePaint.setColor(context.getResources().getColor(R.color.colorPrim
                circlePaint.setStyle(Paint.Style.FILL);
            }

            // 绘制圆
```

```
            if (childLayoutPosition == curPosition) {
                circlePaint.setStyle(Paint.Style.STROKE);
                canvas.drawCircle(leftDecorationWidth / 2, childView.getTop() + childV
            }
            canvas.drawCircle(leftDecorationWidth / 2, childView.getTop() + childView.

            // 绘制竖线 , 第 0 位置上只需绘制 下半部分
            if (childLayoutPosition == 0) {
                // 当前 item position = curPosition 时，绘制下半部分竖线时，颜色设置为浅色
                if (childLayoutPosition == curPosition) {
                    linePaint.setColor(context.getResources().getColor(R.color.colorPr
                }
                canvas.drawLine(startX, bottomStartY, startX, bottomStopY, linePaint);
                // 最后位置上，只需绘制上半部分
            } else if (childLayoutPosition == parent.getAdapter().getItemCount() - 1)
                canvas.drawLine(startX, topStartY, startX, topStopY, linePaint);
            } else {
                // 都要绘制
                canvas.drawLine(startX, topStartY, startX, topStopY, linePaint);
                // 当前 item position = curPosition 时，绘制下半部分竖线时，颜色设置为浅色
                if (childLayoutPosition == curPosition) {
                    linePaint.setColor(context.getResources().getColor(R.color.colorPr
                }
                canvas.drawLine(startX, bottomStartY, startX, bottomStopY, linePaint);
            }
        }

    }

    /**
     * 设置进行中的位置
     *
     * @param recyclerView
     * @param position
     */
    public void setDoingPosition(RecyclerView recyclerView, int position) {
        if (recyclerView == null) {
            throw new IllegalArgumentException("RecyclerView can't be null");
        }
        if (recyclerView.getAdapter() == null) {
            throw new IllegalArgumentException("RecyclerView Adapter can't be null");
        }
        if (position < 0) {
            throw new IllegalArgumentException("position can't be less than 0");
        }
        recyclerView.getLayoutManager().getItemCount();
        if (position > recyclerView.getAdapter().getItemCount() - 1) {
            throw new IllegalArgumentException("position can't be greater than item co
        }
        this.curPosition = position;
    }

    private int dp2Px(int value) {
        return (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, value, con
    }
}
```

```
// MainActivity.java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        RecyclerView recyclerView = findViewById(R.id.rv_recycler);
        recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManage
        List<String> list = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            String data = "test data  - " + i;
            if (i % 2 == 0) {
                data = data + "\n" + "这是测试数据 " + i;
            }
            list.add(data);
        }
        BaseRecyclerAdapter adapter = new BaseRecyclerAdapter<String>(list, R.layout.i

            @Override
            protected void bind(BaseRecyclerAdapter<String> adapter, BaseViewHolder ho
                holder.setText(R.id.tv_title, data);
                holder.setText(R.id.tv_time, new Date().toString());
            }
```

```
        };
        recyclerView.setAdapter(adapter);
        ProgressItemDecoration decoration = new ProgressItemDecoration(this);
        decoration.setDoingPosition(recyclerView, 0);
        recyclerView.addItemDecoration(decoration);
    }
}
```

效果：



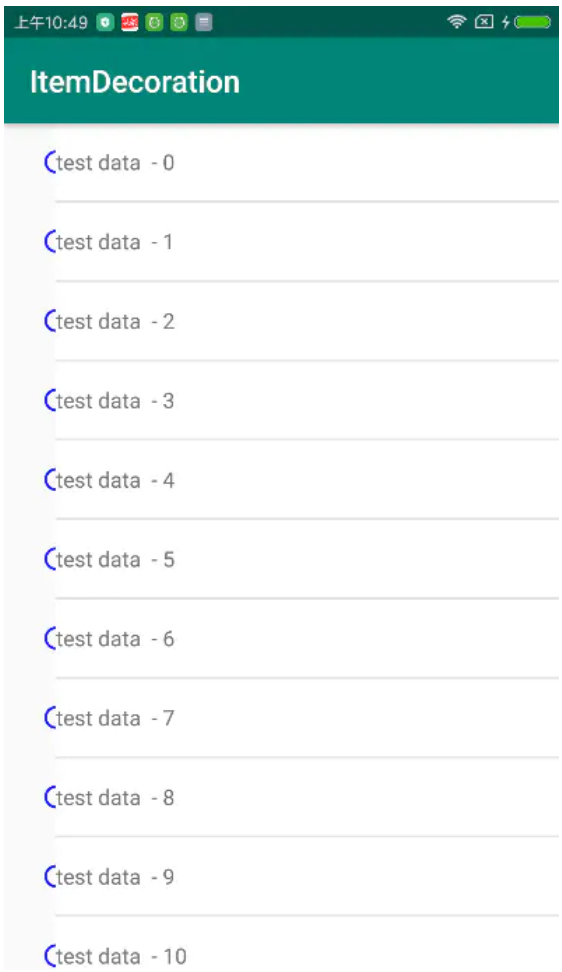

vertical-progress.gif

### 3. onDrawOver()

当我们将上面例子中绘制左侧的空心圆的位置改为： 圆心 x 坐标为 leftDecorationWidth，同时将 item 背景色设置为 白色：

```
@Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        Log.e("ItemOffsets", "getItemOffsets: ");
        outRect.bottom = 5;
        outRect.left = 100;
    }

    @Override
    public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull
        super.onDraw(canvas, parent, state);
        Log.e(TAG, "onDraw: ");
        RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
        for (int i = 0; i < parent.getChildCount(); i++) {
            View childView = parent.getChildAt(i);
            int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
            int bottomDecorationHeight = layoutManager.getBottomDecorationHeight(child
            int left = leftDecorationWidth / 2;
//            canvas.drawCircle(left, childView.getTop() + childView.getHeight() / 2,
            canvas.drawCircle(leftDecorationWidth, childView.getTop() + childView.getH
            // getItemOffsets()中的设置的是 bottom = 5px;所以在 drawRect 时，top 为 childV
            canvas.drawRect(new Rect(
                    leftDecorationWidth,
                    childView.getBottom(),
                    childView.getWidth() + leftDecorationWidth,
                    childView.getBottom() + bottomDecorationHeight
            ), dividerPaint);
        }
    }
```

效果：




draw-leftDecoration-width.png

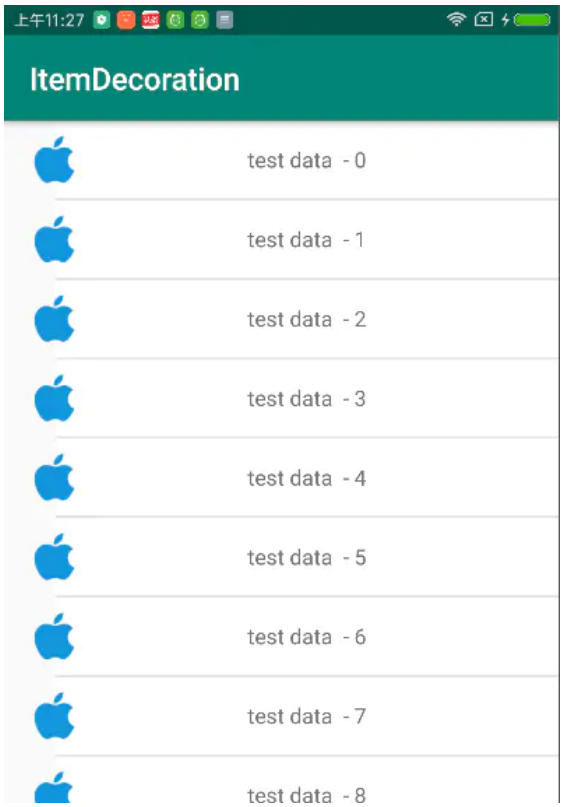
我们发现：绘制的空心圆被 item 遮挡了右边部分，变为不可见了，这是因为在这个绘制的流程中，先调用 ItemDecoration 的 onDraw() 方法，然后再调用 item 的 onDraw() 方法，最后再调用 ItemDecoration 的 onDrawOver() 方法。

因此，当我们想要在 item 的绘制显示一些内容时，将绘制的逻辑写在 onDrawOver() 方法即可。

下面我们实现在 item 与 左侧 decoration 交汇处绘制一个 apple icon ,并将 item 中的内容文本设置为居中显示，代码如下：

```
public class LinearItemDecoration extends RecyclerView.ItemDecoration {

    private static final String TAG = "LinearItemDecoration";
    private Paint paint;
    private Paint dividerPaint;
    private Bitmap iconBitmap;

    public LinearItemDecoration(Context context) {
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5);

        dividerPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        dividerPaint.setColor(Color.parseColor("#e6e6e6"));
        dividerPaint.setStyle(Paint.Style.FILL);

        iconBitmap = BitmapFactory.decodeResource(context.getResources(), R.drawable.i
    }

    @Override
    public void getItemOffsets(@NonNull Rect outRect, @NonNull View view, @NonNull Rec
        super.getItemOffsets(outRect, view, parent, state);
        Log.e("ItemOffsets", "getItemOffsets: ");
        outRect.bottom = 5;
        outRect.left = 100;
    }

    @Override
    public void onDraw(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonNull
        super.onDraw(canvas, parent, state);
        Log.e(TAG, "onDraw: ");
        RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
        for (int i = 0; i < parent.getChildCount(); i++) {
            View childView = parent.getChildAt(i);
            int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
            int bottomDecorationHeight = layoutManager.getBottomDecorationHeight(child
            int left = leftDecorationWidth / 2;
//            canvas.drawCircle(left, childView.getTop() + childView.getHeight() / 2,
//            canvas.drawCircle(leftDecorationWidth, childView.getTop() + childView.ge

            // getItemOffsets()中的设置的是 bottom = 5px;所以在 drawRect 时，top 为 childV
            canvas.drawRect(new Rect(
                    leftDecorationWidth,
                    childView.getBottom(),
                    childView.getWidth() + leftDecorationWidth,
                    childView.getBottom() + bottomDecorationHeight
            ), dividerPaint);
        }
    }

    @Override
    public void onDrawOver(@NonNull Canvas canvas, @NonNull RecyclerView parent, @NonN
        super.onDrawOver(canvas, parent, state);
        RecyclerView.LayoutManager layoutManager = parent.getLayoutManager();
        for (int i = 0; i < parent.getChildCount(); i++) {
            View childView = parent.getChildAt(i);
            int leftDecorationWidth = layoutManager.getLeftDecorationWidth(childView);
            canvas.drawBitmap(iconBitmap, leftDecorationWidth - iconBitmap.getWidth()
                    childView.getTop() + childView.getHeight() / 2 - iconBitmap.getHei
        }
    }
}
```

效果：




drawover-ios.png