

引语

TextView大家应该都不陌生，文本展示控件嘛！就用TextView显示普普通通的文本，OK，很简单，Android入门的都会，没入门的在门缝外看两眼也都会，哈哈，开玩笑。那要是设计在开发需求中要求类似微信聊天表情一样在TextView中插入表情图片呢？有的小伙伴就会说啦，“TextView添加图片我会啊，不就是 `drawableLeft`，`drawableRight` 嘛！”嗯~也行，算是一种方法，可这有一个限制，首先，图片只能在TextView的两端，其次，两端都只能设置一张图片。要是图片要在文本中间呢？无能为力了吧，要是你会使用SpannableString，这个问题也就不难解决了，简直是Just So So。

所以，不论你是否正在经受以上问题的困扰，亦或是还没有经历到，请驻足仔细耐心的看完这篇简短的文章。不仅能够轻松实现以上设计需求，更能收获其他各种炫酷的效果，也许就能帮助你解决现在你所困扰的问题。

首先我们先来了解SpannableString

SpannableString其实和String一样，都是一种字符串类型，同样TextView也可以直接设置SpannableString作为显示文本，不同的是SpannableString可以通过使用其方法setSpan方法实现字符串各种形式风格的显示,重要的是可以指定设置的区间，也就是为字符串指定下标区间内的子字符串设置格式。

setSpan(Object what, int start, int end, int flags)方法需要用户输入四个参数，`what` 表示设置的格式是什么，可以是前景色、背景色也可以是可点击的文本等等，`start` 表示需要设置格式的子字符串的起始下标，同理 `end` 表示终了下标，`flags` 属性就有意思了，共有四种属性：

- `Spanned.SPAN_INCLUSIVE_EXCLUSIVE` 从起始下标到终了下标，包括起始下标
- `Spanned.SPAN_INCLUSIVE_INCLUSIVE` 从起始下标到终了下标，同时包括起始下标和终了下标
- `Spanned.SPAN_EXCLUSIVE_EXCLUSIVE` 从起始下标到终了下标，但都不包括起始下标和终了下标
- `Spanned.SPAN_EXCLUSIVE_INCLUSIVE` 从起始下标到终了下标，包括终了下标

下面我们——解读几种Span常用的格式:

- **ForegroundColorSpan**


设置文字的前景色为淡蓝色

`ForegroundColorSpan`，为文本设置前景色，效果和TextView的setTextColor()类似，实现方法如下：

```
SpannableString spannableString = new SpannableString("设置文字的前景色为淡蓝色");
ForegroundColorSpan colorSpan = new
ForegroundColorSpan(Color.parseColor("#0099EE"));
spannableString.setSpan(colorSpan, 9, spannableString.length(),
Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

设置的区间是9到字符串的最后，也就是图中“淡蓝色”三字。

- **BackgroundColorSpan**



`BackgroundColorSpan`，为文本设置背景色，效果和TextView的setBackground()类，实现方法如下：

```
SpannableString spannableString = new SpannableString("设置文字的背景色为淡绿色");
BackgroundColorSpan colorSpan = new
BackgroundColorSpan(Color.parseColor("#AC00FF30"));
spannableString.setSpan(colorSpan, 9, spannableString.length(),
Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

- **RelativeSizeSpan**

万丈高楼平地起

`RelativeSizeSpan`，设置文字相对大小，在TextView原有的文字大小的基础上，相对设置文字大小，实现方法如下：

```
SpannableString spannableString = new SpannableString("万丈高楼平地起");

RelativeSizeSpan sizeSpan01 = new RelativeSizeSpan(1.2f);
RelativeSizeSpan sizeSpan02 = new RelativeSizeSpan(1.4f);
RelativeSizeSpan sizeSpan03 = new RelativeSizeSpan(1.6f);
RelativeSizeSpan sizeSpan04 = new RelativeSizeSpan(1.8f);
RelativeSizeSpan sizeSpan05 = new RelativeSizeSpan(1.6f);
RelativeSizeSpan sizeSpan06 = new RelativeSizeSpan(1.4f);
RelativeSizeSpan sizeSpan07 = new RelativeSizeSpan(1.2f);

spannableString.setSpan(sizeSpan01, 0, 1, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan02, 1, 2, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan03, 2, 3, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan04, 3, 4, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan05, 4, 5, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan06, 5, 6, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(sizeSpan07, 6, 7, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

- **StrikethroughSpan**

为文字设置删除线

`StrikethroughSpan`，为文本设置中划线，也就是常说的删除线，实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置删除线");
StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spannableString.setSpan(strikethroughSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

看到这有没有小激动，分分钟实现天猫打折优惠效果，有木有？

- **UnderlineSpan**

为文字设置下划线

`UnderlineSpan`，为文本设置下划线，具体实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置下划线");
UnderlineSpan underlineSpan = new UnderlineSpan();
spannableString.setSpan(underlineSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

- **SuperscriptSpan**

为文字设置上标

`SuperscriptSpan`，设置上标，具体实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置上标");
SuperscriptSpan superscriptSpan = new SuperscriptSpan();
spannableString.setSpan(superscriptSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

从效果图可以看出，被设置为上标的文字大小和下面的文本文字大小一样，只要我们稍加修饰，结合 `RelativeSizeSpan` 设置小字体文本作为上标，分分钟实现指数公式有木有，再也不用 $2^2+3^2=13$ 这样缺乏审美的数学公式了，是不是超实用？

- **SubscriptSpan**

为文字设置下标

`SubscriptSpan`，设置下标，功能与设置上标类似，不做过多描述，具体实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置下标");
SubscriptSpan subscriptSpan = new SubscriptSpan();
spannableString.setSpan(subscriptSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

- **StyleSpan**

为文字设置粗体、斜体风格

`StyleSpan`，为文字设置风格（粗体、斜体），和TextView属性textStyle类似，实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置粗体、斜体风格");
StyleSpan styleSpan_B = new StyleSpan(Typeface.BOLD);
StyleSpan styleSpan_I = new StyleSpan(Typeface.ITALIC);
spannableString.setSpan(styleSpan_B, 5, 7, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
spannableString.setSpan(styleSpan_I, 8, 10, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setHighlightColor(Color.parseColor("#36969696"));
textView.setText(spannableString);
```

- **ImageSpan**

在文本中添加🙄（表情）

`ImageSpan`，设置文本图片，实现方法如下：

```
SpannableString spannableString = new SpannableString("在文本中添加表情（表情）");
Drawable drawable = getResources().getDrawable(R.mipmap.a9c);
drawable.setBounds(0, 0, 42, 42);
ImageSpan imageSpan = new ImageSpan(drawable);
spannableString.setSpan(imageSpan, 6, 8, Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setText(spannableString);
```

这一个是不是很炫酷？再加一个解析算法，将文本中特定的文本转换成特定的表情图片，分分钟实现聊天表情显示效果有木有啊朋友们！

- **ClickableSpan**



`ClickableSpan`，设置可点击的文本，设置这个属性的文本可以相应用户点击事件，至于点击事件用户可以自定义，就像效果图显示一样，用户可以实现点击跳转页面的效果，具体实现方法如下：

```
SpannableString spannableString = new SpannableString("为文字设置点击事件");
MyClickableSpan clickableSpan = new MyClickableSpan("http://www.baidu.com");
spannableString.setSpan(clickableSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setMovementMethod(LinkMovementMethod.getInstance());
textView.setHighlightColor(Color.parseColor("#36969696"));
textView.setText(spannableString);
```

```
/*  
*****  
*/
```

```
class MyClickableSpan extends ClickableSpan {  
  
    private String content;
```



```

public MyClickableSpan(String content) {
    this.content = content;
}

@Override
public void updateDrawState(TextPaint ds) {
    ds.setUnderlineText(false);
}

@Override
public void onClick(View widget) {
    Intent intent = new Intent(MainActivity.this, OtherActivity.class);
    Bundle bundle = new Bundle();
    bundle.putString("content", content);
    intent.putExtra("bundle", bundle);
    startActivity(intent);
}
}

```

代码中我们自定义MyClickableSpan类，继承至ClickableSpan，并重写其中一些方法。ds.setUnderlineText()控制是否让可点击文本显示下划线，很明显，在上面代码中我选择了false，不显示下划线。onClick点击事件的具体实现方法写在其中。如上代码，我们重写ClickableSpan的onClick方法实现Activity的跳转效果，并传递跳转数据。

注意：使用ClickableSpan的文本如果想真正实现点击作用，必须为TextView设置setMovementMethod方法，否则没有点击相应，至于setHighlightColor方法则是控制点击是的背景色。

- **URLSpan**

URLSpan，设置超链接文本，其实聪明的小伙伴在讲到ClickableSpan的时候就能实现超链接文本的效果了，重写onClick点击事件就行，也确实看了URLSpan的源码，URLSpan就是继承自ClickableSpan，也和想象中一样，就是重写了父类的onClick事件，用系统自带浏览器打开链接，具体实现方法如下：

```

SpannableString spannableString = new SpannableString("为文字设置超链接");
URLSpan urlSpan = new URLSpan("http://www.jianshu.com/users/dbae9ac95c78");
spannableString.setSpan(urlSpan, 5, spannableString.length(),
    Spanned.SPAN_INCLUSIVE_EXCLUSIVE);
textView.setMovementMethod(LinkMovementMethod.getInstance());
textView.setHighlightColor(Color.parseColor("#36969696"));
textView.setText(spannableString);

```

URLSpan onClick事件的源码如下：

```
@Override
public void onClick(View widget) {
    Uri uri = Uri.parse(getURL());
    Context context = widget.getContext();
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    intent.putExtra(Browser.EXTRA_APPLICATION_ID, context.getPackageName());
    try {
        context.startActivity(intent);
    } catch (ActivityNotFoundException e) {
        Log.w("URLSpan", "Activity was not found for intent, " +
intent.toString());
    }
}
```

除此之外，还有 `MaskFilterSpan` 可以实现模糊和浮雕效果，`RasterizerSpan` 可以实现光栅效果，因为以上两个使用频率不高，而且效果也不是很明显，就不做详细说明，有兴趣的小伙伴不妨去试一试。

SpannableStringBuilder

应该有不少开发的小伙伴知道 `StringBuilder`，可以使用 `append()` 方法实现字符串拼接，非常方便。同样，`SpannableString` 中也有 `SpannableStringBuilder`，顾名思义，就是实现对 `SpannableString` 的一个拼接效果，同样是 `append()` 方法，可以实现各种风格效果的 `SpannableString` 拼接，非常实用。