简单自定义View

这个教程只是关于简单的自定义view，不涉及onMeasure和onLayout的重写。

这里的自定义View的目的是，当一个项目中需要重复相同的view，但是一直写一样的layout，不仅使项目过于冗余，而且也会容易出现问题。比如ui出现不同，或者编译出现问题等等。

这个时候写一个自定义view，在写一个layout时候直接引用就可以避免上述问题了。

1. 首先需要写一个自定义view的layout

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="144dp"
        android:layout_height="24dp"
        android:background="@drawable/label_blue">

        <TextView
            style="@style/CustomLabel"
            android:id="@+id/textField"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

根ViewGroup是ConstraintLayout，背景是一个圆角四方形。ConstraintLayout里面有一个TextView。

关于label_blue我就不列出来了。

2. 第二步，我们需要写关于自定义view的参数，在values文件夹中新建一个attrs.xml文件。代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CustomLabel">
        <attr name="text" format="string" />
    </declare-styleable>

</resources>
```

代码的意思是，在自定义View加入一个叫text的参数，类型为String。
在引用自定义view时要用app: 命名空间才能使用上述参数。

3. 第三步，写自定义View的Class。

```
class CustomLabel : ConstraintLayout {
    private lateinit var customLabel: CustomLabel

    val textView: TextView
        get() {
            return customLabel.findViewById(R.id.textField)
        }

    // 构造器，需要把下面三种方式都写上
    constructor(context: Context) : this(context, null)
    constructor(context: Context, attributeSet: AttributeSet?) : this(context, attributeSet, 0)
    constructor(context: Context, attributeSet: AttributeSet?, defStyleAttr: Int) : super(
        context,
        attributeSet,
        if (defStyleAttr != 0) defStyleAttr else R.attr.customLabel
    ) {
        // 进行初始化，因为有自定义view的参数，需要在这里进行配置
        init(attributeSet, defStyleAttr)
    }
```

```
    private fun init(attributeSet: AttributeSet?, defStyleAttr: Int) {
        val inflater = LayoutInflater.from(context)
        // 获取CustomLabel的View
        customLabel = inflater.inflate(R.layout.custom_layout, this, true) as CustomLabel
        // 获取typedArray
        val typedArray = context.theme.obtainStyledAttributes(
            attributeSet,
            R.styleable.CustomLabel,
            defStyleAttr,
            R.style.CustomLabel
        )

        try {
            // 获取text参数的值
            val src = typedArray.getString(R.styleable.CustomLabel_text)

            if (src.isNullOrEmpty()) {
            // 如果没有关于text的值， 直接设置默认为“hello world”
                textView.text = "hello world"
            } else {
                // 如果有，则直接设置
                textView.text = src
            }
        } finally {
            // 为了让系统使用typedArray，需要在强制重置
            typedArray.recycle()
        }
    }
}
```

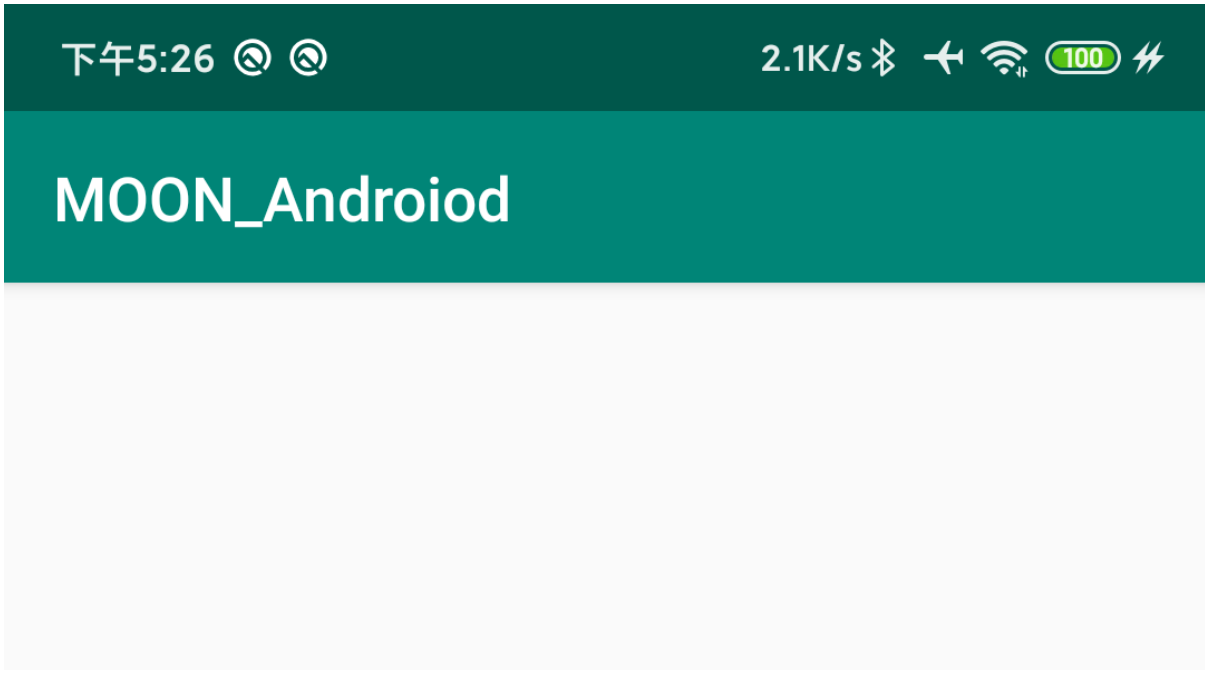
4. 第4步，自定义view的部分已经结束了， 可以在app的activity 中调用了。

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.hyejeanmoon.moon_androiod.ui.CustomLabel
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:text="moon"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />


    </androidx.constraintlayout.widget.ConstraintLayout>

</layout>
```

这样简单的自定义View就结束了。

效果如下图：



moon