MotionLayout简单使用

过渡动画

- 1、 有一个简单的需求，屏幕上左边居中有一个控件，现在要慢慢向右移动 `200dp`，有什么办法呢?

  我们可能会创建一个动画，然后 `translationX(px2dp(200))`，所以我们的代码可能是这样的

  ```
  child.animate()
       .translationX(px2dp(200))
       .start()
  ```

- 2、但是，UI不可能过来跟我们说，我要把这个控件向右移动200个dp，他可能会说要把控件移动到右边，所以，代码就可能会变成这样

  ```
  val parent = child.parent as ViewGroup
  val distance = parent.width - child.width
  child.animate()
      .translationX(distance.toFloat())
      .start()
  ```

- 3、现在有个问题是，UI是让我们从左边移动到右边，现在多出来一步计算的步骤，有没有办法省去呢，当然是可以的，我们可以调整布局参数，让它真正的跑到右边，所以代码可能会变成这样

  ```
  val layoutParams = child.layoutParams as FrameLayout.LayoutParams
  layoutParams.gravity = Gravity.END or Gravity.CENTER_VERTICAL
  child.layoutParams = layoutParams
  ```

- 4、但是这种方式是直接过去的，没有动画，比较生硬，怎么办呢，加一行代码就可以了

  ```
  // 注意，是在布局参数改变之前加
  TransitionManager.beginDelayedTransition(child.parent as ViewGroup)
  val layoutParams = child.layoutParams as FrameLayout.LayoutParams
  layoutParams.gravity = Gravity.END or Gravity.CENTER_VERTICAL
  child.layoutParams = layoutParams
  ```

- 5、为什么在布局参数改变之前加一行这个就可以有动画呢?

  过渡动画的原理:

  ```
  1、两个场景，一个开始场景，一个结束场景，记录场景上的控件的各种参数
  2、根据参数的变化创建动画
  3、播放动画
  ```

  对应 Transition 上的关键函数:

  ```
  1、捕获并记录对应属性
       captureStartValues 和 captureEndValues
  2、基于捕获的值创建对应动画，并播放动画
       createAnimator 和 playTransition
  ```

- 6、如果动画比较复杂，那么我们需要改变的参数就非常多，那么代码就会很多，太麻烦了，所以过渡动画还有另一种使用方式: `TransitionManager.go(scene)`

  > 我们可以用两个 `xml` 表示开始和结束场景，在两个场景之间进行切换
  >
  > 但是 `go` 函数需要我们在每次场景切换之后都要重新绑定数据，这是因为每次都把子控件全部移除了

- 7、有没有什么办法可以不用重新绑定数据，还能在两个场景进行切换呢?

  当然是有的，只要父容器是 `ConstraintLayout`, 就可以使用 `ConstraintSet` 来做过渡动画

  ```
  TransitionManager.beginDelayedTransition(root)
  val constraintSet = ConstraintSet().apply {
      clone(this@ConstraintSetActivity, R.layout.layout_constraintset_end)
  }
  constraintSet.applyTo(root)
  ```

- 8、这样就可以完成过渡动画，并且不用重新绑定数据，但是这样就完美了吗?

  当然不是的，约束布局还不够好

  > 1. 首先它不能停留在任意位置，动画一旦开始就必须等它结束，不能停留在任意位置
  >
  > 2. 其次它不支持触摸反馈，不能让动画跟着手指的滑动慢慢进行
  >
  > 3. 第三个是有布局重复，布局切换的套路代码不得不重复写一遍，还要同时管理两个xml文件，而且两个文件都是在layout文件夹下，第二个布局只是为了做动画用的，并不是真正的布局，没有放到专门管理动画的文件夹下

- 9、为了解决上面的问题，就引入了 `MotionLayout`

  `MotionLayout` 是 `ConstraintLayout` 的子类，所以约束布局能做的事情它都能做

  它的属性只有一个: `app:layoutDescription="@xml/motion_sample"`

  创建一个文件: `res/xml/motion_sample.xml`, 根节点是 `MotionScene`

  这样就有了一个布局文件和一个 `MotionScene` 文件，一个负责管理布局，一个专门用来管理动画

MotionLayout属性介绍

MotionScene节点

```
<MotionScene xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <Transition
        app:constraintSetStart="@+id/start"
        app:constraintSetEnd="@+id/end"/>

    <ConstraintSet android:id="@+id/start"/>

    <ConstraintSet android:id="@+id/end"/>
</MotionScene>
```

Constraint 节点

```
<Constraint android:id="">
      <Motion/>
      <Layout/>
      <Transform/>
      <CustomAttribute/>
      <PropertySet/>
  </Constraint>

运动模型
    弧线路径，时间模型等
布局相关
    注意， width 、 height 和 margin 的命名空间是 android: (beta1 开始)
    而约束相关的命名空间是 app （或 motion ）
动画变换
    做旋转，位移，缩放，海拔等属性
自定义属性
    attributeName 会加上 set/get 反射找到真正的函数名
    比如 backgroundColor 就会调用 setBackgroundColor() 函数
    custom(xxx)Value 对应属性的数据类型
特定的属性
    visibility 、 alpha 等属性
```

时间模型

```
每个控件可以单独设置时间模型
    在 <Motion> 节点中使用 app:transitionEasing
整个 Transition 也可以设置插值器
    在 <Transition> 中使用 app:motionInterpolator
```

Transition 节点

```
<Transition
    app:constraintSetStart="@+id/start"
    app:constraintSetEnd="@+id/end"
    app:duration="1000">

        <OnSwipe />

        <OnClick />

        <KeyPositionSet >

            <KeyAttribute>

                <CustomAttribute/>

            </KeyAttribute>

            <KeyPostion/>

            <KeyCycle/>

            <KeyTimeCycle/>
        </KeyPositionSet>
</Transition>
```

onSwipe

```
<OnSwipe
    app:touchAnchorId="@id/view"
    app:dragDirection="dragDown" />
```

`<OnSwipe>` 添加在 `<Transtion>` 节点中支持的参数

```
touchRegionId 指的是哪一个控件响应触摸事件
touchAnchorId 这个有点诡异，除了在指定的id上可以触摸，别的控件也可以触摸
autoComplete 默认的是 true ， 会根据动画完成的百分比自动到最近的一个状态
dragDirection 拖拽的方向
```

OnClick

```
<OnClick
    app:clickAction="toggle"
    app:targetId="@id/view"/>

targetId 指定控件

clickAction
    toggle 反转状态
    transitionToEnd/Start 通过动画到结束/起始状态
    jumpToEnd/Start 没有动画直接到结束/起始状态
```

KeyPositionSet

- `app:motionTarget` 目标对象ID
- `app:framePosition` 百分比 (0 - 100)

KeyAttribute 属性关键字

```
<KeyAttribute
    app:framePosition="0"
    app:motionTarget="@id/image_film_cover"
    android:elevation="12dp">

    <CustomAttribute
        app:attributeName="BackgroundColor"
        app:customColorValue="@color/colorAccent"/>
</KeyAttribute>
```

可以设置位移，旋转，缩放等属性，同时还可以通过 `CustomAttribute` 添加自定义属性

KeyPosition 位置关键帧

`percentX/Y` 在关键帧时，对应路径的对应百分比

`percentWidth/Height` 在关键帧时，控件大小改变的百分比

`curveFit` 运动路径的样式（直线，曲线等）

`keyPositionType` 坐标系

- `parentRelative`

```
（0， 0）为父容器左上角
（1， 1）为父容器右下角
```

- `deltaRelative`

```
（0， 0）为起始控件中心
（1， 1）为结束控件中心
```

- `pathRelative`

```
（0， 0）为起始控件中心
（1， 0）为结束控件中心
以 X 轴的长度顺时针旋转90度，画相同的长度就是Y轴
```

KeyCycle 和 KeyTimeCycle

通过 3 个 KeyCycle 定义一个准确的循环关键帧

```
<KeyFrameSet>
    <KeyCycle
        android:rotation="0"
        app:motionTarget="@id/fab_favourite"
        app:wavePeriod="0"
        app:framePosition="0"
        app:waveShape="sin"/>
    <KeyCycle
        android:rotation="180"
        app:motionTarget="@id/fab_favourite"
        app:wavePeriod="3"
        app:framePosition="50"
        app:waveShape="sin"/>
    <KeyCycle
        android:rotation="0"
        app:motionTarget="@id/fab_favourite"
        app:wavePeriod="0"
        app:framePosition="100"
        app:waveShape="sin"/>
</KeyFrameSet>
```

`wavePeriod` 循环次数 （`KeyTimeCycle` 表示的每秒循环次数）

`waveShape` 数学模型