00.来源

- LikeView继承自RelativeLayout，是用于响应用户单击和双击的交互，单机暂停或播放视频 / 双击在双击位置上方播放点赞动画。它位于推荐页的ViewHolder子视图内部。

01.init方法

- 在构造方法中会调用init方法。该方法用于初始化手势检测器，并设置在OnTouch中调用手势检测器，进行响应用户的交互。
- 假如是最硬核的，那么我们去onTouchEvent中直接去写逻辑。但是比较麻烦和繁琐，效果也不好，我们这里采用提供的组件，也就是手势检测器：GestureDetector。
- 初始化手势检测逻辑：
  - 我们要去创建一个手势检测器的实例。
  - 我们调用这个GestureDetector类的构造方法，参数是一个匿名内部类继承自`SimpleOnGestureListener`。该匿名内部类内部我们覆写了onDoubleTap双击和onSingleTapConfirmed单击方法。
  - onDoubleTap中通过`addLikeView(e)`来添加点赞图标，然后触发`onLikeListener`的回调。
  - ```
    gestureDetector = GestureDetector(object : SimpleOnGestureListener() {
                override fun onDoubleTap(e: MotionEvent): Boolean {
                    addLikeView(e)
                    onLikeListener!!.onLikeListener()
                    return true
                }

                override fun onSingleTapConfirmed(e: MotionEvent): Boolean {
                    if (onPlayPauseListener != null) {
                        onPlayPauseListener!!.onPlayOrPause()
                    }
                    return true
                }
            })
    ```

    > **这里的onLikeListener是什么?**
    >
    > 它是我们在推荐页的onBindViewHolder中设置的，它会调用ControllerView视图的like爱心按钮的点赞动画。
    >
    > 也就是说，我们双击屏幕，不仅会出现屏幕上的点赞动画，ControllerView层的爱心图标也会执行动画，这是为了与抖音保持一致。
    >
    > ```
    > interface OnLikeListener {
    >     fun onLikeListener()
    > ```
    ```

```
}


private var onLikeListener: OnLikeListener? = null

fun setOnLikeListener(onLikeListener: OnLikeListener?) {
    this.onLikeListener = onLikeListener
}

//VideoAdapter中
// 3. 设置点赞监听器
holder?.binding?.likeview?.setOnLikeListener(object : OnLikeListener
{
override fun onLikeListener() {
if (!it.isLiked) {
  holder?.binding?.controller!!.like()
}
}
})
```

- 单击则会调用onPlayOrPause方法。

  这个onPlayPauseListener!!.onPlayOrPause()是什么?

  - 这个方法我们在playCurVideo中设置，他会进行视频播放器的暂停和播放控制
  - onPlayPauseListener是OnPlayPauseListener接口的实例`private var onPlayPauseListener: OnPlayPauseListener? = null`
  - OnPlayPauseListener是一个接口
    - ```
      interface OnPlayPauseListener {
              fun onPlayOrPause()
          }
      ```
  - onPlayPauseListener通过setOnPlayPauseListener赋值
    - ```
      fun setOnPlayPauseListener(onPlayPauseListener:
      OnPlayPauseListener?) {
              this.onPlayPauseListener = onPlayPauseListener
          }
      ```
  - 在推荐页的playCurVideo播放当前视频逻辑中，我们为其赋值

```
likeView.setOnPlayPauseListener(object:
LikeView.OnPlayPauseListener {
            override fun onPlayOrPause() {
                if (videoView!!.isPlaying()) {
                    videoView?.pause()
                    ivPlay.visibility = View.VISIBLE
                } else {
                    videoView?.play()
                    ivPlay.visibility = View.GONE
                }
            }

        })
```

    - 所以该具体逻辑是：
        - 如果视频是播放状态，那么设置视频暂停，并让播放按钮可见。
        - 如果视频是暂停状态，那么设置视频播放，并让播放按钮不可见。

- 设置OnTouchListener.onTouch回调，在内部会调用手势检测方法，进行View的点击事件消费。
    - view.setOnTouchListener设置onTouchListener。
    - 这个设置的onTouchListener是一个匿名内部类，在该类内部覆写onTouch方法，在该方法中调用gestureDetector.onTouchEvent进行event点击事件处理，并返回true。
    - ```
      view.setOnTouchListener(new View.OnTouchListener() {
          @Override
          public boolean onTouch(View v, MotionEvent event) {
              gestureDetector.onTouchEvent(event);
              return true;
          }
      });
      ```

- 那么对于该LikeView，其事件点击的具体流程是什么样的呢？
    - 事件发生，传递到 LikeView 的 dispatchTouchEvent()
    - 触发 OnTouchListener.onTouch()方法。
    - 内部调用 gestureDetector.onTouchEvent(event)。
    - 其解析原始 MotionEvent，识别单击、双击、滑动等手势，并回调 SimpleOnGestureListener 的对应方法（如 onDoubleTap()）。

> **那么事件分发流程的onTouchEvent呢？为什么不是调用它？**
>
> 其会优先判断是否有onTouch回调，没有才调用onTouchEvent。

> 双击产生了2对Action_Down和Action_UP事件，SimpleOnGestureListener 是如何解析其为双击操作而不是两次单击操作的？
>
> 当第一个ACTION_UP发生时，GestureDetector会记录第一个 `ACTION_DOWN` 的时间戳和坐标。启动一个 **双击超时计时器**（默认 300ms）。并等待可能的第二次点击。如果在定时器超时前接收到第二个ACTION_DOWN，则会检查与首次点击的坐标差是否在阈值内（由getScaledDoubleTapSlop方法定义的阈值），如果满足条件，则触发双击事件，并取消单击事件的触发。
      ```

02.addLikeView方法

- 用户双击会执行addLikeView方法，该方法具体效果是什么呢？该方法用于 **在用户双击的位置上方添加一个点赞图标（`ImageView`）**，并触发动画效果。具体流程如下：
- 其接收一个MotionEvent触摸事件对象，内部存储点击事件的信息，我们通过它获取点击坐标。
  - `private fun addLikeView(e: MotionEvent)`
- 创建ImageView，通过setImageResource设置点赞爱心图标，然后将 ImageView 添加到当前 LikeView中（RelativeLayout）。

```
// 1. 创建 ImageView 并设置资源
val imageView = ImageView(context)
imageView.setImageResource(R.mipmap.ic_like) // 设置点赞图标

// 2. 将 ImageView 添加到当前 LikeView（RelativeLayout）
addView(imageView)
```

- 定位ImageView显示的位置，通过设置其布局参数实现，包括尺寸和位置
  - 尺寸是固定宽高
  - 位置是触摸点水平上移一点距离，避免遮挡手指。
  - ```
    // 3. 设置布局参数（尺寸、位置）
       val layoutParams = LayoutParams(likeViewSize, likeViewSize) // 固定宽高

       // 水平居中：触摸点 X 坐标 - 图标宽度的一半
       layoutParams.leftMargin = e.x.toInt() - likeViewSize / 2

       // 垂直上移：触摸点 Y 坐标 - 图标高度（避免遮挡手指）
       layoutParams.topMargin = e.y.toInt() - likeViewSize

       // 4. 应用布局参数
       imageView.layoutParams = layoutParams
    ```
- `playAnim(imageView)` 触发ImageView的播放动画

03.playAnim播放动画效果

- 该方法为点赞图标（`ImageView`）创建了一个 **组合动画**，在动画结束后自动移除视图。它本质原理是通过创建这个动画集合，然后执行动画集合实现的。
- 初始化动画效果集合（animationSet）
  - 创建动画效果集合animationSet
  - animationSet.addAnimation，添加旋转动画，角度为（-30°, 0°, 30°）中随机选择。
  - 0 - 100ms内，图标进行淡入操作，缩小，变的不透明。：scaleAnim和alphaAnim， 0 - 100ms内从2f缩小到1f，透明度从0f-1f。
  - 100ms - 300ms内，保持动画的显示。
  - 300-800ms内，进行淡出操作，我们进行图标放大、变淡、上移并最终消失操作，：scaleAnim，alphaAnim，translationAnim。300-800ms内，从1f放大到1.8f，透明度从1f-0f，向上位移400px。
    ```

- 设置动画结束回调，也就是AnimationListener中onAnimationEnd。在内部通过Handler.post（removeView(view)）移除视图。
- ```
    // 创建动画效果集合
    val animationSet = AnimationSet(true)

    // 设置开始的动画旋转
    val degrees = angles[Random().nextInt(3)] // 随机旋转角度（-30°, 0°, 30°）
    animationSet.addAnimation(AnimUtils.rotateAnim(0, 0, degrees.toFloat())) // 旋转动画

    //0 - 100ms内从2f缩小到1f，透明度从0f-1f（图标快速弹出并居中）
    animationSet.addAnimation(AnimUtils.scaleAnim(100, 2f, 1f, 0))
   // 初始缩小
    animationSet.addAnimation(AnimUtils.alphaAnim(0f, 1f, 100, 0))
   // 淡入

    //300-800ms内，从1f放大到1.8f，透明度从1f-0f。向上位移400px（图标放大、变淡、上移并最终消失）
    animationSet.addAnimation(AnimUtils.scaleAnim(500, 1f, 1.8f, 300))
   // 再次放大
    animationSet.addAnimation(AnimUtils.alphaAnim(1f, 0f, 500, 300))
   // 淡出
    animationSet.addAnimation(AnimUtils.translationAnim(500, 0f, 0f, 0f, -400f, 300)) // 上移

    // 动画结束回调
    animationSet.setAnimationListener(object : Animation.AnimationListener {
        override fun onAnimationEnd(animation: Animation) {
            Handler().post { removeView(view) } // 安全移除视图
        }
        // ...
    })
  ```

- 通过view.startAnimation(animationSet)启动该ImageView的动画播放效果。这里的view就是上面的ImageView的引用。

> startAnimation是什么？它的工作原理怎样？
  ```