SHOUT · OUR · PASSION · TOGETHER

ODo IT SOPT O

# Android 5차 세미나

SHOUT OUR PASSION TOGETHER
SOPT

00

# 사 전 준 비

# 사전 준비 사항

1. 새로운 프로젝트 생성 후, Gradle에 세 개 라이브러리 미리 추가(복붙)

```
implementation 'com.airbnb.android:lottie:2.6.0-beta19'

implementation "org.jetbrains.anko:anko:0.10.5"

implementation 'de.hdodenhof:circleimageview:2.2.0'
```

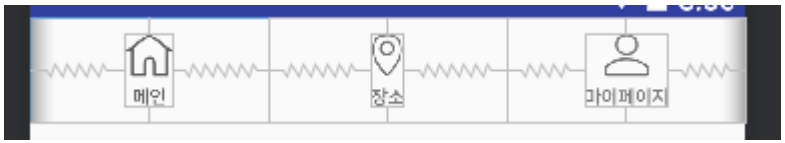
둥근 ImageView

2. Activity 미리 구성해놓기
   - MainActivity : 안드 세미나 때 사용
   - SubActivity : 안드 세미나 때 사용
   - MyPageActivity : 합동 세미나 때 사용

01

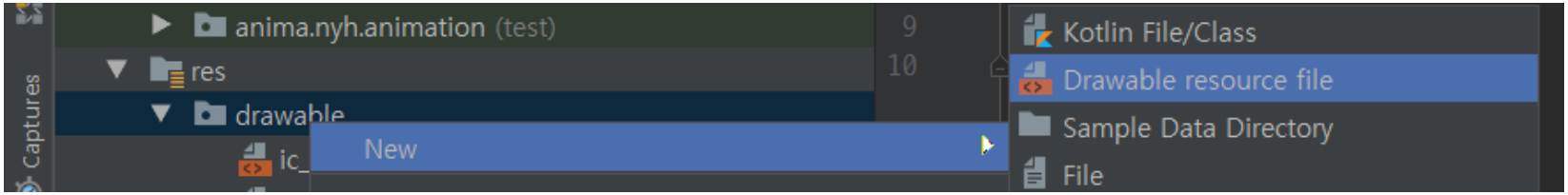
# Shape, Font

## 예전 세미나때 selector 만들었던 것처럼…




```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/tab_main_icon_green" android:state_selected="true" />
    <item android:drawable="@drawable/tab_main_icon_gray" android:state_selected="false" />
</selector>
```

## 1. drawable → New → Drawable resource file로 shape_name_box.xml 만들기!



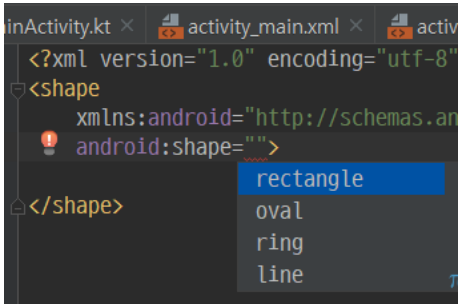
SHOUT OUR PASSION TOGETHER SOPT

# 2. shape의 소요 알아보기 - shape의 유형

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

</shape>
```

- `shape` 로 바꿔 주시고, `android:shape="rectangle"` 를 추가해주세요!
- 자동완성을 통해 shape종류를 알 수 있습니다.

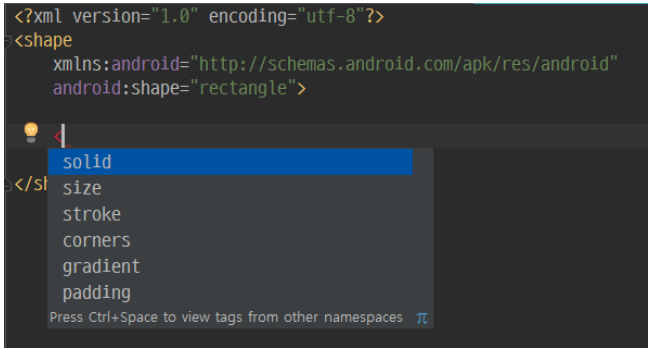
| 값 | 설명 |
|---|---|
| "rectangle" | 포함하는 뷰를 채우는 사각형. 이는 기본 셰이프입니다. |
| "oval" | 포함하는 뷰의 치수에 맞는 타원형 셰이프. |
| "line" | 포함하는 뷰의 너비에 걸쳐 있는 가로선. 이 셰이프를 사용하려면 <stroke> 요소를 통해 선의 너비를 정의해야 합니다. |
| "ring" | 고리형 셰이프. |




- 저는 지금 직사각형 name box를 만들어보려고 하기 때문에 rectangle로!!!
- **shape의 종류에 따라 쓸 수 있는 요소가 다르므로 공식 사이트를 참조해주세요!**

  **https://developer.android.com/guide/topics/resources/drawable-resource?hl=ko**

## 3-1. shape의 내부 요소을 알아보자! – 지극히 개인적으로 자주 쓰는 요소 중심으로!

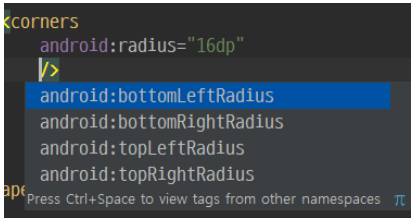



- <solid> : shape에 단색을 채울 때 사용
  (그라데이션은 <gradient> 사용하기~)

```
<solid
    android:color="#E1BEE7"/>
```

- <corners> : shape가 사각형일 때, 둥근 모서리를 만들 때 사용
  (오른쪽 왼쪽 위 아래 개별적으로 둥근 모서리를 줄 수 있습니다.)

```
<corners
    android:radius="16dp" />
```

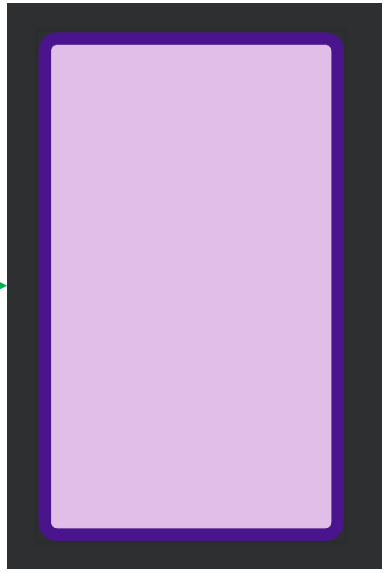


## 3-2. shape의 내부 요소을 더 알아보자! – 지극히 개인적으로 자주 쓰는 요소 중심으로!

- <stroke> : shape의 윤곽선의 두께, 색을 지정할 때 사용
  (dashGap, dashWidth 요소를 통해 점선도 구현 가능!)

```
<stroke
    android:color="#4A148C"
    android:width="16dp" />
```

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

    <solid
        android:color="#E1BEE7"/>

    <corners
        android:radius="16dp" />

    <stroke
        android:color="#4A148C"
        android:width="16dp" />

</shape>
```



preview

예시를 보이기 위해 꾸역꾸역 요소를 꾸겨 넣으면 이렇게 됩니다!

SHOUT OUR PASSION TOGETHER SOPT

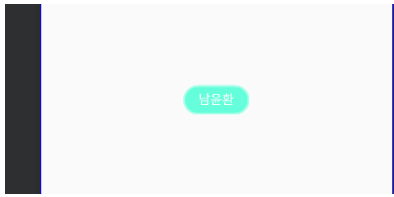
# 4. 직접 한번 만들어보고 적용 시켜 보기 - SubActivity만들고 activity_sub.xml에…

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid
        android:color="#64FFDA"/>
    <corners
        android:radius="24dp" />
    <stroke
        android:color="#A7FFEB"
        android:width="2dp" />
</shape>
```

shape_name_box_rectangle.xml

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


    <RelativeLayout
        android:layout_width="72dp"
        android:layout_height="32dp"
        android:layout_centerInParent="true"
        android:background="@drawable/shape_name_box_rectangle">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="남윤환"
            android:textColor="#FFFFFF" />
    </RelativeLayout>

</RelativeLayout>
```

activity_sub.xml

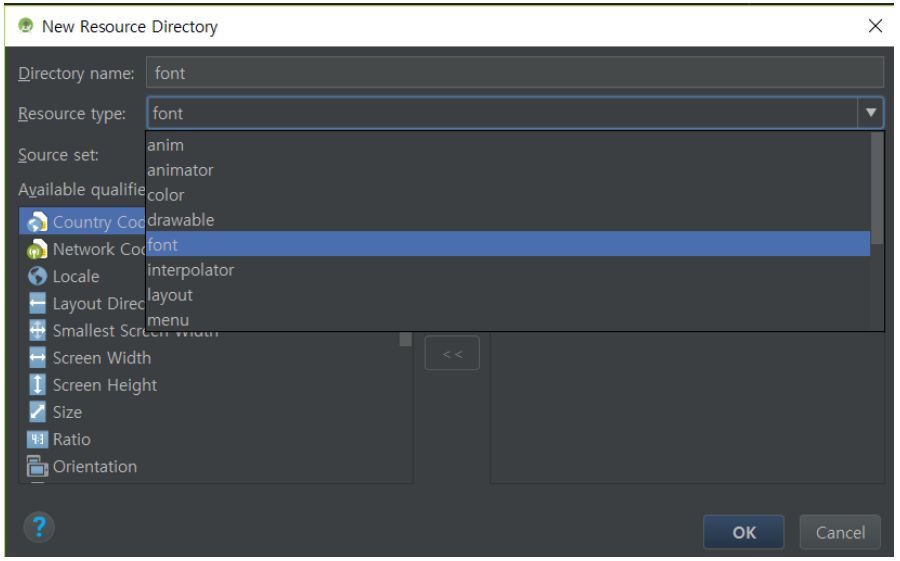



android:background="@drawable/shape_name_box_rectangle">

SHOUT OUR PASSION TOGETHER SOPT

## - 다운받은 폰트를 넣어보자!!!

- res 디렉토리 마우스 오른쪽 클릭 → New → New Resource Directory




- Resource Type을 font로 바꾼 뒤 OK를 눌러주세요.
- 생성된 res/font에 다운받은 폰트 넣기
- EditText의 옵션 fontFamily를 통해 다운받은 폰트 설정

```
android:fontFamily="@font/다운받은폰트파일명"
```

02

# Lottie

SHOUT OUR PASSION TOGETHER SOPT

# 1. Lottie란 무엇인가

- JSON으로 내보낸 Adobe After Effects 애니메이션에 대해 구문 분석을 하고 모바일에 네이티브로 렌더링 하는 Android 및 iOS 용 모바일 라이브러리다!! – Airbnb design

- 쉽게 생각해서, After Effect 라는 툴이 있는데!!! 그걸 통해서 애니메이션을 만들면,
  모바일에 옮기는데 도와주는 라이브러리 입니다. 이해되셨나요?!?!!! 말보다는 실습으로!!!

- 비교되는 것으로 페이스북의 Keyframes가 있습니다!! 궁금하신 분은 찾아보세요~!!!

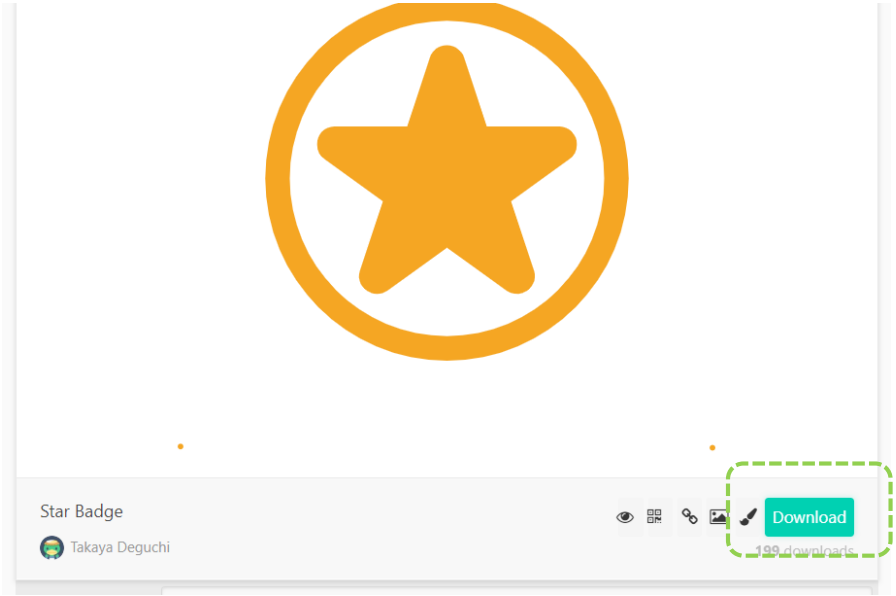
**일단, 라이브러리를 주입시켜봅시다! ➔ https://github.com/airbnb/lottie-android**

**2018/11/15 기준 2.8.0 버전이 최신 버전인데, 우리가 세미나 하는 도중에 Android Studio 버전이 바뀌면서,**
**최신 라이브러리를 쓰려면 마이그레이션 관련 작업이 필요하니까 아래 버전으로 실습하겠습니다!!!**

```
implementation 'com.airbnb.android:lottie:2.6.0-beta19'
```

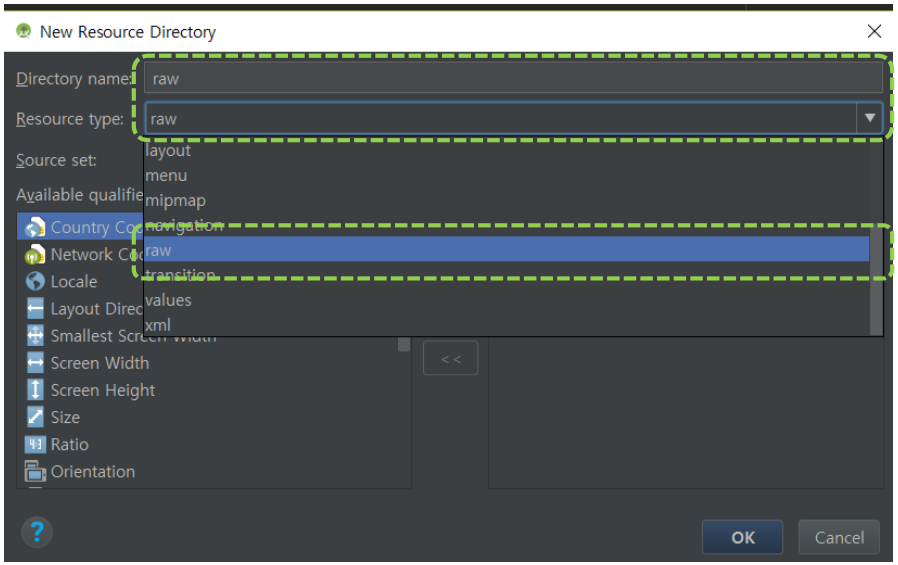
## 2. 실습을 위해 애니메이션이 담긴 JSON 파일을 다운로드 해봅시다!

https://www.lottiefiles.com/ 사이트로 이동해서 원하는 애니메이션 클릭 후




Download 해주세요!

## 3. res 디렉토리 마우스 우 클릭 → New → New Resource Directory




- Resource type을 raw로 한 뒤 OK

## 4. 다운 받은 애니메이션 json 파일을 raw 디렉토리에 넣어주세요!

- 알려드린 Image Resource 추가하는 것 처럼 똑같이 넣으면 됩니다!

# 5. 애니메이션이 들어갈 View 만들기

- 본인이 원하는 곳에 실습하세요! 저는 새로운 프로젝트를 하나 파서! MainActivity에서 하겠습니다!

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

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/lottie_main_act_star"
        android:layout_centerInParent="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:lottie_rawRes="@raw/star_badge"
        app:lottie_loop="false"
        app:lottie_autoPlay="true"/>


</RelativeLayout>
```

<activity_main.xml>

- app:lottie_rawRes 옵션에는 본인이 다운받은 JSON파일 명. `app:lottie_rawRes="@raw/animation_checker"`
- app:lottie_loop 옵션은 반복할 것인지 아닌지. `app:lottie_loop="false"`
- app:lottie_autoPlay 옵션은 자동으로 움직이게 할 것인지. `app:lottie_autoPlay="true"`
- 해당 옵션들은 Programming으로도 설정 할 수 있어요!!

# 5. Animator Listener 달기

- 애니메이션이 시작될 때, 끝날 때, 반복될 때 등 상태를 캐치할 수 있어야 우리가 다양한 구현이 가능하겠죠?!

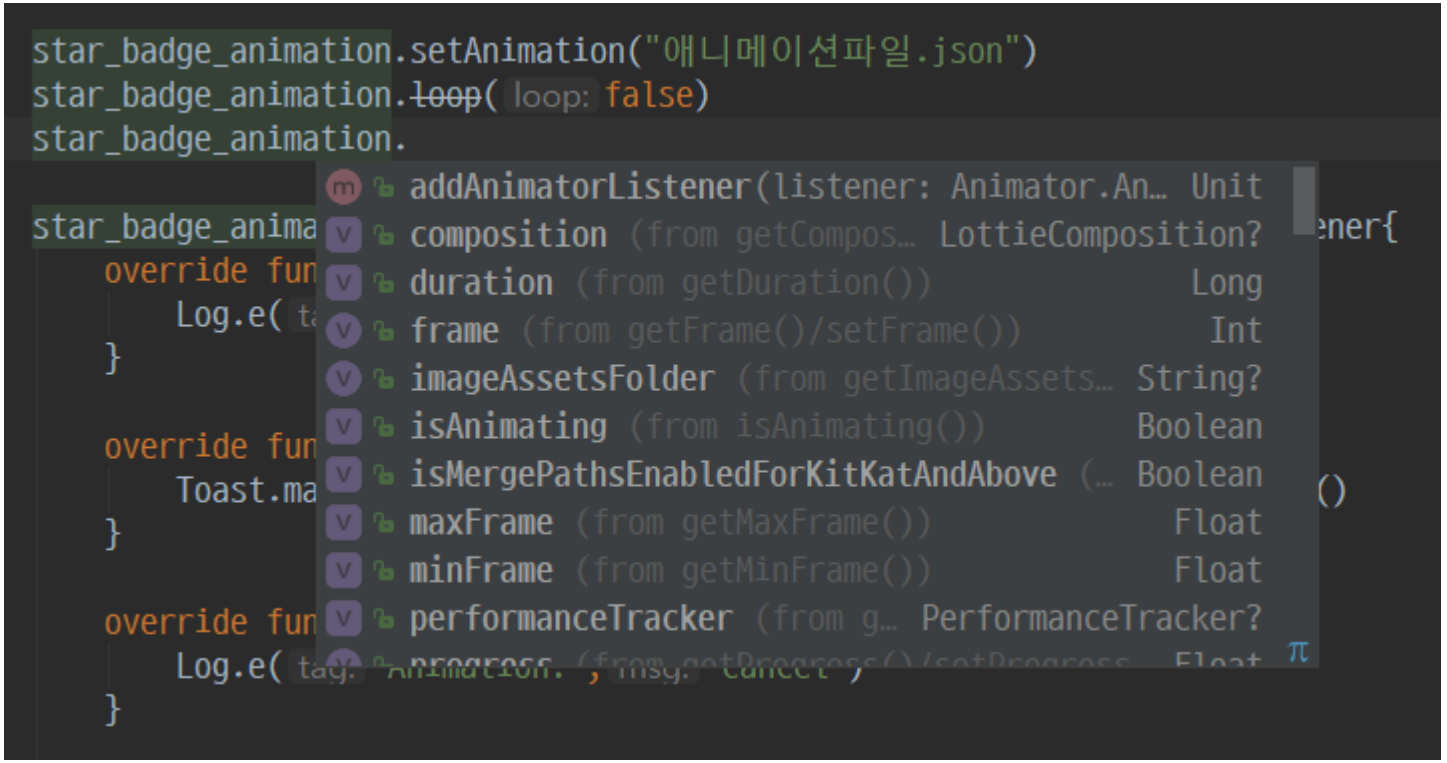
클릭 시 한번 더 재생되는 CODE

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val star_badge_animation : LottieAnimationView = findViewById(R.id.lottie_main_act_star)

        lottie_main_act_star.setOnClickListener {
            star_badge_animation.playAnimation()
        }

        star_badge_animation.addAnimatorListener(object : Animator.AnimatorListener{
            override fun onAnimationRepeat(animation: Animator?) {
                Log.e("Animation:","repeat")
            }

            override fun onAnimationEnd(animation: Animator?) {
                Toast.makeText(application, "끝~",Toast.LENGTH_SHORT).show()
            }

            override fun onAnimationCancel(animation: Animator?) {
                Log.e("Animation:","cancel")
            }

            override fun onAnimationStart(animation: Animator?) {
                Log.e("Animation:","start")
            }
        })
    }
}
```

반복, 끝, 시작, 취소 상태 캐치 리스너

<MainActivity.kt>

## 6. 동적으로 다양한 애니메이션 옵션 접근 가능하니까, 잘 활용하기!



03

# Animation 구현

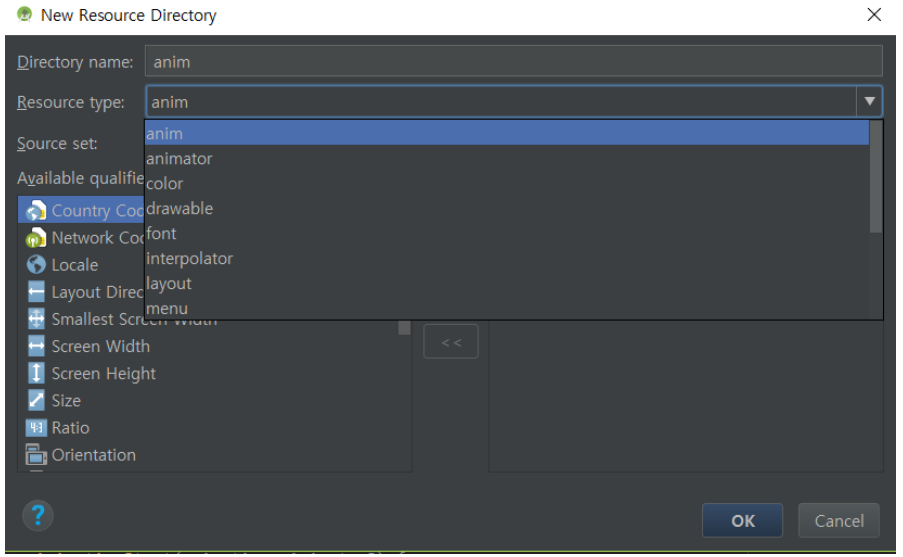
SHOUT OUR PASSION TOGETHER SOPT

## 들어가기에 앞서…

- 애니메이션은 다양하다.
- **Frame Animation, Tween Animation, Property Animation 등..**
- **우리는 Tween Animation을 배워볼 것!**

# 1. 애니메이션의 재사용, 쉬운 편집을 위해! 애니메이션을 xml로 정의

- Tween Animation : 애니메이션 대상에게 애니메이션 연산을 주어 그 결과를 연속적으로 디스플레이 하는 방식.
- 앞서 만든 font, raw 디렉토리처럼 res 디렉토리에 New → New Resource Directory 클릭 후




- Resource type을  anim 선택 후 OK
- 그리고 anim 디렉토리 오른쪽 클릭 후 New Resource File을 통해 xml 파일을 만들어주세요!

## 2. 실습 전 애니메이션 xml 파일에 구성할 속성에 대해 알아봅시다!

- <alpha> : 투명도
- <rotate> : 회전
- <scale> : 크기 확대/축소
- <translate> : 위치 이동

- 위 4가지 속성으로 애니메이션을 구현하는데,
  <set>은 4가지를 동시에 적용되도록 묶을 수 있습니다!!!

SHOUT OUR PASSION TOGETHER SOPT

# 3. 속성에 들어갈 수 있는 공통 옵션에 대해 알아보자!

```
android:interpolator="@android:anim/linear_interpolator"
android:repeatCount="infinite"
android:repeatMode="reverse"
android:fillAfter="true"
android:startOffset="200"
```

- **interpolator :** 애니메이션 효과 중에,
  빠르게 또는 느리게 효과를 줄 수 있다.
- **repeatCount :** 반복 횟수
- **repeatMode :** 반복 모드(restart,reverse)
- **fillAfter :** 애니메이션 효과 후 상태 유지
  기본값은 false
- **startOffset :** 애니메이션 시작 전 대기 시간

- **interpolator 값 정리**
  - ① **accelerate_interpolator**
    **: 점점 빠르게**
  - ② **decelerate_interpolator**
    **: 점점 느리게**
  - ③ **accelerate_decelerate_interpolator**
    **: 점점 빠르게 가다가 느리게**
  - ④ **anticipate_interpolator : 시작 때 당기기**
  - ⑤ **overshoot_interpolator : 종료 때 늘리기**
  - ⑥ **bounce_interpolator**
    **: 종료 위치에서 튀도록**

# 4. <translate> 위치 이동 알아보기

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0%"
        android:toXDelta="100%"
        android:fromYDelta="0%"
        android:toYDelta="0%"
        />
</set>
```

- fromXDelta/fromYDelta : 시작 지점
- toXDelta/toYDelta : 끝 지점

SHOUT OUR PASSION TOGETHER SOPT

# 5. <rotate> 회전 알아보기

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromDegrees="0"
        android:toDegrees="360"
        android:duration="1000"
        />
</set>
```

- pivotX/pivotY : 회전 기준점으로 부터 고정 점 위치
- fromDegrees :  몇 도부터 인지
- toDegrees : 몇 도 까지
- duration : 얼마만에 회전 시킬 것인지

SHOUT OUR PASSION TOGETHER SOPT

## 6. <scale> 확대/축소 알아보기

```
<scale
    android:pivotX="0%"
    android:pivotY="100%"
    android:fromXScale="0.0"
    android:fromYScale="1.0"
    android:toXScale="1.0"
    android:toYScale="1.0">
</scale>
```

- pivotX/pivotY : 회전 기준점으로 부터 고정 점 위치
- fromXScale/fromYScale
  : (1.0일 때 원래 크기) 효과 시작 크기
- toXScale/toYScale
  : 효과 끝 크기

SHOUT OUR PASSION TOGETHER SOPT

# 7. <alpha> 투명도 알아보기

```
<alpha
    android:fromAlpha="0.0"
    android:toAlpha="1.0"
    android:duration="1000"
    />
```

- fromAlpha : 시작 투명도
- toAlpha : 종료 투명도
  (1.0일 때 원래 투명도)

# 실습01

- **애니메이션 xml 파일 만들기 - example_animation.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="500"
        android:fromXDelta="0%p"
        android:fromYDelta="100%p"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toXDelta="0%p"
        android:toYDelta="0%p" />
</set>
```

# 실습01

- **SubActivity의 layout 구성 - activity_sub.xml**

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

    <RelativeLayout
        android:id="@+id/btn_sub_act_show_bottom_bar"
        android:layout_width="72dp"
        android:layout_height="32dp"
        android:layout_centerInParent="true"
        android:background="@drawable/shape_name_box_rectangle">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="클릭!"
            android:textColor="#FFFFFF" />
    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/rl_sub_act_bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="160dp"
        android:layout_alignParentBottom="true"
        android:background="#A7FFEB"
        android:elevation="2dp"
        android:visibility="gone">
    </RelativeLayout>
</RelativeLayout>
```

SHOUT OUR PASSION TOGETHER SOPT

# 실습01

- SubActivity.kt 코드

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import kotlinx.android.synthetic.main.activity_sub.*
import org.jetbrains.anko.toast

class SubActivity : AppCompatActivity() {

    val bottomBarAnimation: Animation by lazy {
        AnimationUtils.loadAnimation(this, R.anim.example_animation)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sub)

        bottomBarAnimation.setAnimationListener(object : Animation.AnimationListener {
            override fun onAnimationRepeat(p0: Animation?) {
            }

            override fun onAnimationEnd(p0: Animation?) {
                toast("애니메이션 끝!")
            }

            override fun onAnimationStart(p0: Animation?) {

            }
        })

        btn_sub_act_show_bottom_bar.setOnClickListener {
            rl_sub_act_bottom_bar.visibility = View.VISIBLE
            rl_sub_act_bottom_bar.startAnimation(bottomBarAnimation)
        }
    }
}
```

SHOUT · OUR · PASSION · TOGETHER

ODo IT SOPT O

# THANK U

SHOUT OUR PASSION TOGETHER
SOPT