August 1st - Layout & Thread basic

Contents

1. Layout

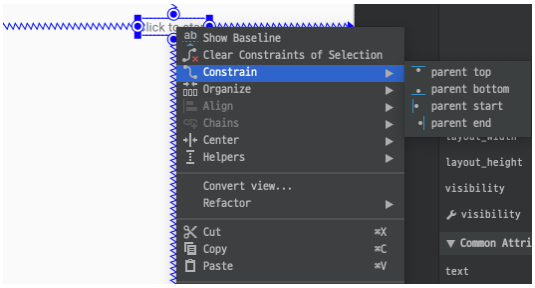
a. FrameLayout, LinearLayout, RelativeLayout, GridLayout

- LayoutParams - 각 ViewGroup 들이 갖고 있는 클래스로써 child View 들이 해당 ViewGroup 에서 얼만큼 크고(onMeasure) 어떻게위치할지(onLayout) 정해주는 역할을 한다. 코드에서 직접 지정할 수도 있으며, xml 에서 android:layout_XXX 같은 속성들이 이를 결정한다.
- FrameLayout - 기본적인 ViewGroup 중 하나이다. child View 들이 놓여질 때 왼쪽 위에 겹치면서 쌓여진다.
- LinearLayout - 한 방향으로 child View 들을 쌓을 때 사용한다. android:orientation 속성으로 가로, 세로 방향을 지정할 수 있으며 weightSum 속성과 layout_weight 속성을 사용하여 childView 들이 LinearLayout 안에서 얼마만큼(%)을 차지할 것인지 결정해줄 수 있다.(LayoutSample 앱 첫 화면 참조)
- RelativeLayout - View 들끼리나 부모 ViewGroup 과 위치 관계를 상대적으로 정의하는 ViewGroup 이다. ConstraintLayout 의 하위 호환이고 잘 안쓰인다.
- GridLayout - 쓰면바보

b. ConstraintLayout 의 장점

ConstraintLayout 은 RelativeLayout 과 비슷하지만, 뷰들의 크기와 위치를 제약을 통해 정의한다는 점에서 조금 다르고 더 화면 구성을 유연하게 할 수 있다. 장점은 다음과 같다.

- 뷰 계층을 평탄화(flatten)할 수 있다. => 뷰 그룹 안에 또 뷰 그룹을 넣어서 만들 레이아웃을 1 단계의 뷰 그룹 계층으로 구성할 수 있다.
- 다른 기본 레이아웃을 모두 대체할 수 있다.
- 자식 뷰의 크기를 가로:세로 비율을 지정할 수 있다.
- 구글이 밀어주는 대세 ViewGroup 이다. => 안드로이드 스튜디오에서 레이아웃 디자인 화면에서 ConstraintLayout 과 관련된 옵션들을 쉽게 이용할 수 있다.



- iOS 에서도 오토레이아웃 이라는 비슷한 개념을 이용해서 뷰를 구성하기 때문에 나중에 iOS 할 때도 쉽다.

ConstraintLayout 의 개념 및 사용법

- 뷰 들끼리의 제약을 걸어서 구성하는 레이아웃
- app:layout_constraintXXX 와 같은 속성으로 정의 가능
-
```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```
- width 와 관련된 제약을 만들면 android:layout_width 가 0dp 로 설정이 되있어야 제약대로 레이아웃이 구성됨 => height 도 마찬가지
- 제약을 만들 때, View 의 Id 를 참조
- chain => 서로 제약을 거는 것 (LayoutSample 2 번째 페이지 참조)
- 부모 뷰 크기에 % 로 크기를 설정 가능 (LayoutSample 3 번째 페이지 참조)
- circle => 반지름과 각도를 정해서 다른 뷰를 기준으로 레이아웃을 배치 가능 (LayoutSample 4 번째 페이지 참조)
- ConstraintSet 이란 클래스로 제약을 프로그래밍적으로 생성 가능 (ConstraintLayout.LayoutParams 와는 다르다)

c. CoordinatorLayout 이란?

- 슈퍼파워를 가진 FrameLayout
- CoordinatorLayout 의 자식 뷰들끼리 서로의 상태를 관찰하며 상태가 변할때마다 반응
- NestedScroll 이벤트를 감지하고 반응
- 매터리얼 디자인을 구현하는 데 많이 쓰임
- CoordinatorLayout.LayoutParams 에 존재하는 Behavior 란 녀석이 핵심

2. Thread

a. 안드로이드의 작업 분류

- 앱이 Foreground 일 때 실행되는 작업 vs Background 일 때 실행되는 작업
- Background 작업을 처리하는 방법

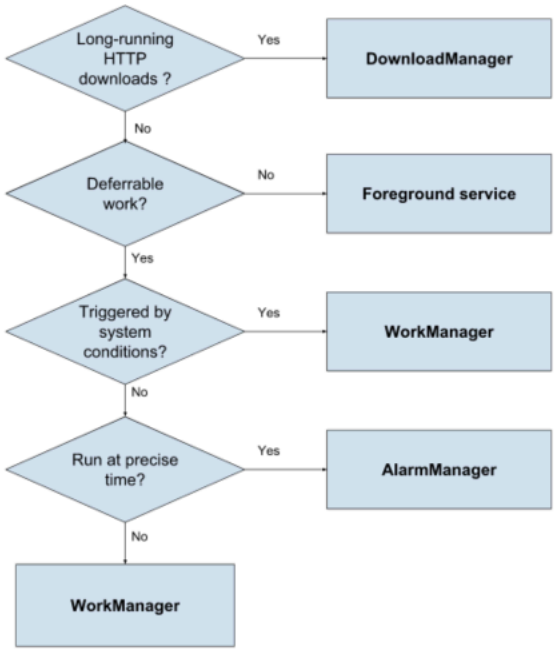


Figure 1. What's the best way to do background work?

-

b. Thread 란?

- Process vs Thread
- Thread 의 중요성
- Android 에서 Thread 를 다루는 방법들
- Thread, Handler, HandlerThread, Looper, AsyncTask, runOnUiThread, MessageQueue
- Thread 와 concurrency
- Thread 를 쉽게 다룰 수 있도록 해주는 라이브러리들
- Android Lifecycle Component, LiveData, RxJava, Kotlin Coroutine, Retrofit etc..