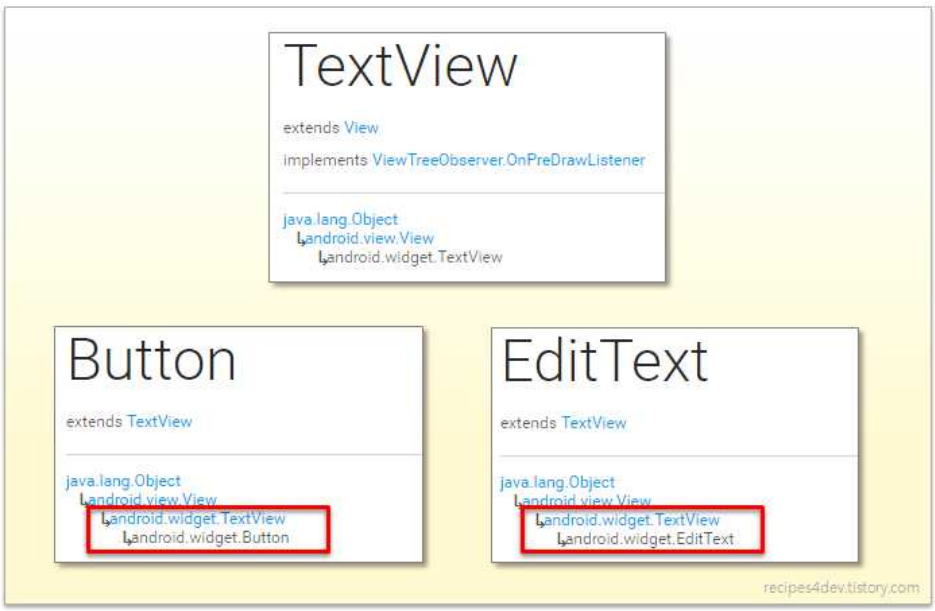
텍스트뷰 기본 사용법. (Android TextView)

1. 안드로이드 TextView

TextView는 안드로이드 UI를 구성함에 있어 화면에 텍스트를 표시하는 기능을 담당하며, 안드로이드에서 제공하는 위젯 중 가장 많이 사용되는 위젯입니다. 텍스트 출력 기능을 가진 기본 위젯이므로, 텍스트와 연관된 기능을 포함하는 Button 또는 EditText의 부모 클래스이기도 합니다.



TextView가 안드로이드에서 제공하는 가장 기본이 되는 위젯인 만큼 사용 방법도 간단합니다. 기본 속성을 사용하여 단순한 문자열(Plain Text)을 출력하는 용도라면, Layout 리소스에 TextView를 정의하고 "text" 속성에 출력될 문자열을 지정하는 것만으로도 사용 가능하니까요. 하지만 간단하다고 해서 확장성이 부족한 것은 아닙니다.

TextView는 View 클래스에서 상속받은 속성과 TextView 자신의 속성을 포함하여 아주 많은 종류의 속성을 제공합니다. 이러한 속성들을 사용하여 기본적으로 단순한 TextView에 아주 많은 확장성을 부여할 수 있는 것이죠.

이제 TextView를 화면에 추가하여 표시하는 방법에 대해 알아보겠습니다.

2. TextView 기본 사용법.

TextView를 사용하여 텍스트를 출력하는 작업은 아주 간단합니다. 하지만 TextView를 마음대로 다루는 건 쉽지 않습니다. 일단 많은 수의 속성들이 존재하고, 폰트 설정 등의 시스템 설정에 따라 스타일이 영향을 받기도 하며, 기기에 따른 Layout에 많은 신경을 써줘야 하는 위젯이기 때문이죠. 그래도 몇 번의 시행착오를 거치다보면 금방 익숙해질 것입니다.

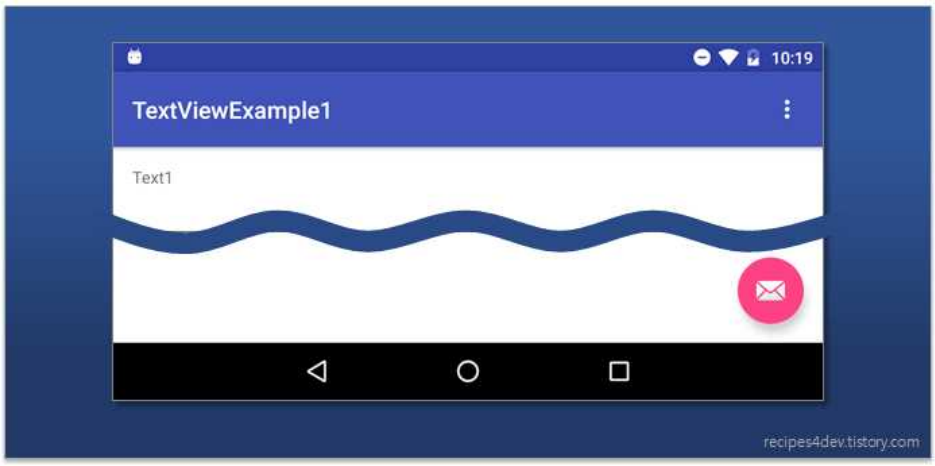
2.1 Layout 리소스 XML에 TextView 추가

TextView를 화면에 표시하기 위해서는 Layout 리소스 XML에 TextView를 추가하면 됩니다. 다음은 TextView를 Activity의 Layout 리소스 XML에 추가한 내용입니다.

[STEP-1] "activity_main.xml" - Layout 리소스 XML에 TextView 추가.

```
<?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 "
    android :paddingBottom="@dimen/activity_vertical_margin "
    android :paddingLeft="@dimen/activity_horizontal_margin "
    android :paddingRight="@dimen/activity_horizontal_margin "
    android :paddingTop="@dimen/activity_vertical_margin "
    app :layout_behavior="@string/appbar_scrolling_view_behavior "
    tools :context="com.recipes4dev.examples.textviewexample1.MainActivity "
    tools :showIn="@layout/activity_main ">

    <TextView
        android :layout_width="wrap_content "
        android :layout_height="wrap_content "
        android :id="@+id/text1 "
        android :text="Text1 " />
</RelativeLayout>
```

여기까지만 작성하고 실행해도 화면에는 문자열 텍스트가 출력되죠. 이렇듯 화면에 문자열을 출력하기 위해 TextView를 사용하는 방법은 매우 간단합니다. 사실 이 정도 코드는 안드로이드 스튜디오에서 프로젝트를 생성할 때 "Blank Activity"를 선택하면 기본적으로 생성되는 코드 수준이니 기본 중의 기본이라 할 수 있습니다.



TextView에 텍스트를 출력할 때, "text" 속성에 문자열을 직접 지정하는 방법 외에 "/res/values/strings.xml"에 저장된 "string" 리소스를 참조하는 방법도 있습니다. 특히, 구글에서는 다국어 지원 등의 이유로 "string" 리소스를 참조하는 방법을 권장하고 있습니다. "string" 리소스를 참조하는 방법에 대해서는 다른 주제를 통해 설명을 하겠습니다.

2.2 Java 소스에서 TextView의 텍스트 변경

앱 실행 시 처음 보여질 텍스트를 지정하기 위해 "text" 속성을 사용하였는데, 만약 앱 실행 중 TextView의 텍스트를 변경하려면 어떻게 해야 할까요? 바로 Layout 리소스 XML에 추가한 TextView에 대한 참조를 획득한 다음, setText() 함수를 호출하는 것입니다.

[STEP-2] "MainActivity.java" - onCreate() 함수에서 TextView의 텍스트 변경

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // ... 코드 계속
        TextView textView1 = (TextView) findViewById(R .id .text1) ;
        textView1 .setText("Text is changed.") ;
    }
}
```

2.3 TextView 텍스트 크기 변경

TextView를 사용할 때 가장 많이 수행하는 작업 중 하나가 바로 TextView에 출력되는 텍스트의 크기를 변경하는 것입니다. TextView의 쓰임에 따라 텍스트 크기를 다양하게 변경할 필요가 있는데, 이 때 "textSize" 속성을 사용합니다.

* android:textSize - 텍스트의 크기 지정.
    > 치수(dimension) 단위 사용 가능. px, dp, sp, in, mm
    > sp(scaled pixel) 단위 사용을 권장. (예. 15sp, 14.5sp, ...)
    > 관련 함수 : setTextSize()

[STEP-3] "activity_main.xml" - TextView의 텍스트 크기 변경

```
<TextView
    android :layout_width="wrap_content "
    android :layout_height="wrap_content "
    android :id="@+id/text1 "
    android :text="Text1 "
    android :textSize="20sp " />
```

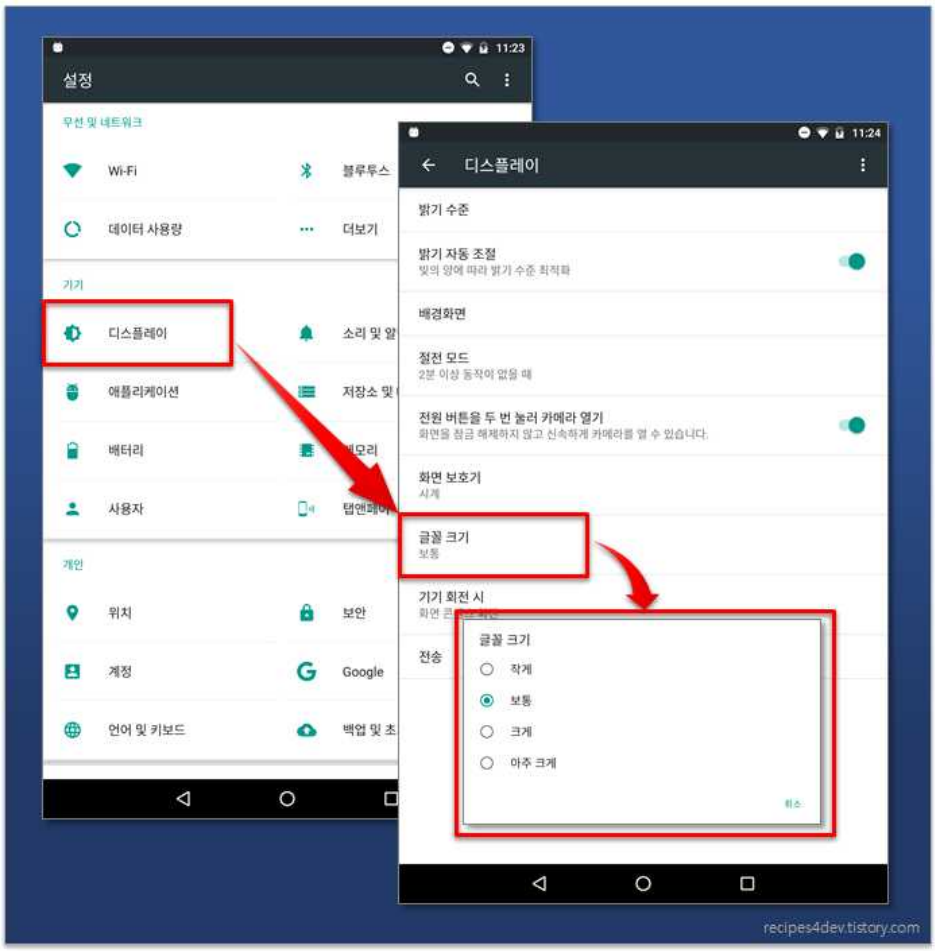
설명에 나와 있듯이 "textSize" 속성 값에는 여러 가지 단위를 사용할 수 있습니다. 어떤 단위의 값을 사용할 것인지는 개발자의 선택이지만, 구글에서는 공식적으로 치수(dimension)와 관련된 곳에는 dp(density-independent pixel)와 sp(scaled pixel)를 사용하도록 권장(recommand)하고 있습니다. 특히, 텍스트 출력과 관련된 단위는 sp를 사용하도록 권장하는데 이유는 시스템 설정에 따른 가독성 때문입니다.

다음 기회에 다른 글을 통해 dimension과 관련한 이슈를 설명하겠지만, 간단히 언급하자면 dp는 현재 사용 중인 디바이스에 의해 결정되지만, sp는 시스템 설정(글꼴 크기)에도 영향을 받습니다. 즉, sp의 "scaled"라는 의미는 디바이스에 의해 결정되는 절대 값이 아니라 시스템 글꼴에 따라 비례한(scaled) 값이 적용된다는 것이죠.

이해가 잘 안되신 다구요? 예제를 통해 sp 단위가 어떻게 적용되는지 확인해보겠습니다.

2.3.1 시스템 글꼴 크기 설정

먼저 sp 단위에 영향을 주는 시스템 글꼴 크기는 넥서스 7 2세대 기기를 기준으로 [설정]-[디스플레이]-[글꼴 크기]에서 설정할 수 있습니다.

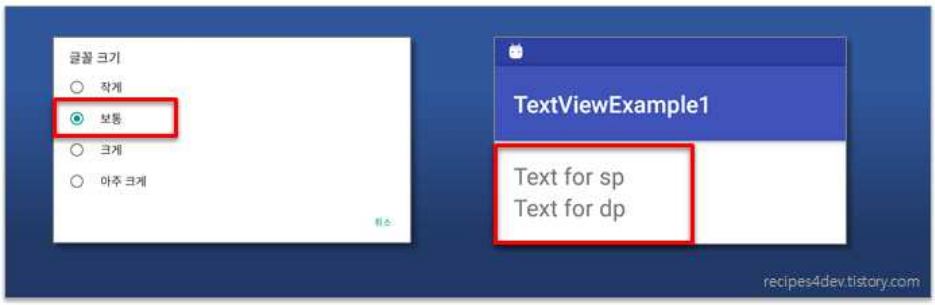


2.3.2 sp와 dp 사용에 따른 출력 결과.

sp와 dp가 시스템 글꼴 크기에 어떻게 영향을 받는지 알아보기 위해 TextView 두 개를 배치한 다음, textSize 값을 각각 "20sp"와 "20dp"로 지정하겠습니다.

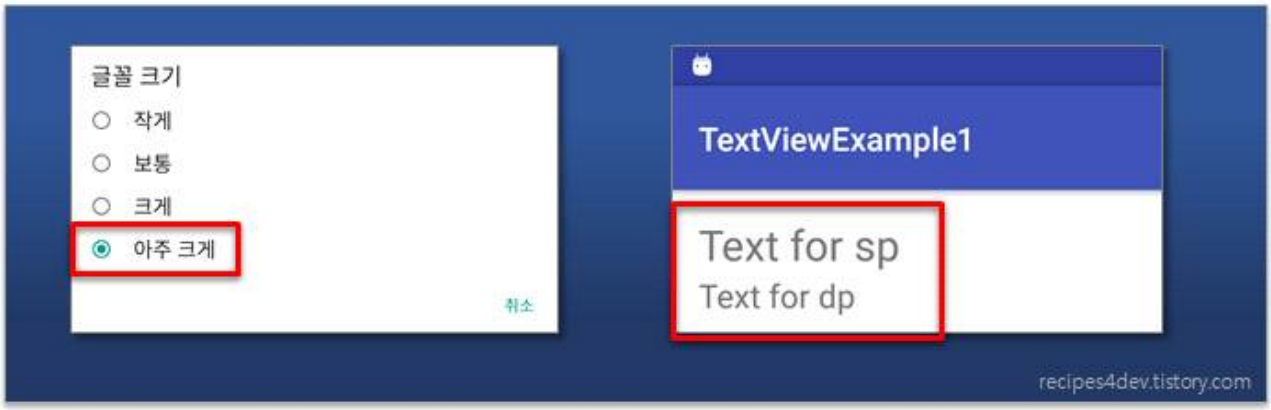
```
<TextView
    android :layout_width="wrap_content "
    android :layout_height="wrap_content "
    android :id="@+id/text1 "
    android :text="Text for sp "
    android :textSize="20sp " />

<TextView
    android :layout_width="wrap_content "
    android :layout_height="wrap_content "
    android :id="@+id/text2 "
    android :text="Text for dp "
    android :layout_below="@id/text1 "
    android :textSize="20dp " />
```

먼저 시스템 글꼴 크기를 "보통"으로 설정했을 때의 출력 결과를 확인하겠습니다.



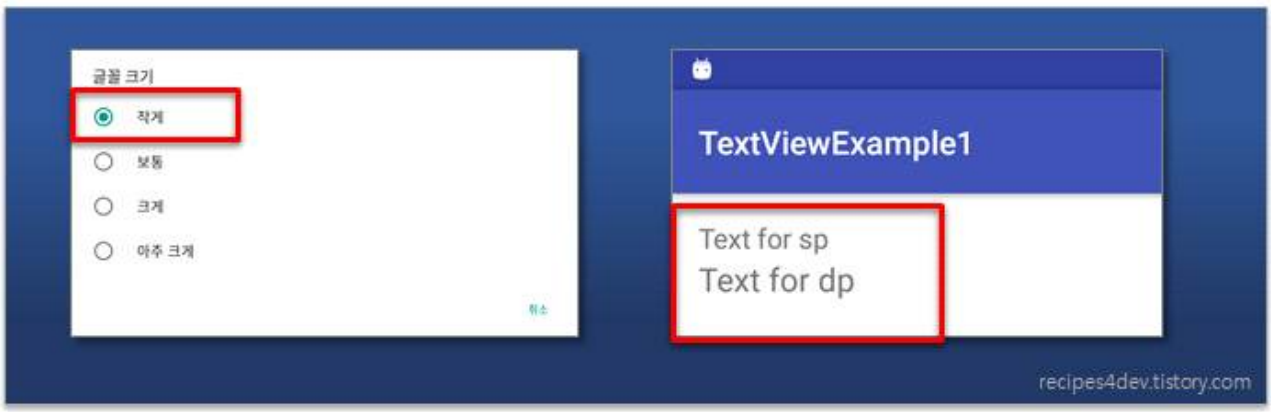
결과에서 알 수 있듯이 sp로 사용한 경우와 dp로 사용한 경우 같은 크기로 출력됩니다.

이제 글꼴 크기를 "아주 크게"로 설정한 다음 출력 결과를 확인하겠습니다.



dp를 사용한 TextView는 이전 결과와 동일하지만, sp를 사용한 TextView는 글꼴 크기 설정에 따라 텍스트가 커진 것을 확인할 수 있습니다.

마지막으로 "작게"로 설정한 다음 출력 결과를 확인하겠습니다.



마찬가지로 dp를 사용한 TextView는 "보통"으로 설정한 결과와 동일하지만, sp 단위를 사용한 TextView의 텍스트는 작게 표시되는 것을 확인할 수 있습니다.

2.3.3 sp 단위를 사용해야 하는 이유.

그런데 이러한 sp는 왜 사용해야 하는 것일까요? 그냥 dp만 사용하면 안될까요? dp 단위를 이해하는

것만도 충분히 복잡한데 말이죠.

가장 큰 이유는 텍스트의 가독성 때문입니다. 보통 화면의 글자들이 작아서 잘 안보이는 사람들을 위해, 안드로이드는 시스템 전체 글꼴 크기를 "크게 또는 "아주 크게"로 변경할 수 있도록 만들었습니다. 반대로 화면의 글자들이 작은 것을 선호하는 사람들을 위해 기본 크기에 비례하여 "작게"로도 변경할 수 있도록 해놓았습니다.

sp를 적용하면 글꼴 크기 설정에 따라 텍스트 크기가 자동으로 조절(scaled)되지만 dp는 이런 설정에 영향을 받지 않죠. 그래서 TextView에는 sp를 사용하길 권장하는 것입니다. (당연하게도 시스템 글꼴 크기 설정과 관계없이 일정한 크기의 텍스트가 출력되길 바란다면 dp 단위를 사용하면 되겠죠.)

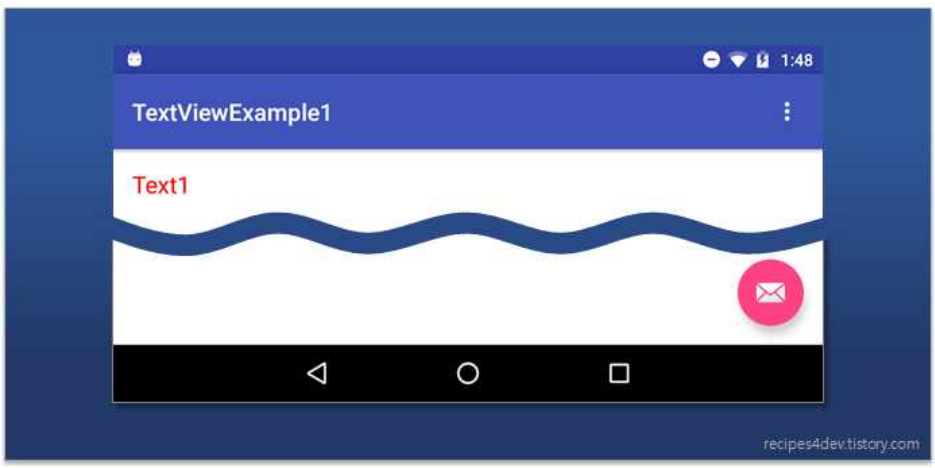
2.4 TextView 텍스트 색상 변경

TextView에 출력되는 텍스트의
색상을 변경하기 위해서는 "textColor" 속성을 사용합니다.

* android:textColor - 텍스트의 색상 지정.
    > 컬러 값 지정. (#rgb, #argb, #rrggbb, #aarrggbb 등의 형식)
    > 관련 함수 : setTextColor()

```
<TextView
    android :layout_width="wrap_content "
    android :layout_height="wrap_content "
    android :id="@+id/text1 "
    android :text="Text1 "
    android :textSize="20sp "
    android :textColor="#FF0000 "/>
```

"#FF0000"은 RGB 순에 따라 빨간색으로 표시되며, 출력 결과는 아래와 같습니다.



3. 참고.

- TextView 속성에 대한 자세한 도움말.
  - [안드로이드 개발 참조문서 TextView 항목] 내용을 참고하세요.
- DIP(Density Independent Pixel) 사용에 대한 구글의 권장 사항.
  - [안드로이드 개발 참조문서. Supporting Different Screen Densities 항목]에서 "DIP(Density Independent Pixel) 사용" 내용을 참고하세요.

4. 출처

이 문서는 아래 블로그 문서를 캡춰하여 정리한 것입니다.
블로그 주소 : http://recipes4dev.tistory.com/
원본 문서 주소 : 안드로이드 텍스트뷰 기본 사용법 (http://recipes4dev.tistory.com/56)