● 리소스 종류: 안드로이드 앱의 리소스들은 모두 res 폴더 하위에 있어야 하며, 리소스별 폴드명이 지정되어 있다. 각 폴더에 리소스 파일을 추가하는 순간, 추가한 리소스를 식별하기 위한 int형 변수가 R.java 팡리에 추가된다. 이때 파일명을 변수명으로 사용하므로, 리소스 파일명은 자바 명명규칙을 위배할 수 없다. 또한, 리소스 파일명에는 대문자를 사용할 수 없다.

drawable	이미지, 이미지와 관련된 XML, 그림을 표현한 XML
layout	화면 UI를 정의한 레이아웃 XML
values	문자열, 색상, 크기 등 여러 가지 값
menu	액티비티의 메뉴를 구성하기 위한 XML
xml	특정 폴더가 지정되어 있지 않은 기타 XML
anim	애니메이션을 위한 XML
raw	바이트 단위로 직접 이용되는 이진 파일
mipmap	앱 아이콘 이미지

● values: 문자열, 배열, 색상, 크기, 스타일, 정보 등 흔히 값이라고 표현되는 리소스는 values 폴더 하위에 위치한다.

strings.xml	문자열 리소스 여러 개를 담는 파일. 파일 내에 <String> 태그로 각 리소스 등록
colors.xml	색상 리소스 여러 개를 담는 파일. 파일 내에 <color> 태그로 각 리소스 등록
styles.xml	스타일 여러 개를 담는 파일. 파일 내에 <style> 태그로 각 리소스 등록
arrays.xml	배열 리소스 여러 개를 담는 파일. 파일 내애 <string-array>, <integer-array> 태그로 각 리소스 등록
dimens.xml	크기 리소스를 담는 파일. 파일 내의 <dimen> 태그로 각 리소스 등록

▶ dimens: 크기. 등록한 크기 리소스는 R.java 파일에 등록된 dimen 변수를 이용하여 사용한다.

```
<resources>
    <dimen name="my_margin">16dp</dimen>
    <dimen name="my_padding">16dp</dimen>
<resources>
```

자바 코드에서 이용하려면 R.dimen.my_padding처럼 사용하면 되고, xml 파일 내에서 속성 값으로 dimen 리소스를 사용하려면 다음과 같이 하면 된다.

```
<Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Button1"
  android:padding="@dimen/my_padding"
  android:layout_margin="@dimen/my_margin" />
```

▶ colors: 색상.

```
<resources>
  <color name="my_background">#FFFF0000</color>
  <color name="my_textColor">#FF00FFFF</color>
</resources>
```

이렇게 등록된 색상 리소스는 R.java파일에 등록된 color 변수를 이용하여 사용한다. 자바 코드에서 이용하려면 R.color.my_background처럼 사용하면 되고, xml 파일 내에서 속성 값으로 dimen 리소스를 사용하려면 다음과 같이 하면 된다.

```
<Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Button1"
  android:background="@color/my_background"
  android:textColor="@color/my_textColor" />
```

▶ styles: 스타일 리소스. 여러 뷰를 레이아웃 XML에 명시하다 보면 같은 속성이 계속 반복되는 경우가 있다. 예를 들면 모든 TextView에 textSize, textColor, textStyle이 같은 값으로 적용되어야 하는 경우이다. 이처럼 같은 속성이 중복될 때 스타일 리소스를 이용하면 중복을 피할 수 있다. 즉, 스타일 리소스는 여러 속성을 하나의 스타일로 묶어 필요한 곳에 적용하기 위해 사용된다.

```
<style name="myStyle">
  <item name="android:textColor">#FF0000FF</item>
  <item name="android:textSize">20dp</item>
  <item name="android:textStyle">bold</item>
</style>
```

정의한 스타일을 레이아웃 XML 파일에서 style 속성을 이용하여 지정해 주면, 3개의 속성이 한꺼번에 적용한 효과가 나타난다.

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="First"
  style="@style/myStyle" />
```

스타일을 정의할 때, 다른 스타일을 상속받아 재정의할 수도 있다. mySubStyle은 parent인 myStyle을 상속 받아 정의하였는데, 이렇게 하면 myStyle의 내용이 그대로 계승되며, mySubStyle에 새로운 내용을 추가하거나, 기존의

것을 바꾸어 사용할 수 있다. mySubStyle에는 TextColor, textSize는 myStyle 내용을 그대로 계승하고, textStyle만 italic으로 바꾼다.

```
<style name="myStyle">
  <item name="android:textColor">#FF0000FF</item>
  <item name="android:textSize">20dp</item>
  <item name="android:textStyle">bold</item>
</style>

<style name"mySubStyle" parent="myStyle">
  <item name="android:textStyle">italic</item>
</style>
```

▶ theme: 테마 리소스. 액티비티 전체 혹은 앱 전체를 위한 스타일(테마 theme)로, values의 하위 폴더인 themes 폴더 아래에 기본적으로 생성된, 테마가 등록된 themes.xml 파일이 있다. <style> 태그로 정의한다. 기본적으로 라이브러리에서 제공하는 테마를 상속받아 Theme.파일이름 이라는 이름의 스타일이 하나 정의되어 있고, 이 테마가 액티비티에 적용되어 있는 것이다.

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.PracticeEventHandling" parent="Theme.MaterialComponents.DayNight.
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariar
        <!-- Customize your theme here. -->
    </style>
</resources>
```

테마 적용은 뷰를 위한 스타일이 아니므로 레이아웃 XML 파일이 아닌 액티비티가 등록되는 AndroidManifest.xml (app > manifests > AndroidManifest)에 설정한다. 액티비티의 <application> 태그에 theme 설정이 themes.xml에 선언한 "Theme.파일이름"로 되어 있어서, 애플리케이션의 모든 액티비티 툴바 색상이 Theme.파일이름에서 지정한 색상으로 나오게 된다.

```
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PracticeEventHandling"
        tools:targetApi="31">
        ...
    </application>

</manifest>
```

화면을 여러 개의 영역으로 구분해 색상을 부여하기 위해, 영역마다 이름이 지정되어 있다. colorPrimary와 colorSecondary는 앱의 브랜드를 표현하기 위한 색상이다. colorPrimary는 툴바와 버든의 백그라운드 색상으로 적용되며, colorSecondary는 체크박스, 라디오버든, 스위치 등의 활성 상태 색상이다. 그리고 statusBarColor는 상태바의 백그라운드 색상으로 사용된다. 또한 colorOnPrimary와 colorOnSecondary는 colorPrimary와 colorSecondary가 적용되는 곳의 포그라운드 색상(전경색)으로 적용된다. 그리고 colorPrimaryVariant와 colorSecondaryVariant는 그림자 색상으로 사용한다. 해당 값을 변경하는 것만으로도 액티비티의 색상을 지정할 수 있다. 이렇게 정의한 테마를 <application> 태그에 적용하면 애플리케이션 내의 모든 엑티비티에 같은 테마가 적용된다. 애플리케이션 내에 엑티비티가 여러 개 있을 때, 이 중 하나의 액티비티에만 적용해야 하는 테마도 있다. 이럴 때는 테마를 AndroidManifest.xml의 <activity> 태그에 설정하면 된다.

```
   <activity
      android:name=".MainActivity"
      android:exported="true"
      android:theme="@style/Theme.Woori2">
```

테마를 액티비티에 설정하면, 툴바의 색상 값뿐 아니라 툴바 전체가 화면에 안 나오게 할 수도 있다. NoActionBar을 상속받아서 아래처럼 설정하면 테마가 적용된 액티비티의 툴바 영역이 안 보이게 된다.

```
    <style name="Theme.Woori2" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">#E91E63</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">#E91E63</item>
        <!-- Customize your theme here. -->
    </style>
```

상태바(status bar)는 스마트폰의 맨 윗줄에 시간이 출력되는 부분이다. 이 영역이 안 보이게 하려면, 화면에 뜨는 액티비티를 이 영역까지 차지하게 하면 된다. 이 또한 테마 설정으로 적용할 수 있다.

```
<style name="Theme.Woori2" parent="Theme.MaterialComponents.DayNight.NoActionBar">
    ...
    <item name="android:windowFullscreen">true</item>
</style>
```

액티비티를 가로 혹은 세로 방향으로 고정해서 사용자가 방향을 전환할 수 없도록 하는 것 또한 AndroidManifest.xm l파일의 〈activity〉 태그 속성 설정으로 할 수 있다. android:screenOrientation="landscape"로 지정하면 가로 방향으로 고정되고, android:screenOrientation="portrait"로 지정하면 세로 방향으로 고정된다.

Resources: 깡쌤의 안드로이드 프로그래밍