안드로이드 스마트폰의 크기가 다양하기 때문에, 여러 크기에 호환하는 UI를 작성하려면 리소스 폴더명에 대한 조건으로 여러 환경에 대응하는 앱을 만들 수도 있고, 개발자가 직접 화면 정보를 추출해서 여러 환경에 대응하게 작성할 수도 있다.

● 리소스 폴더명 조건 명시법: 개발자가 리소스 폴더명만 적절한 규칙에 따라 작성해 놓으면, 시스템에서 어떤 환경의 기기에 어떤 리소스를 적용할 것인지 스스로 결정해 준다.

가장 대표적인 것이 각 스마트폰의 화면 해상도(DPI)에 따라 앱 아이콘 이미지를 달리 보여주는 부분이다. 모듈의 res 폴더를 보면 mipmap폴더가 있는데, 이 폴더의 구조는 다든 리소스 폴더와는 다르다. ic_launcher.png 파일 옆에 (6)이라는 숫자가 보이고, 여섯 개의 파일이 보이며 각 파일명 옆에 (hdpi) 같은 단어가 보인다. 실제 윈도우 탐책 창에서 봤을 때는 mipmap 폴더가 6 개 있으며 mipmap에 대시(—)가 추가되어 그 뒤에 "hdpi", "mdpi" 등이 덧붙여진 폴더명이 보인다. 폴더명이 대시(-) 뒤 단어는 기기 환경에 대한 조건이다. 대시 (-) 뒤의 폴더명을 조건으로, 시스템에서 기기 환경에 맞는 리소스를 어느 폴더의 리소스로 사용할 것인지 스스로 결정한다는 의미이다. 각 mipmap 폴더에는 같이 이름의 리소스 팡ㄹ이 있는데, 이름은 같지만 크기는 각각 다든 파일이다. 이름이 같아서 R.java 팡리에는 리소스 식별자 변수가 하나로 등록된다. 따라서 코드에서는 R.drawable.ic_launcher라고 지정만 해주면 된다. 그럼 시스템에서 알아서 hdpi 스마트폰에는 mipmap-hdpi폴더 밑의 파일을 적용해 주고, xhdpi 스마트폰에서는 mipmap-xhdpi 폴더의 파일을 적용해 준다. 즉, 대시(-)뒤 단어는 기기의 조건을 의미하는 것이다.

리소스 폴더명에 대시(-)로 기기의 조건을 주면 이에 따든 리소스 적용을 시스템에서 알아서 적용해주는 구조이다. 폴더명에 지정할 수 있는 조건은 다음과 같다.

조건	설명	예시
MCC, MNC	모바일 국가 코드, 모바일 네트워크 코드 조건.	mcc310 (미국), mcc310-mnc004(미국 버라이즌)
Language, region	언어 및 지역 조건. 언어는 ISO 639-1의 두 글자, 지역은 ISO 3166-1-alpha-2 코드의 두 글자로 r(소문자)로 시작	en, en-rUS, fr-rFR
Layout Direction	레이아웃 방향 조건. 아랍어 혹은 히브리어 등을 위한 오든쪽에서 왼쪽 형태의 레이아웃을 적용 시 유용하게 이용.	ldrtl은 오든쪽에서 왼쪽, ldltr은 왼쪽에서 오든쪽.
smallestWidth	화면 크기에 대한 조건 중 smallestWidth에 대한 조건. smallestWidth는 화면의 높이와 너비의 짧은 쪽에 대한 조건을 의미하며, 화면 방향과는 관계없음.	sw320dp - 너비든 높이든 화면 방향과 상관없이 작은 쪽의 치수가 320인 경우
Available width	화면의 너비에 대한 조건	w720dp
Available height	화면의 높이에 대한 조건	h720dp
Screen size	화면의 크기를 small, normal, large, xlarge emdd로 나누어 조건 명시.	small은 320x426, normal은 320x470, large는 480x640, xlarge는 720x960 정도의 크기.
Screen aspect	화면의 종횡비 조건.	long은 WQVGA, WVGA,

		FWVGA 같은 긴 화면, notlong은 QVGA, HVGA, VGA 같은 길지 않은 화면.
Screen orientation	화면의 방향 조건. Port는 세로 방향, land는 가로 방향.	
UI mode	기기가 독(dock)에서 추가되거나 제거될 때 대응할 수 있는 방법의 조건. Car은 자동차, desk는 데스크, television은 텔레비전, appliance는 표시되지 않은 제품.	
Night mode	야간 모드에 대응할 수 있는 조건. Night은 야간, nonight는 주간	
Screen pixel density(dpi)	화면 밀도에 대한 조건.	ldpi는 ~120dpi, mdpi는 ~160dpi, hdpi는 ~240dpi, xdpi는 ~320dpi, xxhdpi는 ~480dpi, xxxhdpi는 ~640dip.
Touchscreen type	터치 스크린 지원 여부 조건. Notouch는 터치스크린을 지원하지 않는 기기, finger은 터치스크린 지원하는 기기.	
keyboard availability	키보드에 대한 조건. Keysexposed는 키보드가 노출된 경우, keyshidden은 키보드가 비활성화 상태, keysoft는 소프트 키보드 활성 상태.	
primary text input method	키보드 타입에 대한 조건. Nokey는 하드웨어 키가 없는 기기, gwerty는 하드웨어 쿼터 키보드를 가지는 기기, 12key는 12키 하드웨어 키보드를 가지는 기기.	
Navigation key availability	UI 탐식 키 사용 유무에 대한 조건. Navexposed는 탐색 키를 사용할 수 있는 상황, navhidden은 탐색 키를 사용할 수 없는 상황.	
Primary non-touch navigation method	탐색 키를 지원하는지에 대한 조건. Nonav는 터치 스크린 이외 다든 내비게이션 기능이 없는 경우, dpad는 탐색을 위한 방향 패드가 있는 경우. Trackball은 탐색을 위한 트랙볼에 있는 경우, whell은 탐색을 위한 방향 휠이 있는 경우.	
Platform Version(API level)	기기의 API레벨에 대한 조건.	v7는 API레벨 7 이상의 기기.

또한 위의 조건들을 폴더명에 대시를 이용해 명시할 때 지켜야 할 규칙은 다음과 같다.
▶ 대시로 구분하여 하나의 폴더에 여러 조건을 명시할 수 있다. ex) drawable-en-rUS-land. 이렇게 여러 개의 조건을 하나의 폴더에 명시하면 여러 조건을 모두 만족할 때 해당 폴더의 리소스가 적용된다. And 연산으로 적용되며, Or 연산은 제공하지 않는다.
▶ 대시로 구분하여 하나의 폴더에 여러 조건을 명시할 때, 조건의 순서가 있다. 위에서 설명한 순서가 조건이 나열되는 순서이다. 예를 들면 drawable-hdpi-port는 가능하지만, drawable-port-hdpi는 불가능하다.
▶ 폴더에 서브 폴더로 조건을 세분화하는 건 불가능하다. 예를 들면, res/drawable/drawable-en 같은 폴더 세분화는 불가능하다.
▶ 조건에 대한 대소문자는 구분하지 않는다. 대문자는 가독성을 위한 것일 뿐이다.
▶ 하나의 폴더에는 각 조건마다 하나의 값만 명시할 수 있다. 예를 들면, 언어에 대한 조건을 스페인어와 프랑스어에 대한 조건을 주고 싶을 때 폴더를 두 개 만들어야지, 하나의 폴더에 같은 조건인 언어에 대한 조건을 두 개 줄 수 없다.

◆ 국제화 표현: 앱이 여러 국가 사용자들을 대상으로 만들었다면 여러 언어를 지원해야 하고, 이 부분을 국제화 표현이라고 하는데, 리소스 폴더명 조건으로 해결한다. values 폴더 하위에 strings.xml 파일에 있고, values-ko-rKR 폴더 하위에도 strings.xml 파일에 있다.

```
// values/strings.xml

<string name="hello">Hello!!</string>
```

```
// values-ko-rKR/strings.xml

<string name="hello">안녕하세요!!</string>
```

이렇게 등록 후 자바 코드나 레이아웃 XML 파일에서 R 파일에 등록된 int 변수 hello를 이용하면, 시스템에 알아서 어느 폴더의 리소스를 이용할지 결정하게 된다.
▶ strings.xml파일 만들면서 폴더명 조건 추가해 주기: values 폴더 > New > Values Resources File > File name: strings, Locale - ko: Korean - KR: South Korea
▶ drawable-ko-rKr 폴더 만들기: res > New > Android Resource Directory > name: drawable, resource type: drawable, Locale, ko-rKR. 파일 복사 시 경로를 drawable-ko-rKr로 지정해 주면 됨.

◆ 가로/세로 레이아웃: 사용자 스마트폰의 방향이 가로일 때와 가로일 때 레이아웃을 교체해서 보여주기 위해서는 레아이웃 XML를 교체해서 적용해야 하고, 이때도 layout 폴더에 대시(-)를 이용해 방향에 대한 조건을 명시해 준다. layout 폴더와 layout-land 폴더가 있고, 두 폴더에 같은 이름의 activity_main.xml 파일이 있으면, 세로 방향에서는 layout/activity_main.xml 파일이, 가로 방향에서는 layout-land/activity_main.xml 파일이 적용된다.

● WindowMetrics: 스마트폰의 사이즈를 개발자가 직접 코드로 획득해야 할 때 사용한다. 이때, 코드로 스마트폰의 사이즈 정보를 획득하는 방법은 API Level 30 버전과 이전 버전에 차이가 있다. 30 이전 버전에서 이용하던 DcisplayMetrics 객체의 많은 기능을 30버전부터 지원하고 있지 않기 때문이다. 30버전부터는 WindowMetrics 객체를 이용할 것을 권장한다. 아래 코드는 버전을 고려하여 스마트폰의 가로세로 크기를 획득하는 코드이다.

```
// Build.VERSION.SDK_INT는 앱이 실행되는 기기의 버전 정보이며,
// Build.VERSION_CODES.R은 안드로이드 11 버전, 즉 API Level 30 버전을 의미한다.

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
    WindowMetrics windowMetrics = getWindowManager().getCurrentWindowMetrics();
    int width = windowMetrics.getBounds().width();
    int height = windowMetrics.getBounds().height();
    binding.textView.setText("width : "+width+", height : "+height);
} else {
    Display display = getWindowManager().getDefaultDisplay();
    DisplayMetrics displayMetrics = new DisplayMetrics();
    display.getRealMetrics(displayMetrics);
    int width = displayMetrics.widthPixels;
    int height = displayMetrics.heightPixels;
    binding.textView.setText("width:"+width+", height:"+height);
}
```

● 논리적 단위로 스마트폰 크기 화환성 확보: 크기를 명시할 때 논리적인 단위를 이용하여 스카트폰 크기의 다양성에 대응해야 한다. 앱을 개발할 때 화면에 출력되는 뷰들의 크기와 문자열 크기를 직접 명시해야 하는 경우가 있는데, 이때 크기는 px, dp, sp, in, mm등 다양한 단위로 사용할 수 있다. 단, 안드로이드에서는 dp를 권장하며, 문자열 폰트의 크기는 sp를 권장한다. px로 물리적인 크기를 부여하는 반면 dp 혹은 sp로는 시스템에서 스마트폰의 화면 dpi에 따라 논리적인 크기를 계산해서 적용하므로, 다양한 화면 크기에 호환하는 프로그램을 작성할 수 있기 때문이다.

dp(dip)	Density-Independent Pixels. 스크린의 물리적 밀도에 기초한 단위. 160dpi(dots per inch)에 상대적 수치. 시스템에서 120dpi를 1dpi 스마트폰으로 식별하고, 160dpi = mdpi, 240dpi = hdpi, 320dpi = xhdpi, 480dpi = xxhdpi, 640dpi = xxxhdpi로 식별하여 크기를 적절하게 줄이거나 늘려준다. 계산 로직은 mdpi를 기준으로 한다. mdpi 스카트폰에서는 1dp가 1px을 차

	지한다. ldpi 스카트폰에서는 1dp는 0.75를 곱해서 크기가 감소하며, hdpi 스마트폰에서는 1dp에 1.5를 곱하므로 크게 적영되며, xhdpi 스카트폰에서는 1dp로 주면 2.0를 곱해서 적용한다.
sp(sip)	Scale-independent Pixels. dp와 유사하며 폰트 크기에 적용.
pt	Points. 화면 크기의 1/72를 1pt
px	픽셀
mm	밀리미터
in	인치

dp는 논리 단위로 스마트폰 크기 호환성에 도움을 받을 수 있지만, 이는 레이아웃 XML 파일에서 크기를 명시할 때 사용할 수 있는 단위이지, 자바 코드에서 크기가 명시되는 경우에는 무조건 px 단위로만 지정할 수 있다. 이때 스마트폰 크기 호환성이 문제될 수 있기 때문에, 개발자가 WindowMetrics 혹은 DisplayMetrics 클래스로 스마트폰의 dpi를 판단한 후 직접 계산해주어 hdpi에서는 조금 크게, ldpi에서는 조금 작게 나오게 조정해 주어야 한다. 그런데 만약 이 부분에 귀찮다면, dimen 리소스를 이용하면 된다. dimen 리소스 XML 파일에는 dp 단위로 주고 개발자 코드에서 getDimensions() 함수로 획득하면, 알아서 스마트폰의 dpi 값에 따든 px로 환산된 결괏값이 넘어와서 편하게 이용할 수 있다.

```
// string과 image 설정 후. 스마트폰의 로케일(locale)이 ko이면 태극기와 한국어를,
// en이면 성조기와 영어로 출력한다.

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/flag" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/note" />

</LinearLayout>
```

Resources: 깡쌤의 안드로이드 프로그래밍