**Resource Sets and Configurations**

Devices sometimes change while users are using them, in ways that our application

will care about:

* The user might rotate the screen from portrait to landscape, or vice versa
* The user might put the device in a car or desk dock, or remove it from such a dock
* The user might put the device in a “netbook dock” that adds a full QWERTY keyboard, or remove it from such a dock
* The user might switch to a different language via the Settings application returning to our running application afterwards
* And so on

**how to provide alternative resources for these different**

**scenarios — called “configuration changes”**

Different pieces of Android hardware can have different capabilities, such as:

* Different screen sizes
* Different screen densities (dots per inch)
* Different number and capabilities of cameras
* Different mix of radios (GSM? CDMA? GPS? Bluetooth? WiFi? NFC?
* something else?)

cameras and Bluetooth and WiFi are **features**

That is not the only scenario where resources might need to differ, though. Here are

others:

1. *Screen orientation*: is the screen in a portrait orientation? Landscape?
2. *Screen size*: is this something sized like a phone? A tablet? A television?
3. *Screen density*: how many dots per inch does the screen have? Will we need a

higher-resolution edition of our icon so it does not appear too small?

1. *Keyboard*: what keyboard does the user have (QWERTY, numeric, neither),

either now or as an option?

1. *Other input*: does the device have some other form of input, like a directional pad or click-wheel?

res/values/strings.xml.

res/values-en/ and res/values-es/,

<https://en.wikipedia.org/wiki/ISO_639-1>

**Screen Size and Orientation**

Here, “orientation” refers to how the

device is being held: portrait or landscape.

use -port or -land as resource set qualifiers

e.g., res/layout-land/)

**The Original: Android-Defined Buckets**

• -small for screens at or under 3” in diagonal size

• -normal for screens between 3” and 5” in diagonal size

• -large for screens between 5” and 10” in diagonal size

• -xlarge for screens at or over 10” in diagonal size

**Default Change Behavior**

When you call methods in the Android SDK that load a resource (e.g., the

aforementioned setContentView(R.layout.main)), Android will walk through those

resource sets, find the right resource for the given request, and use it.

But what happens if the configuration changes *after* we asked for the resource? For

example, what if the user was holding their device in portrait mode, then rotates the

screen to landscape? We would want a -land version of our layouts, if such versions

exist. And, since we already requested the resources, Android has no good way of

handing us revised resources on the fly… except by forcing us to re-request those

resources.

So, this is what Android does, by default, to our foreground activity, when the

configuration changes on the fly.

**Destroy and Recreate the Activity**

The biggest thing that Android does is destroy and recreate our activity. In other

words:

• Android calls onPause(), onStop(), and onDestroy() on our original

instance of the activity

• Android creates a brand new instance of the same activity class, using the

same Intent that was used to create the original instance

• Android calls onCreate(), onStart(), and onResume() of the new activity

instance

• The new activity appears on the screen

This may seem… invasive. You might not expect that Android would wipe out a

perfectly good activity, just because the user flicked her wrist and rotated the screen

of her phone. However, this is the only way Android has that guarantees that we will

re-request all our resources.

**Rebuild the Fragments**

If your activity is using fragments, the new instance of the activity will contain the

same fragments that the old instance of the activity does. This includes both static

and dynamic fragments.

By default, Android destroys and recreates the fragments, just as it destroys and

recreates the activities. However, as we will see, we do have an option to tell Android

to retain certain dynamic fragment instances — for those, it will have the new

instance use the same fragment instances as were used by the old activity, instead of

creating new instances from scratch.

Persisting State Information During Changes

in Configuration

So far, you have learned that changing screen orientation destroys an activity and re-creates it. Keep

in mind that when an activity is re-created, its current state might be lost. When an activity is killed,

it fires one or both of the following methods:

➤➤ onPause()—This method is always fired whenever an activity is killed or pushed into the

background.

➤➤ onSaveInstanceState()—This method is also fired whenever an activity is about to be

killed or put into the background (just like the onPause() method). However, unlike the

onPause() method, the onSaveInstanceState() method is not fired when an activity

is being unloaded from the stack (such as when the user pressed the back button) because

there is no need to restore its state later.