UNIVERSITY of WASHINGTON

Android 250 - Lecture 3

Multi-device Support, Styles & Themes

Margaret Maynard-Reid

April 13, 2015

Agenda

- Resource Qualifiers
- Supporting multiple devices
- Localization
- Styles & Themes

Sample Code

- SampleDevices
- SampleLoc
- SampleStyle

Android Stories

- The Next Nexus
- Android Wear on iPhone: How it might happen and what it'll mean
- Facebook's simple trick for serving so many different Android devices

Review From Last Week

- Give a few examples of resources
- Which Drawable type to use to indicate different states of a button?
- What is a 9-patch Drawable?
- What is the difference between /raw vs. /assets resources?

Resource Qualifiers

Resource Qualifiers

- Used to supply alternate resources, to satisfy
  - different conditions on different devices, or
  - different configuration on the same device
- A folder-to-file structure, qualified by hyphens, i.e. *res/layout-land*
- A Resource has the **same name**, but is foldered differently according to it's alternate use

Resource Qualifiers

An example of an .png icon in different drawable folders:

res/ drawable-hdpi/icon.png drawable-ldpi/icon.png drawable-mdpi/icon.png drawable-xhdpi/icon.png	• System will look for "@drawable/icon" • Current system conditions (i.e. what density is reported) will route to the appropriate Resource • Can access the current state via Configuration

Type of Qualifiers

On same device:

- Language
- Screen Orientation - portrait vs. landscape

On different device:

- Screen Size
- Screen Density
- API level…

Link to full list here:

http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

Size Resource Qualifiers

Old Style, there are 4 categories:

1. small >= 426dp x 320dp
2. normal >= 470dp x 320dp
3. large >= 640dp x 480dp
4. xlarge >= 960dp x 720dp

Size Resource Qualifiers

New Style, (Numeric Selectors) since API level 13+

- use the available screen width or height:

- **sw<N>dp** - Smallest Width
  - Does not change with screen orientation
- **w<N>dp** - Available Screen Width
  - Changes with screen orientation
- **h<N>dp** - Available Screen Height
  - Changes with screen orientation

Size Resource Qualifiers

New style example -

- /res/drawable-w480dp
- /res/drawable-w1280dp-h800dp
- /res/drawable-sw480dp
- /res/drawable-sw600dp ← 7” tablet
- /res/drawable-sw720dp ← 10” tablet

Density Resource Qualifiers

Densities

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi ← baseline
- hdpi (high) ~240dpi
- xhdpi (extra-high) ~320dpi
- xxhdpi (extra-extra-high) ~480dpi
- xxxhdpi (extra-extra-extra-high) ~640dpi

Version Resource Qualifiers

Applied only to a particular Android API Level

-v8, -v11, -v14, -v21

/res/drawable-xhdpi-v10/android_logo.png

/res/drawable-xhdpi-v14/android_logo.png

Alternate Resources

- Values are case-insensitive, **lower-case** by convention
- Order of Precedence
  - Strict order of precedence for qualifiers to be appended to the folder name
  - For example, values-mdpi-normal is wrong!
  - Refer to the Android documentation for the full list

http://developer.android.com/guide/topics/resources/providing-resources.html

Demo

- Create resources with qualifiers
- Android Studio Android project view
  - helps manage resources across resource sets

Break

Supporting Multiple Devices

Support Multiple Devices

Things to consider:

1. Screens sizes & density
    a. Modular UI
    b. Screen independence
2. Device Compatibility
    a. Hardware features
    b. API levels

Modular UI Design

Compose reusable UI components

- Use the <include> tag
- Use the <merge> tag
- Use fragments

Screen Independence

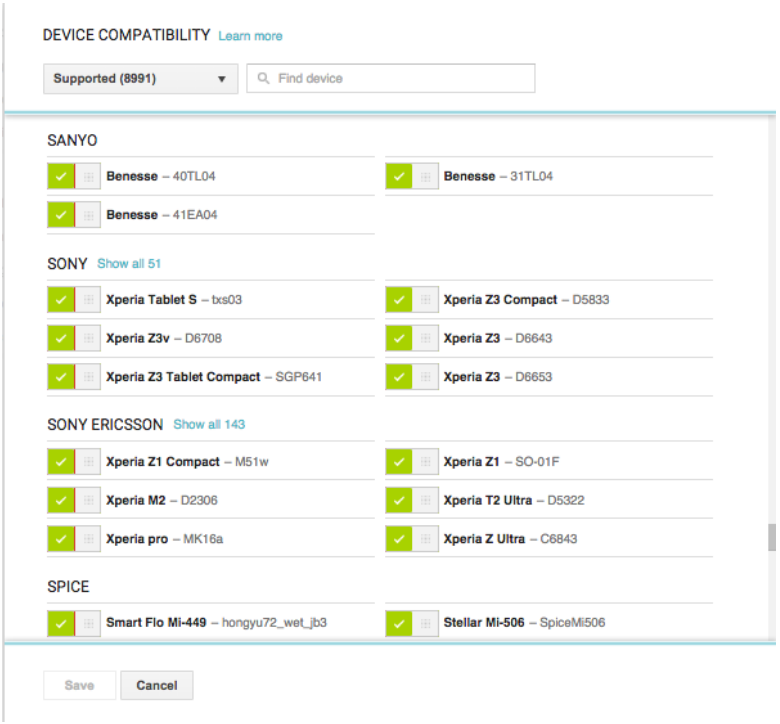
- Don't hardcode dimensions (px)
- Use *wrap_content*, *match_parent*, *dp*
- Use a LinearLayout if applicable
- Define different dimensions (dimens.xml)
- Use qualified layouts for certain screen sizes
- Use the proper drawable for each density
  - Use flexible drawables 9-Patch Drawables for pixel-perfect screens

Device Compatibility

- Google Play has 8991 targetable devices
- Your app can filter device support by
  - API levels
  - Features
  - Manual blacklist

How Does it Work?

- This layer of compatibility is specified in the manifest
- Every feature you add to your application potentially filters out incapable devices



Capability Independence

Add <uses-feature/> entries to AndroidManifest.xml

- Filter out devices missing required features
- Gracefully degrade unsupported features
  - Check for null before usage

http://developer.android.com/guide/topics/manifest/uses-feature-element.html

<uses-feature/> Example

```
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-feature android:name="android.hardware.camera.back" />
<uses-feature android:name="android.hardware.microphone" />
<uses-feature android:name="android.hardware.audio.low_latency" />
```

What might this application do?

What does this "say" and how might we need to handle the following?

```
<uses-feature android:name="android.hardware.camera" required="false" />
```

Tools

- Preview your UI on multiple devices, language, & API:

  Android Studio

  > Graphic Editor

  > Preview all screens / Remove previews

- Use emulators

Demo

- Walk through SampleDevices

Supporting Multiple Cultures

Language

- Localization (I10n)

– Linguistic and communication issues

- Internationalization (i18n)

– Legal and cultural issues

Why should we care?

Locale

Class that contains a language & country description

- ISO 639-1: specifies the language part
- ISO 3166-1 alpha-2: specifies the country part
- These look like:

  en_US, de_DE, fr_FR, en_CA, fr_CA, ...

Available Locales

- Use Locale.getDefault() to get the current Locale (for the *user* of the device)
- Locale.US is almost always available
- Use Locale.getAvailableLocales() to get an array of Locales on the device

Classes with Localization

- DateFormat

  DateFormat.getDateInstance(int style, Locale locale).format(theDate)
- NumberFormat

  NumberFormat.getNumberInstance(Locale locale).format(theNumber);
- Currency

Currency

- Class to represent a currency – Based on ISO 4217
- Offers methods to get the symbol and code
  - getInstance(Locale locale)
  - getInstance(String currencyCode)
  - getCurrencyCode()
  - getSymbol(Locale locale)
- Just use NumberFormat.getCurrencyInstance()

Localize Strings

- Create a new folder with language qualifier
- Create a strings.xml file
- Add localized strings in strings.xml

Demo

- Walk through SampleLoc

Break

Styles & Themes

Styles

- A way to style and centralize view attributes – defined in an XML file located under res/values/
  - XML file needs to contain a <resources> root
  - Style definitions are through <style> elements
- Styles are just another Resource
  - They, in turn, reference other resources
  - This also means they can be qualified...

Styles

- Can be applied to an Activity or a single View
  - When applied to the Application or an individual Activity, the group of styles are called a Theme
  - Themes are organized collections of Styles

Why Styles?

Isn't this just more referencing?

Can't I just edit a Button's background directly?

Yes,but Styles…

- Let you customize your own look and feel without having to change everything – only the things you want to change
- Ease global changes once a new style is defined

What can you style?

- You can style a View or a ViewGroup (Container)
  - All its XML attribute can be styled. i.e. TextView XML attributes
- When styling a Container such as a LinearLayout, the style will *not* be applied to its children automatically

How to Style?

1. Decide on **what** you want to style, i.e. a Button or EditText
2. Decide on **what attributes** to style, i.e. *android:textColor*
3. Create style in styles.xml. To i**nherit Android style**, find that resource reference in the Android style definition

   http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/5.0.2_r1/frameworks/base/core/res/res/values/styles.xml?av=f

4. Two ways to use the style:
   a. **Override** the reference in your **Theme**. Find the reference in current theme: http://developer.android.com/reference/android/R.attr.html
   b. Directly **apply the style** in your layout.xml, i.e.

      style="@style/someStyle"

Defining Styles

Defined Styles can inherit from one another

- Just an XML resource somewhat like CSS
- Properties defined in a parent are used by the children (and can be overridden)
- Uses a dot notation to indicate elements, if referencing your own style

http://developer.android.com/guide/topics/resources/style-resource.html

Applying Styles

- By XML
  - Use the android:theme attribute when applying it to an Activity or your Application
  - Use the style attribute when applying it to a View
- By Code
  - One of the rare cases where you can't do it via code
  - You can, however, inflate a view that has a certain style applied

Styles Example

Style Definition /res/values/styles.xml	Layout Definition /res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?> <resources> <style name="myLayout"> <item name="android:layout_width">match_parent</item> <item name="android:layout_height">wrap_content</item> </style> <style name="myButton"> <item name="android:layout_width">wrap_content</item> <item name="android:layout_height">wrap_content</item> <item name="android:textColor">#FF0000</item> </style> </resources>	*<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" style="@style/myLayout"> <Button style="@style/myButton" android:text="@string/cancel" /> </LinearLayout>*

Platform Styles

- Widget.ActionBar
- Widget.CompoundButton.RadioButton
- Widget.RatingBar
- TextAppearance.Holo.Widget.DropDownHint
- Holo.Light.SegmentedButton

https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/styles.xml

Platform Themes

Themes coalesce a group of styles – Theme

- Theme.Holo
- Theme.Light
- Theme.Holo.Light

Some Oddballs

- Theme.Black
- Theme.Translucent
- Theme.NoDisplay

https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/themes.xml

Platform Themes

What happens if you didn't specify a theme in the AndroidManifest.xml file?

API Level	**Default Theme**
< 11	Theme
11 - 20	Theme.Holo
> 20	Theme.Material

Material Theme

- Dark Version

  @android:style/Theme.Material (dark version)
- Light Version

  @android:style/Theme.Material.Light (light version)
- Dark ActionBar

  @android:style/Theme.Material.Light.DarkActionBar

Material Design Style

- Create style in values-v21 or use appcompat

<style name="AppTheme" parent="android:Theme.Material"></style>

- Material design colors

http://www.google.com/design/spec/style/color.html#color-color-palette

Styles Compatibility

- Use alternative styles.xml
  - Define base styles in res/values/styles.xml,
  - Create another layout under /res/values-v21/styles.xml
  - Then inherit from base styles in res/values-v21/.
- Use the Support Library:
  - v7 Support Libraries r21+

https://developer.android.com/training/material/compatibility.html

SampleStyle

- Walk through SampleStyle app
- See how styles and themes are used in Android

Reminder

- Homework 1 due next week April 20 @6PM.
- No late homework accepted