Mobile Application Development

Application Resources

APPLICATION RESOURCES

Application Resources

- Resources are the additional files and static content that your code uses, such as:
  - bitmaps,
  - layout definitions,
  - user interface strings,
  - and more.

Topics

Providing Resources

Where to save resources, and how to create alternative resources for specific device configurations.

Accessing Resources

How to use the resources you've provided, either by referencing them from your application code or from other XML resources.

PROVIDING RESOURCES

Providing Resources

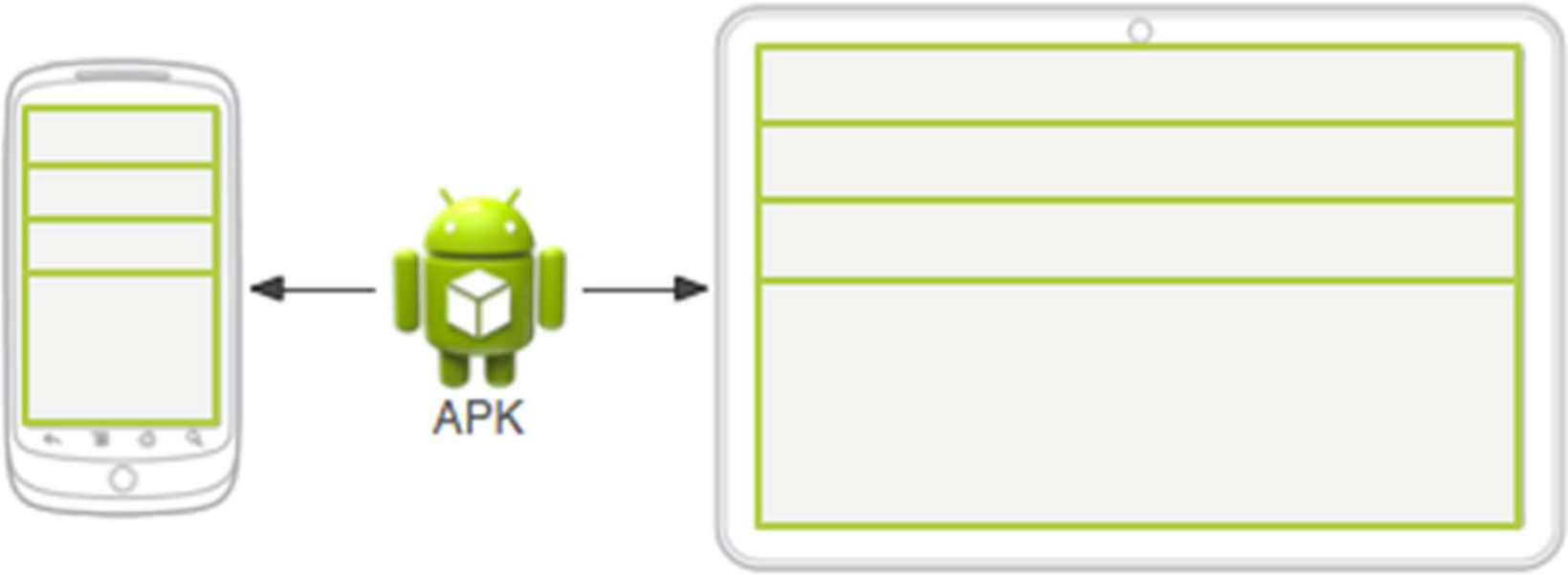
- You should always **externalize** application resources such as images, layouts and strings from your code, so that you can maintain them independently.
- You should also **provide alternative resources** for specific device configurations (i.e., different languages or screen sizes), by grouping them in specially-named resource directories.
- At runtime, Android automatically uses the appropriate resource based on the current configuration.

Default & Alternative Resources

- **Default resources** are those that should be used regardless of the device configuration or when there are no alternative resources that match the current configuration.
- **Alternative resources** are those that you've designed for use with a specific configuration. To specify that a group of resources are for a specific configuration, append an appropriate configuration qualifier to the directory name.

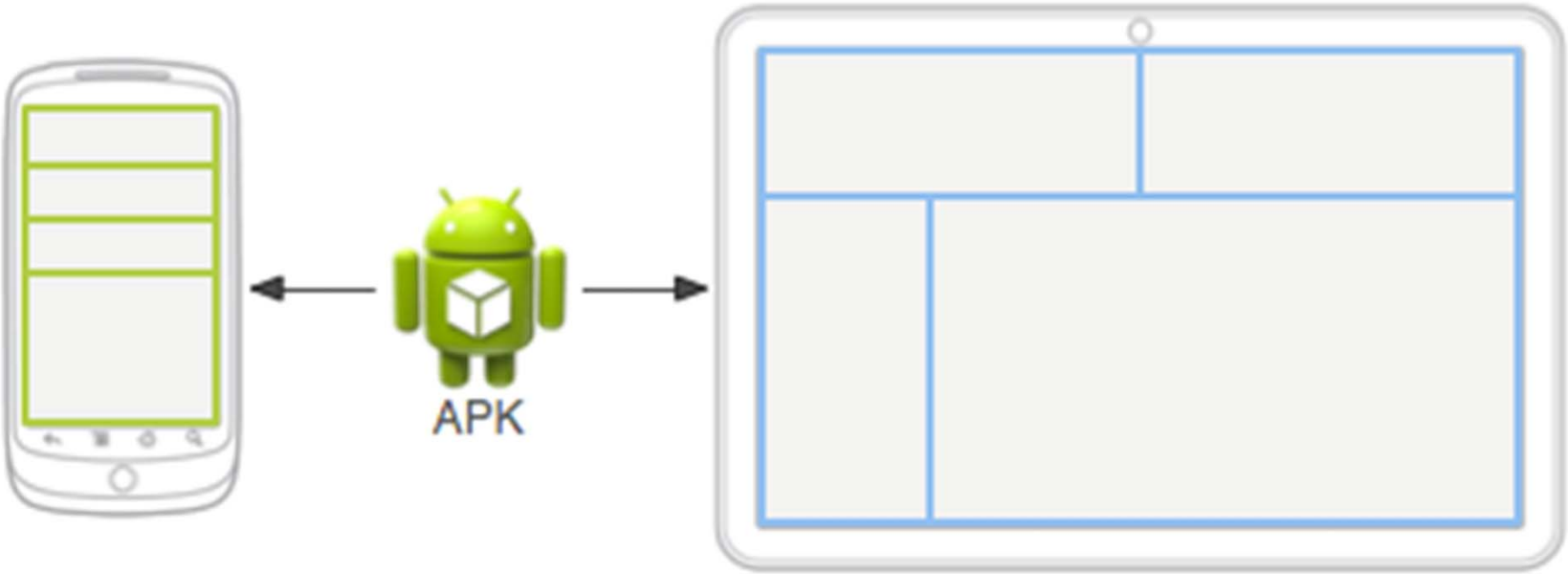
Default Resource

- Two different devices, each using the default layout (the app provides no alternative layouts).



Alternate Resource

- Two different devices, each using a different layout provided for different screen sizes.



Grouping Resource Types

- You should place each type of resource in a specific subdirectory of your project's res/ directory.
- For example, here's the file hierarchy for a simple project:
- Caution: Never save resource files directly inside the res/ directory - it will cause a compiler error.

```
res/
    drawable/
        icon.png
    layout/
        activity_main.xml
    values/
        strings.xml
```

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/animator/

XML files that define property animations.

Reference:
http://developer.android.com/guide/topics/graphics/prop-animation.html

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/anim/

XML files that define tween animations.

Reference:
http://developer.android.com/guide/topics/graphics/view-animation.html

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/color/

XML files that define a color state list.

Reference:
http://developer.android.com/guide/topics/resources/color-list-resource.html

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/drawable/

Bitmap files or XML files that are compiled into drawable resource.

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

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/mipmap/

Drawable files for different launcher icon densities.

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/layout/

XML files that define a user interface layout.

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

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/menu/

XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu.

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

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/raw/

Arbitrary files to save in their raw form.

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/values/

XML files that contain simple values:

- arrays.xml for resource arrays (typed arrays).
- colors.xml for color values
- dimens.xml for dimension values.
- strings.xml for string values.
- styles.xml for styles.

Default Resource Directories

- Resource directories supported inside project res/ directory
  - animator/
  - anim/
  - color/
  - drawable/
  - mipmap/
  - layout/
  - menu/
  - raw/
  - values/
  - xml/

res/xml/

Arbitrary XML files that can be read at runtime.

Providing Alternative Resources

- You should provide **alternative resources** to support specific device configurations.
- To specify that a group of resources are for a specific configuration, append an appropriate **configuration qualifier** to the directory name.

<resources_name>-<config_qualifier>

Providing Alternative Resources

Here are some default and alternative resources:

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

Providing Alternative Resources

- The resource files must be named exactly the same as the default resource files.
- Android detects the current device configuration and loads the appropriate resources for your application.
- Android supports several configuration qualifiers and you can add multiple qualifiers to one directory name, by separating each qualifier with a dash.

Configuration Qualifiers

1. MCC and MNC

- mcc410, mcc410-mnc01
- The mobile country code (MCC), optionally followed by mobile network code (MNC) from the SIM card in the device.

MCC	Country	MNC	Network
410	Pakistan	01	Mobililink
410	Pakistan	03	Ufone
410	Pakistan	04	Zong
410	Pakistan	06	Telenor
410	Pakistan	07	Warid

Configuration Qualifiers

02. Language and Region

- fr, fr-rCA, fr-rFR, en, en-rUS
- The language is defined by a two-letter language code (ISO 639-1), optionally followed by a two letter region code (preceded by lowercase "r").
- The codes are *not* case-sensitive; the r prefix is used to distinguish the region portion. You cannot specify a region alone.

Configuration Qualifiers

03. Layout Direction

- ldrtl or ldltr
- The layout direction of your application. ldrtl means "layout-direction-right-to-left".
- ldltr means "layout-direction-left-to-right" and is the default implicit value.

Configuration Qualifiers

07. Screen Size

- small, normal, large, xlarge
- Screens that are of similar size to a
  - **Small:** 320x426 dp
  - **Normal:** 320x470 dp
  - **Large:** 480x640 dp
  - **Xlarge:** 720x960 dp

Configuration Qualifiers

08. Screen Aspect

- long, notlong
- This is based purely on the aspect ratio of the screen (a "long" screen is wider).
- This is not related to the screen orientation.

Configuration Qualifiers

09. Screen Orientation

- port, land
- **port:** Device is in portrait orientation (vertical)
- **land:** Device is in landscape orientation (horizontal)
- This can change during the life of your application if the user rotates the screen.

Configuration Qualifiers

12. Screen Pixel Density (dpi)

- ldpi, mdpi, hdpi, xhdpi, nodpi, tvdpi
- **ldpi:** Low-density screens; approximately 120dpi.
- **mdpi**: Medium-density (on traditional HVGA) screens; approximately 160dpi.
- **hdpi:** High-density screens; approximately 240dpi.
- **xhdpi:** Extra high-density screens; approximately 320dpi. Added in API Level 8

Configuration Qualifiers

18. Platform Version (API level)

- v3, v4, v7, etc.
- The API level supported by the device.
- For example, v1 for API level 1 (devices with Android 1.0 or higher) and v4 for API level 4 (devices with Android 1.6 or higher).

Configuration Qualifiers

Other Qualifiers:

- 04. Smallest Width: (sw320dp, sw600dp, etc) The system will use resources only when the smallest dimension of available screen is at least what is specified - does not change when the screen's orientation changes.
- 05. Available Width: (w720dp, w1024dp, etc) Specifies a minimum available screen width, in dp units at which the resource should be used - configuration value will change when the orientation changes between landscape and portrait to match the current actual width.
- 06. Available Height: (h720dp, h1024dp, etc) Specifies a minimum available screen height, in dp units at which the resource should be used - configuration value will change when the orientation changes between landscape and portrait to match the current actual width.

Configuration Qualifiers

Other Qualifiers:

- 10. UI mode: (car, desk, television, appliance)
- 11. Night mode: (night, notnight)
- 13. Touchscreen Type: (notouch, finger)
- 14. Keyboard Availability: (keysexposed, keyshidden, keyssoft)
- 15. Primary Text Input Method: (nokeys, qwerty, 12key)
- 16. Navigation Key Availability: (navexposed, navhidden)
- 17. Primary Non-touch Navigation Method: (nonav, dpad, trackball, wheel)

Qualifier Name Rules

- You can specify multiple qualifiers for a single set of resources, separated by dashes. For example, drawable-en-rUS-land applies to US-English devices in landscape orientation.
- The qualifiers must be in the order .
- Alternative resource directories cannot be nested.
- Only one value for each qualifier type is supported. For example, if you want to use the same drawable files for Spain and France, you cannot have a directory named drawable-rES-rFR/. Instead you need two resource directories, such as drawable-rES/ and drawable-rFR/

Always Provide Default Resource

- In order to provide the best device compatibility, **always provide default resources** for the resources your application needs to perform properly.
- Then **create alternative resources for specific device configurations** using the configuration qualifiers.

ACCESSING RESOURCES

Accessing Resources

- Once you provide a resource in your application, you can apply it by referencing its resource ID.
- All resource IDs are defined in your project's R class, which automatically generated.
- For each type of resource, there is an R subclass (for example, R.drawable for all drawable resources), and for each resource of that type, there is a static integer (for example, R.drawable.icon).
- This integer is the resource ID that you can use to retrieve your resource.

Resource ID

A resource ID is always composed of:

- The resource type: Each resource is grouped into a "type," such as string, drawable, and layout.
- The resource name, which is either: the filename, excluding the extension; or the value in the XML android:name attribute, if the resource is a simple value (such as a string).

Accessing Resource

There are two ways you can access a resource:

- In XML
- In code

Accessing Resource (XML)

- In XML: Use special XML syntax that also corresponds to the resource ID defined in your R class, such as:

  @string/hello

- string is the resource type and hello is the resource name.

- You can use this syntax in an XML resource any place where a value is expected that you provide in a resource.

Accessing Resource (XML)

Here is the syntax to reference a resource in an XML resource:

```
@[<package_name>:]<resource_type>/<resource_name>
```

- <package_name> is the name of the package in which the resource is located (not required when referencing resources from the same package)
- <resource_type> is the R subclass for the resource type
- <resource_name> is either the resource filename without the extension or the android:name attribute value in the XML element (for simple values).

Accessing Resource (Code)

- Use a static integer from a sub-class of your R class, such as:

  R.string.hello

- string is the resource type and hello is the resource name.

- There are many Android APIs that can access your resources when you provide a resource ID in this format.

Accessing Resource (Code)

Here's the syntax to reference a resource in code:

```
[<package_name>.]R.<resource_type>.<resource_name>
```

- <package_name> is the name of the package in which the resource is located (not required when referencing resources from your own package).
- <resource_type> is the R subclass for the resource type.
- <resource_name> is either the resource filename without the extension or the android:name attribute value in the XML element (for simple values).

SIMPLE RESOURCES TYPES

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>

String string = getString(R.string.hello);

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <color name="translucent_red">#80ff0000</color>
</resources>

Resources res = getResources();
int color = res.getColor(R.color.opaque_red);

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/translucent_red"
    android:text="Hello"/>
```

res/values/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="textview_height">25dp</dimen>
    <dimen name="textview_width">150dp</dimen>
    <dimen name="ball_radius">30dp</dimen>
    <dimen name="font_size">16sp</dimen>
</resources>
```

```
Resources res = getResources();
float fontSize = res.getDimension(R.dimen.font_size);
```

```
<TextView
    android:layout_height="@dimen/textview_height"
    android:layout_width="@dimen/textview_width"
    android:textSize="@dimen/font_size"/>
```

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

res/values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomText" parent="@style/Text">
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#008</item>
   </style>
</resources>

<EditText
   style="@style/CustomText“
   android:layout_width="fill_parent“
   android:layout_height="wrap_content“
   android:text="Hello, World!" />
```

References

- Android Developers: Application Resources Guide
  - http://developer.android.com/guide/topics/resources/index.html
- Mobile Country Codes (MCC) and Mobile Network Codes (MNC)
  - http://www.mcc-mnc.com/
- Two-letter ISO 639-1 Language Codes
  - http://www.loc.gov/standards/iso639-2/php/code_list.php

Q & A