Android Getting Started

Hello World!

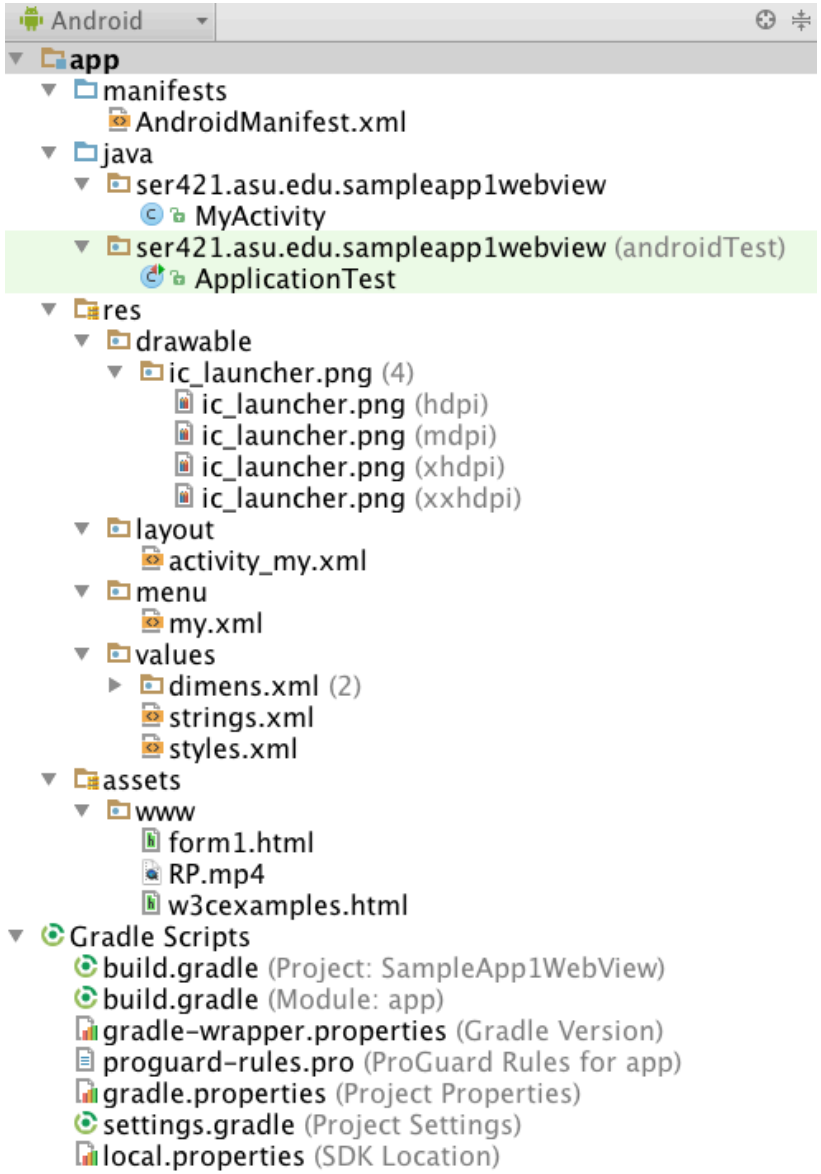
Runtime and Build System

Tools

Hello World!

- Create a new project in Android Studio
- Select the Basic (empty) layout
- Let Android Studio create the project
- Run it - empty screen with "Hello World!"
  - Note the full set of buttons are there
  - In the source you have an Activity and an ApplicationTest
    - The test doesn't do anything now, we'll come back to this
    - main (FirstActivity) has 4 imports and 3 methods
      - Note the methods all start with "on..." - tips you off it is a framework (IoC)
      - setContentView loads a resource (R) into the viewable area
    - open AndroidManifest.xml - metadata describing the application - includes services, permissions, activities...more to come

Understanding App Structure



1. AndroidManifest.xml – next slide
2. Java code: A Main *Activity*, & test code
3. Resources – in 2 slides
4. Various XML files
5. Assets
6. Gradle stuff (stay tuned…)

AndroidManifest.xml

- metadata describing the application - includes services, permissions, activities...more to come

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ser421.asu.edu.sampleapp1ser421" >
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".FirstActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Root package of your app

See drawable, strings & styles.xml

Activity relative to package

Telling Android this is the "main", The launching point for the app

Resources

Android has its own way of handling resources

- res dir: each subdirectory has files, either XML or png
  - drawable/ic_launcher.png is the launching image - the android robot
    - A different version of it for different categories of drawables (phone versus tablet)
- XML files?
  - layout – activity_first.xml – suffix matches activity string (FirstActivity)
    - The IDE will bring up a visual editor
    - Note the @+id references in the XML, what gives?
  - menu – first.xml – default menus, just as you would expect
  - values:

    dimens.xml - "dp" or "dip" are device-independent pixels, basically scaling to your device

    strings.xml - defines strings you use throughout your UI as resources

    - Look in activity_first.xml, you see references not literals – why?

    styles.xml - Android's notion of CSS - in our app it simply create a theme – responsiveness

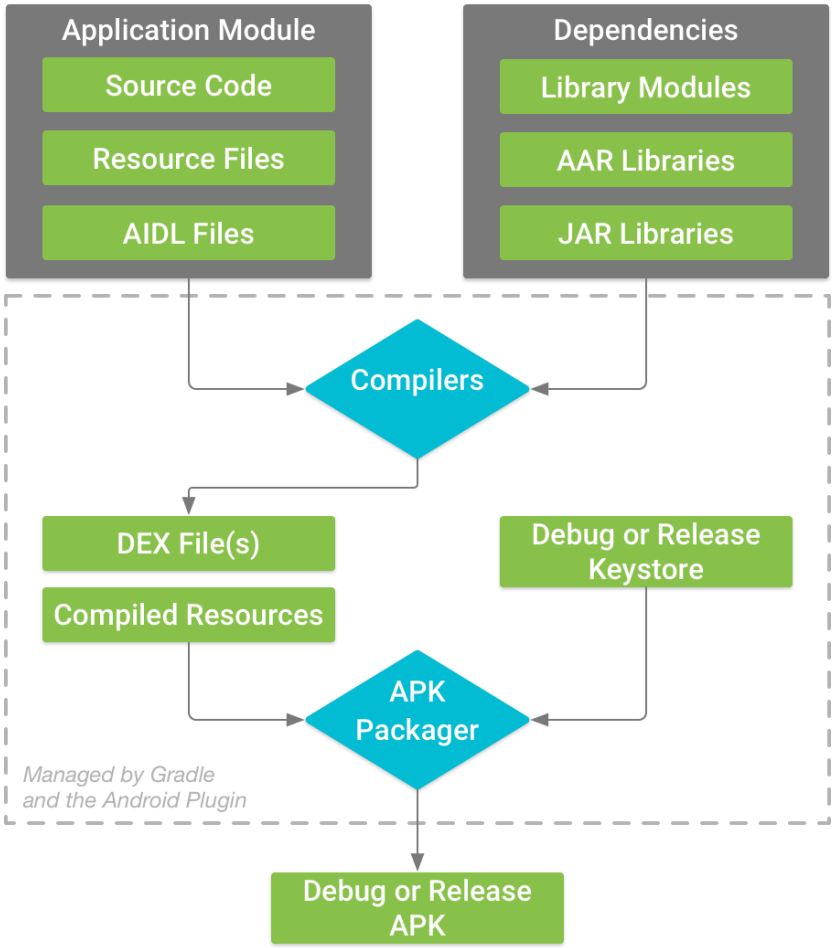
Resources are handled in this funky way to help w/ "responsiveness",
Or the ability to handle media rendering on different types of devices
http://developer.android.com/guide/topics/resources/providing-resources.html

Tools

- AVDs - Android Virtual Devices (AVDs)
  - downloaded with the SDK
    - You can get the SDK with Android Studio or have one on your system separately
    - It is possible to develop and run Android apps without an IDE through command-line tools
    - Install AVD definitions through the menu when configuring your SDK
    - It usually takes many minutes before the emulator finishes starting the virtual device (either that or my laptop melts).

Android Builds

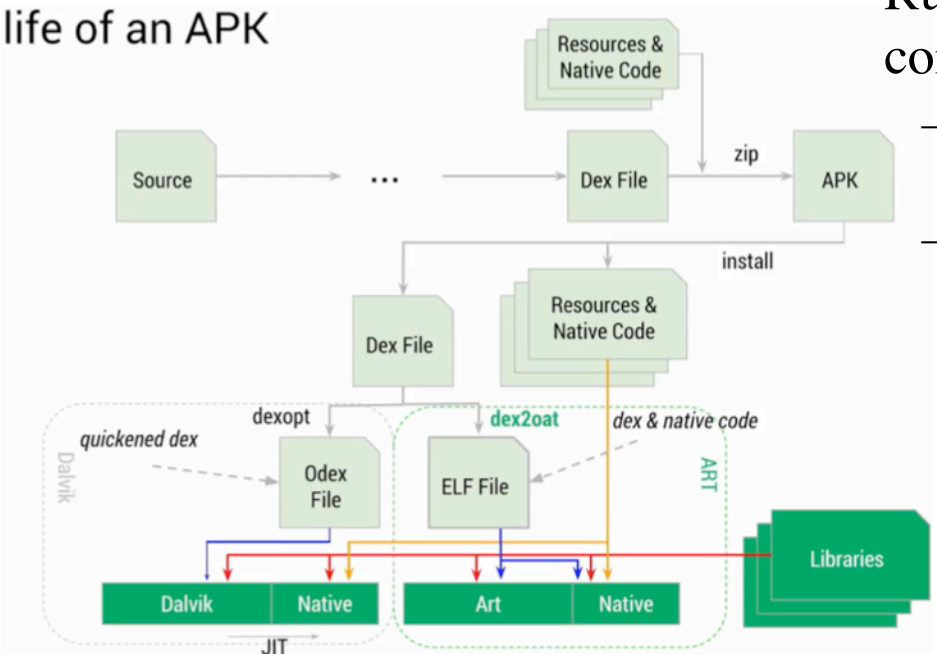
- Android Studio will create the apk
  - Look in build/outputs/apk
  - Default is to create debug version
- Internally, builds done by Gradle
  - Gradle is like Ant, or maybe more like Maven as it manages dependencies
  - Gradle is built on Groovy
    - Inspect build.gradle, notice the Java-like syntax
  - You can use the Gradle wrapper "gradlew" (or "gradlew.bat") program to do a command-line build (or just use "gradle" without a GUI wrapper)
  - You maintain Gradle tools separate from the SDK, which gets gnarly!



https://developer.android.com/sdk/installing/studio-build.html

Android Runtime

- The "classic" Android runtime is based on *Dalvik*, using a Just-In-Time (JIT) approach to compilation
  - Creates DEX files (specialized bytecode)
  - Packaged into an APK
- Starting in KitKat (4.4), the ART (Android Runtime) introduced Ahead-Of-Time compilation (AOT)
  - Still uses DEX files, but downstream this is optimized via ELF files
  - KitKat supported both Dalvik and ART, but from Lollipop (5.0) forward, it is all ART



http://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l