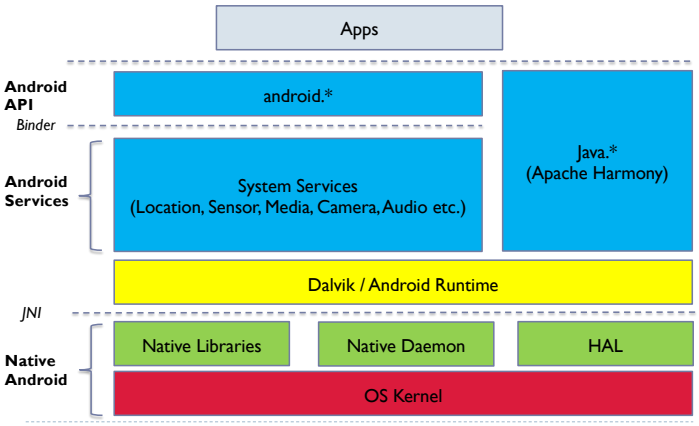
Android using Java

Amit Gulati, amit.gulati@gmail.com

1

Android Architecture



9

Android Architecture

10

10

Android Architecture

- OS Kernel
  - Linux based kernel
  - ~150 patches to Linux Kernel
  - Kernel Services
    - Process management
    - Memory management
    - Driver model
    - Networking Stack
    - Security
    - Etc.

11

11

Android Architecture

- OS Kernel
  - Androidism
    - Power Management
    - Low Memory Killer
    - Binder (IPC)
    - Ashmem
    - Alarm
    - Logger

12

12

Android Architecture

- Androidism
  - Low Memory Killer
    - OOM adjustments allow user space to control part of the kernel's OOM killing policies.
    - The OOM adjustments (oom_adj) range from -17 to 15
    - Lets user-space specify a set of memory thresholds where processes with a range of **oom_adj** values will get killed.

13

13

Android Architecture

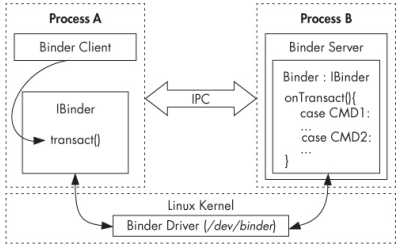
- Androidism
  - Low Memory Killer
    - Each process gets a "oom_adj" value between -17 to 15

```
radio     132   33    99224  24744 ffffffff afd0c51c S com.android.phone
system    136   33    88136  27024 ffffffff afd0c51c S com.android.systemui
app_13    140   33    94528  30700 ffffffff afd0c51c S com.android.launcher
app_6     179   33    93548  26264 ffffffff afd0c51c S android.process.acore
app_19    187   33    84312  21432 ffffffff afd0c51c S com.android.deskclock
app_24    200   33    82968  20056 ffffffff afd0c51c S com.android.protips
app_5     211   33    83516  20540 ffffffff afd0c51c S com.android.music
app_2     220   33    84008  21288 ffffffff afd0c51c S com.android.quicksearchbox
app_0     229   33    86484  22484 ffffffff afd0c51c S android.process.media
app_15    244   33    95600  21784 ffffffff afd0c51c S com.android.mms
app_28    265   33    85976  22948 ffffffff afd0c51c S com.android.email
root      325   41    732    348   c003da38 afd0c3ac S /system/bin/sh
root      352   325   888    324   00000000 afd0b45c R ps
# cat /proc/229/oom_adj
7
# cat /proc/211/oom_adj
11
# cat /proc/132/oom_adj
-12
```

14

Android Architecture

- Androidisms
  - Binder
    - Linux provides various IPC mechanisms like files, signals, sockets, pipes, semaphores, shared memory, message queues.
    - For Android the pre-existing were not deemed appropriate



15

15

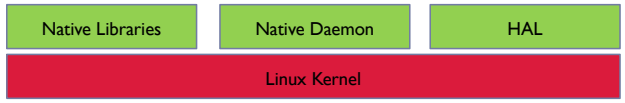
Android Architecture

- Androidisms
  - Android Mechanisms based on Binder
    - *Intents*
    - *Messengers*
    - *ContentProviders*
    - *AIDL*

16

16

Android Architecture

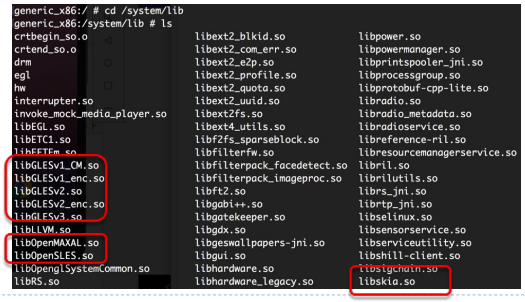


17

17

Android Architecture

- Native Libraries
  - ~100 dynamically loaded libraries,
  - */system/lib* directory



18

18

Android Architecture

- Native Daemons
  - Written in native language
    - Do not require Virtual Runtime
  - Part of the system startup
  - Continue to run throughout the lifetime of the system
  - Examples
    - rild
    - mediaserver
    - adb
    - installd

19

19

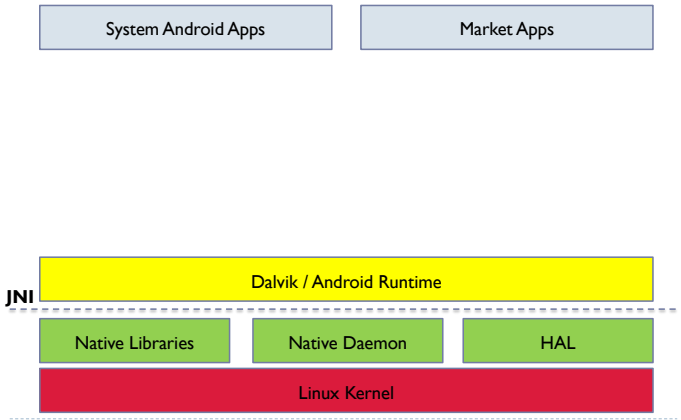
Android Architecture

- HAL
  - Decouples Android Platform software from hardware
  - Interfaces that hardware vendor implements
  - Hardware vendor implements HAL and Device Drivers.

20

20

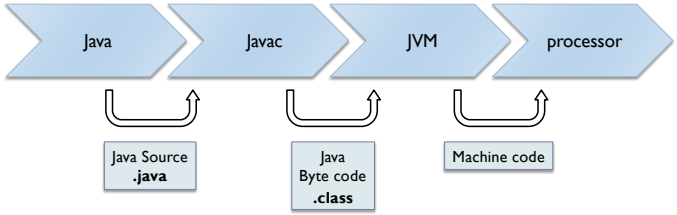
Android Architecture



21

Android Architecture

- Java Virtual Machine
  - Java byte code requires an JVM that converts the byte code to native instructions

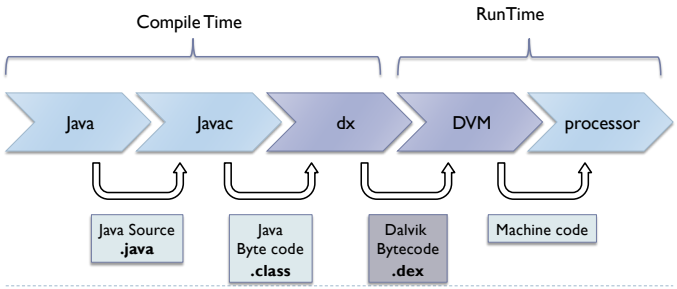


22

22

Android Architecture

- Dalvik Virtual Machine
  - Default with Kitkat and below
  - Optimized for mobile and embedded devices
  - Runs Dalvik byte code or "**.dex**" files.

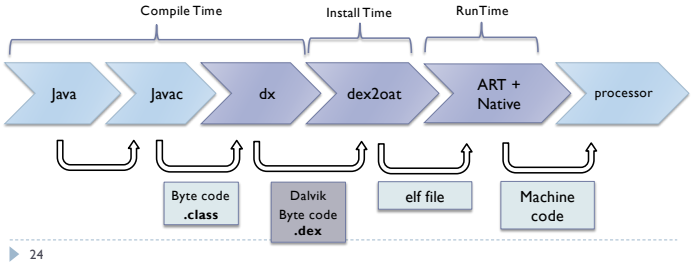


23

23

Android Architecture

- ART
  - Default with Lollipop and above
  - Ahead-of-time (AOT)
  - Improved Garbage Collection
  - 64bit support



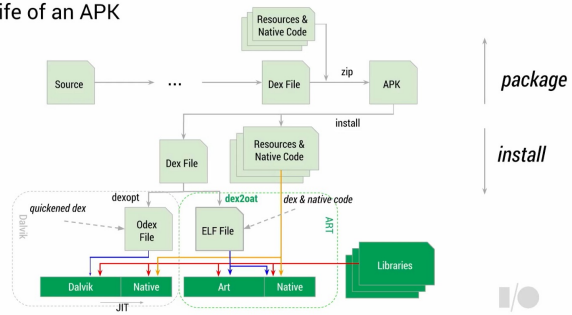
24

24

Android Architecture

- Android Runtime

The life of an APK



25

25

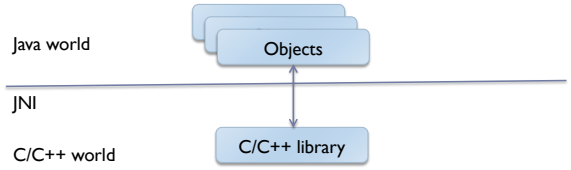
Android Architecture

- Android Runtime
  - ART (N+)
    - Ahead-of-time (AOT) + JIT
    - Improved Garbage Collection

26

Android Architecture

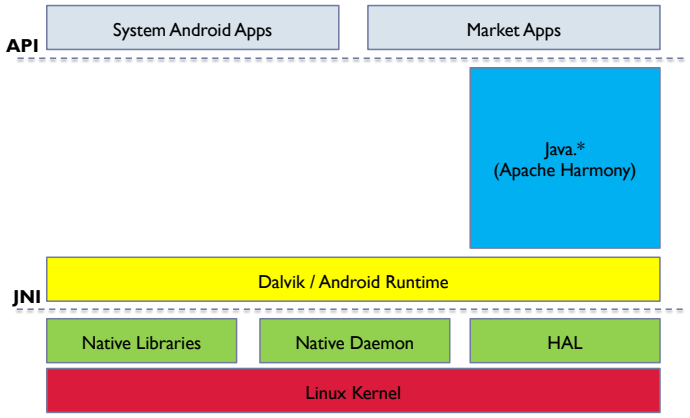
- JNI (Java Native Interface)
  - JNI is the bridge between Java layer and native layer.



  - Android relies on JNI to enable Java-coded services and components to interface with Android's native components.

27

Android Architecture



28

28

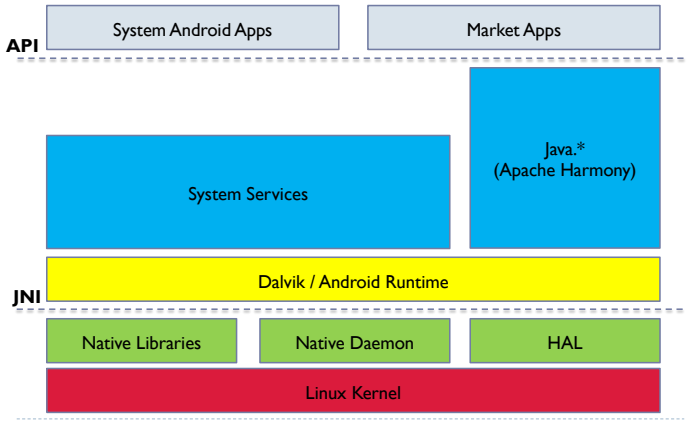
Android Architecture

- Java Runtime Libraries
  - Core Libraries
    - Core libraries based on Java SE 5 (mostly from Apache Harmony), with many differences.
    - Bundled in **android.jar**
    - No support
      java.applet, java.awt, java.lang.management and javax.management (JMX), java.rmi and javax.rmi, javax.accessibiliy,javax.activity, javax.imageio, javax.naming (JNDI), javax.print, javax.security.auth.kerberos, javax.security.auth.spi,javax.security.spi, javax.security.sasl, javax.sound, javax.swing, javax.transaction, javax.xml (except for javax.xml.parsers), org.ietf,org.omg, org.w3c.dom.* (subpackages)

29

29

Android Architecture



30

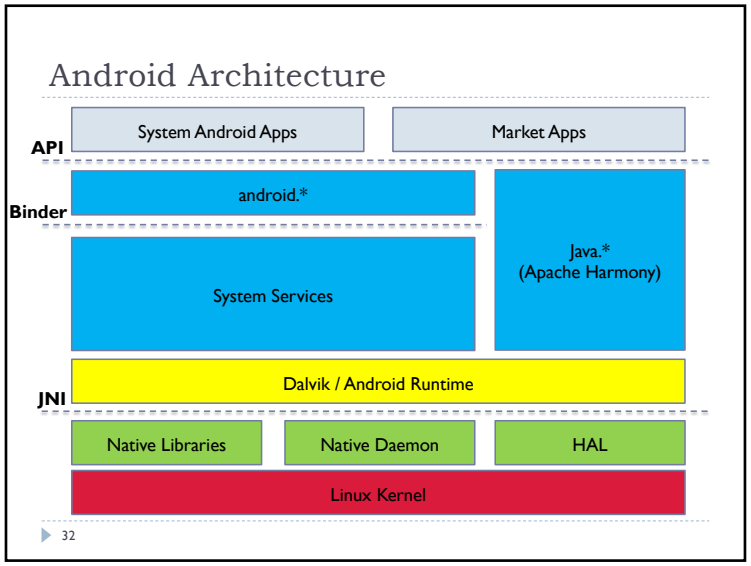
30

Android Architecture

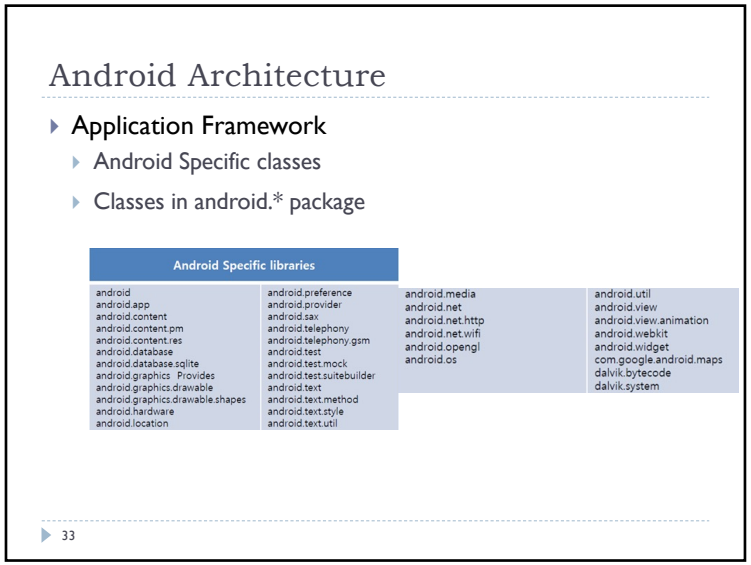
- System Services
  - Implement most of the fundamental Android features
    - Display and touch screen support, telephony, and network connectivity, notifications, location, sensor.
    - 79+ in number
    - Mostly Implemented in Java
  - Each system service defines a remote interface that can be called from other services and applications
  - Android IPC (Binder) based communication
    - Provides OO mechanism in which applications talk to System Services

31

31



32



33

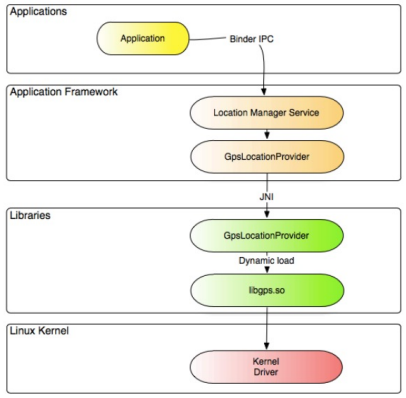
Android Architecture

- Applications
  - System Applications
    - Pre-installed on the phone and are shipped along with the phone
    - Cannot be uninstalled or changed
    - Mounted to /system folder
    - Example
      - Contacts, Phone, Browser
  - User-installed Applications
    - Applications installed by user from various sources
    - Mounted to /data folder

34

34

Layer Interaction



35

35

Android Studio

Amit Gulati, amit.gulati@gmail.com

59

Android Studio

- Installation
  - Download location

https://developer.android.com/studio

Android Studio provides the fastest tools for building apps on every type of Android device.

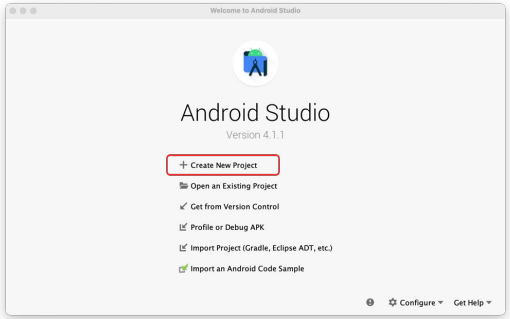
DOWNLOAD ANDROID STUDIO

60

60

Android Studio

- Start Android Studio
  - Select "Create New Project"

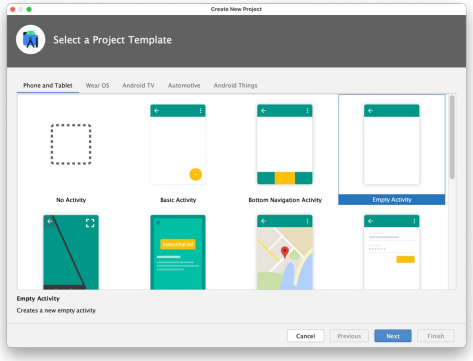


61

61

Android Studio

- Project Template

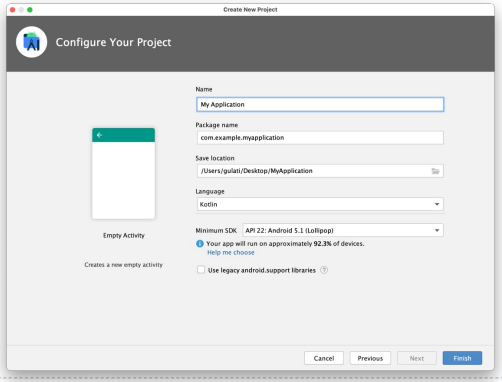


62

62

Android Studio

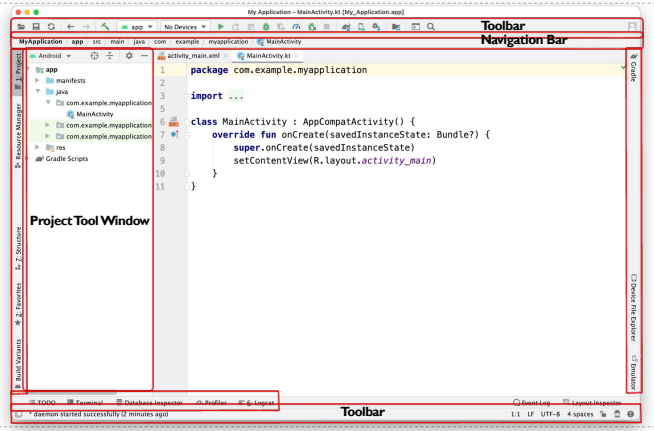
- Project Settings



63

63

Android Studio



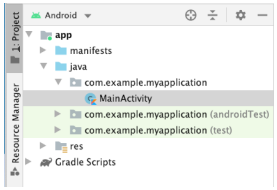
64

64

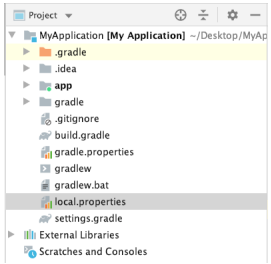
Android Studio

- Project Tool Window

Android View



Project View

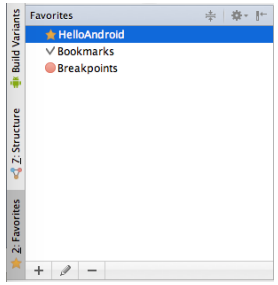


65

65

Android Studio

- Favorite Tool Window
  - Logically group references to related files
  - Contains 3 sections
    - Favorites
      - Files can be added to Favorites sections.
    - Bookmarks
      - Allow you to navigate to any line in a file.
    - Breakpoints



Favorite (⌥ ⇧ f)

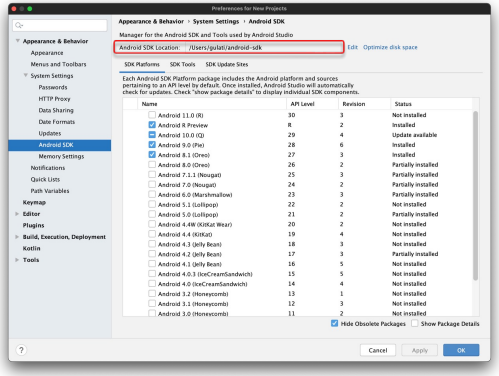
Bookmark (fn F3)

66

66

Android Studio

- SDK Manager



67

67

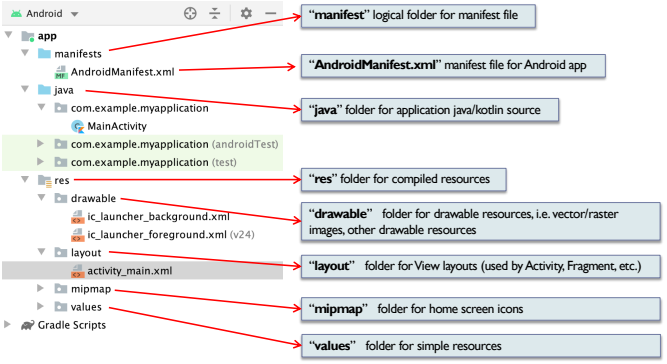
Android Project

amit.gulati@gmail.com

68

Android Studio

- Android View

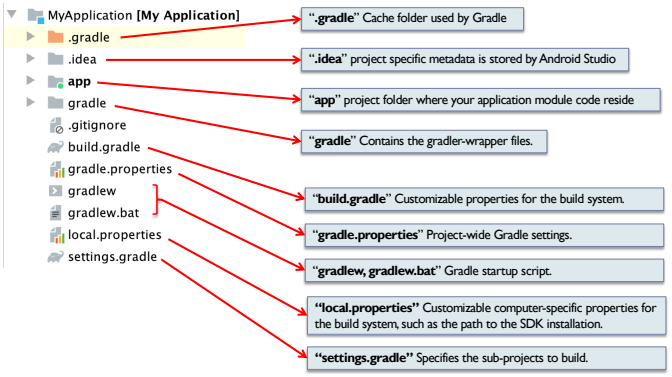


69

69

Android Studio

- Project View: File/Folder structure

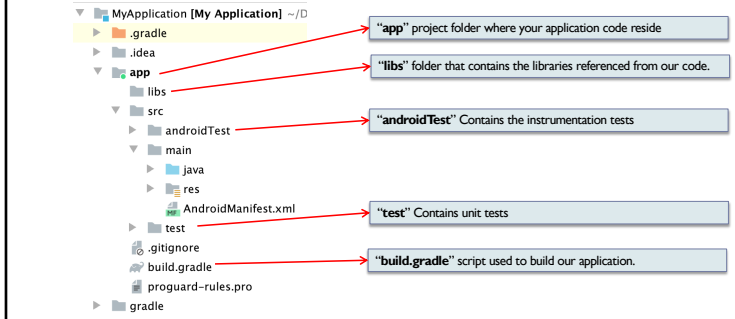


70

70

Android Studio

- Project View: File/Folder structure

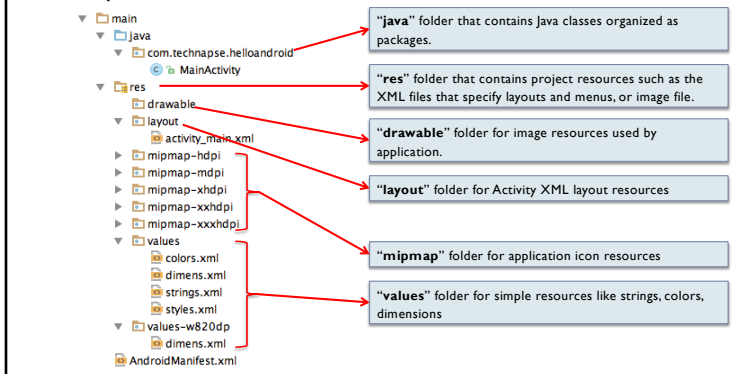


71

71

Android Studio

- Project View: File/Folder structure



72

72

Android Studio

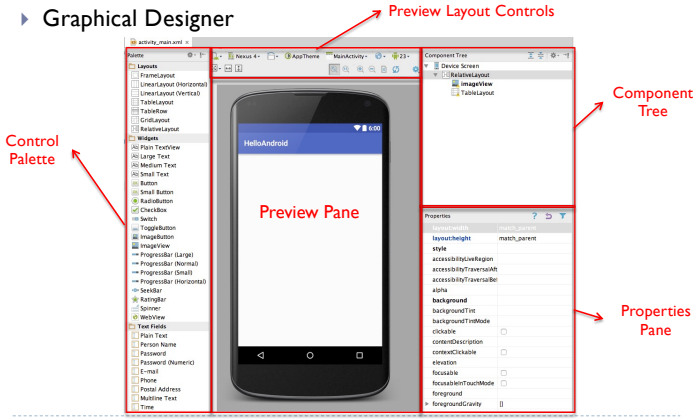
- Build Project
  - Shortcuts
    - ⌘ + F9
- Execute Project
  - Shortcuts
    - ⌘ + F10
    - Ctrl + R

73

73

Android Studio

- Graphical Designer



74

74

Android Studio

- Useful Shortcuts

Ctrl + /	This comments each line of the selected code. To block comments, use Ctrl + Shift + /
Ctrl + Alt + I	Indents the selected code.
Ctrl + Alt + O	Optimizes the imports
Command + O	Open File
Command + '+'	Expand folded code
Command + '-'	Collapse folded code
Command + 'n'	Code Generation • Constructor • Getter/Setter • Override methods • toString

75

75

Android Tools

Amit Gulati, amit.gulati@gmail.com

76

Android Tools

- Android SDK includes the following tools that help in the development process
  - Android Emulator (emulator)
  - Android Debug Bridge (adb)
  - Android Monitor
  - Android Asset packaging Tool (aapt)
  - hierarchyviewer
  - traceView
  - systrace
  - and more….

77

77

Android Tools

- Android Virtual Device
  - For testing Android Applications you need
    - Physical Device that runs Android
    - Virtual Device that runs Android Emulator
  - An Android Virtual Device (AVD) represents a device configuration.
    - A hardware profile (screen size, memory, external storage etc.)
    - Version of Android OS you want to run.
  - Android Virtual Device (AVD) Manager allows us to create/manage AVDs

78

78

Android Tools

- Android Emulator
  - Emulator runs on a Virtual Device
  - Provides a sandboxed version of Linux Kernel and the entire Android Stack to emulate environment found on Android device.
  - However, limited/no support for the following
    - Multi-Touch
    - Sensors
    - Camera
    - Connectivity API

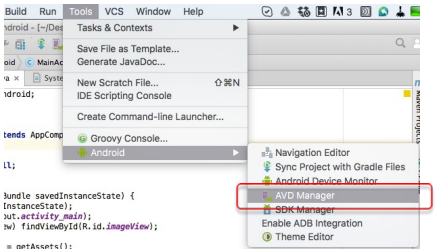
79

79

Android Tools

- AVD Manager
  - Toolbar Shortcut

  - Menu

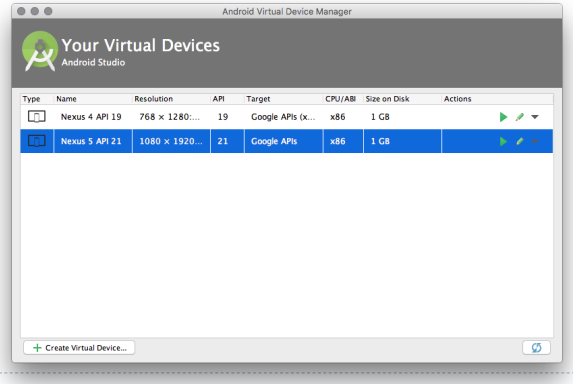


80

80

Android Tools

- AVD Manager

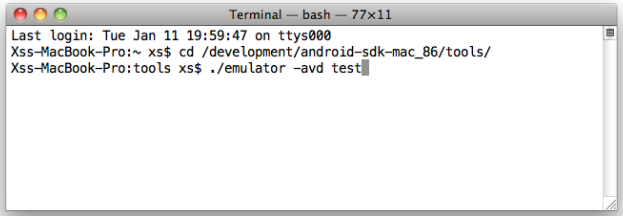


81

81

Android Tools

- Running the Android Emulator
  - Using the command line
    - Change Directory to the SDK tools directory
      - *cd /**<Path to SDK>**/tools*
    - Use the "emulator" command tool and specify an AVD.
      - ./emulator –AVD **<name of the AVD>**



82

82

Android Tools

- Geny Motion Android Emulator
  - Really Fast alternative to Android Emulator provided by Google.
  - Free for personal use

https://www.genymotion.com/#!/

83

83

Android Tools

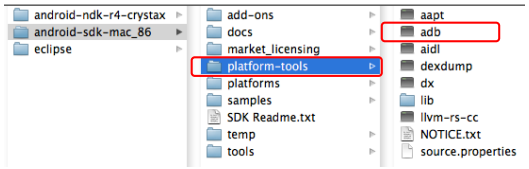
- Android Debug Bridge (ADB)
  - Manage the state of an emulator instance or an actual device.
  - ADB uses a command line interface, using which you can do a lot of things with the emulator.
    - Install application packages on emulator or device.
    - Push file to and pull files from from emulator or device.
    - Run shell commands on the device. Etc.

84

84

Android Tools

- Android Debug Bridge (ADB)



85

85

Android Tools

- Android Debug Bridge (ADB)
  - Check Devices/Emulators attached to/running on development machine

adb devices

```
Terminal — bash — 80×24
Xss-MacBook-Pro:~ xs$ cd /development/android-sdk-mac_86/
Xss-MacBook-Pro:android-sdk-mac_86 xs$ ls
SDK Readme.txt          market_licensing        samples
add-ons                 platform-tools          temp
docs                    platforms               tools
Xss-MacBook-Pro:android-sdk-mac_86 xs$ cd platform-tools/
Xss-MacBook-Pro:platform-tools xs$ ls
NOTICE.txt              aidl                    lib
aapt                    dexdump                 llvm-rs-cc
adb                     dx                      source.pro
Xss-MacBook-Pro:platform-tools xs$ ./adb devices
List of devices attached
100096cbd568    device
emulator-5554   device
```

86

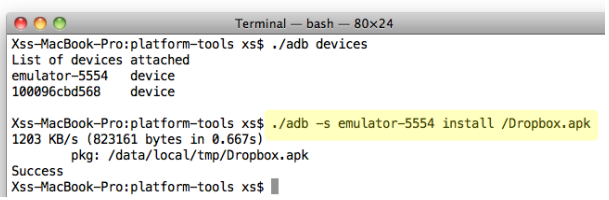
86

Android Tools

- Android Debug Bridge (ADB)
  - Installing Applications to Device or Emulator

adb –s <*name of device or emulator*> **install** <*name of adb file*>

  - Example: Installing Dropbox.apk



87

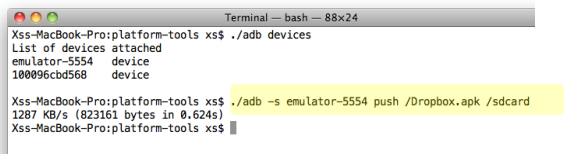
87

Android Tools

- Android Debug Bridge (ADB)
  - Push Files to Device/Emulator

adb –s <*name of device or emulator*> ***push*** <*path of file on desktop*> <*path on device*>

  - Example : Pushing Dropbox.apk to device external storage



  - Pull Files from Device/Emulator

adb –s <*name of device or emulator*> ***pull*** <*path on device*> <*path on desktop*>

88

88

Android Tools

- ADB Options
  - **pull** <device path> <local path> To transfer a file to your computer.
  - **push** <local path> <device path> To transfer a file to your device or emulator.
  - **install** <application package> To install an application package
  - **install** -r <package to reinstall> To reinstall an application.
  - **devices** To list all Android devices currently connected
  - **reboot** To restart an Android device programmatically
  - **start-server** To launch the ADB server communicating with devices and emulators
  - **kill-server** To terminate the ADB server
  - **bugreport** To print the whole device state (like dumpsys)
  - **help** To get an exhaustive help with all options and flags available

89

89

Application Essentials

Amit Gulati, amit.gulati@gmail.com

90

Android Runtime

- Fully Multitasking OS
  - Multiple processes can execute simultaneously, via CPU time slicing.
  - Multiple Android applications can run simultaneously

```
u0_a32    2165  1460  1263828 45900    ep_poll 78ab70e3866a S com.android.deskclock
u0_a12    2210  1460  1252948 40552    ep_poll 78ab70e3866a S android.ext.services
u0_a50    2250  1460  1254920 41332    ep_poll 78ab70e3866a S com.android.printspooler
u0_a14    2312  1460  1289376 69096    ep_poll 78ab70e3866a S com.android.launcher3
u0_a10    2337  1460  1263520 53128    ep_poll 78ab70e3866a S android.process.media
u0_a53    2416  1460  1272472 55768    ep_poll 78ab70e3866a S com.android.messaging
u0_a15    2448  1460  1253864 40276    ep_poll 78ab70e3866a S com.android.managedprovisionir
u0_a2     2523  1460  1258108 46028    ep_poll 78ab70e3866a S com.android.providers.calendar
u0_a26    2597  1460  1262044 44784    ep_poll 78ab70e3866a S com.android.calendar
u0_a27    2620  1460  1261252 41720    ep_poll 78ab70e3866a S com.android.camera2
u0_a35    2638  1460  1271820 49692    ep_poll 78ab70e3866a S com.android.email
u0_a40    2661  1460  1289664 57912    ep_poll 78ab70e3866a S com.android.gallery3d
```

91

91

Android Runtime

- Process Isolation
  - Every application runs in its own process
  - Application UID (App ID)
    - Android automatically assigns a unique UID, often called an *app ID*, to each application at installation

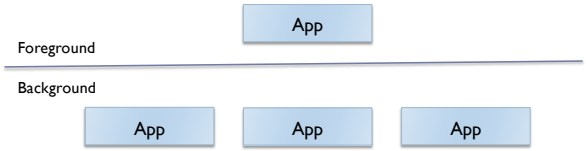
```
u0_a14    2417  1460  1289376 69096    ep_poll 78ab70e3866a S com.android.launcher3
u0_a10    2337  1460  1263520 53128    ep_poll 78ab70e3866a S android.process.media
u0_a53    2416  1460  1272472 55768    ep_poll 78ab70e3866a S com.android.messaging
u0_a15    2448  1460  1253864 40276    ep_poll 78ab70e3866a S com.android.managedprovisioning
u0_a2     2523  1460  1258108 46028    ep_poll 78ab70e3866a S com.android.providers.calendar
u0_a26    2597  1460  1262044 44784    ep_poll 78ab70e3866a S com.android.calendar
u0_a27    2620  1460  1261252 41720    ep_poll 78ab70e3866a S com.android.camera2
u0_a35    2638  1460  1271820 49692    ep_poll 78ab70e3866a S com.android.email
u0_a40    2661  1460  1289664 57912    ep_poll 78ab70e3866a S com.android.gallery3d
shell     2707  1437  8304   1704  sigsuspend 7614f6a7a8c7 S /system/bin/sh
root      2712  2707  8304   1748  sigsuspend 7fa32706f8c7 S /system/bin/sh
root      2755  2     0      0     worker_thr 0000000000 S kworker/u8:2
root      3086  2     0      0     worker_thr 0000000000 S kworker/u8:0
root      3212  2712  9868   1776           0 7841e5438507 R ps
generic_x86_64:/ #
```

92

92

Android Runtime

- Active vs Background Applications
  - Active application
    - Application that is currently displaying content on the screen and has focus
  - Background application
    - Application process is still in Memory but is not displaying content on the screen.
    - Application has restricted access to CPU



93

93

Android Runtime

- File System support
  - ext2, ext3, and ext4 (used by Linux systems)
  - vfat file system (used by Windows-based systems)
    - Used primarily for SD cards
  - YAFFS and YAFFS2 (needed to support NAND chips)

94

94

Android Runtime

- Persistent Storage
  - Options for File Storage
    - Internal Storage
    - External Storage

Internal Storage	External Storage
All phones have Internal Storage. Built-in Storage Limited in Size Always available (no need to be mounted) Secured Storage	External card slot Size depends on the size of the card. May not be mounted and hence not available Un-Secure

95

95

Android Runtime

- Directory Structure
  - File System and common folders
    - **/** : Root folder
    - **/system** : holds most of the Operating System (OS) files, including system applications, libraries, fonts, executables etc.
    - **/data** : Contains user data
    - **/sdcard** : soft link to the /mnt/sdcard and refers to the SD card on the device.
    - **/vendor** : Contains files specific to the vendor of the device

96

96

Android Runtime

- Directory Structure
  - **/data** folder contains the user data
  - **/data/app**
    - Apps that are installed by the user will be placed under this location.
  - **/data/data**
    - Contains the private data of all the applications
    - Every application is given a private folder in /data/data/<application package>

```
/data/data
generic_x86_64:/data/data # ls
android                     com.android.fallback              com.android.providers.settings
android.ext.services        com.android.gallery3d             com.android.providers.telephony
android.ext.shared          com.android.gesture.builder       com.android.providers.userdictionary
com.android.backupconfirm   com.android.htmlviewer           com.android.proxyhandler
com.android.bookmarkprovider com.android.inputdevices         com.android.quicksearchbox
com.android.calculator2     com.android.inputmethod.latin     com.android.sdksetup
com.android.calendar        com.android.keychain              com.android.server.telecom
com.android.camera2         com.android.launcher3             com.android.settings
com.android.captiveportallogin com.android.location.fused     com.android.sharedstoragebackup
com.android.carrierconfig   com.android.managedprovisioning   com.android.shell
```

97

97

Android Runtime

- Directory Structure
  - **/system**
    - folder contains libraries, system binaries, and other system-related files
  - **/system/app**
    - contains system apps and preinstalled apps

```
generic_x86_64:/data/app # cd /system/app/
generic_x86_64:/system/app # ls
BasicDreams         EasterEgg           NetSpeed
BookmarkProvider    Email               OpenWnn
Browser2            EmulatorSmokeTests  PacProcessor
Calendar            ExactCalculator     PicoTts
Camera2             ExtShared           PrintRecommendationService
CaptivePortalLogin  Fallback            PrintSpooler
CertInstaller       Gallery2            Protips
```

98

98

Android Runtime

- Directory Structure
  - **/sdcard**
    - Partition that contains the data present on the SD card of the device
  - **/sdcard/Android/data/<application package>**
    - Private location in sdcard for application to write data

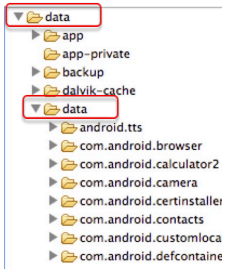
```
generic_x86_64:/sdcard/Android/data # ls
com.android.gallery3d
generic_x86_64:/sdcard/Android/data #
```

99

99

Android Runtime

- File System Isolation
  - Applications are restricted to access to access to just their home directories
  - Internal folder for applications
    - /data/data/<application package>
  - External folder for applications
    - /sdcard/Android/data/<application package>



100

100

Application Essentials

- Physical Structure
  - An Android application is bundled as a "**.apk**"file.
  - .apk if a zip file

101

101

Application Essentials

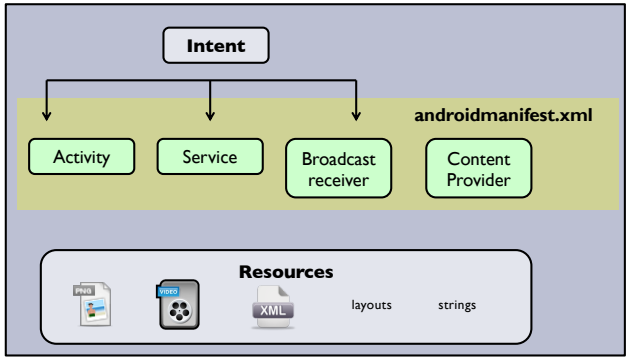
- What is an Android Application ?
  - Collection of loosely coupled Application Components
    - Activity
    - Service
    - Content Provider
    - Broadcast Receiver
  - Bound together by the Application Manifest.
  - Communicate with each other via Intent objects.
  - Use Resources
    - Almost everything other than Java code in Android is a resource.
    - Common Resources: String, Images, Audio, Video, Layouts etc.

102

102

Application Essentials

- What is an Android Application ?



103

103

Application Essentials

- Activity
  - Activities form the Presentation layer of an Android application.
  - A Single Screen in your application is an Activity.
    - Covers the whole screen of device
  - Activity object is given a blank window to draw.
    - Activity object is not capable of drawing in a window.

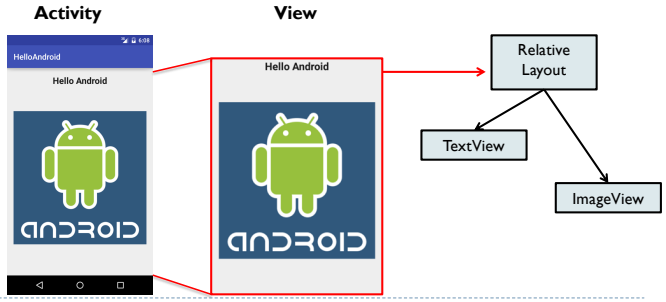
Simple Activity

104

104

Application Essentials

- Activity
  - Activity attaches a **View** object to itself
    - A View object can draw in the window.

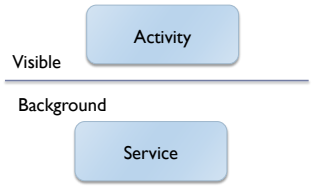


105

105

Application Essentials

- Service
  - Service is an Android Application Component that runs in the background.



  - Accomplish tasks that are to be run even when none of the Activities associated with an application are visible or running.

107

107

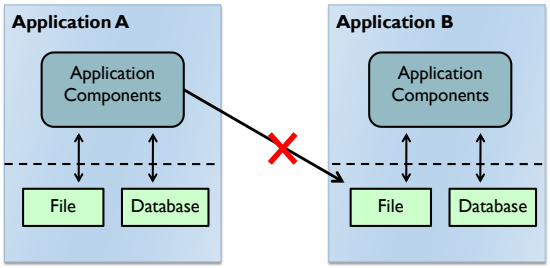
Application Essentials

- Service
  - Service can be scheduled for execution in the background at a later point in time or periodically.
  - Common Use-cases
    - Playing music in the background
    - Syncing Application data with remote server.
    - Run scheduled tasks

108

Application Essentials

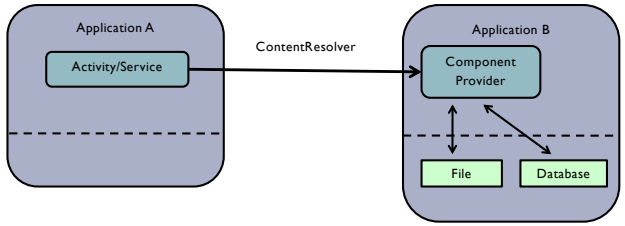
- Content Provider
  - Android Applications run in a sand-boxed environment.
    - An application does not have direct access to files and directories of other applications.



109

Application Essentials

- Content Provider
  - Allows sharing data between applications
    - By implementing content provider, an application can share its data with other applications



110

110

Application Essentials

- Content Provider
  - Android ships with a number of content providers for common data
    - call log, contacts, browser information, audio and video media etc.

111

111

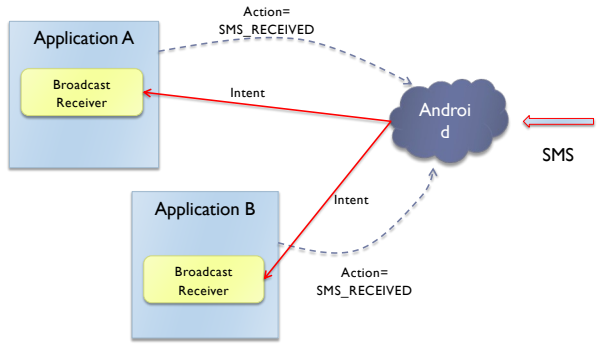
Application Essentials

- Broadcast Receiver
  - System and Application broadcast events.
    - Android System generates broadcasts when certain state changes occur.
    - Applications can generate broadcasts.
  - Applications that want to listen for broadcasts
    - Implement Broadcast receiver component
    - Register the Broadcast Receiver for the Broadcast Action.

112

Application Essentials

- Broadcast Receiver



113

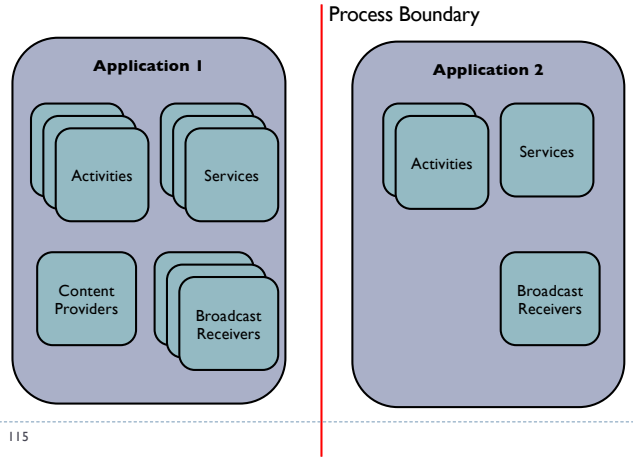
Application Essentials

- Broadcast Receiver
  - Application with broadcast receiver does not have to be running or in background
    - Launched automatically in the background in order to execute Broadcast Receiver.
    - Executed as a result of a specific broadcast is raised by the System or an Application.
  - Broadcast announcements generated by system may include:
    - Airplane mode
    - Phone call received.
    - Low battery
    - USB connection
    - Etc.

114

114

Application Essentials

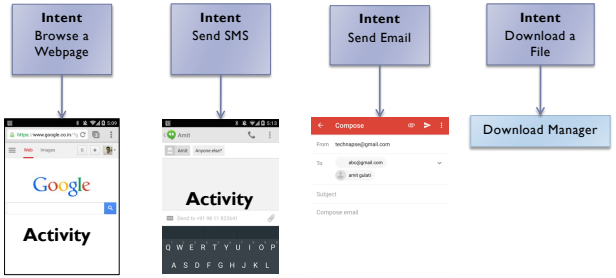


115

115

Application Essentials

- Intent
  - Abstract concept that represent
    - Operation or action to be performed.
  - Actions are performed via Application components

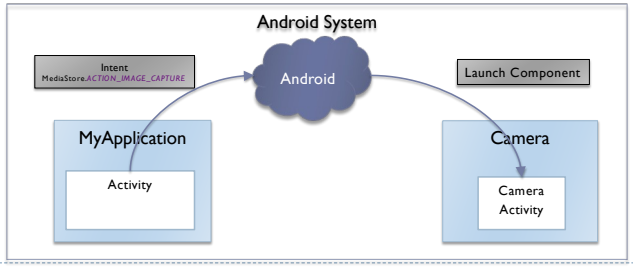


116

116

Application Essentials

- Intent
  - Primary way for launching Application components.
  - Components that are launched using Intent may be part of other applications



117

117

Application Essentials

- Intent
  - Example

```
var intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
startActivity(intent)
```


119

Application Essentials

- Intent
  - Primary way for launching Application components.
  - Primary way in which system / third party application send events to applications.
  - Communication between application components

122

Application Essentials

- Why Intents ?
  - Intent is a new concept for Application Development
    - Different from other systems in terms of system design
  - Primary Goals of using Intent
    - Re-usability & Loose Coupling

123

123

Application Essentials

- Why Intents ?
  - Re-usability & Loose Coupling
    - Use of components written by other programmers without knowing much about them.
    - To use the Camera Activity in android

```
Intent camera =
        new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

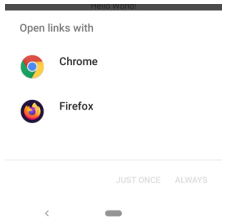
124

124

Application Essentials

- Why Intents ?
  - Re-usability & Loose Coupling
    - Intents are bound to Application components at Run-time
    - In case of multiple components for an intent the user can select which one to use.

```
var intent = Intent(Intent.ACTION_VIEW)
intent.data = Uri.parse("http://www.google.com")
startActivity(intent)
```

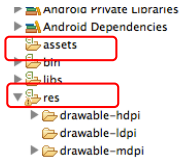


125

125

Application Essentials

- Android Application Resources
  - Resources are things that are used by your application other than source code.
    - Images (Icons, Splash Screen, etc.)
    - XML files
    - Media files (Audio, Video etc.)
    - Strings, Arrays, Styles and Themes etc.
  - Two folders for adding resources
    - **res**
    - **assets**

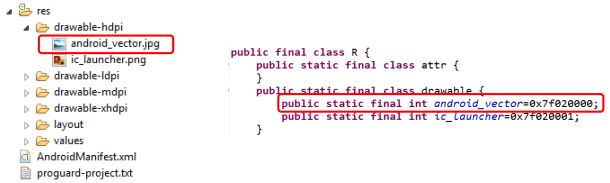


130

130

Application Essentials

- Android Application Resources
  - "res" folder resources
    - Android Asset Packaging Tool (AAPT) processes resources added to the "res" folder in project.
    - A unique entry (ID) for each resource is placed in the R.java file.
    - Resource can then be accessed in an Android project with the ID.

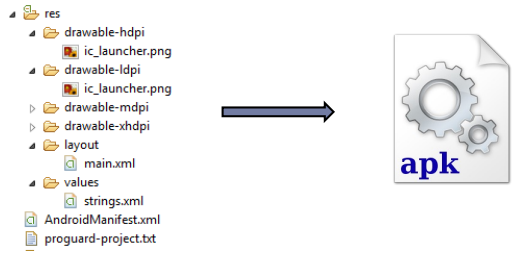


131

131

Application Essentials

- Android Application Resources
  - Resources added to the application project are packaged together in the application bundle (.apk) that is installed on the device.



132

132

Application Essentials

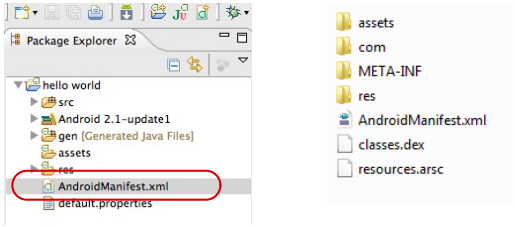
- Android Manifest
  - Every Android application package contains an Application Manifest file (androidmanifest.xml).
  - Contains application meta-data, that is used by Android system.
  - Application Manifest file serves the following purpose
    - Application information i.e. Name, Version, icon, etc are defined in the manifest file.
    - Used to specify hardware and software dependencies of the application.
    - Used to specify permissions required by the application.
    - Used to define the different Application components present in the application package.

133

133

Application Essentials

- Android Manifest
  - Manifest file is located at the root level in the Android project hierarchy.
  - Manifest file is packaged along with executable code and resources in the .apk file.



134

134

Application Essentials

- A sample manifest file

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.androidbook.multimedia" android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name"
        android:debuggable="true">
        <activity android:name=".MultimediaMenuActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="AudioActivity"></activity>
        <activity android:name="StillImageActivity"></activity>
        <activity android:name="VideoPlayActivity"></activity>
        <activity android:name="VideoRecordActivity"></activity>
    </application>
    <uses-permission android:name="android.permission.WRITE_SETTINGS"></uses-permission>
    <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
    <uses-permission android:name="android.permission.SET_WALLPAPER"></uses-permission>
    <uses-permission android:name="android.permission.CAMERA"></uses-permission>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8"></uses-sdk>
    <uses-feature android:name="android.hardware.camera"></uses-feature>
</manifest>
```

135

135

Activities

Amit Gulati, amit.gulati@gmail.com

143

Activity

- **android.app.Activity** class
  - An activity is represented by the **android.app.Activity** class.
  - **Activity** class is generally sub-classed or extended to create an Activity for Application

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

144

144

Activity

- **Activity** sub-classes must to the following
  - Implement the onCreate method and load a Content View

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

145

145

Activity

- **Activity** must be registered with the Android System
  - Be Declared in the AndroidManifest.xml.

androidmanifest.xml

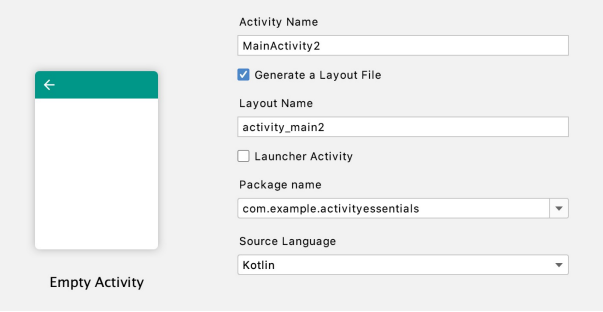
```
<activity
    android:name="com.example.testing.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

146

146

Activity

- Creating a new Activity using Android Studio



147

147

Activity

- Starting a New Activity.
  - Method that is used to start an activity such that it will not return any information back to the parent activity.

```
public void startActivity (Intent intent)
```

intent, Intent object that will be used to create the activity.

- No apparent relationship between current activity and newly launched activity.

- Terminating an Activity

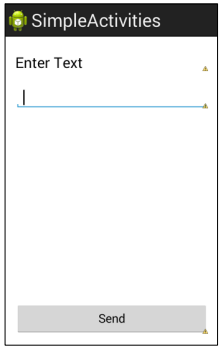
```
void finish ()
```

148

148

Activity and Layout

- Activity layout



res\layout\activity_main.xml

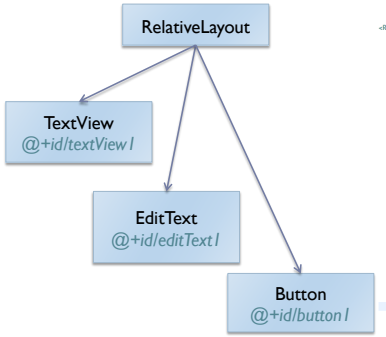
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="Enter Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="21dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignRight="@+id/editText1"
        android:text="Send" />

</RelativeLayout>
```

149

149

Activity and Layout

- Activity layout is a View Tree



res\layout\activity_main.xml

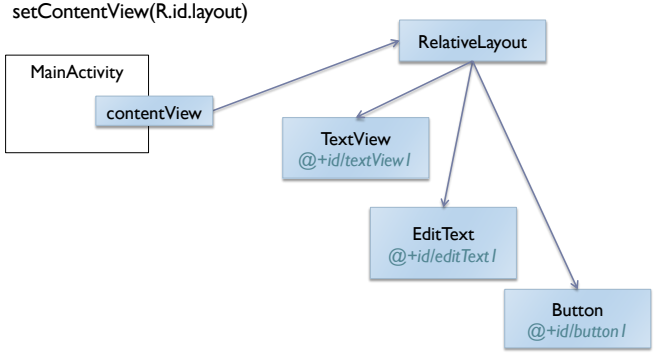
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="Enter Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="21dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignRight="@+id/editText1"
        android:text="Send" />

</RelativeLayout>
```

150

150

Activity and Layout

- Activity attaches a View to itself
  - setContentView(R.id.layout)

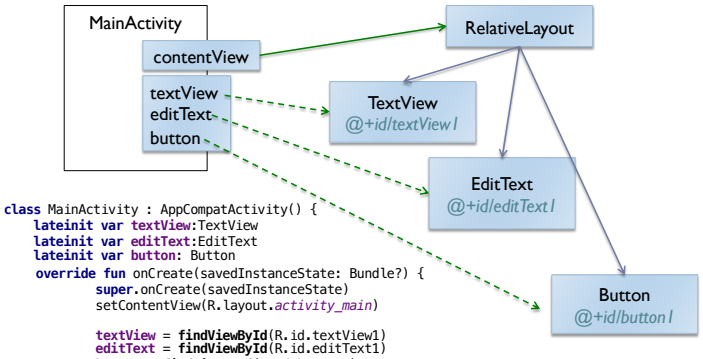


151

151

Activity and Widgets

- Activity and View References




```
class MainActivity : AppCompatActivity() {
    lateinit var textView:TextView
    lateinit var editText:EditText
    lateinit var button: Button
    override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)

            textView = findViewById(R.id.textView1)
            editText = findViewById(R.id.editText1)
            button = findViewById(R.id.button1)
```

152

152

Activity and Widgets

- Activity and View References
  - To interact with Views in the content view hierarchy, we require references to those objects.
  - Method that allows us to get references to View objects in Activities content view

View **findViewById** (int id)

id,	ID of the View for which we require a reference.

153

153

Activity and Widgets

- Button Event Handling
  - Button widget provides an attribute in XML "**android:onClick**", that's used for specifying name of the handler method in Activity.

activity_main.xml

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignRight="@+id/editText1"
    android:text="Send"
    android:onClick="send" />
```

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun send(view:View) { }
}
```

154

154

Activity and Widgets

- View Event Handling using Listener
  - Listener object is attached to the View on which events need to be handled.
    - Example

```
var button:Button
button = findViewById(R.id.mainButton)
button.setOnClickListener(View.OnClickListener {


})
```

155

155

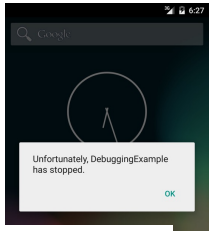
Debugging, Logging and Static Analysis using Android Studio

Amit Gulati, amit.gulati@gmail.com

156

Runtime Errors

- Unfortunately, Application has stopped
  - When application in Android device crashes
    - Dialog box on the screen.
    - Stack Trace of the crashed application is dumped on the Logcat.



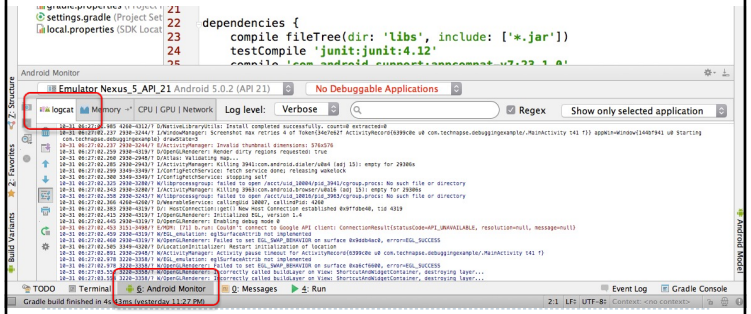

```
E/AndroidRuntime: FATAL EXCEPTION: main
E/AndroidRuntime: Process: com.technapse.debuggingexample, PID: 4372
E/AndroidRuntime: java.lang.RuntimeException: Unable to start activity ComponentInfo{com
om.technapse.debuggingexample.MainActivity}: java.lang.NullPointerException: Attempt to invoke
widget.Button.setOnClickListener(android.view.View$OnClickListener)' on a null object reference
E/AndroidRuntime:     at android.app.ActivityThread.performLaunchActivity(ActivityThread

E/AndroidRuntime:     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2360)
E/AndroidRuntime:     at android.app.ActivityThread.access$800(ActivityThread.java:144)
E/AndroidRuntime:     at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1278)
E/AndroidRuntime:     at android.os.Handler.dispatchMessage(Handler.java:102)
E/AndroidRuntime:     at android.os.Looper.loop(Looper.java:135)
E/AndroidRuntime:     at android.app.ActivityThread.main(ActivityThread.java:5221)
```

157

157

Runtime Errors

- Android Monitor
  - Android Monitor is part of Android Studio
  - Most important tab is the LogCat

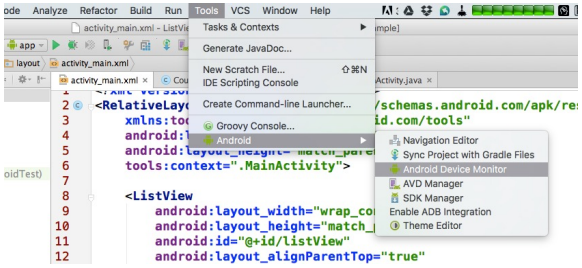


158

158

Runtime Errors

- Android Device Monitor
  - Android Device Monitor or DDMS also has Logcat.
  - Opens in a separate window rather than Android Studio
  - Tools => Android => Android Device Monitor

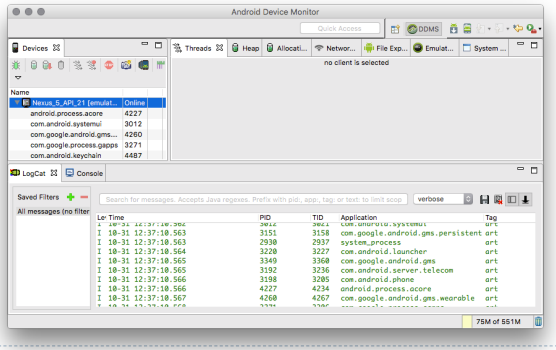


159

159

Runtime Errors

- Android Device Monitor
  - Logcat window



160

Runtime Errors

- Android Stack Trace

```
10-31 07:46:35.041 4596-4596/? D/AndroidRuntime: Shutting down VM
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: FATAL EXCEPTION: main
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: Process: com.technapse.debuggingexample, PID: 4596
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: java.lang.RuntimeException: Unable to start
 activity ComponentInfo{com.technapse.debuggingexample/com.technapse.debuggingexample.MainActivity}:
  java.lang.NullPointerException: Attempt to invoke virtual method 'void android.widget.Button
 .setOnClickListener(android.view.View$OnClickListener)' on a null object reference
```

Time Stamp | Process ID/ Thread ID | Log level/Tag | Message

161

Runtime Errors

- Android Stack Trace

```
10-31 07:46:35.041 4596-4596/? D/AndroidRuntime: Shutting down VM
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: FATAL EXCEPTION: main
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: Process: com.technapse.debuggingexample, PID: 4596
10-31 07:46:35.041 4596-4596/? E/AndroidRuntime: java.lang.RuntimeException: Unable to start
 activity ComponentInfo{com.technapse.debuggingexample/com.technapse.debuggingexample.MainActivity}:
  java.lang.NullPointerException: Attempt to invoke virtual method 'void android.widget.Button
 .setOnClickListener(android.view.View$OnClickListener)' on a null object reference
```

Time Stamp | Process ID/ Thread ID | Log level/Tag | Message

162

Runtime Errors

- Android Stack Trace

```
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: FATAL EXCEPTION: main
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: Process: com.technapse.debuggingexample, PID: 4372
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: java.lang.RuntimeException: Unable to start activity
 ComponentInfo{com.technapse.debuggingexample/com.technapse.debuggingexample.MainActivity}: java.lang
 .NullPointerException: Attempt to invoke virtual method 'void android.widget.Button.setOnClickListener(android.view
 .View$OnClickListener)' on a null object reference
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.performLaunchActivity
 (ActivityThread.java:2298)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.handleLaunchActivity
 (ActivityThread.java:2360)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.access$800(ActivityThread.java:144
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread$H.handleMessage(ActivityThread
 .java:1278)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.os.Handler.dispatchMessage(Handler.java:102)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.os.Looper.loop(Looper.java:135)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.main(ActivityThread.java:5221)
  <2 internal calls>
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run
 (ZygoteInit.java:899)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:694)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:  Caused by: java.lang.NullPointerException: Attempt to invoke
 virtual method 'void android.widget.Button.setOnClickListener(android.view.View$OnClickListener)' on a null object
 reference
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.technapse.debuggingexample.MainActivity.onCreate
 (MainActivity.java:22)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.Activity.performCreate(Activity.java:5937)
```


163

Runtime Errors

- Android Stack Trace

```
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: FATAL EXCEPTION: main
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: Process: com.technapse.debuggingexample, PID: 4372
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime: java.lang.RuntimeException: Unable to start activity
 ComponentInfo{com.technapse.debuggingexample/com.technapse.debuggingexample.MainActivity}: java.lang
 .NullPointerException: Attempt to invoke virtual method 'void android.widget.Button.setOnClickListener(android.view
 .View$OnClickListener)' on a null object reference
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.performLaunchActivity
 (ActivityThread.java:2298)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.handleLaunchActivity
 (ActivityThread.java:2360)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.access$800(ActivityThread.java:144
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread$H.handleMessage(ActivityThread
 .java:1278)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.os.Handler.dispatchMessage(Handler.java:102)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.os.Looper.loop(Looper.java:135)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.ActivityThread.main(ActivityThread.java:5221)
  <2 internal calls>
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run
 (ZygoteInit.java:899)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:694)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:  Caused by: java.lang.NullPointerException: Attempt to invoke
 virtual method 'void android.widget.Button.setOnClickListener(android.view.View$OnClickListener)' on a null object
 reference
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at com.technapse.debuggingexample.MainActivity.onCreate
 (MainActivity.java:22)
10-31 06:40:19.737 4372-4372/? E/AndroidRuntime:     at android.app.Activity.performCreate(Activity.java:5937)
```

164

164

Logging

- Logcat
  - Logcat tool dumps a log of system messages which includes
    - Stack trace when the device throws an error.
    - Log messages written from user applications.
- Viewing logcat messages
  - To dump the messages on the terminal (Command Line) just run the command "**adb logcat**".
    - Not the best way to view log messages.
  - Use the Logcat view provided Android Studio "Android Monitor" or the "Android Device Monitor"

165

165

Logging

- Writing log messages from applications
  - **android.util.Log** is a logging class you can use to dumpmessages to the logcat.
  - Methods of the Log class

Name of Method	Description	Color used in Logcat
int **v** (String tag, String msg)	Write verbose messages.	Black
int **d** (String tag, String msg)	Write debug messages.	Blue
int **i** (String tag, String msg)	Write info messages.	Green
int **w** (String tag, String msg)	Write warning messages.	Orange
int **e** (String tag, String msg)	Write error messages.	Red

166

166

Debugging

- Debugging using Android Studio
  - Android Studio enables you to debug apps running on the emulator or on an Android device.
  - Android Studio builds a debuggable version of your app, connects to a device or to the emulator, installs the app and runs it.
  - Setting Breakpoints
    - Click on the gray are in the editor, also known as Gutter

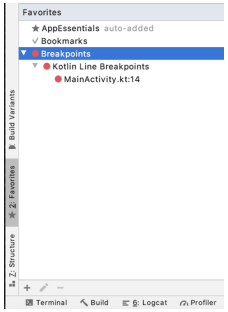
```
10  class MainActivity : AppCompatActivity() {
11
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContentView(R.layout.activity_main)
15
```

167

167

Debugging

- Debugging using Android Studio
  - Viewing all Breakpoints

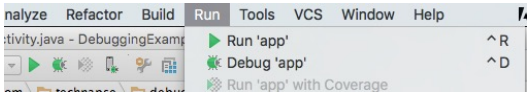


168

168

Debugging

- Debugging using Android Studio
  - Run the app in Debug mode



  - Shortcut

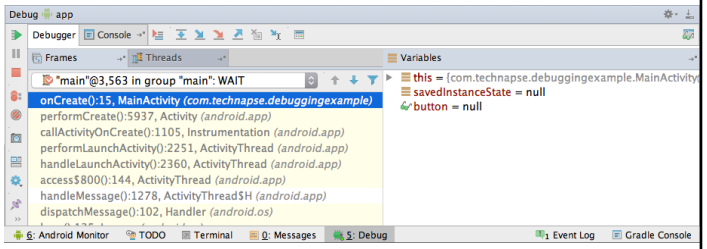
Ctrl + D

169

169

Debugging

- Debugging using Android Studio
  - Debug Window

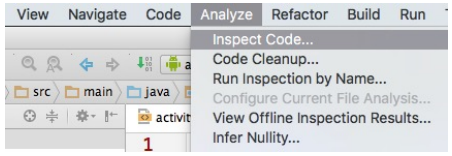


170

170

Static Analysis

- Android Lint
  - Lint is an Android specific code analysis tool.
  - Android Lint uses its knowledge of the Android frameworks to look deeper into your code and find problems that the compiler cannot.
  - Running Lint



171

171

Activity Life-Cycle

Amit Gulati, amit.gulati@gmail.com

172

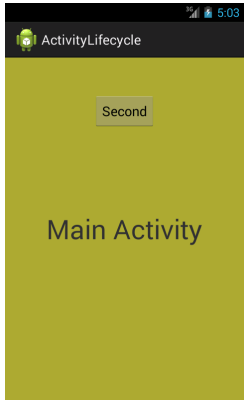
Activity Life-Cycle

- Active/Running
  - Activity is in the foreground of the screen.
  - The user is interacting with the Activity.
  - Example:
    - **MainActivity** is fully visible, focused and touch events will go to this Activity.
    - MainActivity is Active

Activity Stack

MainActivity

Launcher



173

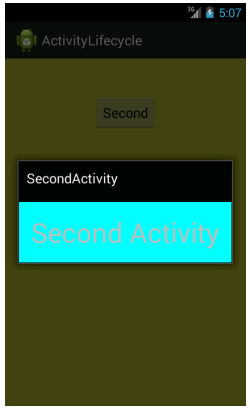
173

Activity Life-Cycle

- Paused/Visible
  - Activity is visible but does not have focus.
    - Obscured by another activity but not fully.
  - Paused activity does not receive user input events.
  - Example:
    - MainActivity is Paused.

Activity Stack

SecondActivity
MainActivity
Launcher



174

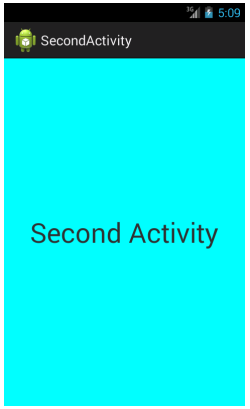
174

Activity Life-Cycle

- Stopped
  - Activity is fully obscured by another activity.
  - Activity in stop mode retains all state information and remains in memory.
  - Example:
    - MainActivity is now Stopped.

Activity Stack

SecondActivity
MainActivity
Launcher



175

175

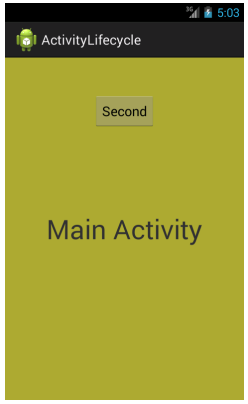
Activity Life-Cycle

- Terminated/Destroyed
  - Activity is terminated, if
    - Back button is pressed by user.
    - System requires memory and Activity is not visible to user
    - **finish**() method called in Activity
  - Example:
    - SecondActivity is terminated.

Activity Stack

MainActivity

Launcher



176

176

Activity Life-Cycle

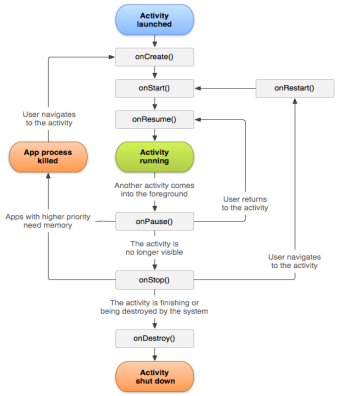
- Activity Lifecycle methods
  - Methods called by the Android runtime when activity state changes.
  - Following functions are called as a result of activity state transition

    void **onCreate**(Bundle savedInstanceState)
    void **onStart**()
    void **onRestart**()
    void **onResume**()
    void **onPause**()
    void **onStop**()
    void **onDestroy**()

177

177

Activity Life-Cycle Flowchart

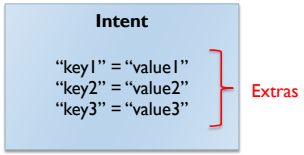


178

178

Activity and Data Flow

- Passing data from one activity to another
  - We can attach extra values to an Intent.
  - Any extra value attached to the Intent will be received by the receiver Activity.
  - Intent Extras are nothing by key, value pairs attached to Intent.
    - blble across process boundaries



179

179

Activity and Data Flow

- Passing data from one activity to another
  - Adding primitives as extras to Intent

    Intent **putExtra**(String *name*, [int, float, double, String] *value*)

  - Adding Object as extra to Intent

    Intent **putExtra**(String *name*, Parcelable *p*)

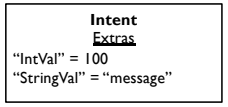
180

180

Activity and Data Flow

- Passing data from one activity to another
  - Adding values as extra to Intent

```
var intent = Intent(this, SecondActivity::class.java)
intent.putExtra("IntVal", 100)
intent.putExtra("StringVal", "message")
```



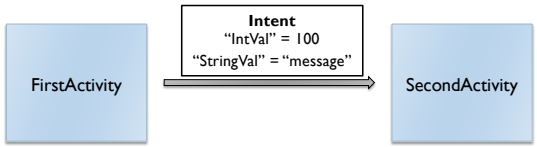
181

181

Activity and Data Flow

- Passing data from one activity to another
  - Launching Activity and passing Intent with Extras

startActivity(intent)



182

182

Activity and Data Flow

- Passing data from one activity to another
  - Getting access to Launching Intent

Intent **getIntent** ()

- method of Activity class
- provides access to the Intent that was used for launching the Activity.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    //get access to intent that launched this Activity
    var launchingIntent = getIntent()
}
```

183

183

Activity and Data Flow

- Passing data from one activity to another
  - Getting access to Launching Intent and data attached to it

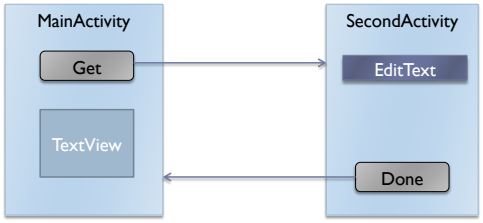
```
//get access to intent that launched this Activity
val launchingIntent = getIntent()
//reading attached data from the launching intent
if (launchingIntent.extras != null) {
    val intVal = launchingIntent.getIntArrayExtra("IntVal")
    val stringVal = launchingIntent.getStringExtra("StringVal")
} else {
    Log.i("MainActivity", "No Extras")
}
```

184

184

Activity and Data Flow

- Returning a Result back to previous Activity
  - Often we want an Activity to return a value back to the Previous Activity



185

185

Activity and Data Flow

- Returning a Result back to previous Activity
  - Start an activity that returns a result value

public void **startActivityForResult** (Intent *intent*, int *requestCode*)

requestCode, An integer that uniquely identifies the result request made to a sub-activity.

- Method is used in-place of calling startActivity().
- Allows the Launched Activity to return a value back to the Launching Activity.

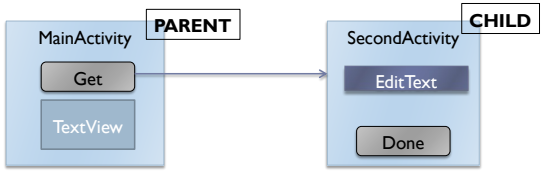
186

186

Activity and Data Flow

- Returning a Result back to previous Activity
  - Start an activity that returns a result value

```
private val SECOND_ACTIVITY_ID = 101
fun launchSecond(view: View) {
    val intent = Intent(this, SecondActivity::class.java)
    startActivityForResult(intent, SECOND_ACTIVITY_ID)
}
```



187

187

Activity and Data Flow

- Returning a Result back to previous Activity
  - Method that allows a child Activity to send an Intent back to the parent Activity.

void **setResult** (int *resultCode*, Intent *data*)

resultCode,	*RESULT_OK*,	User successfully selected/entered the data.
	RESULT_CANCELLED,	User decided not to select/ enter data.
data,	Intent object that contains data to be sent back (attached as extras to the Intent)	

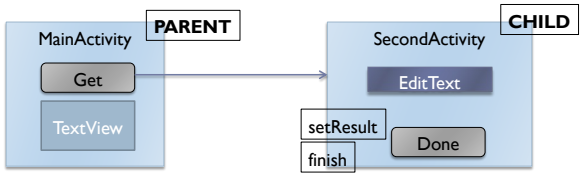
188

188

Activity and Data Flow

- Returning a Result back to previous Activity
  - Child Activity returning data back to Parent

```
val intent = Intent()
intent.putExtra("key", "value")
setIntent(intent)
finish()
```



189

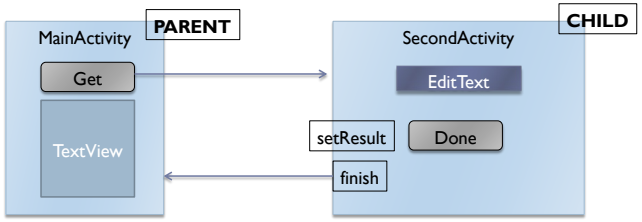
189

Activity and Data Flow

- Returning a Result back to previous Activity
  - Going back to previous Activity (back button).

void **finish**()

  - This method terminates the current Activity.
  - Results are propagated back to the parent Activity.



190

190

Activity and Data Flow

- Returning a Result back to previous Activity
  - Method that is executed in the Parent Activity when the Child calls **setResult**() and then **finish**()

protected void **onActivityResult** (int *requestCode*, int *resultCode*, Intent *data*)

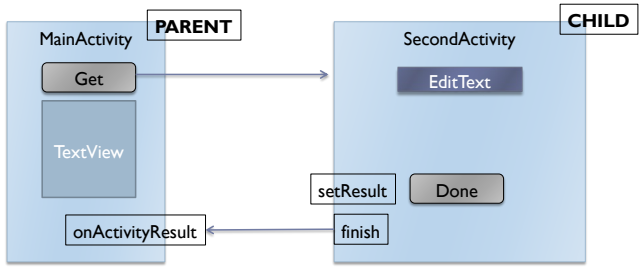
requestCode,	paramater that was passed to the startActivityForResult,
to	uniquely identify the request.
resultCode,	Result value as set by the sub-activity using the
setResult()	method.
	RESULT_OK, the operation ended successfully.
	RESULT_CANCELED, child activity failed.
data,	Intent object holding the data for parent.

191

191

Activity and Data Flow

- Returning a Result back to previous Activity
  - Method that is executed in the Parent Activity when the Child calls **setResult**() and then **finish**()



192

192

Activity State

Amit Gulati, amit.gulati@gmail.com

198

Activity State

- Activity is terminated in the following Scenarios
  - System runs low on memory
    - As a result it may terminate Activities that are not visible to the user.
  - Configuration Change
    - Device Orientation Change
- Activity Termination Scenario must be handled by the programmer.
  - Preserve Activity instance data before Activity terminated
  - Android system provides a mechanism that allows the programmer to save the state of an Activity (before it is terminated), so that it can be later restored.

199

199

Activity State

- Preserve Activity data before Activity terminated
  - Methods that is called before an Activity is killed because of system killing a process.

void **onSaveInstanceState** (Bundle *outState*)

outState, Bundle object can be used to write key-value pair that will be made available to the activity in the onCreate() method, if it is restarted by the system.

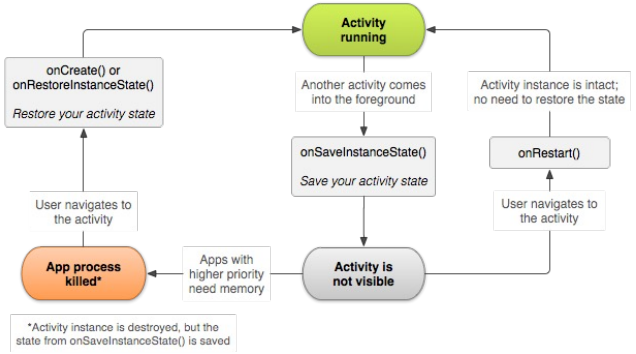
```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    outState.putString("key", "value")
    outState.putInt("key", 0)
}
```

200

200

Activity State

- Activity Termination Scenario



201

201

Activity State

- Retrieve Activity state when Activity re-starts
  - When the system restarts the Activity that was killed, the bundle object is passed to the Activity in the **onCreate**() method.
  - Example

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    if (savedInstanceState != null) {
        val stringVal = savedInstanceState.getString("StringVal")
        var intVal = savedInstanceState.getInt("IntVal")
    }
```

202

202

Activity State

- Handling configuration changes yourself
  - In some cases you may not want the Activity to be terminated for a configuration change.
  - You can declare that your Activity will handle configuration changes itself.
  - In the manifest file add the following attribute to the <activity> registration

android:configChanges

"orientation"
"screenSize"
"keyboardHidden"
"layoutDirection"

203

203

Activity State

- Handling configuration changes yourself
  - Implement the onConfigurationChanged function

public void **onConfigurationChanged**(Configuration *newConfig*)

204

204

Activity State

- Preserving Objects
  - Retain an object during configuration change using Activity API
    - Override the method, and return reference to object you want to preserve

```
public Object onRetainNonConfigurationInstance()
```

    - Once new Activity created, get access to the retained object

```
public Object getLastNonConfigurationInstance()
```

205

205

Fragments

Amit Gulati, amit.gulati@gmail.com

222

Why Fragments?

- Android screen Sizes
  - Multiple Screen sizes are available for phones.
  - New form factors emerging
    - Tablet
    - Auto
    - Wear
  - Activity not the best representative of the User Interface
    - Activity treats the whole screen as a representation for a single screen and a single task that the user can accomplish.
    - Activity not a scalable and flexible mechanism for building apps across devices.

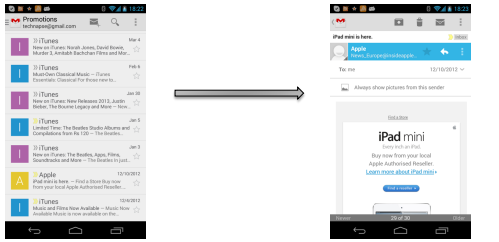
223

223

Why Fragments?

- Activity and Phone
  - Each Screen is represented by an Activity.
  - A user navigates between Activities.
    - On smaller device Activity navigation is fine.

Email List Activity **Email Detail Activity**



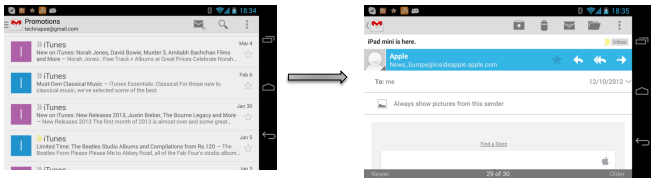
224

224

Why Fragments?

- Activity and Tablet
  - On tablets, Activity navigation is not the best use of Screen space

Email List Activity **Email Detail Activity**



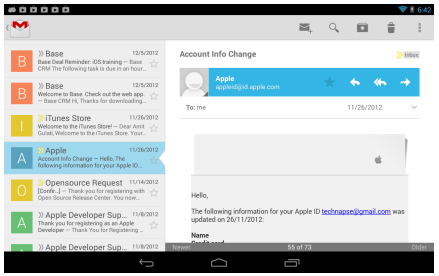
225

225

Why Fragments?

- Activity and Tablet
  - Enough room to display multiple content in the same Screen

List **Detail**



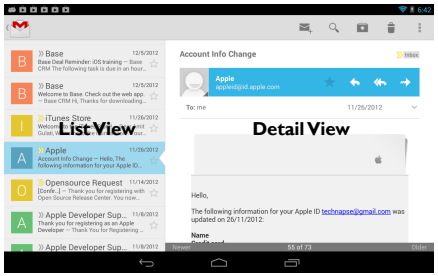
226

226

Why Fragments?

- Activity and Tablet
  - Split screen configuration also possible via multiple Views in Activity

Email Split Activity



227

227

Why Fragments?

- Activity and Tablet
  - Disadvantages of Single Activity – multiple View for configuring split screen.
    - Single Activity class handles the logic for two Screens.
    - Activity code becomes large.
    - Reuse of View (part of the Activity) challenging.
    - Leads to code management issues.
  - What is required is something like a mini-Activity or a sub-Activity
    - So that we will be able to split a Single Activity in to self contained sub-components.

228

228

What is a Fragment?

- Fragment
  - Represented by android.app.**Fragment** class
  - More like a sub-activity
  - Has its own XML layout, lifecycle, and receives its own input events.

FirstFragment.java

```
public class FirstFragment
       extends Fragment {

   public FirstFragment() {

   }
}
```

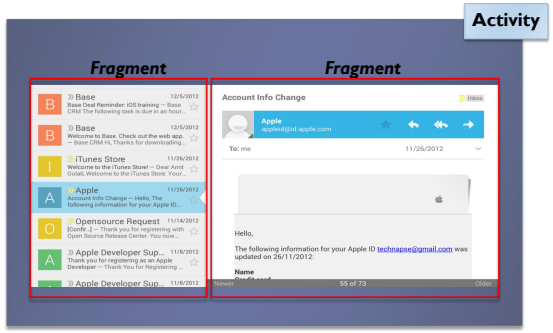
fragment_first.xml

229

229

What is a Fragment?

- Fragment
  - Fragments are always embedded inside an Activity



230

230

What is a Fragment?

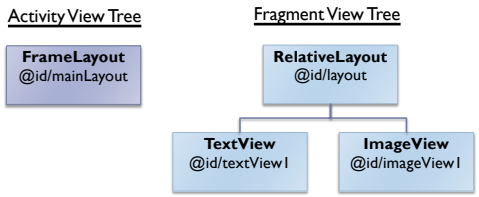
- Fragment
  - Fragments are not independent Android components.
    - Not registered in the manifest.
    - Cannot be launched via Intents.
    - Android Runtime is not even aware of Fragments
  - Fragments are always embedded inside an Activity
    - Activity provides a ViewGroup container in which the View of the Fragment is added.
    - For the Android View/Drawing System, Fragment looses its Identity once it is added to the Activity, as the View tree of Activity gets updated with the Fragments View Tree.

231

231

What is a Fragment?

- Fragment
  - Fragments are always embedded inside an Activity
    - By default Activity View tree and Fragment View tree are independent entitied.

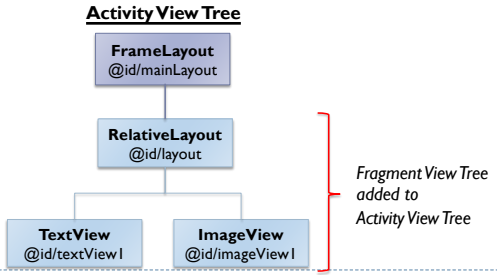


232

232

What is a Fragment?

- Fragment
  - Fragments are always embedded inside an Activity
    - When Fragment is added to activity, activities view tree is updated with the fragment view tree.
    - This is the only way to display Fragment on the Screen

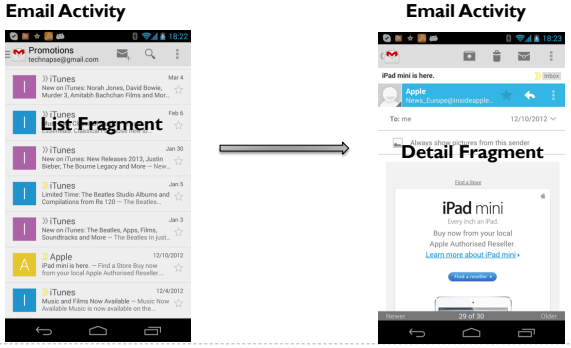


233

233

What is a Fragment?

- Fragment
  - Application running on a phone

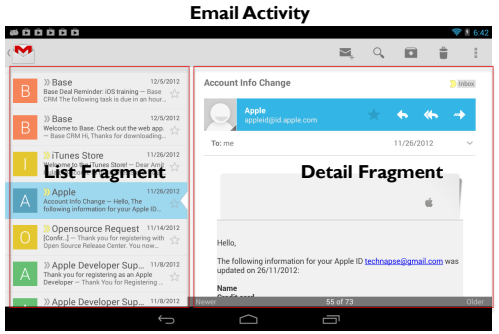


234

234

What is a Fragment?

- Fragment
  - Application running on a tablet



235

235

Fragments

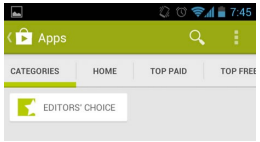
- Benefits Fragments?
  - Light weight
  - Create flexible UI's that can be configured for Tablets and Handsets.
  - Smaller independent modules which are easier to manage and Re-use
    - Fragment represents an independent modular section of Activity.
    - Independent Fragments can be re-used in multiple projects.

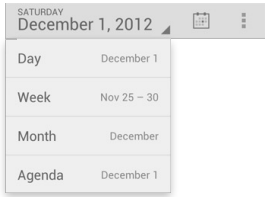
236

236

Android and Fragments

- Fragments are used all over the Android SDK
  - Better Screen Navigation
    - ViewPager uses Fragments
    - Create Tabbed UI
    - Create Navigation List based Interface
  - Dialog Boxes
  - Preferences UI





237

237

Fragments

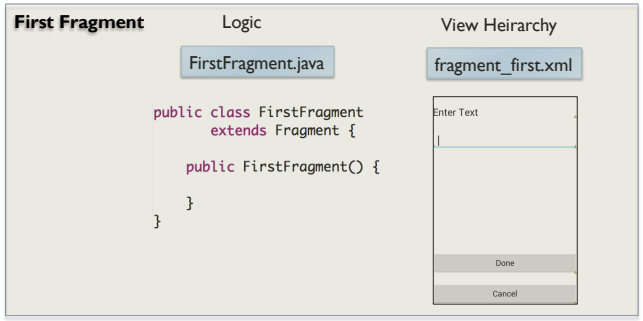
- Types of Fragments
  - **UI Fragments**, present a View on the screen.
  - **Background Fragments** do not present a View on the screen.

238

238

Fragments

- UI Fragments
  - A UI Fragment requires a Java class and XML Layout.

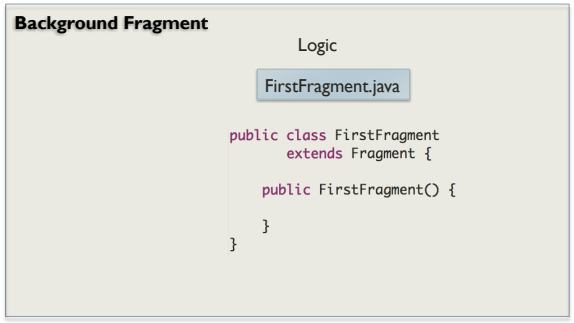


239

239

Fragments

- Background Fragments
  - A Background Fragment only requires a Java class.

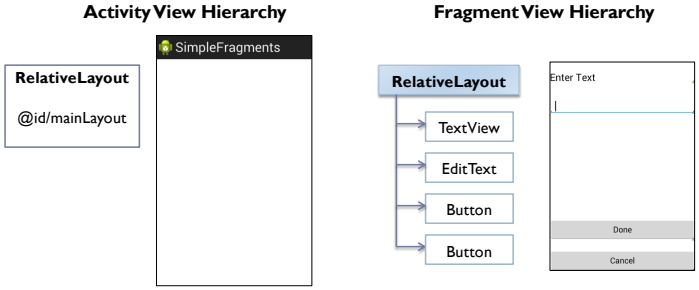


240

240

Fragments

- UI Fragments
  - Activity provides a container in its View hierarchy where the Fragments View hierarchy can be attached.

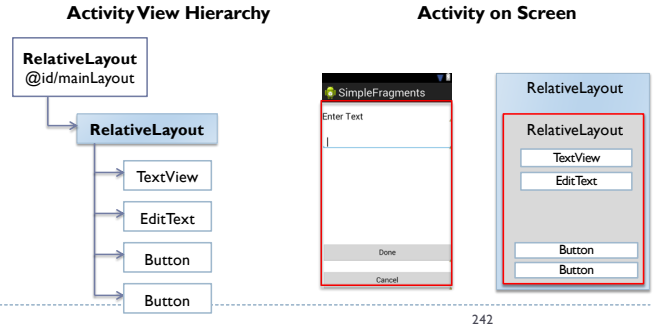


241

241

Fragments

- UI Fragments
  - When a Fragments View is to be displayed on the screen it is **attached** to the Activity's hierarchy.

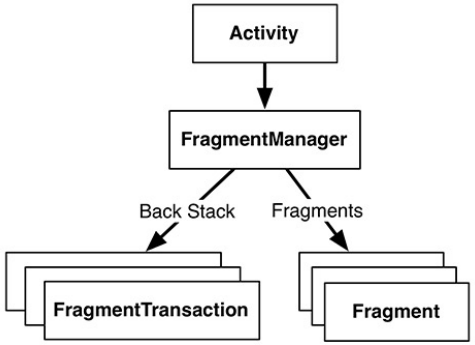


242

242

Managing Fragments

- FragmentManager



266

266

Managing Fragments

- FragmentManager
  - Every Activity (beginning Android Honeycomb 3.0) has a Fragment Manager.
  - Used for Managing Fragments in the Activity.
  - Common tasks for which FragmentManager object is used for
    - Finding Fragments in an Activity, given a tag or id.
    - Creating a Fragment Transaction, so that Fragments can be added/removed/replaced from the Activity.
    - Managing the Fragments in the Activity back-stack.

267

267

Managing Fragments

- FragmentManager
  - Get access to the FragmentManager

FragmentManager **getFragmentManager**()

package **android.app**;

- Example

```
FragmentManager manager =
            getFragmentManager();
```

FragmentManager **getSupportFragmentManager**()

- Example

package **android.support.v4.app**;

```
FragmentManager manager =
            getSupportFragmentManager();
```

268

268

Managing Fragments

- FragmentManager
  - Begin a Fragment Transaction, to dynamically add/remove/replace Fragment from an Activity

FragmentTransaction **beginTransaction**()

- Any changes to Fragments in an Activity can be made only after this method call.
- A fragment transaction can only be created/committed prior to an activity saving its state.
  - Do not commit a Transaction, before onStart(), or onResume(), or After onPause() or onStop().

270

270

Fragment Operations

- FragmentTransaction
  - Allows us to dynamically add/remove/replace Fragments from an Activity.
  - Provides ability to save the Fragment operations on the Back stack so that user can navigate back.
  - FragmentTransaction is created using the FragmentManager.

```
FragmentManager manager =
            getFragmentManager();

FragmentTransaction trans = manager.beginTransaction();
```

271

271

Fragment Operations

- FragmentTransaction
  - Adding UI Fragment to Activity

FragmentTransaction **add**(int *containerViewId*, Fragment *fragment*, String *tag*)

containerViewId,	Id of the container in the Activity, where Fragments View will be added.
fragment,	Fragment object to be added to Activity.
tag,	Tag to be attached to the Fragment.

272

272

Fragment Operations

- FragmentTransaction
  - Adding Background Fragment to Activity

FragmentTransaction **add**(Fragment *fragment*, String *tag*)

fragment,	Fragment object to be added to Activity.
tag,	Tag to be attached to the Fragment.

  - Background Fragment does not have a View.
  - Hence, it does not require the Id of the container in Activity.

273

273

Fragment Operations

- FragmentTransaction
  - Removing a Fragment from Activity

FragmentTransaction **remove**(Fragment fragment)

fragment,	Fragment object to be removed from Activity.

  - Removing a Fragment from Activity, lead to destroying of the Fragment View and Fragment object itself
  - Fragment object is detached from Activity.

274

274

Fragment Operations

- FragmentTransaction
  - Detach a Fragment from User Interface

FragmentTransaction **detach**(Fragment fragment)

fragment, Fragment object to be removed from Activity.

- Fragment's View is removed from Activity View container.
- Fragment's View gets destroyed.
- Fragment object is not removed from Activity.
- Same as when a Fragment is put on the back stack.
  - Fragment is removed from the UI, however its state is still being actively managed by the fragment manager.

275

275

Fragment Operations

- FragmentTransaction
  - Attach a previously detached Fragment to the User interface

FragmentTransaction **attach**(Fragment fragment)

fragment, Fragment object to be Attached .

- Re-attach a fragment after it had previously been detached from the UI with detach(Fragment).
- View hierarchy is re-created, attached to the UI, and displayed.

276

276

Fragment Operations

- FragmentTransaction
  - Commit a Fragment transaction

    FragmentTransaction **commit**()
    - Schedules a commit of the fragment transaction.
    - The commit does not happen immediately; it will be scheduled as work on the main thread to be done the next time that thread is ready.

277

277

Fragment Transaction and Back

- Fragments are not aware of Back button
  - Activity is removed from the Stack if back button is pressed.
  - This can cause disconnected User Experience.
  - Example

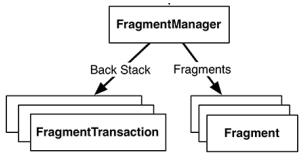
What happens when the user presse the back button?

279

279

Fragment Transaction and Back

- Fragments can be added to the FragmentManager back stack.
  - FragmentManager not only manages Fragments but also a Fragment back stack.
  - Fragment Back Stack contains Fragment Transactions.
  - Fragment Manager first unwinds the FragmentTransactions in the back stack when back button is pressed.



280

280

Fragment Transaction and Back

- Saving the FragmentTransaction to the Back Stack
  - Method in FragmentTransaction class that adds the transaction to the back stack

FragmentTransaction **addToBackStack**(String *name*)

name, An optional name for this back stack state, or null.

- Must be called before **commit**.

281

281

Navigating the Back Stack

- FragmentManager allows you to programmatically control the back stack.
  - Get a count of items in the Back Stack

```
public int getBackStackEntryCount ()
```

  - Pop the Back Stack

```
public void popBackStack ()
```

282

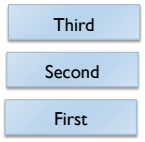
282

Navigating the Back Stack

- FragmentManager allows you to programmatically control the back stack.
  - Pop the Back Stack

```
public void popBackStack (String name, int flags)
```

name, Name of the transaction to pop from back stack.
flag, 0, or POP_BACK_STACK_INCLUSIVE




```
FragmentManager manager =
            getFragmentManager();
manager.popBackStack("Second", 0);
```

283

283

Fragment State

- Saving / Restoring State of Fragment
  - Fragments can be retained so that Fragment is not destroyed even when the enclosing Activity is destroyed.

public void **setRetainInstance** (boolean retain)

retain, if true, Fragment object is not destroyed.

286

286

Fragment Life-Cycle

- Life-Cycle of Fragment
  - Mostly follows the Life-Cycle of an Activity
  - Lifecycle callback methods are called by the hosting Activity rather than the Android System.
    - Activity callbacks are called by the ActivityManager (Android System component)
  - Fragment object may be in 4 different states
    - Simple Java Object
    - Attached to an Activity
    - Displayed in Activity
    - Destroyed

287

287

Fragment Life-Cycle

- Fragment Object
  - Statically – Using XML file
    - <fragment> tag is parsed, Fragment object is created
    - **onInflate**() method is called.

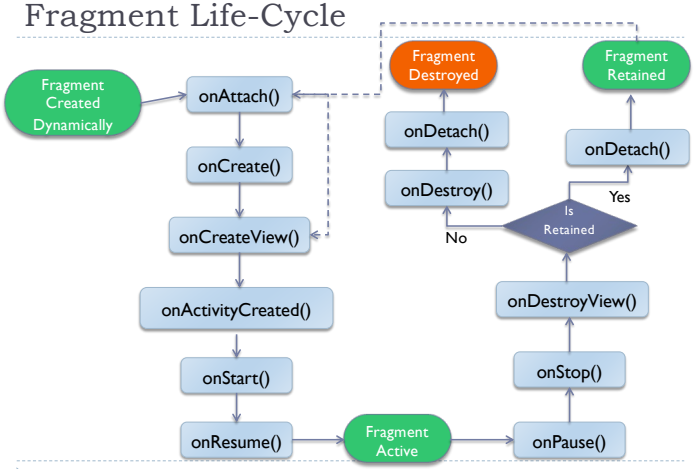
public void **onInflate** (Context *cxt*, AttributeSet *attrs*, Bundle *savedInstanceState*)

ctx,	Context that is inflating the Fragment.
AttributeSets,	Attributes of fragment element that was used for inflating the fragment.
savedInstanceState,	In case fragment is being recreated by system, the bundle object will contain the saved values.

288

288

Fragment Life-Cycle

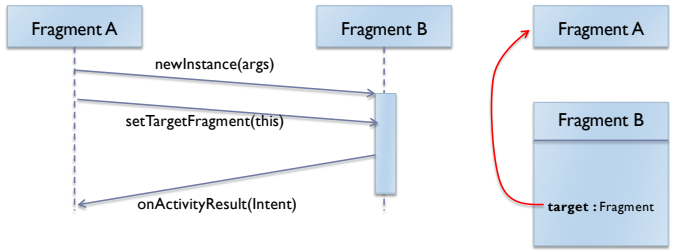


289

289

Fragment Communication

- Fragment communication using Target Fragment
  - Setup target relationship between 2 fragments.
  - A fragment can send data to its target fragment.



304

304

Fragment Communication

- Fragment communication using Target Fragment
  - Method used for setting up target relationship

void **setTargetFragment** (Fragment fragment, int requestCode)

fragment,	The fragment that is the target of this one.
requestCode,	Optional request code

305

305