Agenda

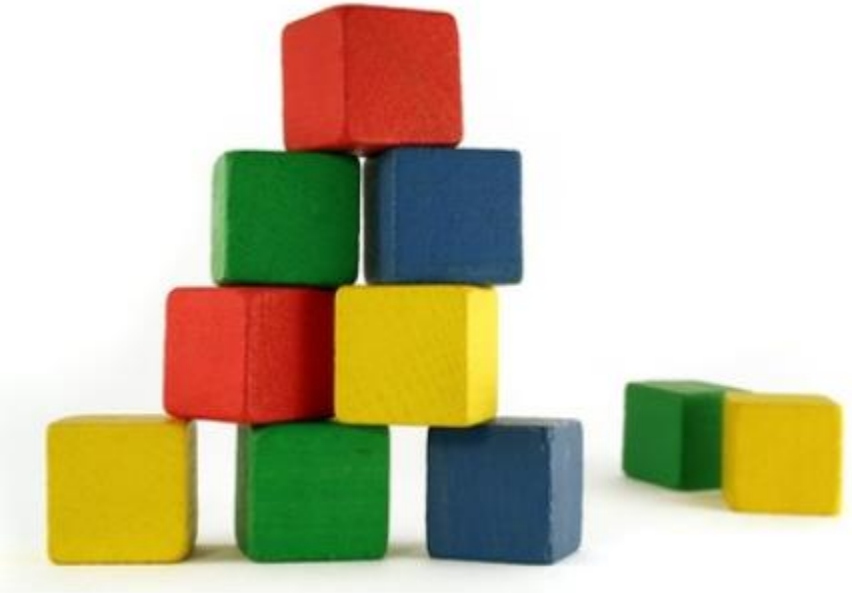
- Introduction
- Fundamentals
- Development
- Publishing App in Google Play

Introduction

- Android is a software stack for mobile devices
- Includes an operating system, middleware and key applications
- Developed by Open Handset Alliance led by Google.
- Unveiling of the Android platform was announced on 5th November 2007 with the founding of OHA, a consortium of 86 hardware, software and telecom companies
- Free and open source, based on modified Linux kernel
- More than 50, 000 Android apps available
- In 2012 first quarter, Android had a 59% share of the smartphone market
- Android License
  - Android is released under version 2 of the Apache Software License (ASL)
  - The Apache license allows manufacturers and mobile operators to innovate using the platform

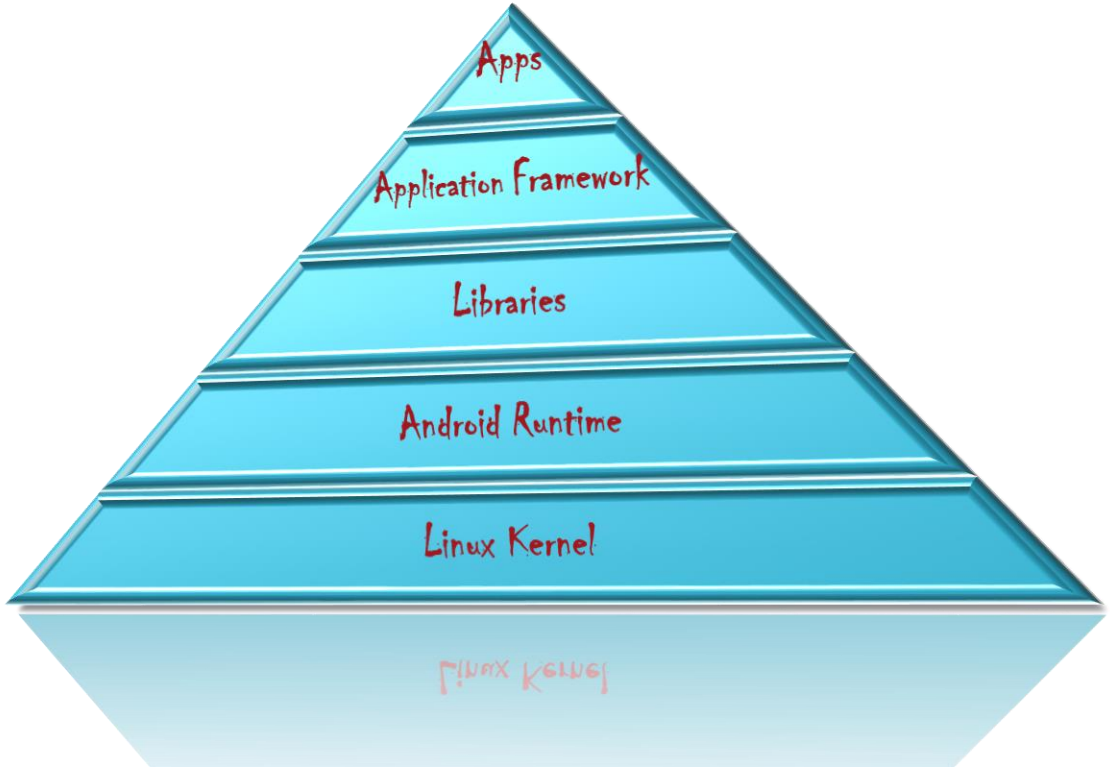
Introduction

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source Webkit engine
- Optimized graphics
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)



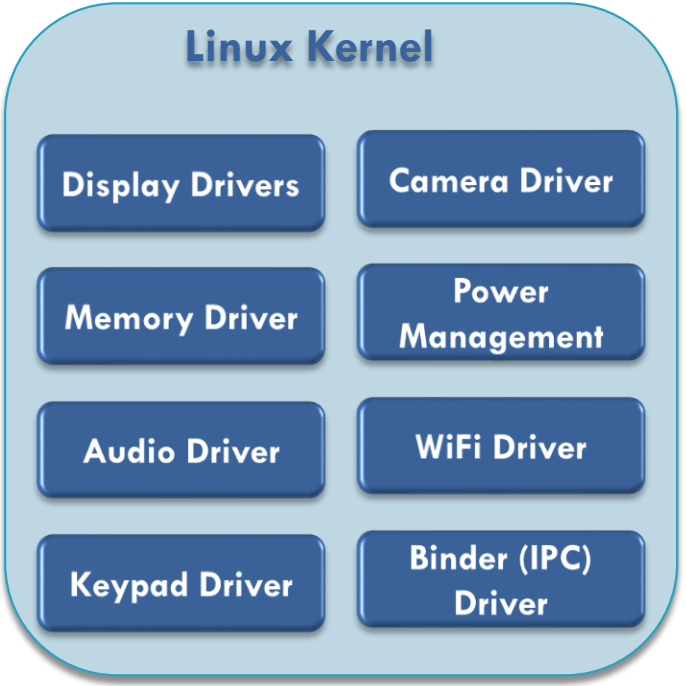
Fundamentals

Android Architecture



Linux Kernel

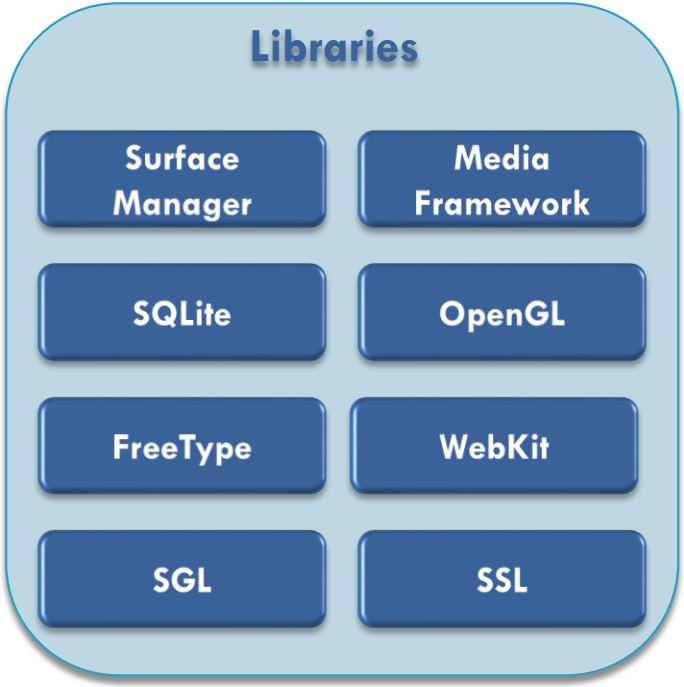
- Works as a HAL (Hardware Abstraction Layer)
- Performs
  - Device driver management
  - Memory Management
  - Process management
  - Networking
- ASHMEM (Android SHared y) is used to share data between processes
- Uses LRU to manage proces



What is LRU?

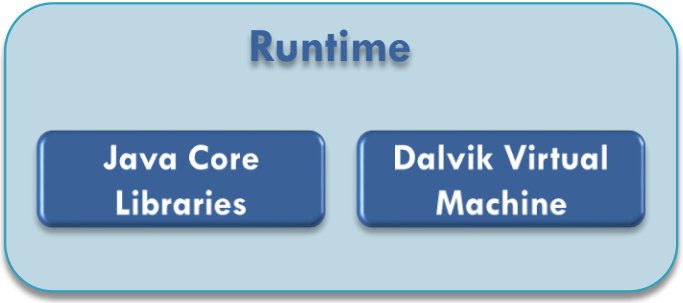
Libraries

- C/C++ libraries interfaced with Java
- Surface manager (handling UI Windows)
- 2D and 3D graphics
- Media codecs
- SQLite database
- Web browser



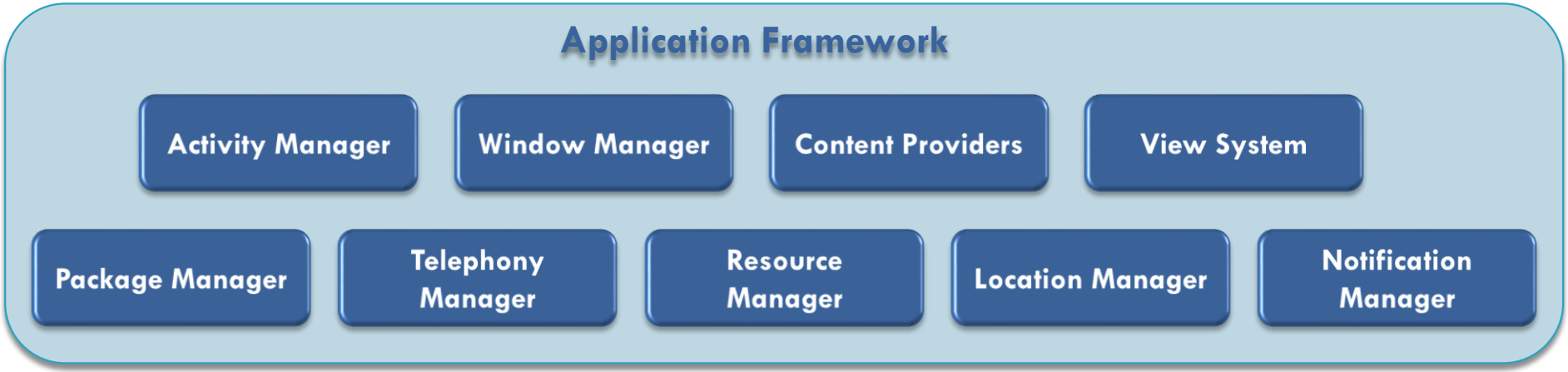
Android Runtime

- Dalvik VM – optimized for embedded systems
  - Optimized for Embedded Systems
  - .Java to .Class to .Dex files
  - Compact and efficient than class files
  - Limited memory and battery power
- Java Core libraries



Application Framework

- API interface
- OOTB Managers



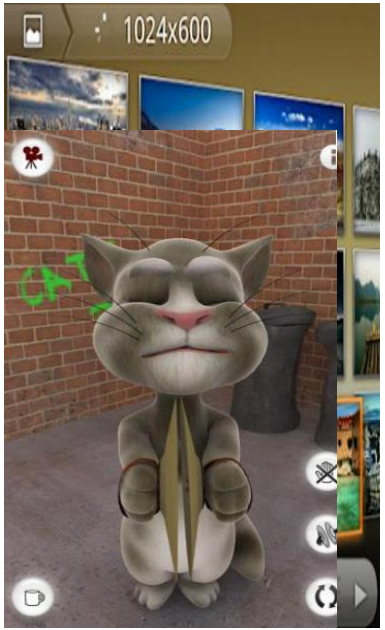
Applications

- Built-in Apps

Home Page

lace t

Browser



Gallery

The list is endless...
And Many more.....

Design considerations

Application Fundamentals

- Application Sandbox
- Based on Least privilege principle
- Multi-user Linux system with userId for each application
- Permissions granted on the basis of userId
- Each process has its VM
- Applications are deployed onto devices as .apk files

Application Building Blocks

- Activity
- Fragments
- Intent
- Service
- Content Provider
- Broadcast receiver
- Action Bar
- Views
- Layouts
- Resources

Activity

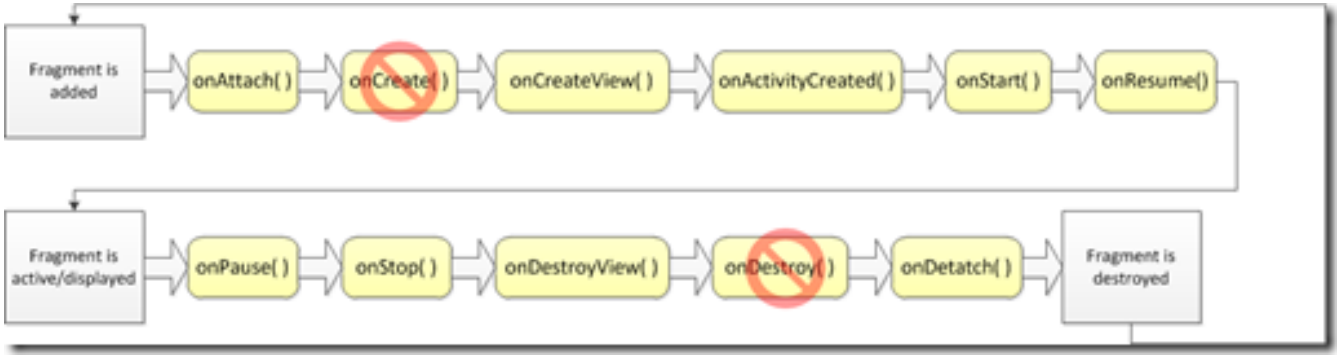
- Typically corresponds to one UI screen.
- One activity would serve as the entry point to the application.
  - <activity android:name=".activity.SplashActivity“ android:screenOrientation="portrait">
  - <intent-filter>
  - <action android:name="android.intent.action.MAIN" />
  - <category android:name="android.intent.category.LAUNCHER" />
  - </intent-filter>
  - </activity>
- Activities history are maintained on a Activity Stack which makes navigating between activities simple.
- Activities use Widgets called Views to draw on screen.
- Views can be defined in XMLs as well as Java code.

Activity Lifecycle

Fragments

- A Fragment represents a behavior or a portion of user interface in an Activity.
- Fragments decompose application functionality and UI into reusable modules.
- Fragments have their own lifecycle, state, and back stack.
- A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
- You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup.
- Fragments require API Level 11 or greater but android provides support libraries for previous version of android.

Fragment Lifecycle



Intent

- Abstract description of an operation to be performed
- An Activity can call startActivity(Intent) to start another Activity or call startActivityForResult() to expect result back from the Activity being started, OnActivityResult() method handles the returned result.
- Can be used to communicate with a background service using Context.startService(Intent) and Context.bindService(Intent, ServiceConnection, int)
  - Intent intent = new Intent(SplashActivity.this, HomePageActivity.class);
  - startActivity(intent);
- The primary pieces of information in an intent are:
  - action -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
  - data -- The data to operate on, such as a person record in the contacts database, expressed as Uri.
- Can be used to send broadcasts using Context.sendBroadcast(Intent)

Service

- Application component for long running operations while not interacting with user.
- Provide functionality for other applications to use.
- Service is a single instance and is managed by Android itself.
- Must have a corresponding <service> declaration in its package's AndroidManifest.xml.
- Runs in the main thread of the hosting process and not in a separate thread.
- Can be started with Context.startService() and Context.bindService().
- Content is represented by URI and MIME type.

Content provider

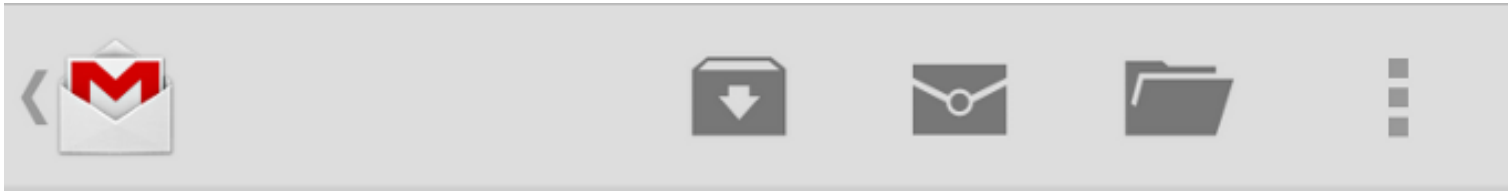
- Provide content to applications through ContentResolver interface
  - Eg. Photo gallery, Contacts etc.
- Required only if you need to share data between applications
- Methods available to query, insert, update and delete data from a Content provider
  - query(Uri, String[], String, String[], String) which returns data to the caller
  - insert(Uri, ContentValues) which inserts new data into the content provider
  - update(Uri, ContentValues, String, String[]) which updates existing data in the content provider
  - delete(Uri, String, String[]) which deletes data from the content provider

Broadcast Receiver

- Components which receive and act on broadcast events
- Can start an Activity in response to the broadcast received
- Can also use NotificationManager to alert the user
- Do not have a user interface
- Base class for code that will receive intents sent by sendBroadcast().
- Can be registered either dynamically using Context.registerReceiver() or statically through <receiver> tag in AndroidManifest.xml.
- Using LocalBroadcastReceiver, the intents broadcast never go outside of the current process.

ActionBar

- Provides a dedicated space for giving your app an identity and indicating the user's location in the app.
- Makes important actions prominent and accessible in a predictable way (such as Search).
- Supports consistent navigation and view switching within apps (with tabs or drop-down lists).
- The ActionBar APIs were first added in Android 3.0 (API level 11) but they are also available in the Support Library for compatibility with Android 2.1 (API level 7) and above.



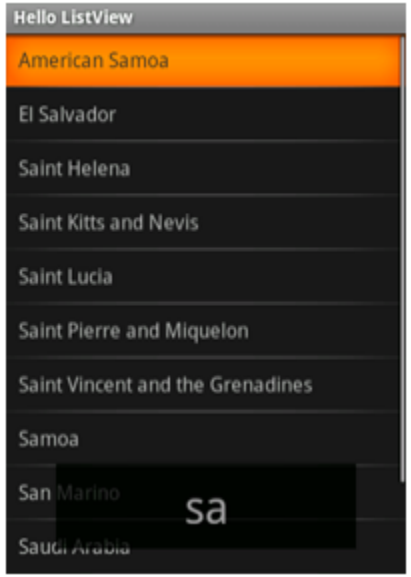
Views

- Basic building block for user interface components (Android base class View)
- Occupies a rectangular area on the screen and is responsible for drawing and event handling
- Base for *widgets*, used to create interactive UI components (buttons, text fields, etc.)
- Can be added from code or in XML layout files
- Specialized subclasses that act as controls or are capable of displaying text, images, or other content – TextView, ProgressBar, ImageView, Button etc.
- Operations
  - Set Properties
  - Set Focus
  - Set up event Listeners
  - Set Visibility

Layouts

- Container of all UI components, defines the layout structure (Android ViewGroup class)

ListView




```
public class HelloListView extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item, COUNTRIES));

      ListView lv = getListView();
      lv.setTextFilterEnabled(true);

      lv.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
          // When clicked, show a toast with the TextView text
          Toast.makeText(getApplicationContext(), ((TextView) view).getText(),
              Toast.LENGTH_SHORT).show();
        }
      });
    }
}
```

Tasks

- Task is what the user experiences as an application
- Task is a group of activities from a single or multiple apps arranged in a stack
- The activity on top of the stack is the activity currently running
- On BACK key hit, current activity is popped and previous activity comes on top
- Activity which began the task is called the root activity
- All activities in a task move as a unit, entire task can move to background or foreground

Android Manifest file

- Must have file for all applications
- Name must be AndroidManifest.xml
- It should be present in the application root directory
- Defines
  - Java package for the application (used as unique identifier for the app)
  - Application components, activities, services, broadcast receivers and content providers
  - Permissions required by the application
  - Permissions that other applications should have to interact with this application
  - Instrumentation classes that provide profiling for the app during development
  - Minimum level of Android API required by the application
  - Libraries that the application must be linked against

Android Manifest file

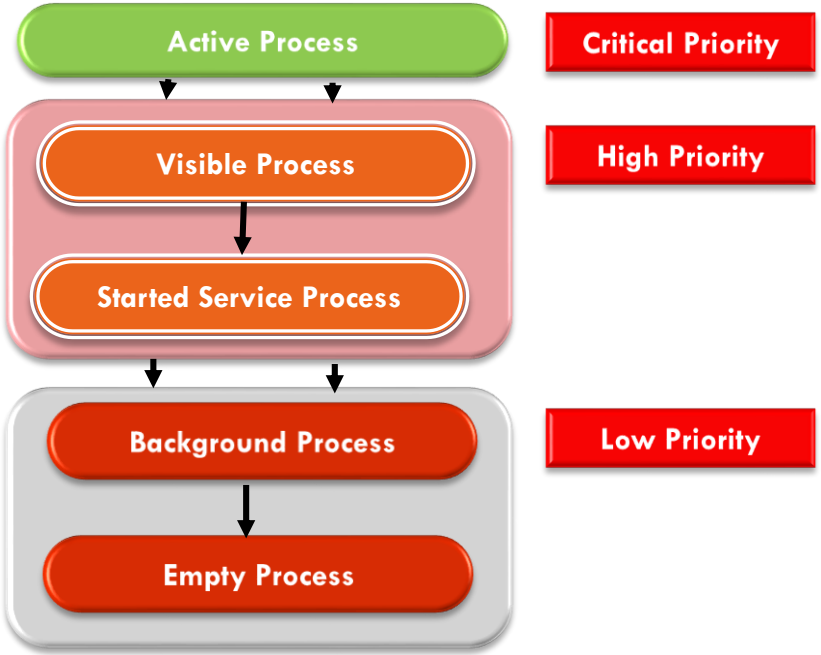
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.nagarro"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:icon="@drawable/splash"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Black.NoTitleBar" >
        <uses-library android:name="com.google.android.maps" />

        <activity
            android:name=".activity.SplashActivity"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".activity.PrepareRequestTokenActivity"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />

                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />

                <data
                    android:host="callback"
                    android:scheme="x-oauthflow-twitter" />
            </intent-filter>
        </activity>
```

Process and Threads

- Components run in a process
- A single UI thread is responsible for hosting running activity
- Components can be declared in manifest file to run in different processes using the process attribute, default process is set in the <application> element in AndroidManifest.xml
- All user actions and lifecycle notifications run in the main thread of the process
- No component should perform long running or blocking operations on the main thread
- Separate threads should be spawned for long running operations like networking or computational loops
- Android utility classes for creating threads – Looper, Handler, HandlerThread

Process Lifecycle

- **Foreground Process** - required for what the user is currently doing
- **Visible Process** - that doesn't have any foreground components, but still can affect what the user sees on screen (after onPause).
- **Service Process** - holding a Service - not directly visible to the user- relevant tasks
- **Background Process** - holding an Activity - not visible to the user - can get killed at any time (onStop has been called)
- **Empty Process** - doesn't hold any active application components (Cache to improve start-up time)



Development

- Java SDK
- Android SDK
- Eclipse IDE
- Android Development Tools (ADT) plugin for Eclipse

Android SDK Tools

- **android** - Lets you manage AVDs, projects, and the installed components of the SDK
- **Dalvik Debug Monitor Server (ddms)** - Lets you debug Android applications
- **dmtracedump** - Generates graphical call-stack diagrams from trace log files
- **Draw 9-patch** - Allows you to easily create a NinePatch graphic using a WYSIWYG editor
- **Android Emulator (emulator)** - A QEMU-based device-emulation tool that you can use to design, debug, and test your applications in an actual Android run-time environment
- **Hierarchy Viewer (hierarchyviewer)** - Lets you debug and optimize the user interface
- **hprof-conv** - Converts the HPROF file that is generated by the Android SDK tools to a standard format so you can view the file in a profiling tool of your choice
- **layoutopt** - Lets you quickly analyze and optimize your application's layouts for efficiency

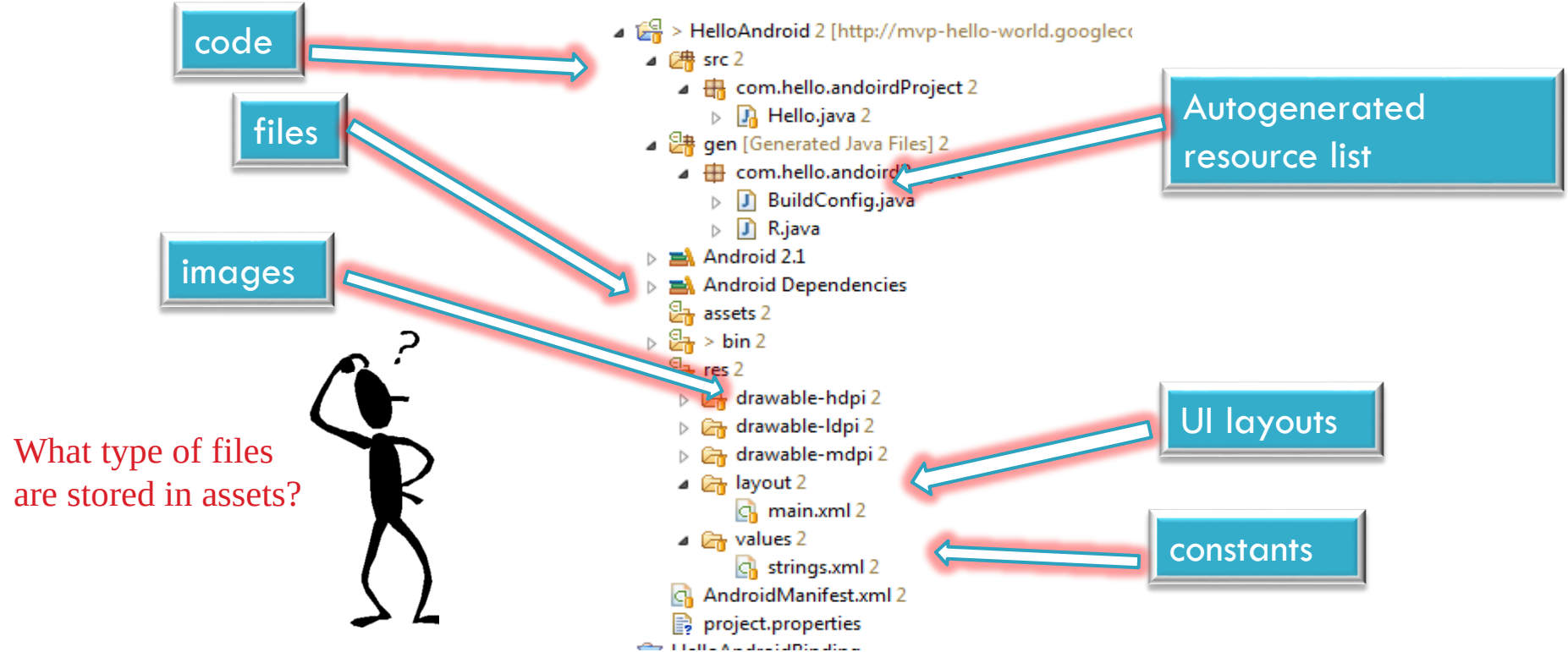
Why do we need Draw 9-patch?

Android SDK Tools

- **monkey** - Runs on the emulator or device and generates pseudo-random streams of user events to stress-test applications that you are developing, in a random yet repeatable manner.
- **monkeyrunner** - Provides an API for writing programs that control an Android device or emulator from outside of Android code.
- **proGuard** - Shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names.
- **sqlite3** – to access the SQLite data files created and used by Android applications.
- **traceview** - graphical viewer for execution logs saved by your application.
- **zipalign** - used to align .apk files after they have been signed.

Is there a tool for memory profiling and performance measurement?

Android Project Structure



Localization

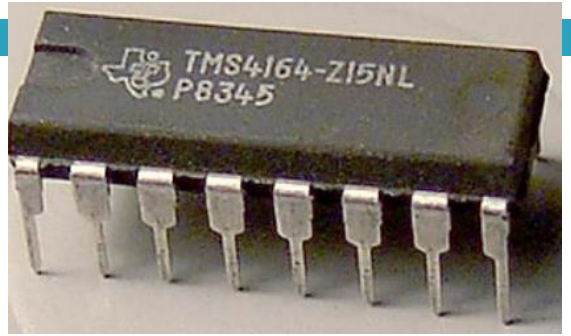
- Application where used should handle text, audio files, numbers, currency, and graphics in ways appropriate to the selected locale
- Locale based resources like Strings.xml, images etc.
- res/values/strings.xml as default, res/values-en/strings.xml for english
- res/drawable/ as default, res/drawable-ja/ for Japanese
- Alternate layouts for languages with long words (eg. German) - res/layout-de/main.xml
- Getting Locale –
  - String locale = context.getResources().getConfiguration().locale.getDisplayName();
- Date/number formatting per locale in Java code

Supporting Multiple Screens

- screen sizes are divided into small, normal, large and extra large
- Screen densities are divided into low (ldpi), medium (mdpi), high (hdpi) and extra high (xhdpi)
- Orientation could be Landscape or Portrait
- Dp (Density-independent Pixels) us
- Sp used to define text sizes
- Application can provide alternative resources for any of the generalized sizes and densities
- Automatic scaling done by Android at runtime if app does not provide alternative resources
- drawable-hdpi/, drawable-xhdpi/ and so on should contain bitmaps for different density screens
- layout-xlarge/, layout-large/ and so on should contain layouts for different screen sizes
- Land and port qualifiers for Landscape and portrait orientations

Data Storage - Internal Storage

- Application can save files directly on the device's internal storage
- Files remain private to the application
- By default, inaccessible to other applications and the user
- Files are removed on un-installation of application



What is the directory structure of internal storage?

Data Storage - External Storage

- A shared external storage can be used to save files
- It could be removable storage media like SD card or internal storage
- Files are accessible to other apps and user
- User can remove the storage media or can remove the files

When should we use External storage then?

How do we find out if external storage media is there?

Data Storage - SQLite Database

- SQLite database is fully supported by Android
- Database is accessible by any class in the application
- Not accessible from outside the application
- SQLiteOpenHelper to be sub-classed, tables can be created from overridden onCreate() method
- getWritableDatabase() and getReadableDatabase() for writing and reading from the database – they return SQLiteDatabase instance with methos for DB operations
- SQLiteDatabase.query() methods to execute queries
- SQLiteQueryBuilder for building complex queries
- Cursor object is returned pointing to rows returned

What do you mean by complex queries?

How do we create a SQLite database?

How can I query the data from outside the application?

Data Storage - SQLite Database

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE = "CREATE TABLE "
            + DICTIONARY_TABLE_NAME + " (" + KEY_WORD
            + " TEXT, " + KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```

Data Storage - Network Connection

- Web based services can be used to store and access data
- Network when available on the device can be used
- Android exposes classes and methods to do operations over the network in the packages
  - java.net.*
  - android.net.*

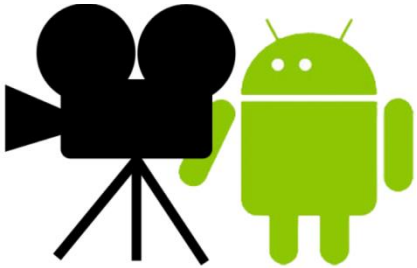
How do we publish services on the web?

Can we connect to services on the intranet?

What are the design and security considerations if we access services over the web?

Which classes and methods are available?

Camera Integration



- Android supports capturing Images using Camera API or via Intent
- Manifest Declarations when using Camera API :
  - Access Camera hardware using <uses-permission android:name=”android.permission.CAMERA”/
  - <uses-feature android:name="android.hardware.camera" />
  - <uses-permission android:name="*android.permission.WRITE_EXTERNAL_STORAGE*"/>
- Intent for Camera Image Capture : MediaStore.ACTION_IMAGE_CAPTURE
- Intent for Camera video Capture : MediaStore.ACTION_VIDEO_CAPTURE
- Android provides android.hardware.Camera class to integrate with Camera

How do we get the image object?

Can the app control the image resolution and size?

Camera Integration

- Cropping Images
  - Android supports intent to crop the captured image
  - Intent to crop captured image : "com.android.camera.action.CROP”
- Camera App selector
  - In a device there can be more than one camera apps (default camera, camera 360)
  - If you want the user to be shown a list of applicable camera applications to handle the request, then wrap your intent within another intent called ACTION_CHOOSER

Facebook Integration

- Facebook provides an android sdk to integrate with it
- This SDK provides following features:
  - User can post on her own wall
  - Import contacts
  - Send requests
  - Post on friends wall
- Android SDK can be downloaded from git repository : https://github.com/facebook/facebook-android-sdk
- For further reference please refer: https://developers.facebook.com/docs/mobile/android/build/

Maps Integration

- Google provides powerful API for Location and Map integration in Android Apps
- For same Android SDK with Google API's is required
- AndroidManifest.xml configuration for the map integration
  - <uses-library android:name="com.google.android.maps" />
- For map integration the activity needs to extend com.google.android.maps.MapActivity
- The layout of this activity should include <com.google.android.maps.MapView> to show the map
- To open the map, there is a unique map key for each system
- Follow the link https://developers.google.com/android/maps-api-signup?hl=en-US to generate map key.

What are geopoints?

What are pins and overlay?

If user moves the map, when shall we refresh the data?

Maps Integration

- Set Location Pins on Map
- You can search any address by using Geocoder api provided by Google.
- Google map provides the option to highlight the location using your own pins.
- For the same you need to create you own itemized overlay by extending ItemizedOverlay class
- Create your overlay item by using geopoints

App Publishing

- The Android system requires that all installed applications be digitally signed with a certificate whose private key is held by the application's developer.
- The important points to understand about signing Android applications are:
  - All applications must be signed with a suitable private key. The system will not install an application th not signed.
  - Self-signed certificates can be used to sign your applications. No certificate authority is needed.
  - The system tests a signer certificate's expiration date only at install time.
  - Keytool and Jarsigner — to generate keys and sign your application .apk files.
  - zipalign tool to optimize the final APK package.

How do we contact Google to host the application in Google Play?

References

- http://developer.android.com
- http://en.wikipedia.org/wiki/Android_(operating_system)