Chapter 1
Android Development Introduction

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html
&
Unlocking Android
by Frank Ableson, Charlie Collins, and Robi Sen.
ISBN 978-1-933988-67-2
Manning Publications, 2009.

Chapter 1 - Goals

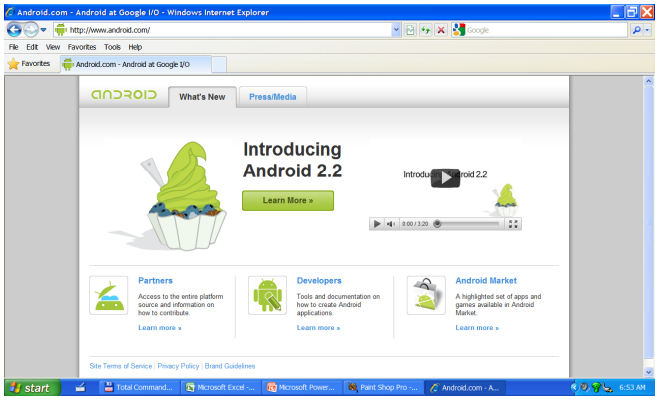
THE BIG PICTURE

1. *What is Android?*
2. *Overview development environment*

Chapter 1 - Resources

Android's web page

http://www.android.com/



3

What is Android?

- **Android** is an open-source software platform created by Google and the *Open Handset Alliance.*
- It is primarily used to power mobile phones.
- It has the capability to make inroads in many other (non-phone) embedded application markets.

4

What is Android?

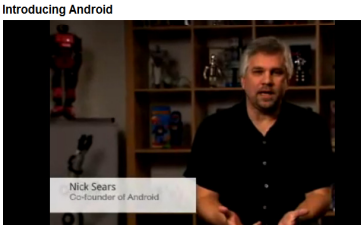
- **Android™** consists of a complete set of software components for mobile devices including:
  - an operating system,
  - middleware, and
  - embedded key mobile applications
  - a large market.

5

Why Android?

Listen from the project creators/developers (2.19 min)

- Nick Sears. Co-founder of Android
- Steve Horowitz. Engineering Director
- Dam Morrill. Developer
- Peisun Wu. Engineering Project Manager
- Erick Tseng. Project Manager
- Iliyan Malchev. Engineer
- Mike Cleron. Software Manager
- Per Gustafsson. Graphics Designer.
- etc…



- http://www.youtube.com/watch?v=6rYozIZOgDk&eurl=http://www.android.com/about/&feature=player_embedded
- You will hear statements such as "…currently it is too difficult to make new products … open software brings more innovation … choices … lower costs … more applications such as family planner, my taxes, understand my wife better, … "

6

What is Open Handset Alliance?

- Quoting from ***www.OpenHandsetAlliance.com*** page
- "… **Open Handset Alliance™**, a group of 47 technology and mobile companies have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.
- Together we have developed Android™, the first complete, open, and free mobile platform.
- We are committed to commercially deploy handsets and services using the Android Platform. "

7

Open Handset Alliance Members

Operators	Software Co.	Commercializat.	Semiconductor	Handset Manf
China Mobile China Unicom KDDI Corp. NTT DoCoMo Sprint Nextel T-Mobile Telecom Italia Telefónica Vodafone Softbank … Ericsson	Ascender Corp. eBay Esmertec Google LivingImage NMS Comm. Nuance Comm. PacketVideo SkyPop SONiVOX … Borqs	Aplix Noser Engineering Astonishing Tribe Wind River Systems Omron Software … Teleca	Audience Broadcom Corp. Intel Corp. Marvell Tech. Group Nvidia Corp. Qualcomm SiRF Tech. Holdings Synaptics Texas Instr. AKM Semicond. ARM Atheros Comm … EMP	ACER ASUS HTC LG Motorola Samsung ASUSTek Garmin Huawei Tech LG Samsung … Sony Ericsson Toshiba

8

See Android Developers

http://www.youtube.com/watch?v=7Y4thikv-OM

Short video (4 min.)
Showing *Dave Bort* and *Dan Borstein*, two members of the Android Open Source Project talk about the project.

9

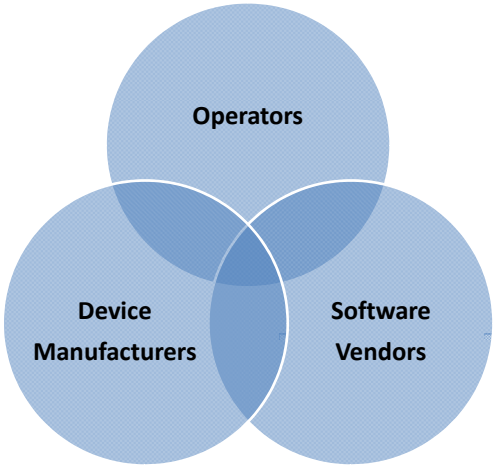
The Android Platform

Again, what did they say about Android?

- Android is a *software environment* built for mobile devices.
- It is ***not*** *a hardware platform.*
- Android includes:
  - Linux kernel-based OS,
  - a rich UI,
  - telephone functionality,
  - end-user applications,
  - code libraries,
  - application frameworks,
  - multimedia support, ...
- User applications are built for Android in Java.

10

Android's Context: Mobile Market Player$



Stakeholders:

Mobile **network operators** want to lock down their networks, controlling and metering traffic.

Device manufacturers want to differentiate themselves with features, reliability, and price points.

Software vendors want complete access to the hardware to deliver cutting-edge applications.

11

The Maturing Mobile Experience

Electronic tools of a typical business warrior

Not so long ago …	Today
1. Phone 2. Pager 3. PDA Organizer 4. Laptop 5. Portable music player 6. No Internet access / limited access	1. Smartphone 2. Laptop (perhaps!)

Tomorrow ?

12

The Maturing Mobile Experience

I want my 2015 Smartphone to act as …

Trying to answer: Tomorrow ?

1. Phone
2. Pager
3. PDA Organizer
4. High Quality Camera (still & video)
5. Portable music player
6. Portable TV / Video Player / Radio
7. Laptop
8. Play Station
9. GPS
10. Golf Caddy (ball retriever too)
11. Book Reader (I don't read, It reads to me)
12. Car / Home / Office Key
13. Remote Control (Garage, TV, …)
14. Credit Card / Driver's License / Passport
15. Cash on Demand
16. Cook, house chores
17. Psychologist / Mentor / Adviser
18. ????

13

Android vs. Competitors

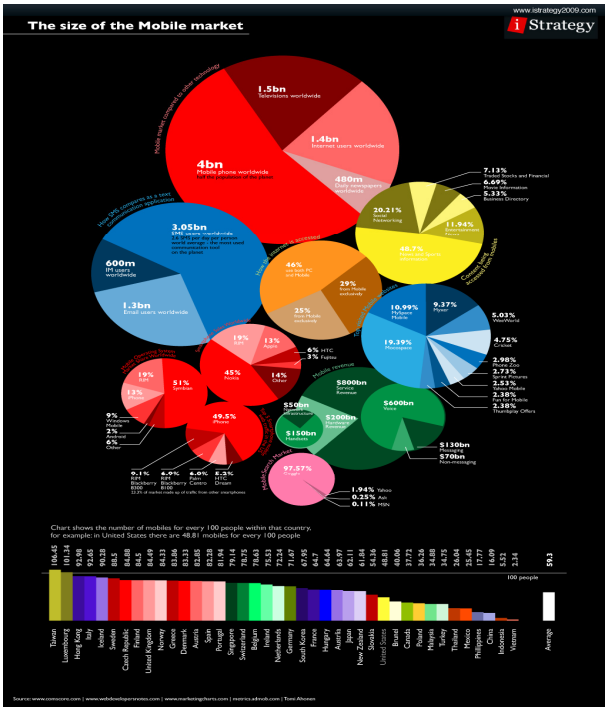
	1. Apple Inc. 2. Microsoft 3. Nokia 4. Palm 5. Research In Motion 6. Symbian

14

The Size of the Mobile Market

http://gizmodo.com/5489036/cellphone-overshare

[see appendix]



Android Components (Stack)

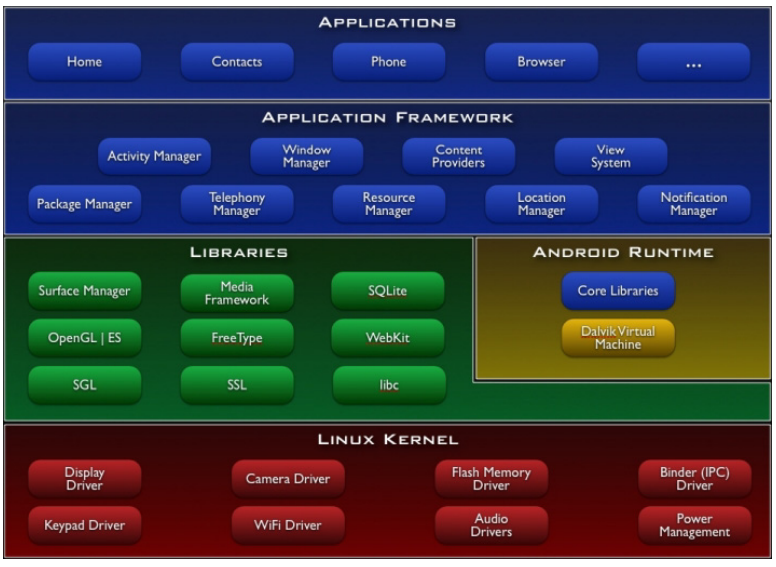
- The Android stack includes a large array of features for mobile applications.
- *It would be easy to confuse Android with a general purpose computing environment.*
- All of the major components of a computing platform are included.

Android Components

- **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** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES specification (hardware acceleration optional)
- **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, 4G, and Wi-Fi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

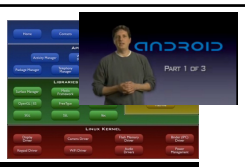
17

Android Components



18

Android Components



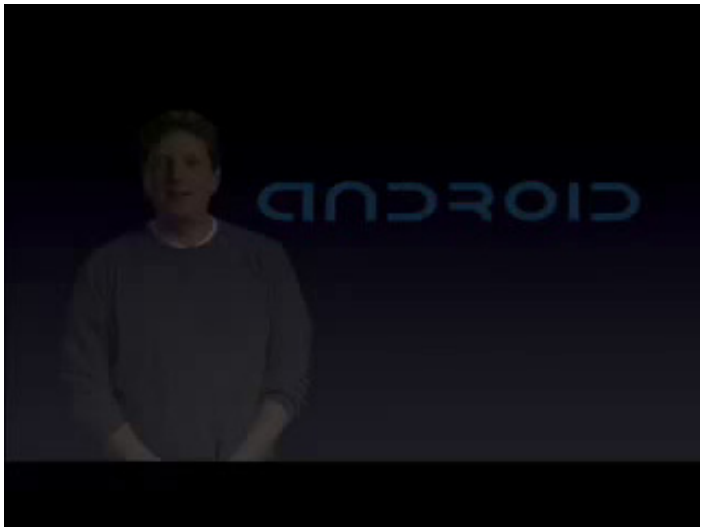
Video 1/3: Android's Architecture
Presented by Mike Cleron, Google Corp. (13 min)

Available at: http://www.youtube.com/watch?v=QBGfUs9mQYY

19

Android Components

Video 2/3: Application's Life Cycle
Presented by Mike Cleron, Google Corp. (8 min)



Available at: http://www.youtube.com/watch?v=fL6gSd4ugSI&feature=channel

20

Android Components

Video 3/3: Android's API
Presented by Mike Cleron, Google Corp. (7 min)

Available at: http://www.youtube.com/watch?v=MPukbH6D-lY&feature=channel

21

Android Application Framework

Video:

Inside the
Android Application Framework
(about 52 min)

Presented by Dan Morrill – Google
At Google Developer Conference
San Francisco - 2008
Available at:
http://sites.google.com/site/io/inside-the-android-application-framework

Android is designed to be fast, powerful, and easy to develop for. This session will discuss the Android application framework in depth, showing you the machinery behind the application framework.

explains the life-cycle of an android apk. very good!

22

Android Components

Video:

An Introduction to Android

(about 52 min)

Presented by Jason Chen – Google

At Google Developer Conference

San Francisco - 2008

Available at:

http://www.youtube.com/watch?v=x1ZZ-R3p_w8

23

Why use Linux for a phone?

- Linux kernel is a **proven** core platform.
- **Reliability** is more important than performance when it comes to a mobile phone, because voice communication is the primary use of a phone.
- Linux provides a **hardware abstraction** layer, letting the upper levels remain unchanged despite changes in the underlying hardware.
- As new **accessories** appear on the market, **drivers** can be written at the Linux level to provide support, just as on other Linux platforms.

24

Dalvik Virtual Machine

- **User applications**, as well as **core Android applications**, are written in Java programming language and are compiled into *byte codes*.
- Android *byte codes are interpreted at* runtime by a processor known as the *Dalvik virtual machine.*

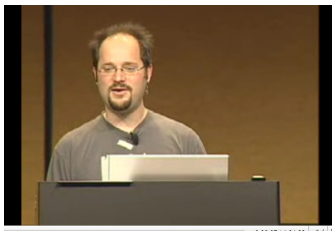
25

Why another JavaVirtual Machine?

- Android bytecode files are logically equivalent to Java bytecodes, but they permit Android to
  - run its applications in its own virtual environment that is free from Sun's licensing restrictions and
  - an open platform upon which Google, and potentially the open source community, can improve as necessary.

26

Dalvik Virtual Machine



Video (61 min)

Dalvik VM Internals

Presented by Dan Borstein

At Google Developer – 2008

San francisco

Available at:

http://www.youtube.com/watch?v=ptjedOZEXPM

27

Inside Android: Intents

- An important and recurring theme of Android development is the **Intent**.
- An Intent in Android describes *what you want to do*.
- This may look like
  - *"I want to look up a contact record,"* or
  - *"Please launch this website,"* or
  - *"Show the Order Confirmation Screen."*
- Intents are important because they facilitate navigation and represent the most important aspect of Android coding.

28

Intents & IntentFilters

- An **Intent** is a declaration of need.
- An **Intent** is made up of various pieces including:
  - desired *action* or *service*,
  - *data*, and
  - *category* of component that should handle the intent and instructions on how to launch a target activity.
- An **IntentFilter** is a trigger, a declaration of capability and interest in offering assistance to those in need.
- An **IntentFilter** may be generic or specific with respect to which Intents it offers to service.

29

Intents & IntentFilters

- An **intent** is an abstract description of an operation to be performed.
- Its most significant use is in the *launching of activities*, where it can be thought of as the glue between activities.
- The primary pieces of information in an intent are:

Action	Data
The general action to be performed, such as: ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.	The data to operate on, such as a person record in the contacts database, expressed as a Uri.

30

Intents & IntentFilters

Some examples of Intent's action/data pairs are:

ACTION_VIEW ***content://contacts/1*** -- Display information about the person whose identifier is "1".
ACTION_DIAL ***content://contacts/1*** -- Display the phone dialer with the person filled in.
ACTION_VIEW ***tel:123*** -- Display the phone dialer with the given number filled in
ACTION_DIAL ***tel:123*** -- Display the phone dialer with the given number filled in.
ACTION_EDIT ***content://contacts/1*** -- Edit information about the person whose identifier is "1".
ACTION_VIEW ***content://contacts/*** -- Display a list of people, which the user can browse through.

31

Dissecting Intents

1. **Component name** The name of the component that should handle the intent (for example "com.example.project.app.MyActivity1").

2. **Action** A string naming the action to be performed — or, in the case of broadcast intents, the action that took place and is being reported (for example: ACTION_VIEW, ACTION_CALL, ACTION_TIMEZONE_CHANGED, …).

3. **Data** The URI of the data to be acted on and the MIME type of that data (for example tel:/216 555-1234 , "http://maps.google.com", ...).

4. **Category** A string containing additional information about the kind of component that should handle the intent (for example CATEGORY_BROWSABLE, CATEGORY_LAUNCHER, …).

5. **Extras** *Key-value pairs* for additional information that should be delivered to the component handling the intent.

6. **Flags** of various sorts.

32

Delivering Intents

- An Intent object is passed to **Context.startActivity()** or **Activity.startActivityForResult()** to launch an activity or get an existing activity to do something new (*asynchronous* & *synchronously* respectively).
- An Intent object is passed to **Context.startService()** to initiate a service or deliver new instructions to an ongoing service.
- An intent can be passed to **Context.bindService()** to establish a connection between the calling component and a target service. It can optionally initiate the service if it's not already running.

33

Intent Resolution

Intents can be divided into two groups:

- ***Explicit intents*** designate the target component by its name, typically used for an activity starting a subordinate service or launching a sister activity.
- ***Implicit intents*** do not name a target (the field for the component name is blank). Implicit intents are often used to activate components in other applications. Late binding applies.

Whenever possible Android delivers an explicit intent to an instance of the designated target class.

34

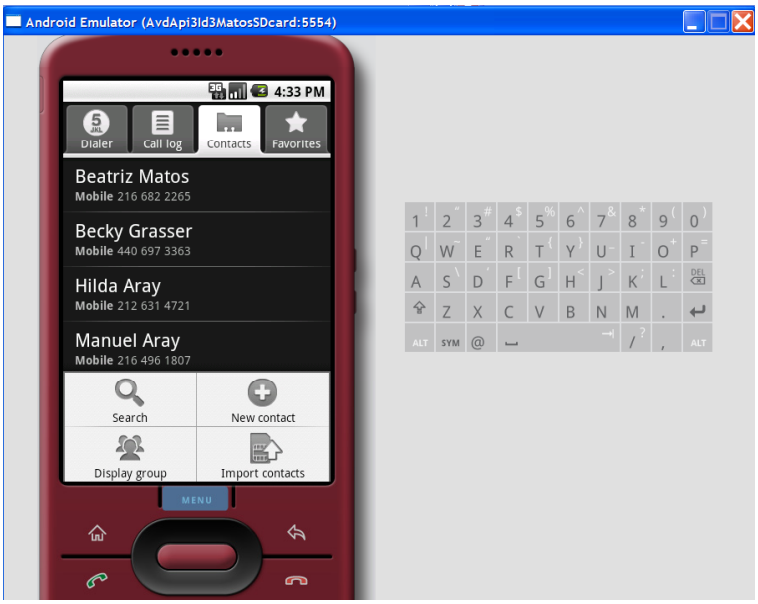
Example of Intent (1)

- Following fragments calls an **Intent** whose job is to invoke a built-in task (*ACTION_VIEW*) and explore the *Contacts* available in the phone.

```
Intent myIntent = new Intent(
    Intent.ACTION_VIEW,
    Uri.parse("content://contacts/people"));

startActivity(myIntent);
```

35

Example of Intent (1)



Intent uses ACTION_VIEW to see Contacts.

36

Example of Intent (1)

- Complete code to see Contacts.

```
package matos.cis493;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

public class AndDemo1 extends Activity {
   /** show contact list */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      Intent myIntent = new Intent( Intent.ACTION_VIEW,Uri.parse( "content://contacts/people"));
      startActivity(myIntent);
   }
}
```

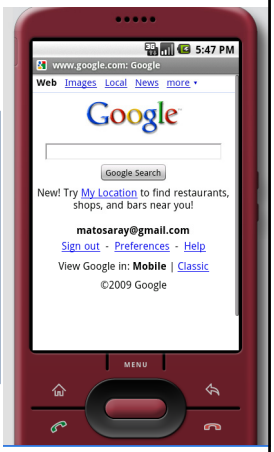
37

Example of Intent (2)

- Following **Intent** uses built-in task (*ACTION_VIEW*) to explore a web page

  (see new Uri value)

```
Intent myIntent = new Intent(
    Intent.ACTION_VIEW,
    Uri.parse("http://www.google.com"));

startActivity(myIntent);
```



38

Example of Intent (3)

- Following **Intent** uses built-in task (*ACTION_VIEW*) to make a phone call

  (see new Uri value)

```
Intent myIntent = new Intent(
    Intent.ACTION_VIEW,
    Uri.parse("tel:/216 555-1234"));

startActivity(myIntent);
```

39

IntentFilters

- The **IntentFilter** defines the relationship between the Intent and the application.
- **IntentFilters** can be specific to the data portion of the Intent, the action portion, or both.
- **IntentFilters** also contain a field known as a *category. A category helps classify* the action.
- For example, the category named

  CATEGORY_LAUNCHER

  instructs Android that the Activity containing this IntentFilter should be visible in the home screen.

40

IntentFilters

- When an **Intent** is dispatched, the system evaluates the available Activities, Services, and registered BroadcastReceivers and routes the Intent to the *most appropriate* recipient (see next Figure).

41

IntentFilters

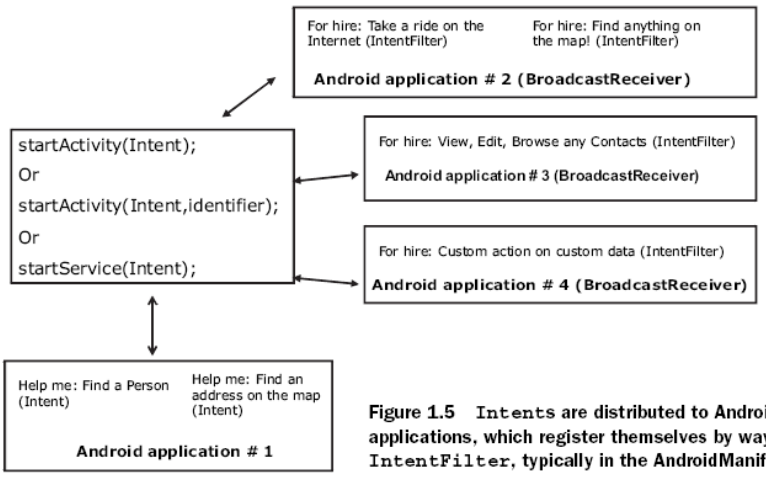


Figure 1.5 `Intents` are distributed to Android applications, which register themselves by way of the `IntentFilter`, typically in the AndroidManifest.xml file.

42

IntentFilters

- To inform the system which implicit intents they can handle, *activities*, *services*, and *broadcast receivers* can have one or more intent filters.
- Each filter describes a capability that the component is willing to receive.
- An explicit intent is always delivered to its target, no matter what it contains; the filter is not consulted.
- But an implicit intent is delivered to a component only if it can pass through one of the component's filters.

43

IntentFilters

- IntentFilters are often defined in an application's AndroidManifest.xml with the <intent-filter> tag.

```
<intent-filter . . . >
      <action android:name="code android.intent.action.MAIN" />
       <category android:name="code android.intent.category.LAUNCHER" />
       <category android:name="android.intent.category.BROWSABLE" />
       <data android:type="video/mpeg" android:scheme="http" . . . />
       <data android:type="audio/mpeg" android:scheme="http" . . . />
      . . .
</intent-filter>
```

44

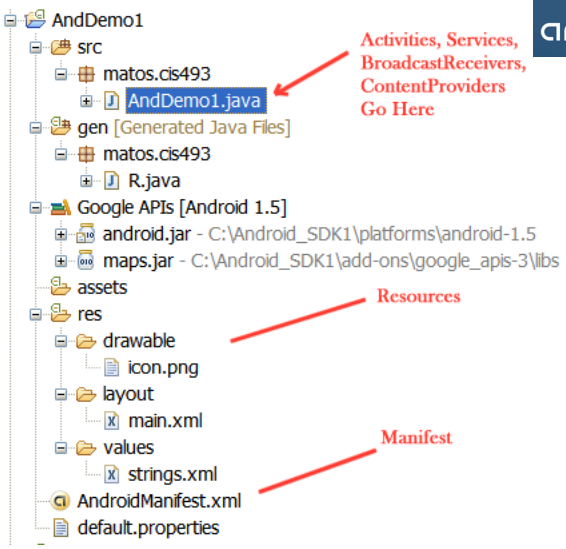
Android Applications

- Each **Android** application runs in its own Linux process.
- An application consists of a combination of software components including:
  - Activities
  - Services
  - Broadcast Receivers
  - Content Providers

45

Android Applications

Structure of a typical Android Application



46

Android Services

- A **Service** is an application component that runs in the background, not interacting with the user, for an indefinite period of time.
- Each service class must have a corresponding **<service>** declaration in its package's AndroidManifest.xml.
- Services can be started/stopped with
  - **Context.startService()** and
  - **Context.bindService()**.
  - stopService(…) and unbindService(…)

47

Android Services

- **Services**, like other application objects, run in the main thread of their hosting process.
- This means that, if your service is going to do any CPU intensive (such as MP3 playback) or blocking (such as networking, RSS exchange) operations, it should spawn its own thread in which to do that work

48

Android Services

Service1 Class

```
package matos.service;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class Service1 extends Service implements Runnable {
 private int counter = 0;
 @Override
 public void onCreate() {
       super.onCreate();
       Thread aThread = new Thread(this);
       aThread.start();
 }

 public void run() {
       while (true) {
              try {
                 Log.i("service1", "service1 firing : # " + counter++);
                 Thread.sleep(10000);  //this is where the heavy-duty computing occurs
              } catch (Exception ee) {
                 Log.e("service1", ee.getMessage());
              }
       }
  }
  @Override
  public IBinder onBind(Intent intent) {
     return null;
  }
}
```

49

Android Services

```
// Service1Driver
package matos.service;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
public class Service1Driver extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      // invoking the service
      Intent service1Intent = new Intent( this,  Service1.class );
      startService( service1Intent );
     // do some work here….
   }
}// Service1Driver
```

50

Android Services

Service1Demo Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="matos.service"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".Service1Driver"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <service  android:name="Service1" android:enabled="true" >
    </service>
  </application>
  <uses-sdk android:minSdkVersion="3" />
</manifest>
```


Android Services

Debugging - Log Cat

07-01 02:49:46.097: INFO/ActivityManager(583): Displayed activity matos.service /.Service1 Driver

07-01 02:49:51.277: DEBUG/dalvikvm(724): GC freed 1575 objects / 81280 bytes in 138ms

07-01 02:49:55.831: INFO/service1(767): service1 firing : # 1
07-01 02:50:05.839: INFO/service1(767): service1 firing : # 2
07-01 02:50:15.847: INFO/service1(767): service1 firing : # 3
07-01 02:50:25.857: INFO/service1(767): service1 firing : # 4

Android Broadcast Receiver

What is a BROADCASTRECEIVER?

- If an application wants to receive and respond to a *global event*, such as the phone ringing or an incoming text message, it must register as a **BroadcastReceiver**.
- An application registers to receive Intents by announcing in the AndroidManfest.xml file its **IntentFilterS**.
- If the receiver is registered in the AndroidManifest.xml file, it does not have to be running in order to be triggered.
- When the global event occurs, the application is started automatically upon notification of the triggering event. All of this housekeeping is managed by the Android OS itself.
- An application may register at runtime via the Context class's **registerReceiver** method.

53

Android Broadcast Receiver

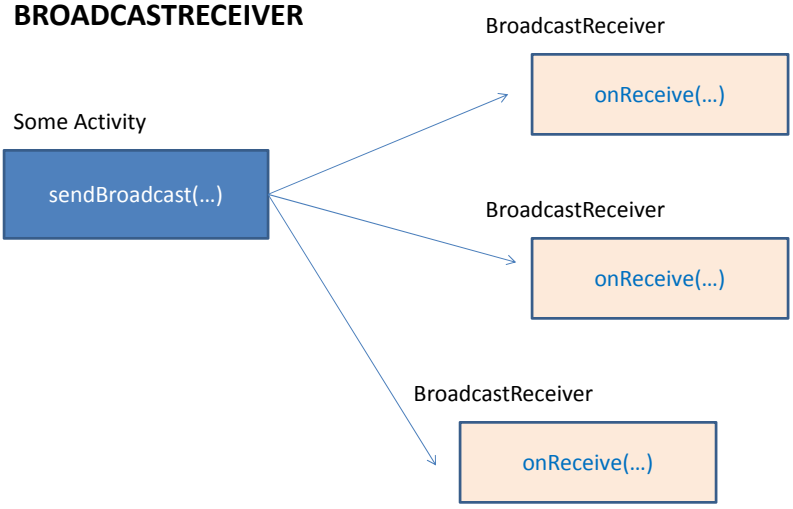
BROADCASTRECEIVER and UI.

- Like Services, BroadcastReceivers *do not have a UI*.
- Of even more importance, the code running in the onReceive method of a BroadcastReceiver should make no assumptions about persistence or long-running operations.
- If the BroadcastReceiver requires more than a trivial amount of code execution, it is recommended that the code initiate a request to a *Service* to complete the requested functionality.

54

Android Broadcast Receiver

BROADCASTRECEIVER



55

Android Broadcast Receiver

Intents vs. Broadcasts

- Starting an Activity with an Intent is a foreground operation that modifies what the user is currently interacting with.
- Broadcasting an Intent is a background operation that the user is not normally aware of.

56

Android Broadcast Receiver

Type of Broadcasts

There are two major classes of broadcasts that can be received:

- **Normal broadcasts** (sent with **sendBroadcast**) are completely *asynchronous*. All receivers of the broadcast are run in an undefined order, often at the same time. This is more efficient, but means that receivers cannot use the result or abort APIs included here.
- **Ordered broadcasts** (sent with **sendOrderedBroadcast**) are *delivered to one receiver at a time*. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the **android:priority** attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order.

57

Android Broadcast Receiver

Broadcast Receiver Life Cycle

- A process that is currently executing a **BroadcastReceiver** (that is, currently running the code in its **onReceive(Context, Intent)** method) is considered to be a foreground process and will be kept running by the system except under cases of extreme memory pressure.
- Once you return from **onReceive**(), the BroadcastReceiver is no longer active, and its hosting process is only as important as any other application components that are running in it.
- This means that for longer-running operations you will often use a **Service** in conjunction with a **BroadcastReceiver** to keep the containing process active for the entire time of your operation.

58

Android Broadcast Receiver

Broadcast Receiver Example (1/5). Intercept arriving SMS

```
package matos.broadcastreceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;

import android.util.Log;

import android.app.Activity;
import android.os.Bundle;

public class MySMSMailBox extends Activity {
// intercepts reception of new text-messages
```

59

Android Broadcast Receiver

Broadcast Receiver Example (2/5). Intercept arriving SMS

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   // define instance of local broadcast receiver
   MySMSMailBoxReceiver mySmsReceiver = new MySMSMailBoxReceiver();

   // receiver's filter will accept event: ...SMS_RECEIVED
   IntentFilter filter = new IntentFilter(
                         "android.provider.Telephony.SMS_RECEIVED");

   // tell Android OS this receiver is ready to go
   registerReceiver(mySmsReceiver, filter);
}
```

60

Android Broadcast Receiver

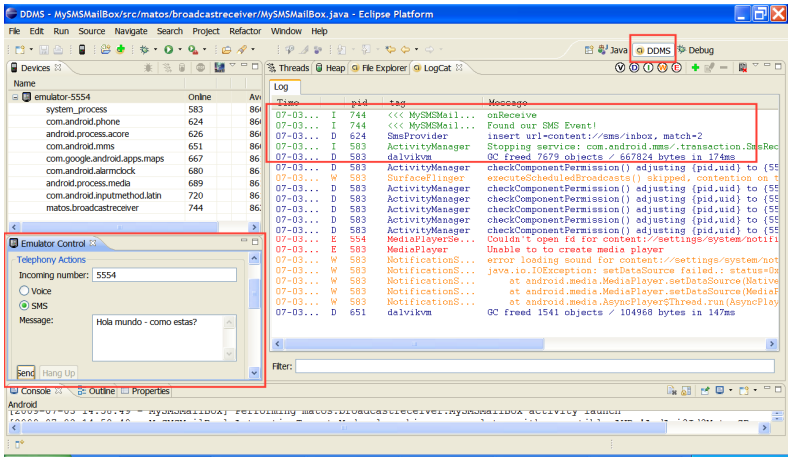
Broadcast Receiver Example (3/5). Intercept arriving SMS

```
// this is the custom made broadcast receiver. Its onReceive method
// is fired when the filter matches the SMS_RECEIVED event
public class MySMSMailBoxReceiver extends BroadcastReceiver {
      public static final String tag = "<<< MySMSMailBox >>>";

      @Override
      public void onReceive(Context context, Intent intent) {
         Log.i(tag, "onReceive");
         // checking global event signaling arrival of text-message
         if (intent.getAction().equals(
               "android.provider.Telephony.SMS_RECEIVED")) {
               Log.i(tag, "Found our SMS Event!");
               // you have intercepted the SMS
               // do something interesting with it. Bye!
         }
      }// onReceive

} // BroadcastReceiver
}
```

61

Android Broadcast Receiver

Broadcast Receiver Example (4/5). Intercept arriving SMS



62

Android Broadcast Receiver

Broadcast Receiver Example (5/5). Intercept arriving SMS

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="matos.broadcastreceiver"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MySMSMailBox"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <receiver android:name="MySMSMailBoxReceiver" >
         <intent-filter>
           <action
                android:name = "android.provider.Telephony.SMS_RECEIVED"/>
         </intent-filter>
      </receiver>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

Android Content Provider

- **Content providers** *store* and *retrieve* data and make it accessible to *all applications*.
- *They are the only way to share data across Android applications.* There's no common storage area that all Android packages can access.
- Android ships with a number of content providers for common data types (audio, video, images, personal contact information, and so on).

64

Android Content Provider

- ContentProviders are a data layer providing data abstraction for its clients and centralizing storage and retrieval routines in a single place.
- A **ContentProvider** may provide data to an Activity or Service in the same application's space as well as an Activity or Service contained in other applications.
- A ContentProvider may use any form of data storage mechanism available on the Android platform, including files, SQLite databases, or even a memory-based hash map if data persistence is not required.

65

Android Content Provider

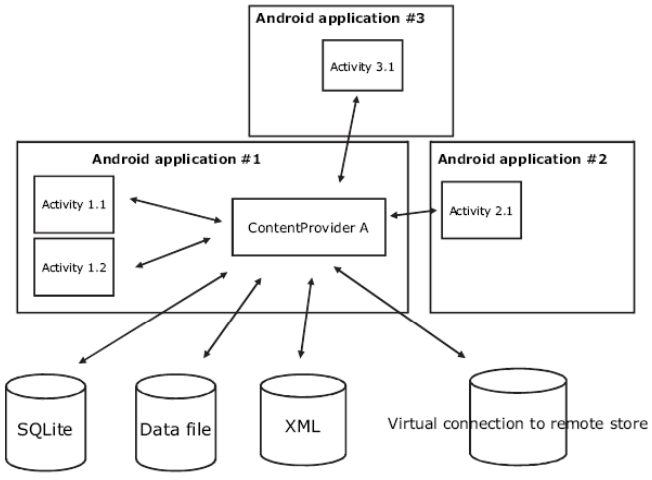


Figure 1.6 The content provider is the data tier for Android applications and is the prescribed manner in which data is accessed and shared on the device.

Android Content Provider

The data model

- Content providers expose their data as a *simple table* on a database model, where each *row* is a record and each *column* is data of a particular type and meaning.
- For example, information about people and their phone numbers might be exposed as follows:

_ID	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	(425) 555 6677	425 555 6677	Kirkland office	Bully Pulpit	TYPE_WORK
44	(212) 555-1234	212 555 1234	NY apartment	Alan Vain	TYPE_HOME
45	(212) 555-6657	212 555 6657	Downtown office	Alan Vain	TYPE_MOBILE
53	201.555.4433	201 555 4433	Love Nest	Rex Cars	TYPE_HOME

67

Android Content Provider

URIs

- Each content provider exposes a public **URI** that uniquely identifies its data set.
- A content provider that controls multiple data sets (multiple tables) exposes a separate URI for each one.
- All URIs for providers begin with the string "**content://**".
- Android defines CONTENT_URI constants for all the providers that come with the platform. For example
  - android.provider.Contacts.Phones.CONTENT_URI
    android.provider.Contacts.Photos.CONTENT_URI
  - android.provider.CallLog.Calls.CONTENT_URI
    android.provider.Calendar.CONTENT_URI
- The **ContentResolver** method takes an URI as its first argument. It's what identifies which provider the ContentResolver should talk to and which table of the provider is being targeted.

68

Android Content Provider

Querying a Content Provider

- You need three pieces of information to query a content provider:
  - The URI that identifies the provider
  - The names of the data fields you want to receive
  - The data types for those fields
- If you're querying a particular record, you also need the ID for that record.
- A query returns a Cursor object that can move from record to record and column to column to read the contents of each field. It has specialized methods for reading each type of data.

69

Android Content Provider

Example: Posting a query to the Contact list (1/2)

```
package matos.cis493;
import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;
import android.provider.Contacts.People;
import android.content.ContentUris;
import android.database.Cursor;


public class AndDemo1 extends Activity {
    /** queries contact list */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);


        // Use the ContentUris method to produce the base URI for the contact with _ID == 23.
        Uri myPerson1 = ContentUris.withAppendedId(People.CONTENT_URI, 23);

        // use the "people" content provider to explore all your contacts
         Uri myPerson2 = Uri.parse("content://contacts/people");

        // Then query for this specific record using method: managedQuery
        // args: (Uri uri, String[] projection, String selection,
        //       String[] selectionArgs, String sortOrder)

         Cursor cur = managedQuery(myPerson2, null, null, null, null);

        // do something with the cursor here

    }
}
```

70

Android Content Provider

Example: Posting a query to the Contact list (2/2)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="matos.cis493"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".AndDemo1"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
    <uses-permission android:name="android.permission.READ_CONTACTS">
    </uses-permission>

</manifest>
```

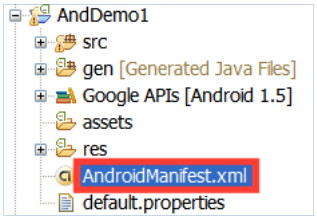
71

Android Manifest xml File

- Every application must have an AndroidManifest.xml file

  (with precisely that name) in its root directory.

- The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code.



72

Android Manifest xml File

These are the only legal elements; you cannot add your own elements or attributes.

<action>	<permission>
<activity>	<permission-group>
<activity-alias>	<permission-tree>
<application>	<provider>
<category>	<receiver>
<data>	<service>
<grant-uri-permission>	<uses-configuration>
<instrumentation>	<uses-library>
<intent-filter>	<uses-permission>
<manifest>	<uses-sdk>
<meta-data>	

73

Android Manifest xml File

Among other things, the manifest does the following:

- It names the *Java package for the application*. The package name serves as a unique identifier for the application.
- It describes the components of the application — the *activities*, *services*, *broadcast receivers*, and *content providers* that the application is composed of.
- It names the *classes* that implement each of the components and publishes *their capabilities* (for example, which Intent messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will *host application components*.
- It declares which *permissions* the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It lists the *Instrumentation* classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
- It declares the minimum level of the *Android API* that the application requires.
- It lists the *libraries* that the application must be linked against.

74

Android Manifest xml File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="matos.earthquake"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/yellow_circle" android:label="@string/app_name">


        <activity android:name=".AndQuake"
                  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=".SatelliteMapping">  </activity>

        <service  android:name="AndQuakeService" android:enabled="true" >
        </service>

        <receiver android:name="AndQuakeAlarmReceiver" >
           <intent-filter>
             <action
                  android:name = "ALARM_TO_REFRESH_QUAKE_LIST"/>
           </intent-filter>
        </receiver>
    </application>

    <uses-library android:name="com.google.android.maps" />
    <uses-permission android:name="android.permission.INTERNET" />

</manifest>
```

75

Example. Currency converter

Implementing a simple currency converter:

USD – Euro – Colon (CR)

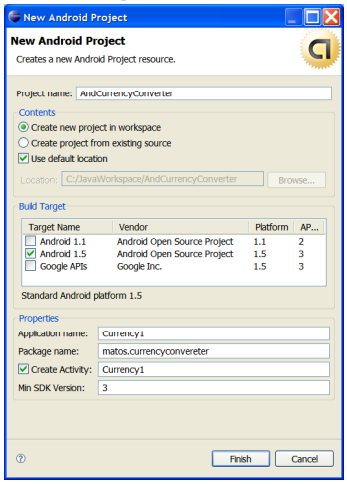
Note. Naive implementation using the rates

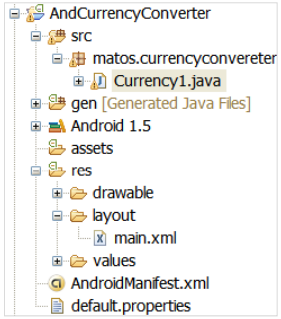
1 Costa Rican Colon = 0.001736 U.S. dollars

1 Euro = 1.39900 U.S. dollars

76

Example. Currency converter





77

Example. Currency converter

```
package matos.currencyconvereter;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Currency1 extends Activity {
      // naive currency converter from USD to Euros & Colones
     final double EURO2USD = 1.399;
     final double COLON2USD = 0.001736;

     // GUI widgets
     Button btnConvert;
     Button btnClear;
     EditText txtUSDollars;
     EditText txtEuros;
     EditText txtColones;
```

78

Example. Currency converter

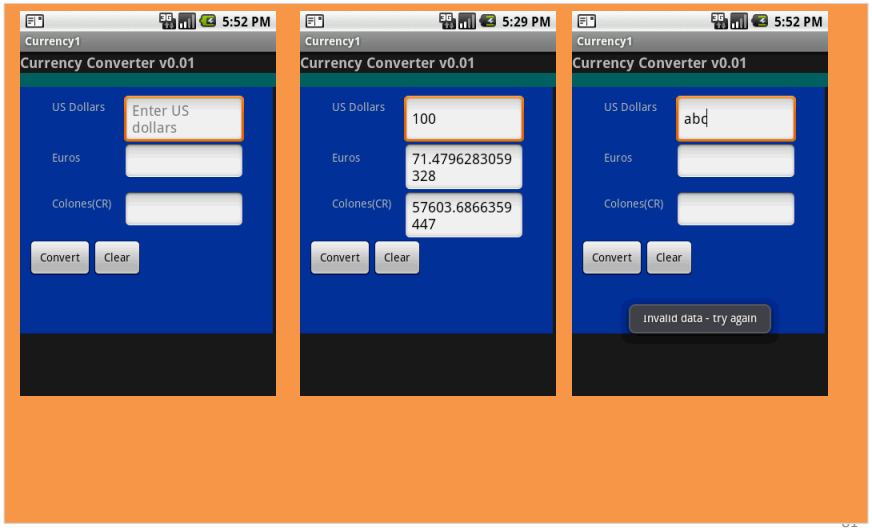
```
@Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       // bind local controls to GUI widgets
       txtUSDollars = (EditText)findViewById(R.id.txtUSDollars);
       txtUSDollars.setHint("Enter US dollars");
       txtEuros = (EditText)findViewById(R.id.txtEuros);
       txtColones = (EditText)findViewById(R.id.txtColones);

      // attach click behavior to buttons
       btnClear = (Button)findViewById(R.id.btnClear);
       btnClear.setOnClickListener(new OnClickListener() {
            // clear the text boxes
           @Override
           public void onClick(View v) {
                  txtColones.setText("");
                  txtEuros.setText("");
                  txtUSDollars.setText("");
           }
       });
```

79

Example. Currency converter

```
       // do the conversion from USD to Euros and Colones
       btnConvert = (Button) findViewById(R.id.btnConvert);
       btnConvert.setOnClickListener(new OnClickListener() {
          @Override
          public void onClick(View v) {
              try {
                   String usdStr = txtUSDollars.getText().toString();
                   double usd = Double.parseDouble( usdStr );
                   String euros = String.valueOf( usd / EURO2USD );
                   String colones = String.valueOf( usd / COLON2USD );
                   txtEuros.setText(euros);
                   txtColones.setText(colones);
               } catch (Exception e) {
                   Toast.makeText(v.getContext(), "Invalid data - try again" ,
                                 Toast.LENGTH_SHORT).show();
               }
         }
         });// setOnClick...
   }// onCreate

}// class
```

Example. Currency converter



Example. Currency converter

Resource: res/ layout/main.xml (1/2)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget47"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/an
    droid"
>

<TextView
android:id="@+id/caption1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Currency Converter v0.01"
android:textSize="18sp"
android:textStyle="bold"
>
</TextView>

<TextView
android:id="@+id/greenFiller1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff006666"
>
</TextView>
```

```
<AbsoluteLayout
android:id="@+id/absLayout"
android:layout_width="316px"
android:layout_height="308px"
android:background="#ff003399"
>

<TextView
android:id="@+id/usdCaption"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="US Dollars"
android:layout_x="40px"
android:layout_y="15px"
>
</TextView>

<EditText
android:id="@+id/txtUSDollars"
android:layout_width="150px"
android:layout_height="wrap_content"

android:layout_x="130px"
android:layout_y="10px"
>
</TextView>
```

Example. Currency converter

Resource: res/ layout/main.xml (2/2)

```
<EditText
android:id="@+id/txtEuros"
android:layout_width="150px"
android:layout_height="wrap_content"
android:layout_x="130px"
android:layout_y="70px"
>
</EditText>

<TextView
android:id="@+id/colonCaption"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Colones(CR)"
android:layout_x="40px"
android:layout_y="135px"
>
</TextView>
<EditText
android:id="@+id/txtColones"
android:layout_width="150px"
android:layout_height="wrap_content"
android:layout_x="130px"
android:layout_y="130px"
>
</EditText>
```

```
<Button
android:id="@+id/btnConvert"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" Convert "
android:layout_x="10px"
android:layout_y="190px"
>
</Button>

<Button
android:id="@+id/btnClear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" Clear "
android:layout_x="90px"
android:layout_y="190px"
>
</Button>

</AbsoluteLayout>

</LinearLayout>
```

83

Example. Currency converter

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="matos.currencyconvereter"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        <activity android:name=".Currency1"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                   android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

84

Additional Resources

Google Developer Conference

San Francisco – 2009

Web page: http://code.google.com/events/io/

85