Android Introduction

Platform Overview

What is Android?

- Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

Phones

HTC G1,
Droid,
Tattoo

Motorola Droid (X)

Suno S880

Samsung Galaxy

Sony Ericsson

Tablets

Velocity Micro Cruz

Gome FlyTouch

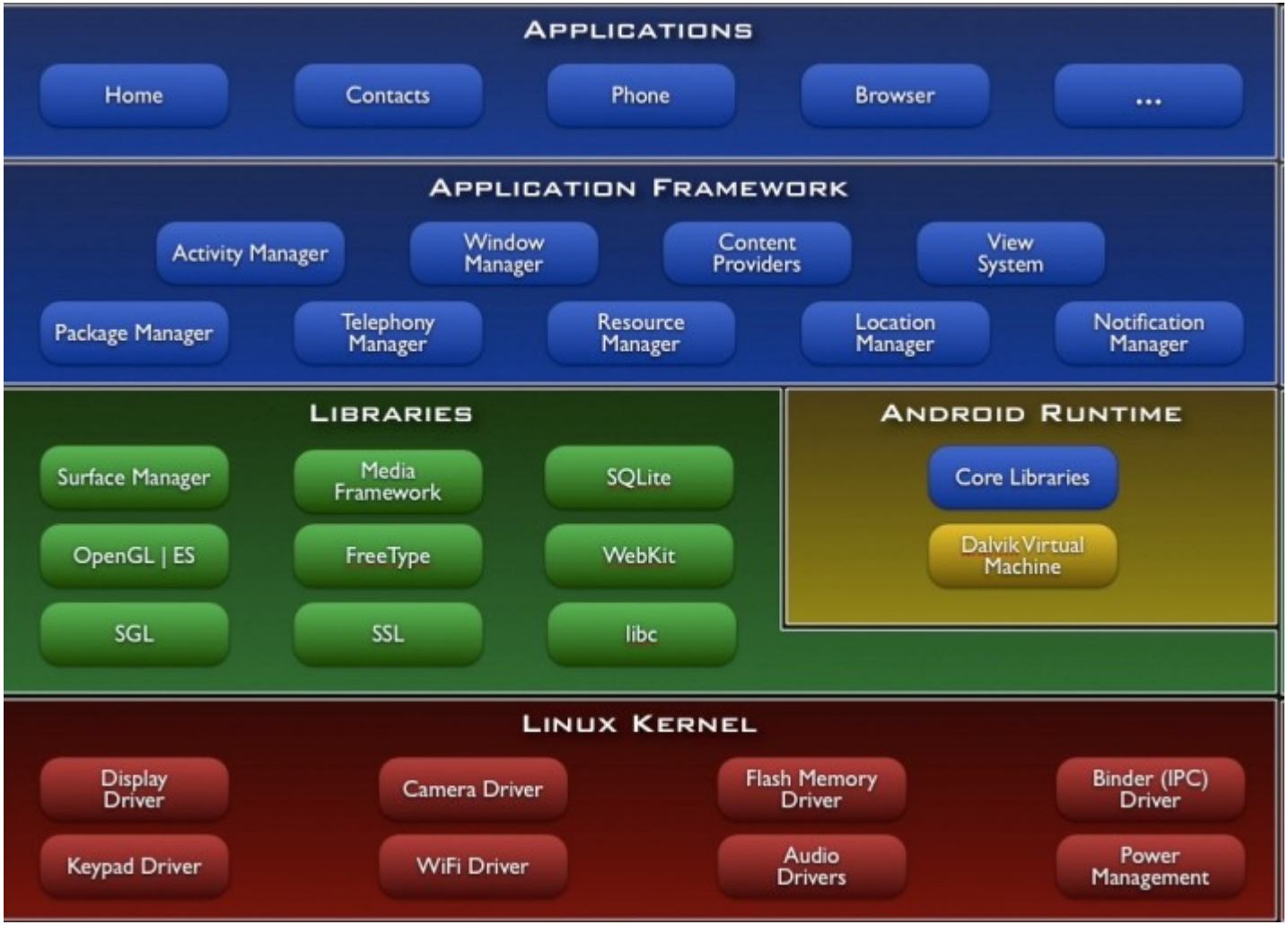
Acer beTouch

Dawa D7

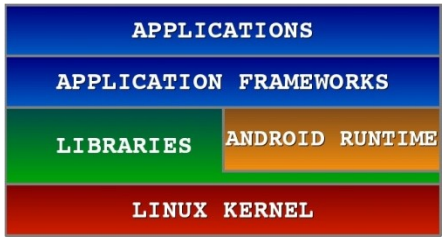
Toshiba Android SmartBook

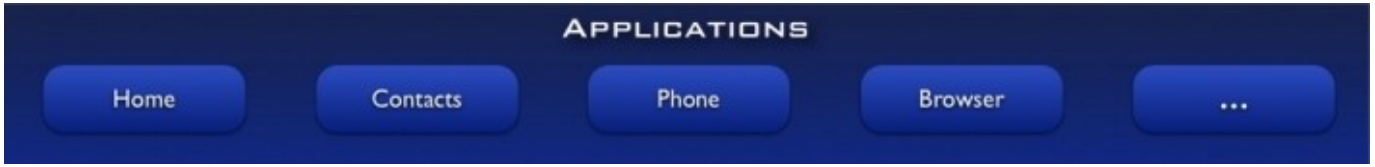
Cisco Android Tablet

Architecture



Android S/W Stack - Application

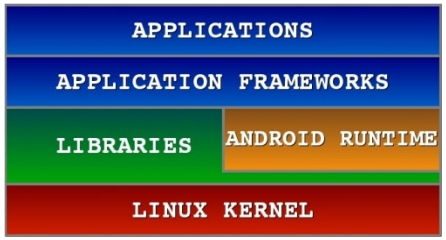




- Android provides a set of core applications:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc

- All applications are written using the Java language.

Android S/W Stack – App Framework



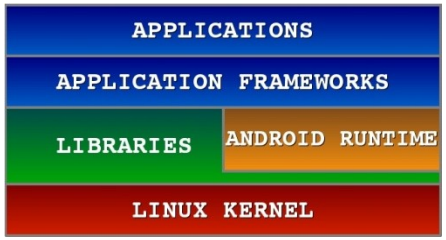
- Enabling and simplifying the reuse of components
  - ✓ Developers have full access to the same framework APIs used by the core applications.
  - ✓ Users are allowed to replace components.

Android S/W Stack – App Framework (Cont)

- Features

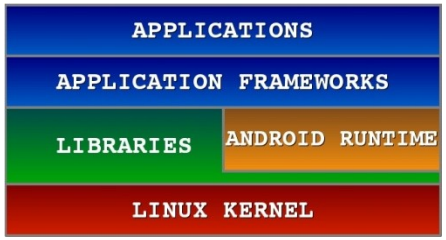
Feature	Role
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized strings, graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

Android S/W Stack - Libraries



- Including a set of C/C++ libraries used by components of the Android system
- Exposed to developers through the Android application framework

Android S/W Stack - Runtime



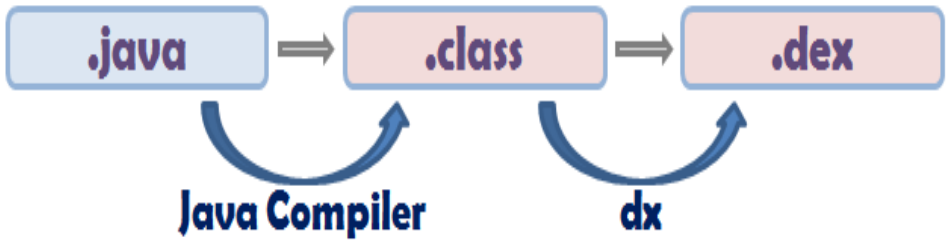
- Core Libraries
  - ✓ Providing most of the functionality available in the core libraries of the Java language
  - ✓ APIs
    - Data Structures
    - Utilities
    - File Access
    - Network Access
    - Graphics
    - Etc

Android S/W Stack – Runtime (Cont)

- Dalvik Virtual Machine
  - ✓ Providing environment on which every Android application runs
    - Each Android application runs in its own process, with its own instance of the Dalvik VM.
    - Dalvik has been written such that a device can run multiple VMs efficiently.
  - ✓ Register-based virtual machine

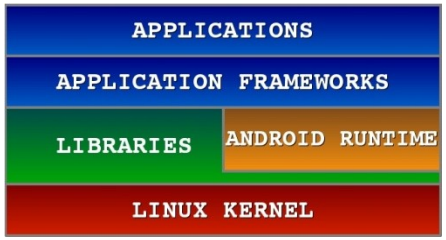
Android S/W Stack – Runtime (Cont)

- Dalvik Virtual Machine (Cont)
  - ✓ Executing the Dalvik Executable (.dex) format
    - .dex format is optimized for minimal memory footprint.
    - Compilation



  - ✓ Relying on the Linux Kernel for:
    - Threading
    - Low-level memory management

Android S/W Stack – Linux Kernel



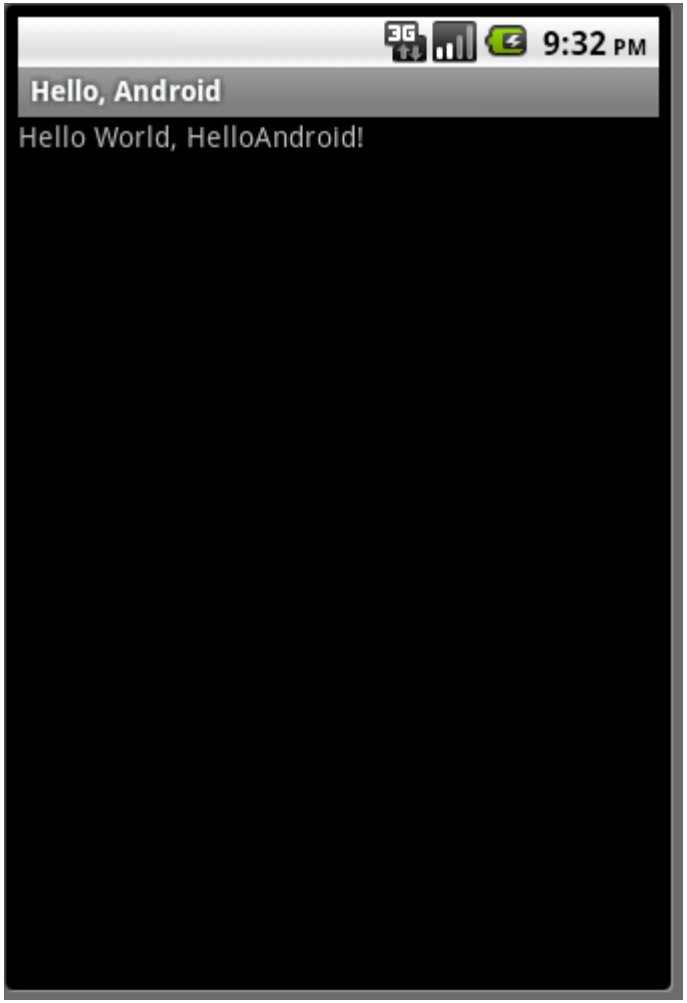
- Relying on Linux Kernel 2.6 for core system services
  - ✓ Memory and Process Management
  - ✓ Network Stack
  - ✓ Driver Model
  - ✓ Security
- Providing an abstraction layer between the H/W and the rest of the S/W stack

Android Introduction

Hello World

Goal

- Create a very simple application
- Run it on a real device
- Run it on the emulator
- Examine its structure



Google Tutorial

- We will follow the tutorial at:

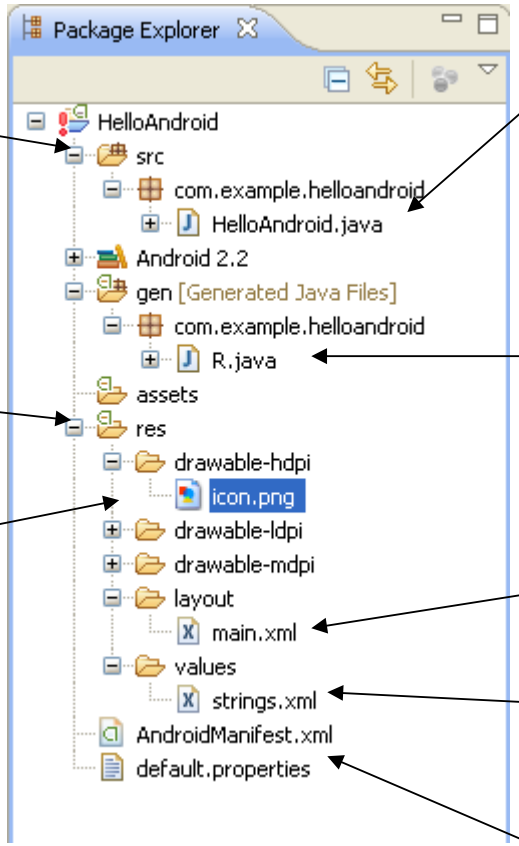
http://developer.android.com/resources/tutorials/hello

- Start Eclipse (Start -> All Programs -> Eclipse)
- Create an Android Virtual Device (AVD)
- Create a New Android Project

Package Content

All source code here

Java code for our activity



All non-code resources

Generated Java code
Helps link resources to Java code

Images

Layout of the activity

Strings used in the program

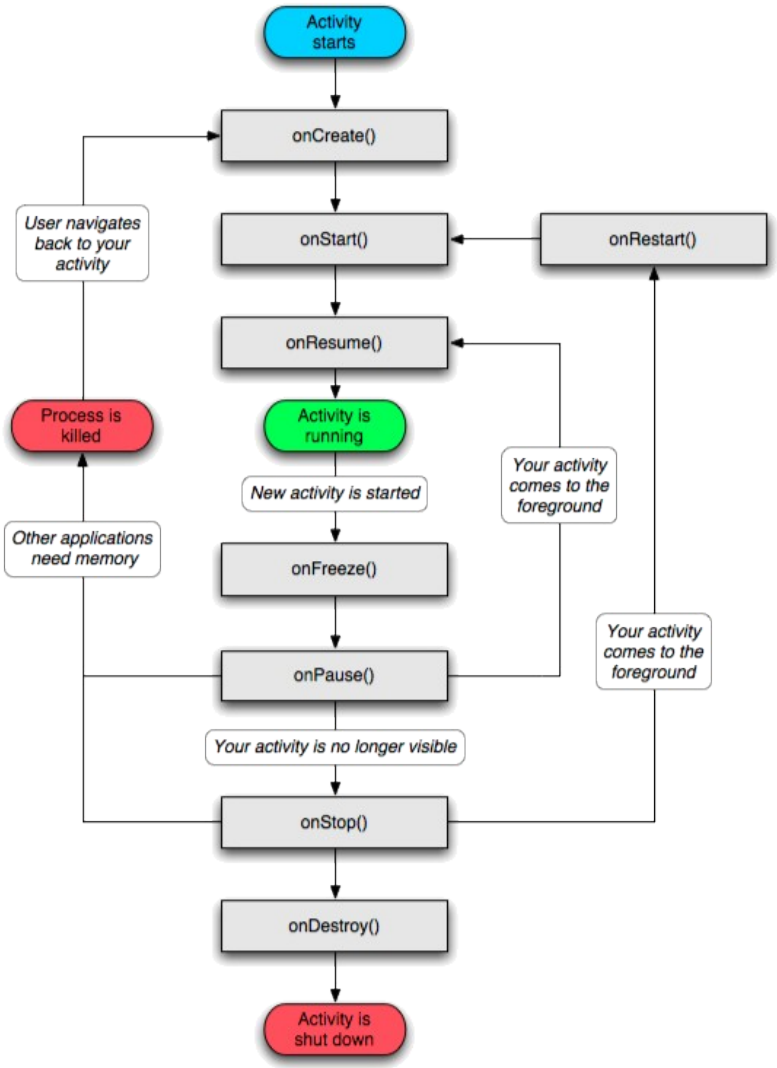
Android Manifest

Android Manifest

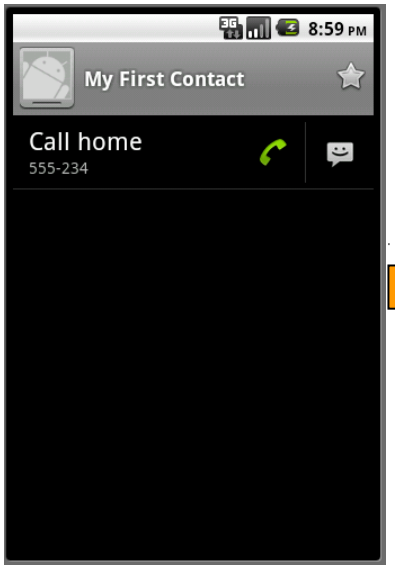
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
   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

    </application>


</manifest>
```

Activity

- An Android activity is focused on a single thing a user can do.
- Most applications have multiple activities



Activities start each other



Revised HelloAndroid.java

```
package com.example.helloandroid;

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

 public class HelloAndroid extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      TextView tv = new TextView(this);
      tv.setText("Hello, Android – by hand");
      setContentView(tv);
   }
 }
```

Inherit from the Activity Class

Set the view “by hand” – from the program

Run it!

/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello"
    />
</LinearLayout>
```

Further redirection to
/res/values/strings.xml

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroid – by resources!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

HelloAndroid.java

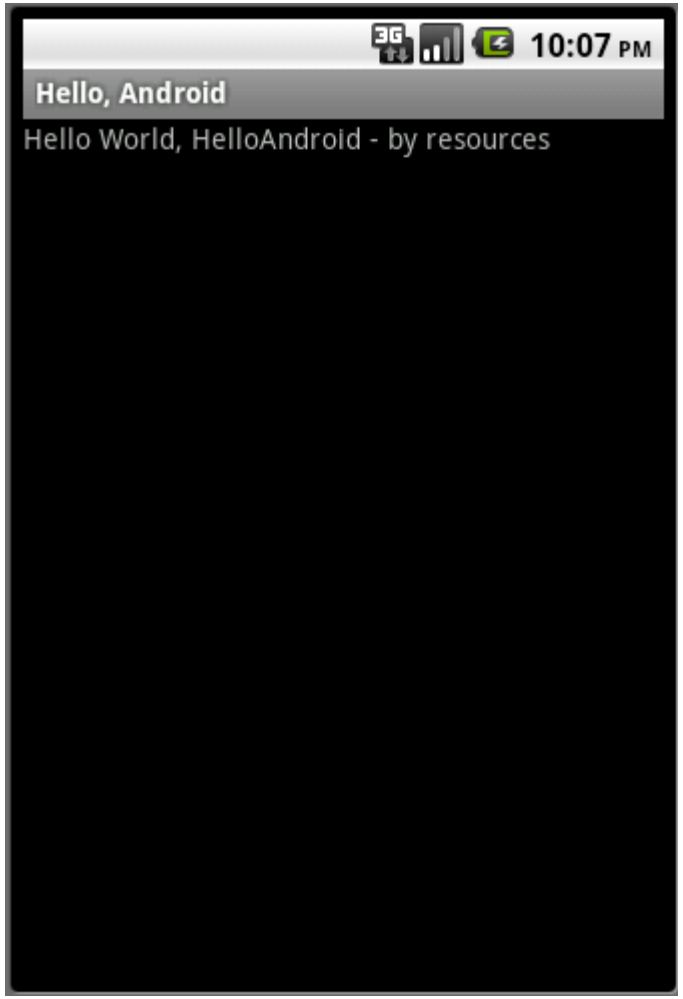
```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
public class HelloAndroid extends Activity {

        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        }
}
```

Set the layout of the view as described in the main.xml layout

/gen/R.java

```
package com.example.helloandroid;

  public final class R {
      public static final class attr {
      }
      public static final class drawable {
          public static final int icon=0x7f020000;
      }
      public static final class id {
          public static final int textview=0x7f050000;
      }
      public static final class layout {
          public static final int main=0x7f030000;
      }
      public static final class string {
          public static final int app_name=0x7f040001;
          public static final int hello=0x7f040000;
      }
  }
```

Run it!


10:07 PM
Hello, Android
Hello World, HelloAndroid - by resources

Introduce a bug

```
package com.example.helloandroid;

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

  public class HelloAndroid extends Activity {
      /** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          Object o = null;
          o.toString();
          setContentView(R.layout.main);
      }
  }
```

Run it!


10:11 PM
Alarm Clock
Browser
Calculator
Camera
Sorry!
The application Hello, Android (process com.example. helloandroid) has stopped unexpectedly. Please try again.
Force close
Music
Phone
Settings
Spare Parts

Android Introduction

Application Fundamentals

Goal

- Understand applications and their components
- Concepts:
  - activity,
  - service,
  - broadcast receiver,
  - content provider,
  - intent,
  - AndroidManifest

Applications

- Written in Java (it's possible to write native code – will not cover that here)
- Good separation (and corresponding security) from other applications:
  - Each application runs in its own process
  - Each process has its own separate VM
  - Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported)

Application Components

- Activities – visual user interface focused on a single thing a user can do
- Services – no visual interface – they run in the background
- Broadcast Receivers – receive and react to broadcast announcements
- Content Providers – allow data exchange between applications

Activities

- Basic component of most applications
- Most applications have several activities that start each other as needed
- Each is implemented as a subclass of the base Activity class

Activities – The View

- Each activity has a default window to draw in (although it may prompt for dialogs or notifications)
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

Services

- Does not have a visual interface
- Runs in the background indefinitely
- Examples
  - Network Downloads
  - Playing Music
  - TCP/UDP Server
- You can bind to a an existing service and control its operation

Broadcast Receivers

- Receive and react to broadcast announcements
- Extend the class BroadcastReceiver
- Examples of broadcasts:
  - Low battery, power connected, shutdown, timezone changed, etc.
  - Other applications can initiate broadcasts

Content Providers

- Makes some of the application data available to other applications
- It’s the only way to transfer data between applications in Android (no shared files, shared memory, pipes, etc.)
- Extends the class ContentProvider;
- Other applications use a ContentResolver object to access the data provided via a ContentProvider

Intents

- An intent is an Intent object with a message content.
- Activities, services and broadcast receivers are started by intents. ContentProviders are started by ContentResolvers:
  - An activity is started by Context.startActivity(Intent intent) or Activity.startActivityForResult(Intent intent, int RequestCode)
  - A service is started by Context.startService(Intent service)
  - An application can initiate a broadcast by using an Intent in any of Context.sendBroadcast(Intent intent), Context.sendOrderedBroadcast(), and Context.sendStickyBroadcast()

Shutting down components

- Activities
  - Can terminate itself via finish();
  - Can terminate other activities it started via finishActivity();
- Services
  - Can terminate via stopSelf(); or Context.stopService();
- Content Providers
  - Are only active when responding to ContentResolvers
- Broadcast Receivers
  - Are only active when responding to broadcasts

Android Manifest

- Its main purpose in life is to declare the components to the system:

```
<?xml version="1.0" encoding="utf-8"?>
 <manifest . . . >
    <application . . . >
       <activity android:name="com.example.project.CputActivity"
               android:icon="@drawable/small_pic.png"
               android:label="@string/cputLabel"
               . . .  >
       </activity>
       . . .
    </application>
 </manifest>
```

Intent Filters

- Declare Intents handled by the current application (in the AndroidManifest):

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest . . . >
    <application . . . >
      <activity android:name="com.example.project.CputActivity"
                android:icon="@drawable/small_pic.png"
                android:label="@string/cputLabel"
                . . .  >
        <intent-filter . . . >
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter . . . >
          <action android:name="com.example.project.BOUNCE" />
          <data android:mimeType="image/jpeg" />
          <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </activity>
      . . .
    </application>
  </manifest>
```

Shows in the Launcher and is the main activity to start

Handles JPEG images in some way

Android Introduction

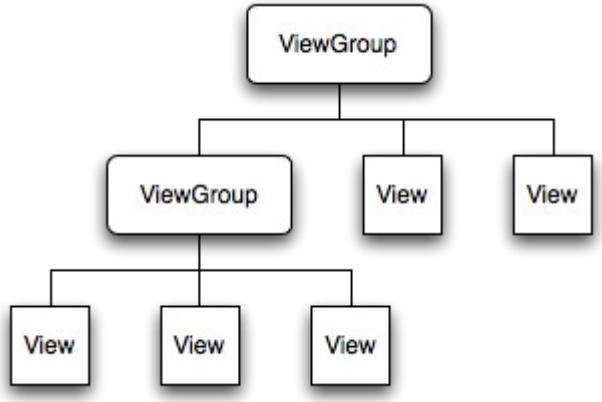
Graphical User Interface

Goal

- Familiarize with the main types of GUI components
- Concepts:
  - Layouts
  - Widgets
  - Menus

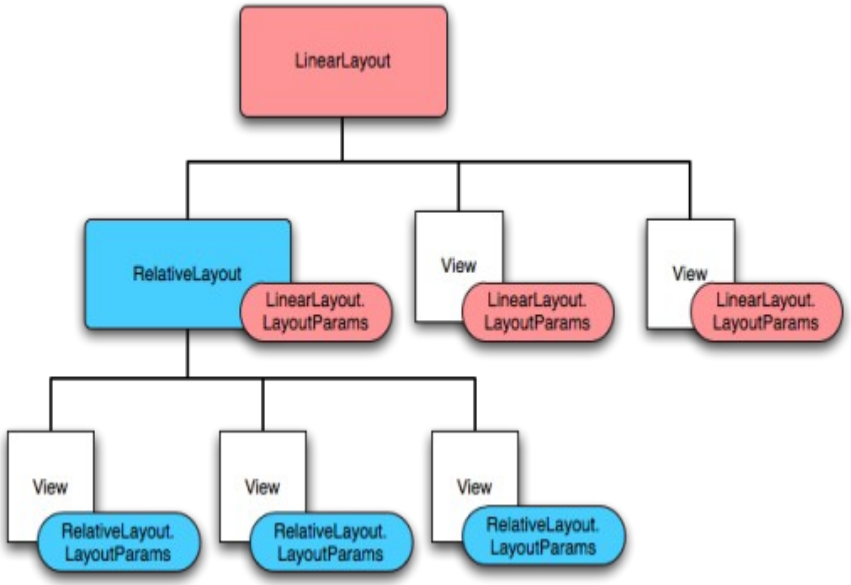
View Hierarchy

- All the views in a window are arranged in a tree
- You show the tree by calling setContentView(rootNode) in the activity



Layout

- Defines how elements are positioned relative to each other (next to each other, under each other, in a table, grid, etc.)
- Can have a different layouts for each ViewGroup



Widgets

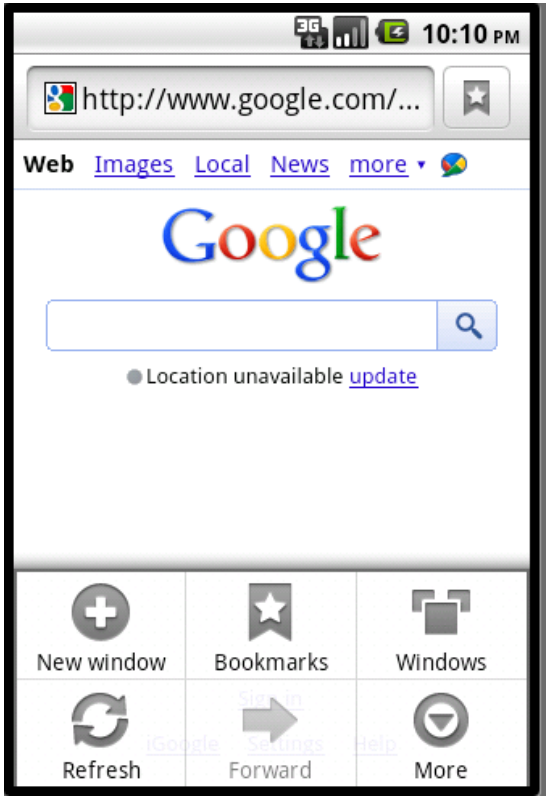
- All are View objects
- Examples:
  - TextFields
  - EditFields
  - Buttons
  - Checkboxes
  - RadioButtons
  - etc.

UI Events

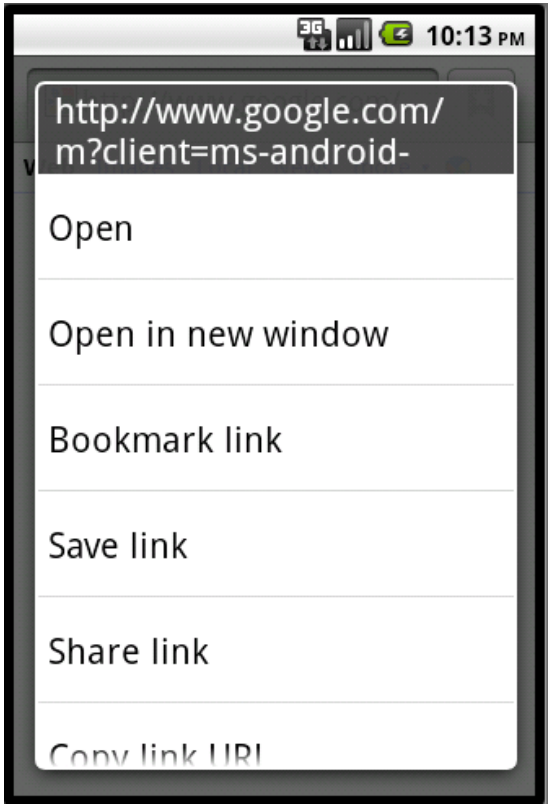
- Usually handled by defining a Listener of the form On<something>Listener and register it with the View
- For example:
  - OnClickListener() for handling clicks on Buttons or Lists
  - OnTouchListener() for handling touches
  - OnKeyListerner() for handling key presses
- Alternatively, Override an existing callback if we implemented our own class extending View

Menus

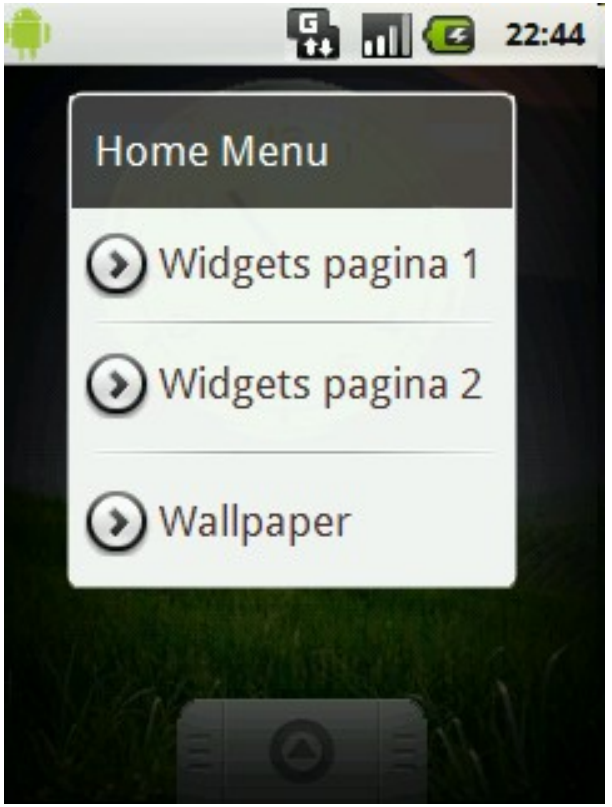
- Options Menu



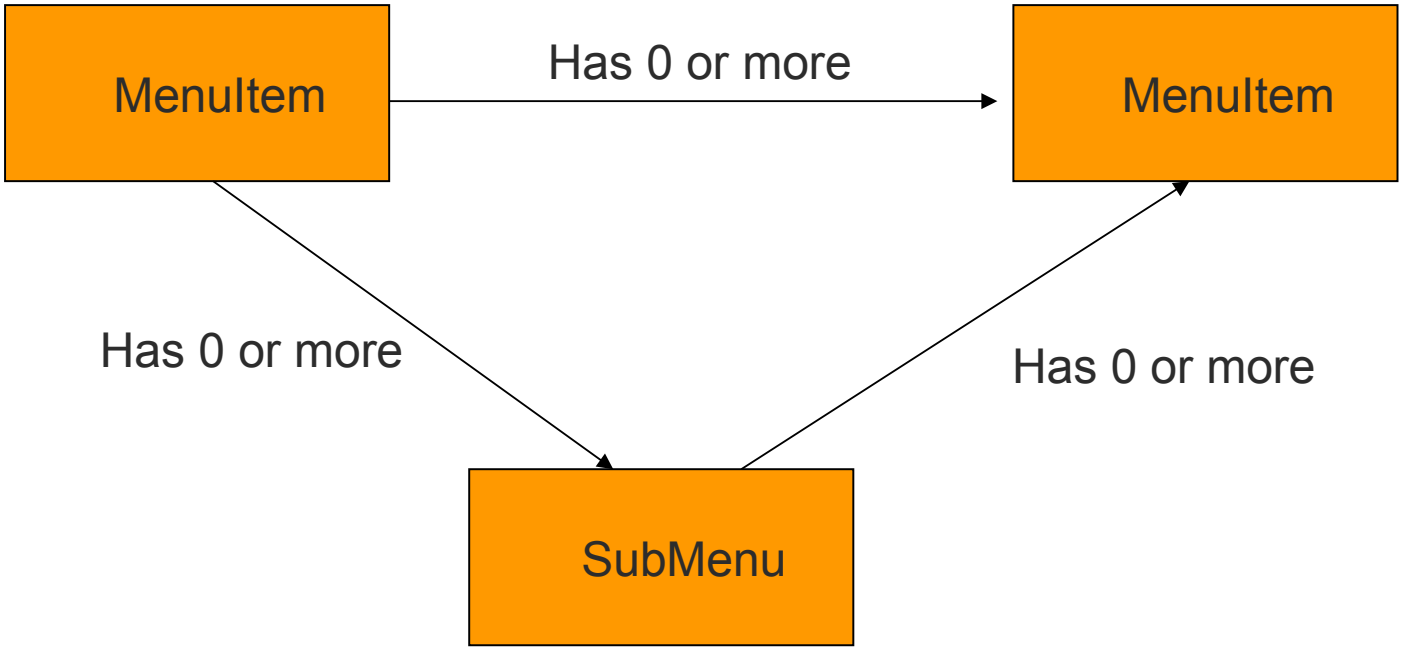
- Context Menu



- Sub-menu



Menus (continued)



Android Introduction

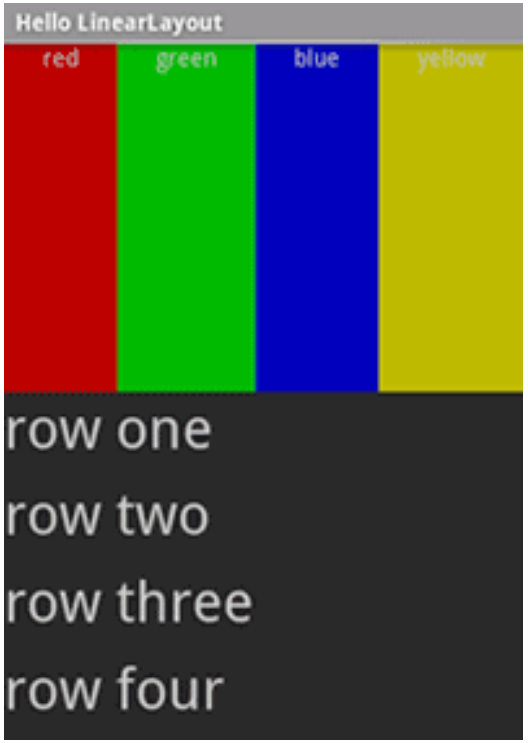
Hello Views

Goal

- Familiarize with the main types of GUI components
- Concepts:
  - Layouts
  - Widgets
  - Menus

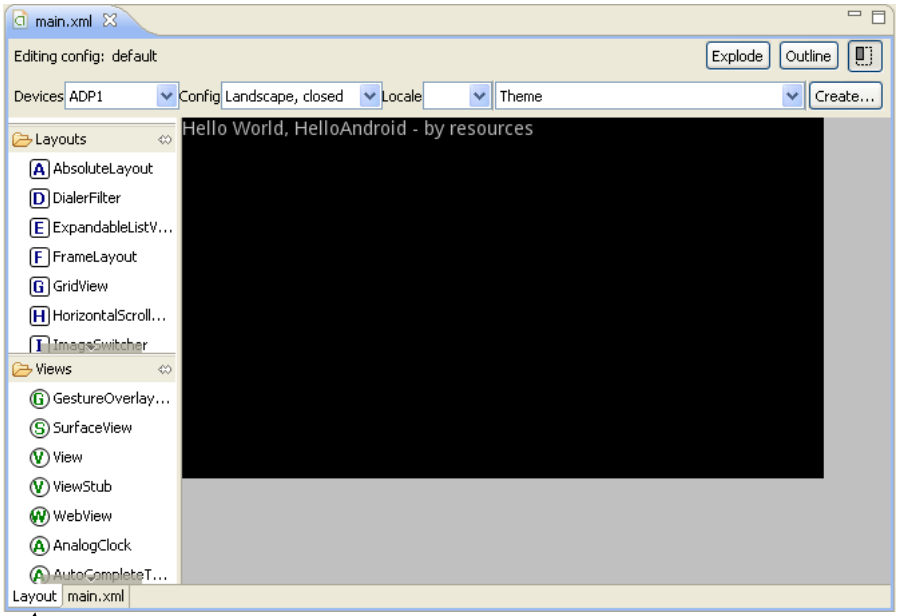
Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
        <LinearLayout
        xmlns:android="http://schemas.android.com/apk/re
        s/android"
           android:orientation="vertical"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent">

          <LinearLayout
             android:orientation="horizontal"
             android:layout_width="fill_parent"
             android:layout_height="fill_parent"
             android:layout_weight="1">
             <TextView
                android:text="red"
                android:gravity="center_horizontal"
          [……................]
         </LinearLayout>

          <LinearLayout
           android:orientation="vertical"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:layout_weight="1">
           <TextView
              android:text="row one"
              android:textSize="15pt"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_weight="1"/>
           <TextView
              android:text="row two"
              android:textSize="15pt"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_weight="1"/>
        [……..……..]
          </LinearLayout>

        </LinearLayout>
```



http://developer.android.com/resources/tutorials/views/hello-linearlayout.html

One Layout, two views



XML File vs Layout Preview

Relative Layout

-
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```

Table Layout

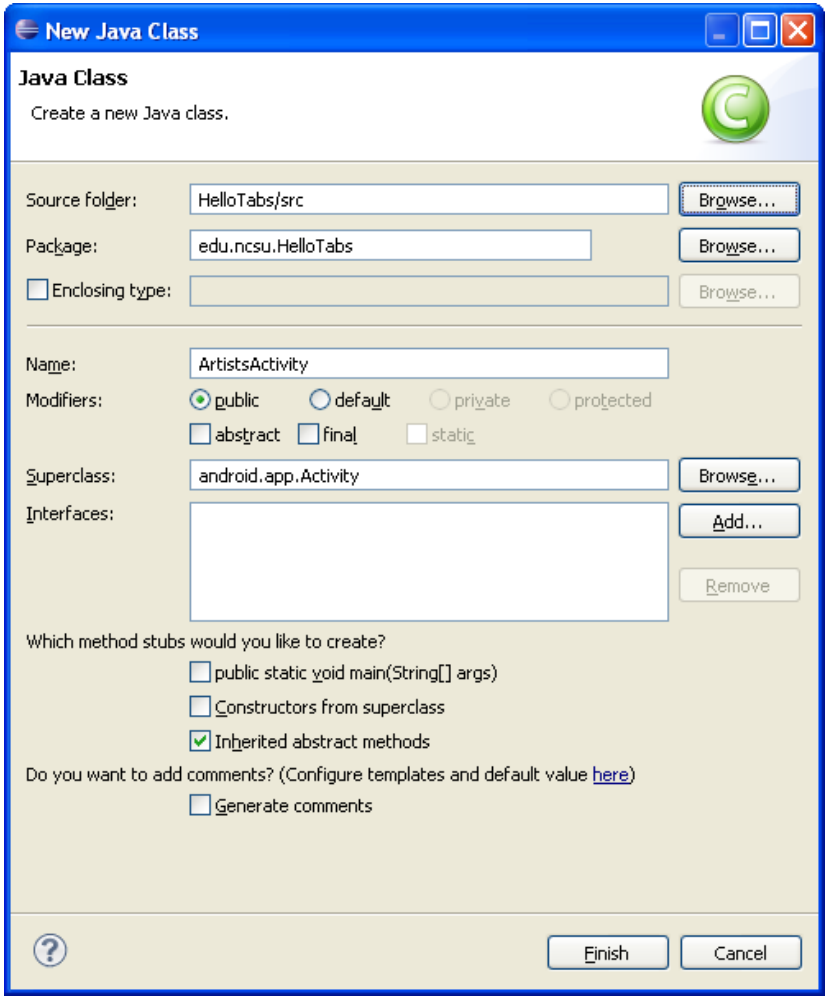
```
<?xml version="1.0" encoding="utf-8"?>
        <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:stretchColumns="1">

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Open..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-O"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Save..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-S"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Save As..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-Shift-S"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <View
              android:layout_height="2dip"
              android:background="#FF909090" />

        [.........................]
        </TableLayout>
```

TabLayout

- One activity per tab
- Create new Project HelloTabs

Create three new activities

- Right click on HelloTabs Package Name -> New -> Class
- Right click on the new class, Source -> Override/Implement Methods -> Check OnCreate();



Fill in the OnCreate() method

```
public class ArtistsActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            TextView textview = new TextView(this);
            textview.setText("This is the Artists tab");
            setContentView(textview);
        }
    }
```

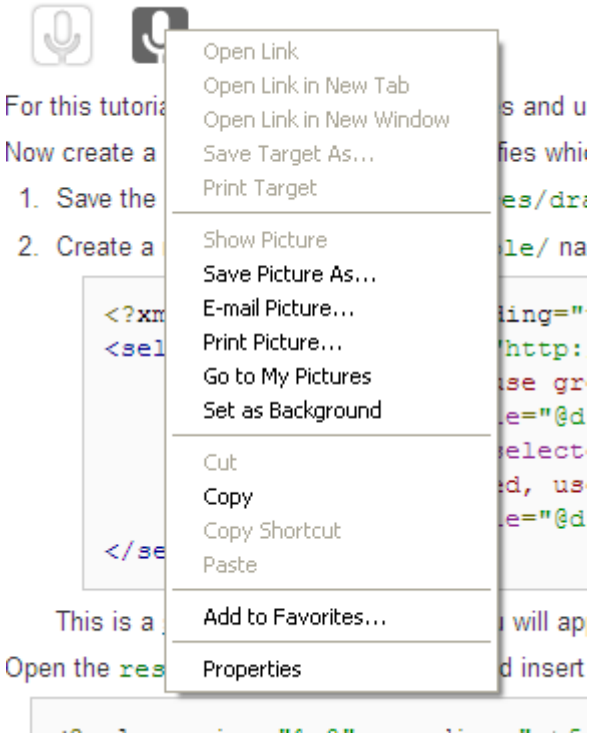
Quick and dirty
“by hand”
like in
HelloWorld

Copy and Paste ArtistsActivity into two more activities:

- AlbumsActivity and
- SongsActivity

Copy the icons

- Right click -> Save As,
- Make ./res/drawable
- move the icons into ./res/drawable



Create ./res/drawable/ic_tab_artists.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <selector xmlns:android="http://schemas.android.com/apk/res/android">
        <!-- When selected, use grey -->
        <item android:drawable="@drawable/ic_tab_artists_grey"
             android:state_selected="true" />
        <!-- When not selected, use white-->
        <item android:drawable="@drawable/ic_tab_artists_white" />
    </selector>
```

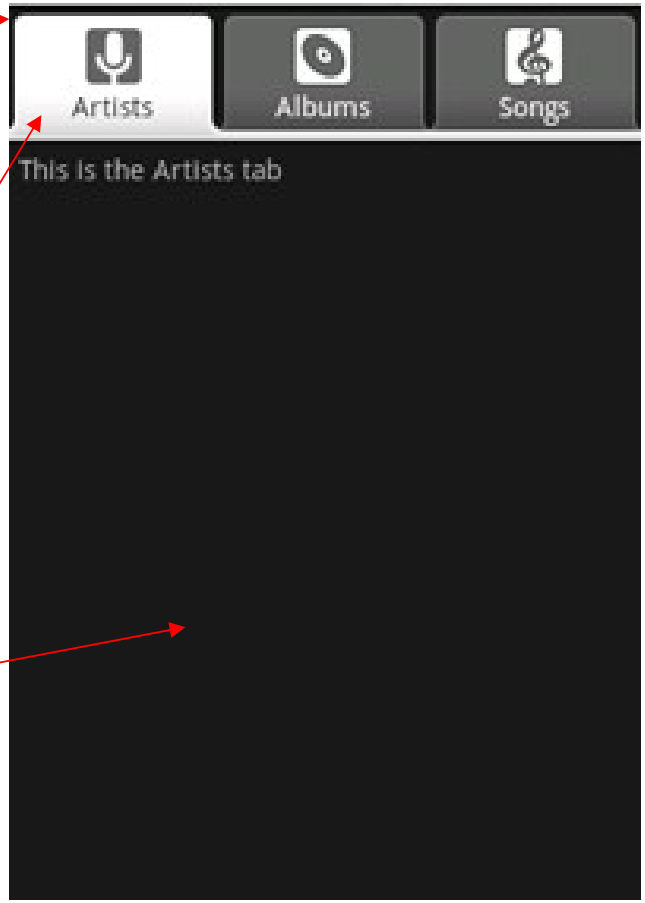
StateListDrawable object that displays different images for different states of a View

Make copies or the xml files for the other two tabs:

- Copy the xml file:
  - ic_tab_artists.xml ->
    - ic_tab_albums.xlm ->
    - ic_tab_songs.xml

Main Layout

- <?xml version="1.0" encoding="utf-8"?>
  <TabHost
  xmlns:android="http://schemas.android.com/apk/res/a
  ndroid"
  android:id="@android:id/tabhost"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <LinearLayout
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="5dp">
  <TabWidget
  android:id="@android:id/tabs"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content" />
  <FrameLayout
  android:id="@android:id/tabcontent"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="5dp" />
  </LinearLayout>
  </TabHost>



OnCreate() for HelloTabs (main activity)

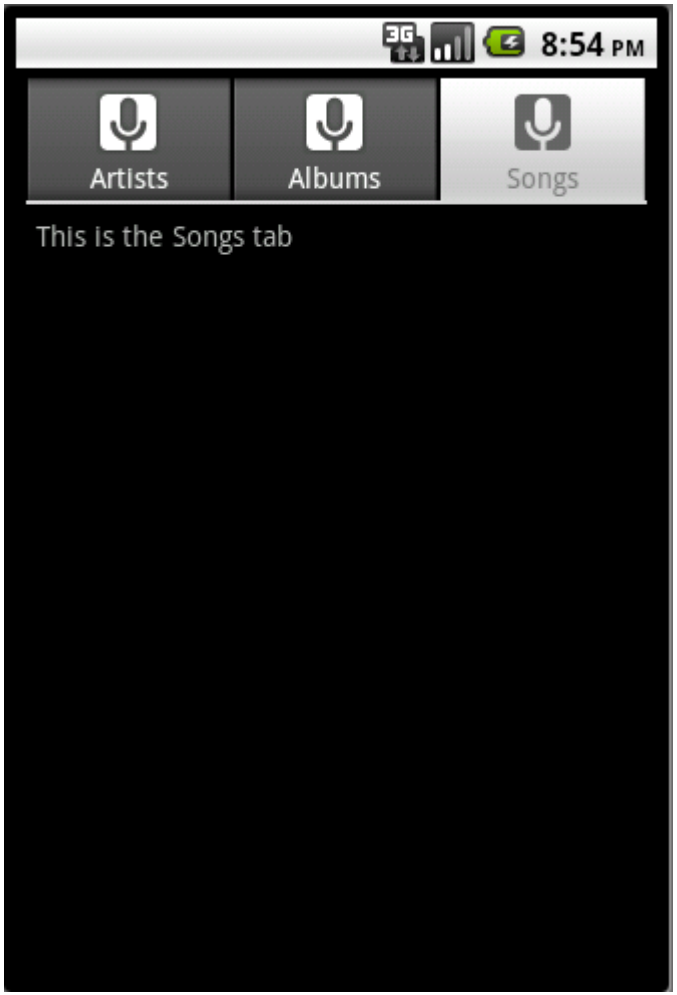
```
public void onCreate(Bundle savedInstanceState)

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Resources res = getResources(); // Resource object to get Drawables
    TabHost tabHost = getTabHost();  // The activity TabHost
    TabHost.TabSpec spec;  // Resusable TabSpec for each tab
    Intent intent;  // Reusable Intent for each tab

    // Create an Intent to launch an Activity for the tab (to be reused)
    intent = new Intent().setClass(this, ArtistsActivity.class);

    // Initialize a TabSpec for each tab and add it to the TabHost
   spec = tabHost.newTabSpec("artists").setIndicator("Artists",
                res.getDrawable(R.drawable.ic_tab_artists))
                .setContent(intent);
      tabHost.addTab(spec);

      // Do the same for the other tabs
      [.........................]
      tabHost.setCurrentTab(2);
    }
```

Main Activity is a TabActivity – has a TabHost

Builder mapping the resources to the tab

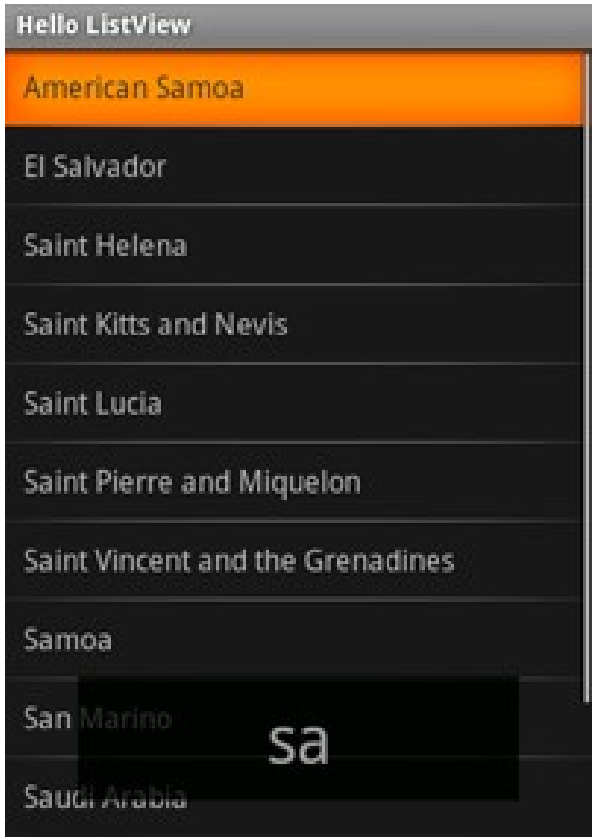
Select Tab 2

Run it!



List View

- List of scrollable items
- Application will inherit from ListActivity rather than Activity
- Create ./res/layout/list_item.xml
  - Layout for each item



Override the OnCreate method

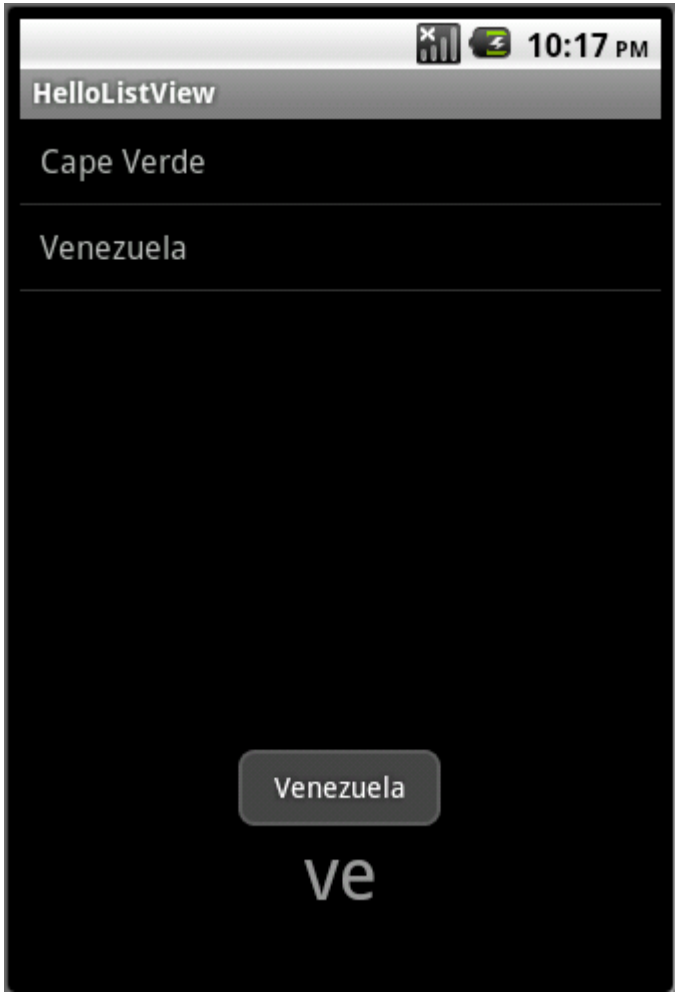
```
public class HelloListView extends ListActivity {
   /** Called when the activity is first created. */

@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();
    }
  });
}
}
```

Setup the list for this application, with this layout and this content

Enables filtering by keyboard

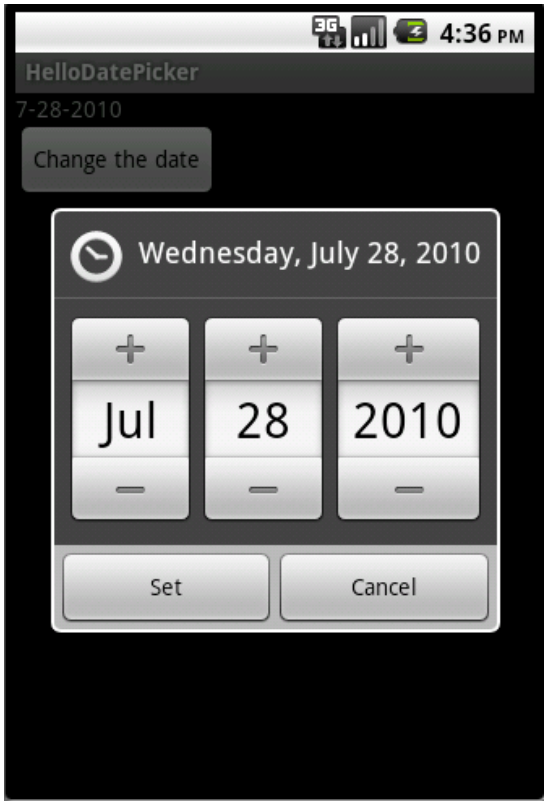
Small Toast showing the text in the clicked item for a short time

Run it!



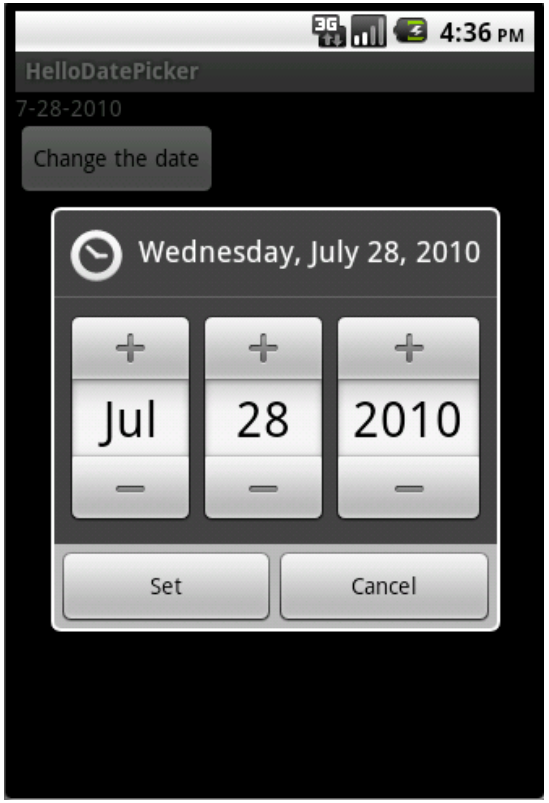
Date Picker

- Will display a dialogbox allowing to change the date



Layout

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="vertical">
      <TextView android:id="@+id/dateDisplay"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text=""/>
      <Button android:id="@+id/pickDate"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Change the date"/>
  </LinearLayout>
```



OnCreate( )

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // capture our View elements
        mDateDisplay = (TextView) findViewById(R.id.dateDisplay);
        mPickDate = (Button) findViewById(R.id.pickDate);

        // add a click listener to the button
        mPickDate.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID);
            }
        });

        // get the current date
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);

        // display the current date (this method is below)
        updateDisplay();
    }
```

updateDisplay( )

```
// updates the date in the TextView
    private void updateDisplay() {
        mDateDisplay.setText(
            new StringBuilder()
                    // Month is 0 based so add 1
                    .append(mMonth + 1).append("-")
                    .append(mDay).append("-")
                    .append(mYear).append(" "));
    }
```

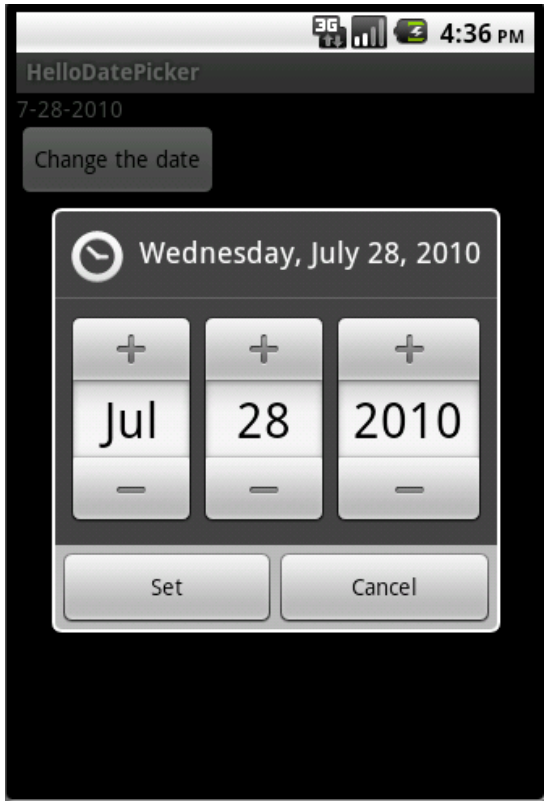
DatePickerDialog.OnDateSetListener( )

```
// the callback received when the user "sets" the date in the dialog
    private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener() {

            public void onDateSet(DatePicker view, int year,
                                  int monthOfYear, int dayOfMonth) {
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateDisplay();
            }
        };
```

onCreateDialog( )

```
@Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case DATE_DIALOG_ID:
            return new DatePickerDialog(this,
                        mDateSetListener,
                        mYear, mMonth, mDay);
        }
        return null;
    }
```

Run it!


4:36 PM
HelloDatePicker
7-28-2010
Change the date
Wednesday, July 28, 2010
Jul
28
2010
Set
Cancel