Android
Date /Time – TabWidget - ActionBar

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html

Date/Time Selection Widgets

Date & Time

Users can set time and date values using either of the Android mechanisms:

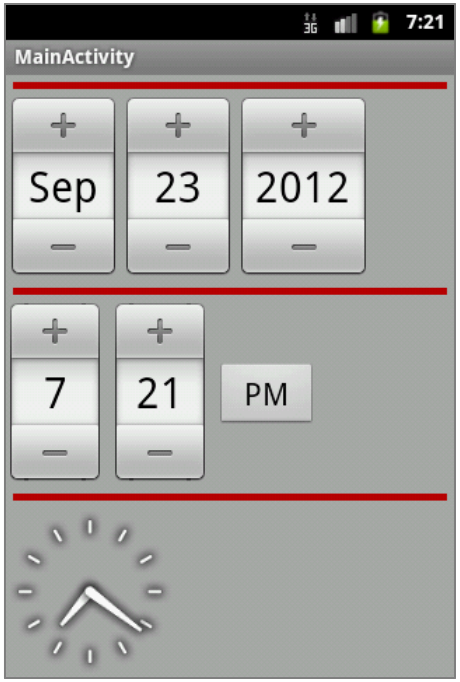
Through UI widgets: **DatePicker**, **TimePicker**

DialogBoxes: **DatePickerDialog**, **TimePickerDialog**

DatePicker

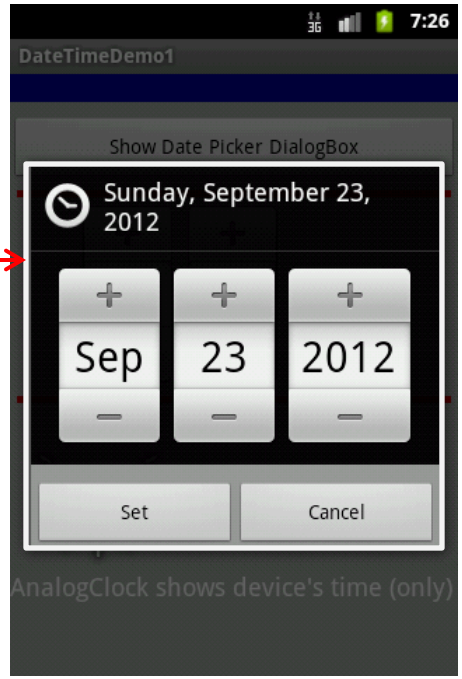
TimePicker

AnalogClock
Shows the device's time



UI Widgets

DatePickerDialog



Dialog Box

Date/Time Selection Widgets

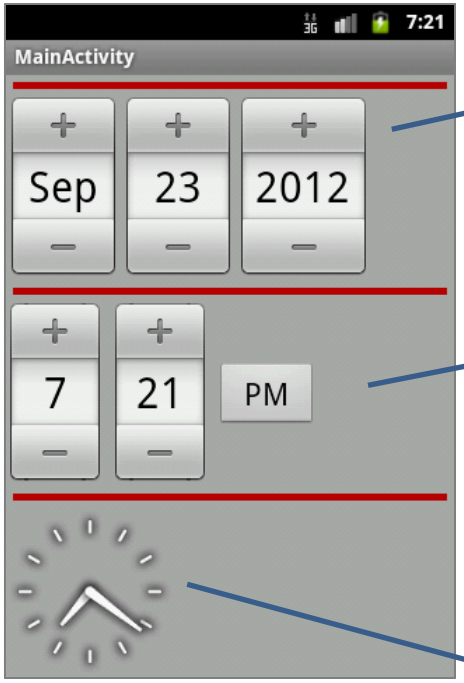
Reacting to UI Date & Time Changes

You may use the callbacks: **OnDateChangedListener** or **OnDateSetListener**

to react to changes made on the UI date (time) widgets

DatePicker

TimePicker



`OnDateChangedListener or OnDateSetListener`

`OnTimeChangedListener or OnTimeSetListener`

Cannot be changed !!!

Date/Time Selection Widgets

Time Selection

The widgets **TimePicker** and **TimePickerDialog** let you:

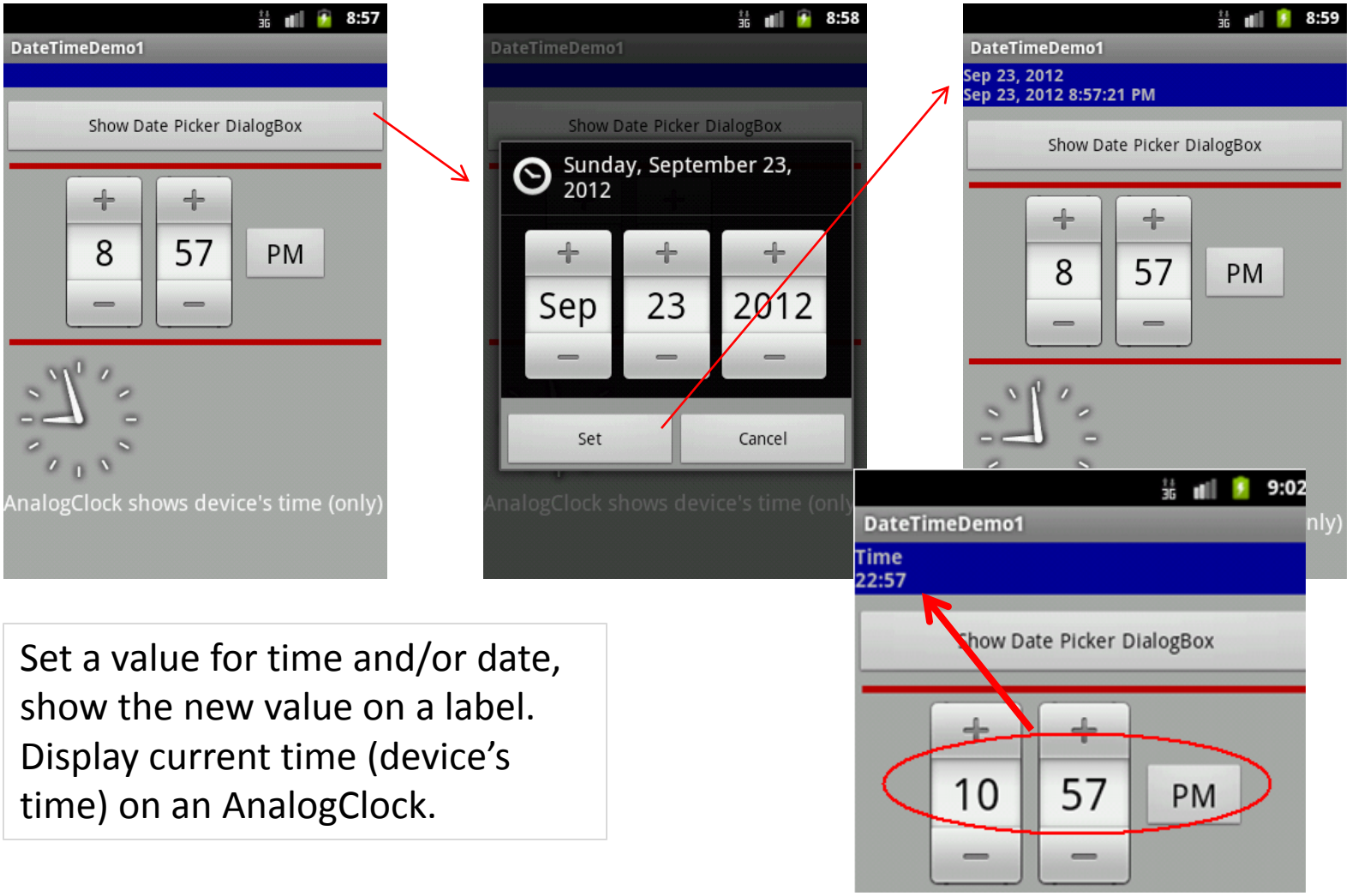
1. set a time [hour, minutes] where:
   hour (***0*** through ***23***) and a ***minute*** (***0*** through ***59***)

2. An AM/PM toggle.

3. provide a callback object:
   OnTimeChangedListener or **OnTimeSetListener**
   to be notified of when the user has chosen a new time.

Date/Time Selection Widgets

Example1: Using Time/Date Widgets

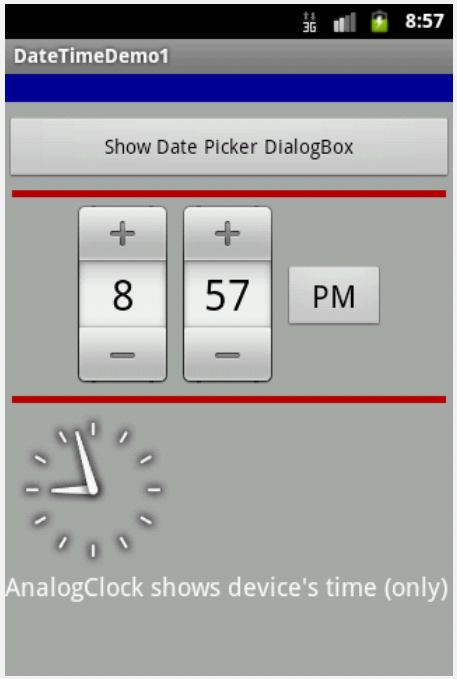


Set a value for time and/or date, show the new value on a label. Display current time (device's time) on an AnalogClock.

Date/Time Selection Widgets

Example1: Using Time/Date Widgets

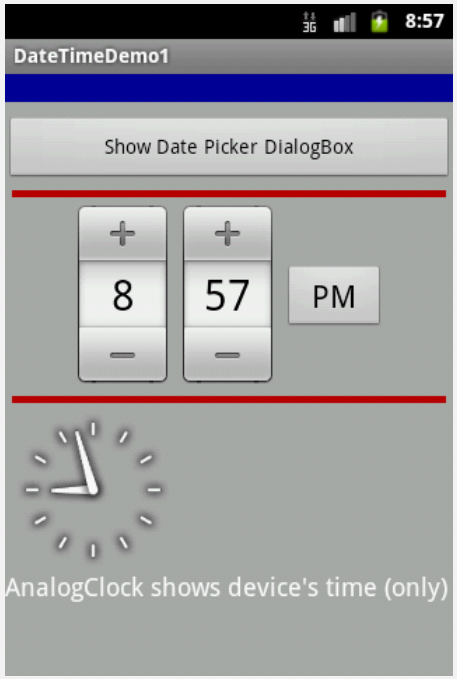
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/widget28"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@android:color/darker_gray"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/lblDateAndTime"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dip"
        android:background="#ff000099"
        android:singleLine="false"
        android:textStyle="bold" />

    <Button
        android:id="@+id/btnDate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Show Date Picker DialogBox" />

    <View
        android:layout_width="match_parent"
        android:layout_height="5dip"
        android:layout_margin="5dip"
        android:background="#ffbb0000" />
```



Date/Time Selection Widgets

Example1: Using Time/Date Widgets

```
    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

    <View
        android:layout_width="match_parent"
        android:layout_height="5dip"
        android:layout_margin="5dip"
        android:background="#ffbb0000" />


    <AnalogClock
        android:id="@+id/analogClock1"
        android:layout_width="126dp"
        android:layout_height="114dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AnalogClock shows device's time (only)"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```



Date/Time Selection Widgets

Example1: Using Time/Date Widgets

```
public class DateTimeDemo1 extends Activity {

   TextView lblDateAndTime;
   Calendar myCalendar = Calendar.getInstance();

   TimePicker time1;

   @Override
   public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);

      // show results here
      lblDateAndTime = (TextView) findViewById(R.id.lblDateAndTime);

      // connect to TimePicker widget already on the UI
      time1 = (TimePicker) findViewById(R.id.timePicker1);
      time1.setOnTimeChangedListener(new OnTimeChangedListener() {
         @Override
         public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            String newTime =  "Time\n" + hourOfDay + ":" + minute;
            lblDateAndTime.setText(newTime);

         }
      });
```

Date/Time Selection Widgets

Example1: Using Time/Date Widgets

```
    // ////////////////////////////////////////////////////
    // show Date Picker DialogBox on top of current UI
    Button btnDate = (Button) findViewById(R.id.btnDate);
    btnDate.setOnClickListener(new View.OnClickListener() {
       public void onClick(View v) {
          DatePickerDialog dateDialog = new DatePickerDialog(
                DateTimeDemo1.this,
                datePicker,
                //first time around show today's date
                myCalendar.get(Calendar.YEAR),
                myCalendar.get(Calendar.MONTH),
                myCalendar.get(Calendar.DAY_OF_MONTH));
          dateDialog.show();
       }
    });

 }// onCreate
```

Date/Time Selection Widgets

Example1: Using Time/Date Widgets

```
    // Date listener - gets user's supplied new value of date
   DatePickerDialog.OnDateSetListener datePicker = new
    DatePickerDialog.OnDateSetListener() {
      public void onDateSet(DatePicker view, int year, int monthOfYear,
                            int dayOfMonth) {

         myCalendar.set(Calendar.YEAR, year);
         myCalendar.set(Calendar.MONTH, monthOfYear);
         myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);

         Date date = myCalendar.getTime();

         String strDate = DateFormat.getDateInstance().format(date);
         String strDateTime = DateFormat.getDateTimeInstance().format(date);

         lblDateAndTime.setText( strDate + "\n" + strDateTime );
      }
   };

} // class
```

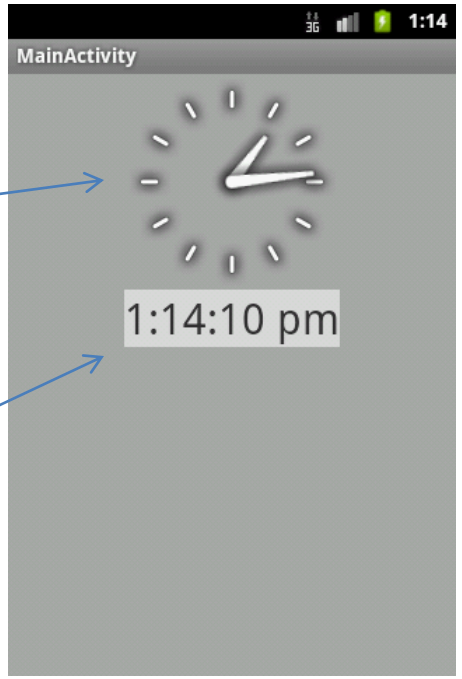
Date/Time Selection Widgets

Other Time Widgets

Android provides a **DigitalClock** and **AnalogClock** widgets.

Automatically update with the passage of time (no user intervention is required).

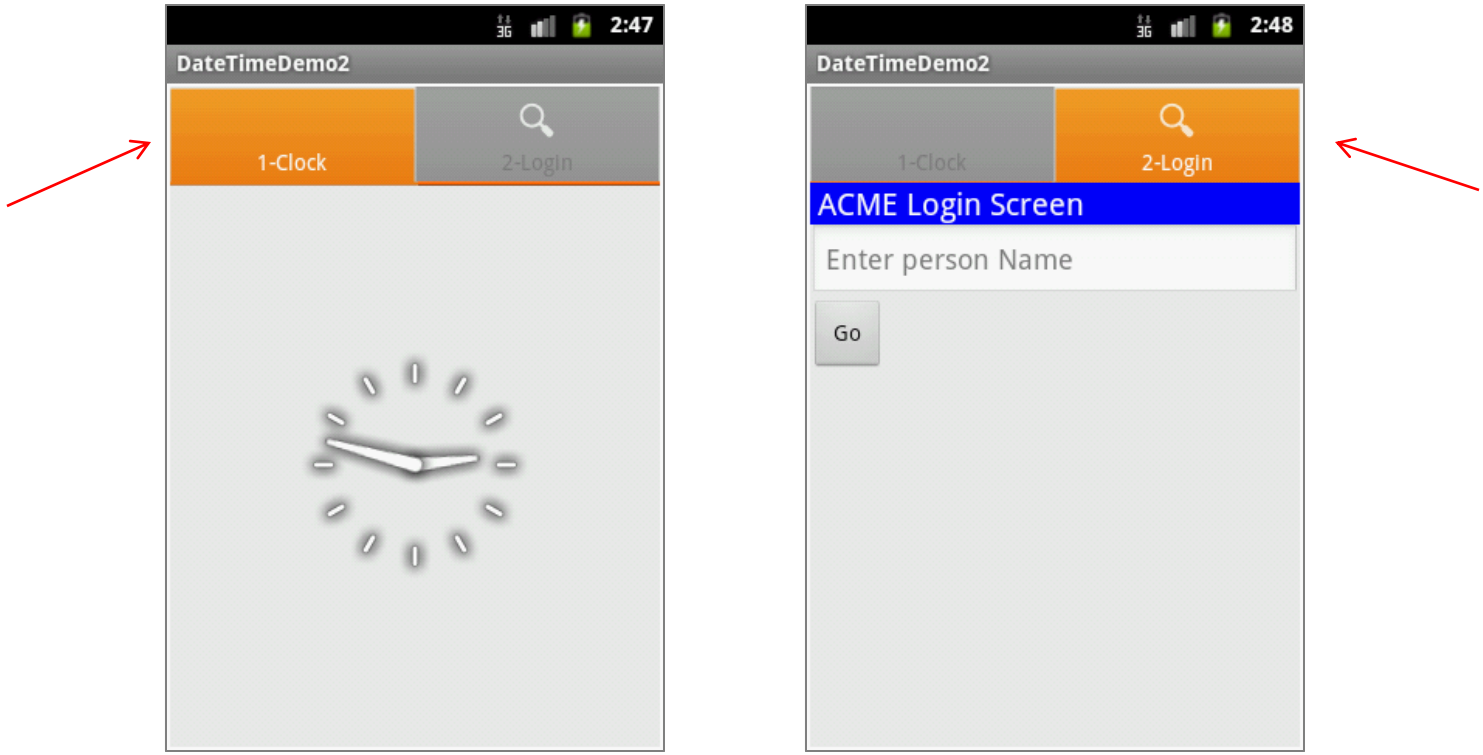
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@android:color/darker_gray">

    <AnalogClock
        android:id="@+id/analog"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >
    </AnalogClock>

    <DigitalClock
        android:id="@+id/digital"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/analog"
        android:layout_centerHorizontal="true"
        android:background="#FFdddddd"
        android:textSize="30sp" >
    </DigitalClock>
</RelativeLayout>
```



Tab Selection Widget

Tab Selector

1. Handheld devices usually offer limited screen space.
2. Their UI design should be effective and simple.
3. Complex apps having many visual elements could benefit from the **Tab Widget** which maintains the awareness of the pieces but shows only a few fragments at the time.



Tab Selection Widget

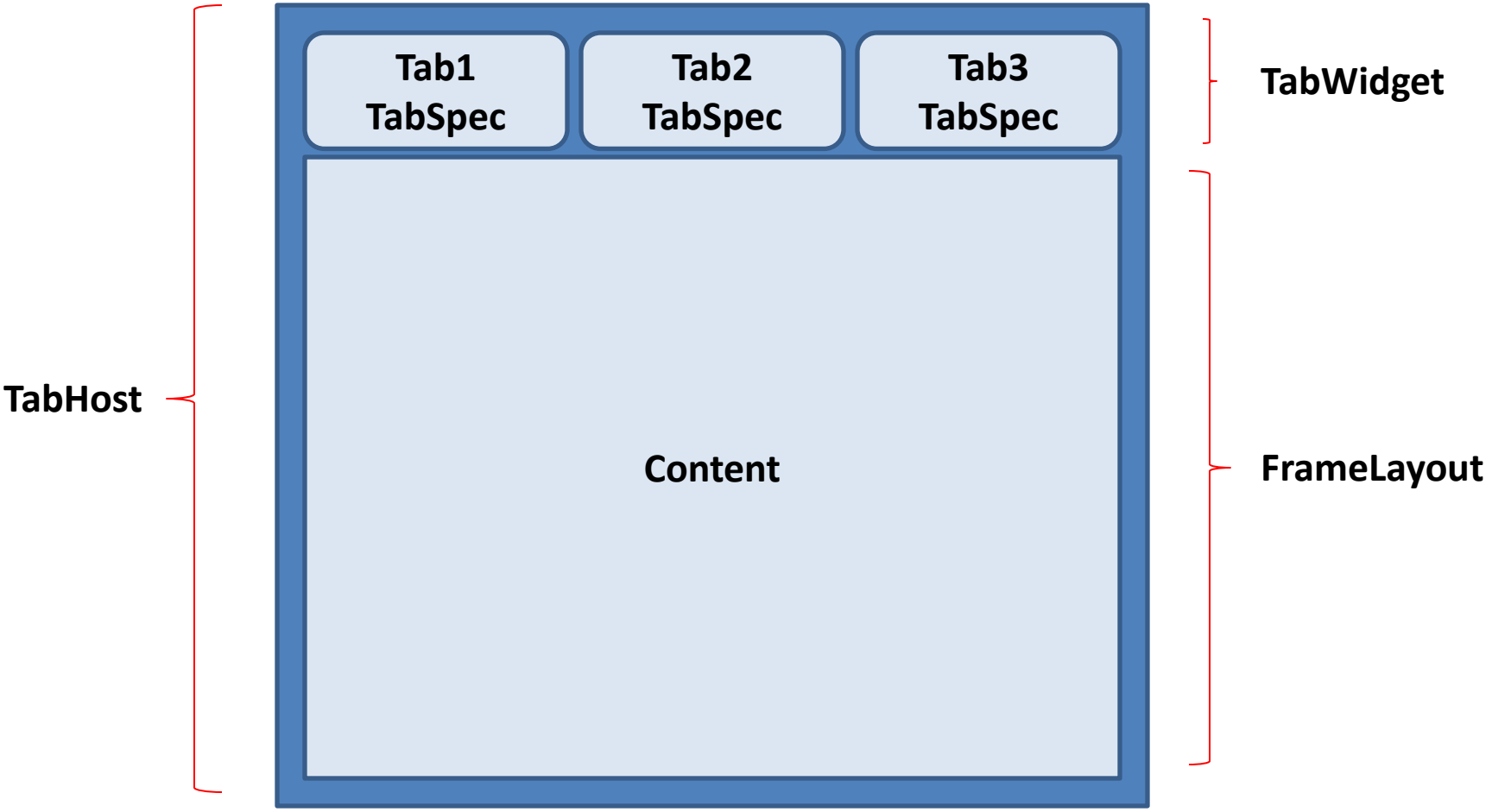
Tab Selector – Components

A Tabbed UI consists of three pieces that you need to set:

1. **TabHost** is the main container for the tab buttons and tab contents
2. **TabSpec** implements the row of tab buttons, which contain text labels (and optionally contain icons)
3. **FrameLayout** is the container for the tab contents

Tab Selection Widget

Tab Selector – Components



Tab Selection Widget

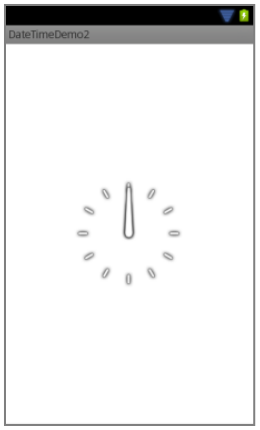
Example2: Using Tabs - A handcrafted solution

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="2dip">

    <TabHost
        android:id="@+id/tabhost"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#ffeeeeee">

        <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:paddingTop="62dip" >

            <include layout="@layout/main_tab1" />

            <include layout="@layout/main_tab2" />

        </FrameLayout>
    </TabHost>

</LinearLayout>
```

*You may enter here the actual layout specification, or (better) use the <**include**> tag to refer to an external layout assembled in a separated xml file.*

Details in next pages…

Tab Selection Widget



Example2: Using Tabs

This is the layout specification in **main_tab1.xml**.
It defines an analog clock.
It is injected in the main.xml via `<include layout="@layout/main_tab1" />`)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <AnalogClock
        android:id="@+id/tab1Clock"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center_horizontal" />

</LinearLayout>
```

Tab Selection Widget

Example2: Using Tabs

This is **main_tab2.xml**.
It defines a *LinearLayout* holding a *label*, a textBox, and a *button*.

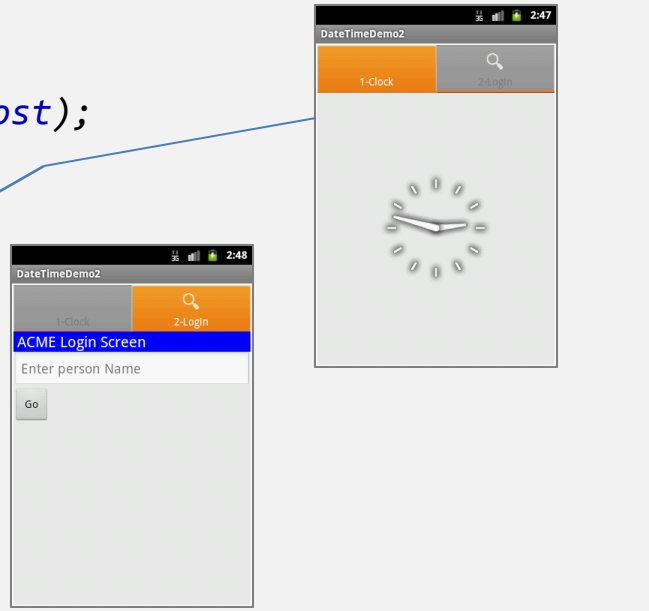
Inserted in main.xml using <include layout=@Layout/... >


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab2"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text=" ACME Login Screen"
        android:textColor="@android:color/white"
        android:textSize="20sp"  />

    <EditText
        android:id="@+id/tab2TxtPerson"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Enter person Name"
        android:inputType="textCapWords"
        android:textSize="18sp" />

    <Button
        android:id="@+id/tab2BtnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Go " />

</LinearLayout>
```

Tab Selection Widget

Example2: Using Tabs

```
public class DateTimeDemo2 extends Activity {
   TabHost tabhost;

   @Override
   public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);

      tabhost = (TabHost) findViewById(R.id.tabhost);

      tabhost.setup();

      TabHost.TabSpec tabspec;

      tabspec = tabhost.newTabSpec("screen1");
      tabspec.setContent(R.id.tab1);
      tabspec.setIndicator("1-Clock",null);
      tabhost.addTab(tabspec);

      tabspec = tabhost.newTabSpec("screen2");
      tabspec.setContent(R.id.tab2);
      tabspec.setIndicator("2-Login",
                  getResources().getDrawable(R.drawable.ic_action_search));
      tabhost.addTab(tabspec );

      tabhost.setCurrentTab(0);
```



Tab Selection Widget

Example2: Using Tabs

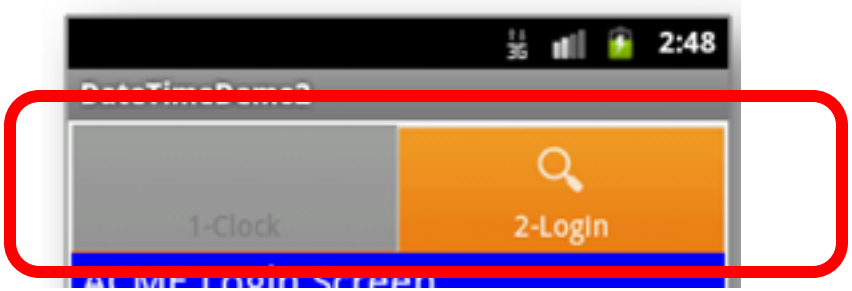
```
      // wiring UI widgets shown in the various user-screens
       AnalogClock clock1 = (AnalogClock) findViewById(R.id.tab1Clock);

      Button btnGo = (Button) findViewById(R.id.tab2BtnGo);
      btnGo.setOnClickListener(new OnClickListener() {

         public void onClick(View arg0) {
            EditText txtPerson = (EditText) findViewById(R.id.tab2TxtPerson);
            String theUser = txtPerson.getText().toString();
            txtPerson.setText("Hola " + theUser);

         }
      });

   }// onCreate

}// class
```

Tab Selection Widget

HINT

Example2: Using Tabs

You may decorate the tab indicator Including text and image as shown below:




```
tabspec = tabhost.newTabSpec("screen2");

  tabspec.setContent(R.id.tab2);

  tabspec.setIndicator("2-Login",
          getResources().getDrawable(R.drawable.ic_action_search ));

tabhost.addTab(tabspec);
```

Note: Many icons available at android-sdk-folder\docs\images\icon-design

Tab Selection Widget

Example2: Using Tabs

You may want to add a listener to monitor the selecting of a particular tab. Add this fragment to the ***onCreate*** method.

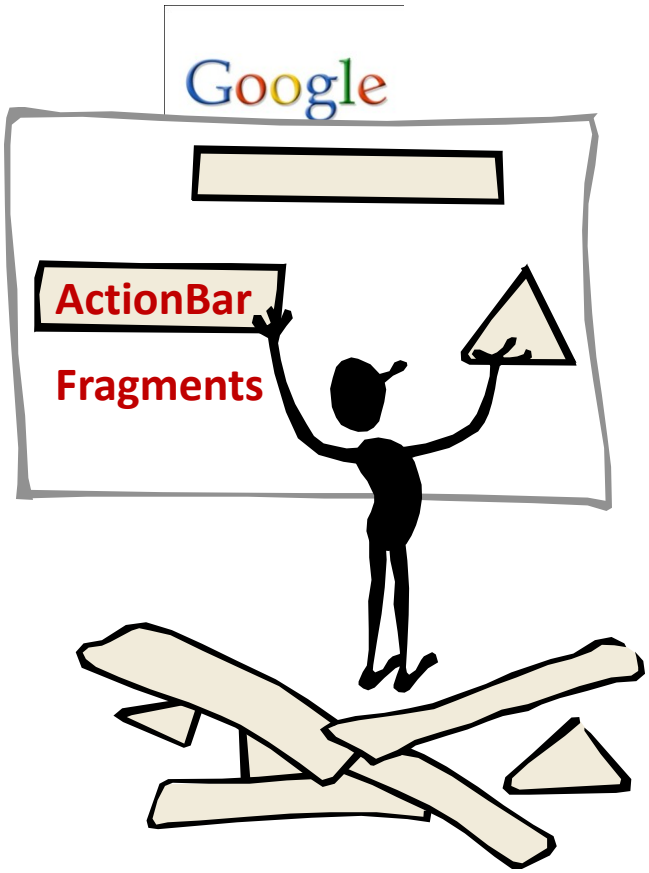
```
// tabs.setCurrentTab(0);
// you may also use
tabs.setCurrentTabByTag("screen1");

    tabs.setOnTabChangedListener(new OnTabChangeListener() {
        @Override
        public void onTabChanged(String tagId) {
        // do something useful with the selected screen
        String text =  "Im currently in: " + tagId
                    + "\nindex: " + tabs.getCurrentTab();

         Toast.makeText(getApplicationContext(), text, 1).show();
        }
    });
```

This fragment returns:

```
Im currently in: tag1
index: 0
```

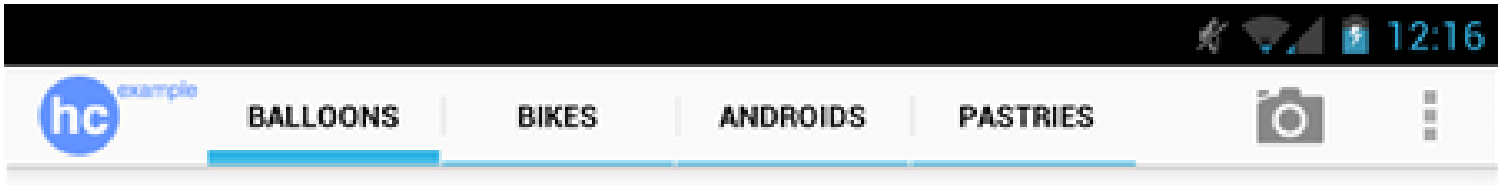
New way of doing things…

“In previous versions of Android, tabs could be implemented using a **TabWidget** and **TabHost** … As of Android 3.0, however, you should use either **NAVIGATION_MODE_TABS** … along with the **ActionBar** class”.



Why?
It is very desirable to obtain a more common ‘look-&-feel’ appeal across applications and devices.

This commonality should make the user experience simpler and more enjoyable.



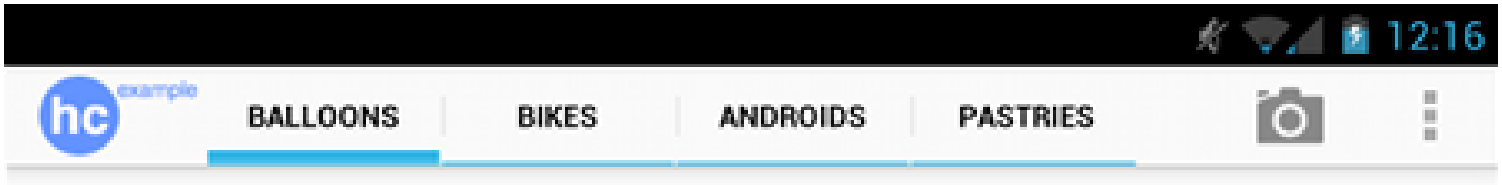
ActionBar

The *action bar* is a dedicated view-design at the top of each screen that is generally persistent throughout the app.

It provides several key functions:

1. Makes important actions prominent and accessible in a predictable way (such as *New* or *Search*).
2. Supports consistent navigation and view switching within apps.
3. Reduces clutter by providing an action overflow for rarely used actions.
4. Provides a dedicated space for giving your app an identity

Reference:
http://developer.android.com/guide/topics/ui/actionbar.html#Tabs
http://developer.android.com/design/patterns/actionbar.html



ActionBar

Beginning with Android 3.0 (API level 11), the action bar appears at the top of an activity's window when the activity uses the system's **Holo** theme (or one of its descendant themes), which is the default.

You may otherwise add the action bar by calling

requestFeature (FEATURE_ACTION_BAR)

or by declaring it in a custom theme with the windowActionBar property.

Reference:
http://developer.android.com/guide/topics/ui/actionbar.html#Tabs
http://developer.android.com/design/style/iconography.html (download ActionBar Icon Package)

Fragments

Fragments

- A **Fragment** represents a behavior or a portion of user interface in an **Activity**.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- 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 think of a fragment as a modular section of an activity capable of processing its own input events.

Reference:
http://developer.android.com/guide/components/fragments.html

Fragments

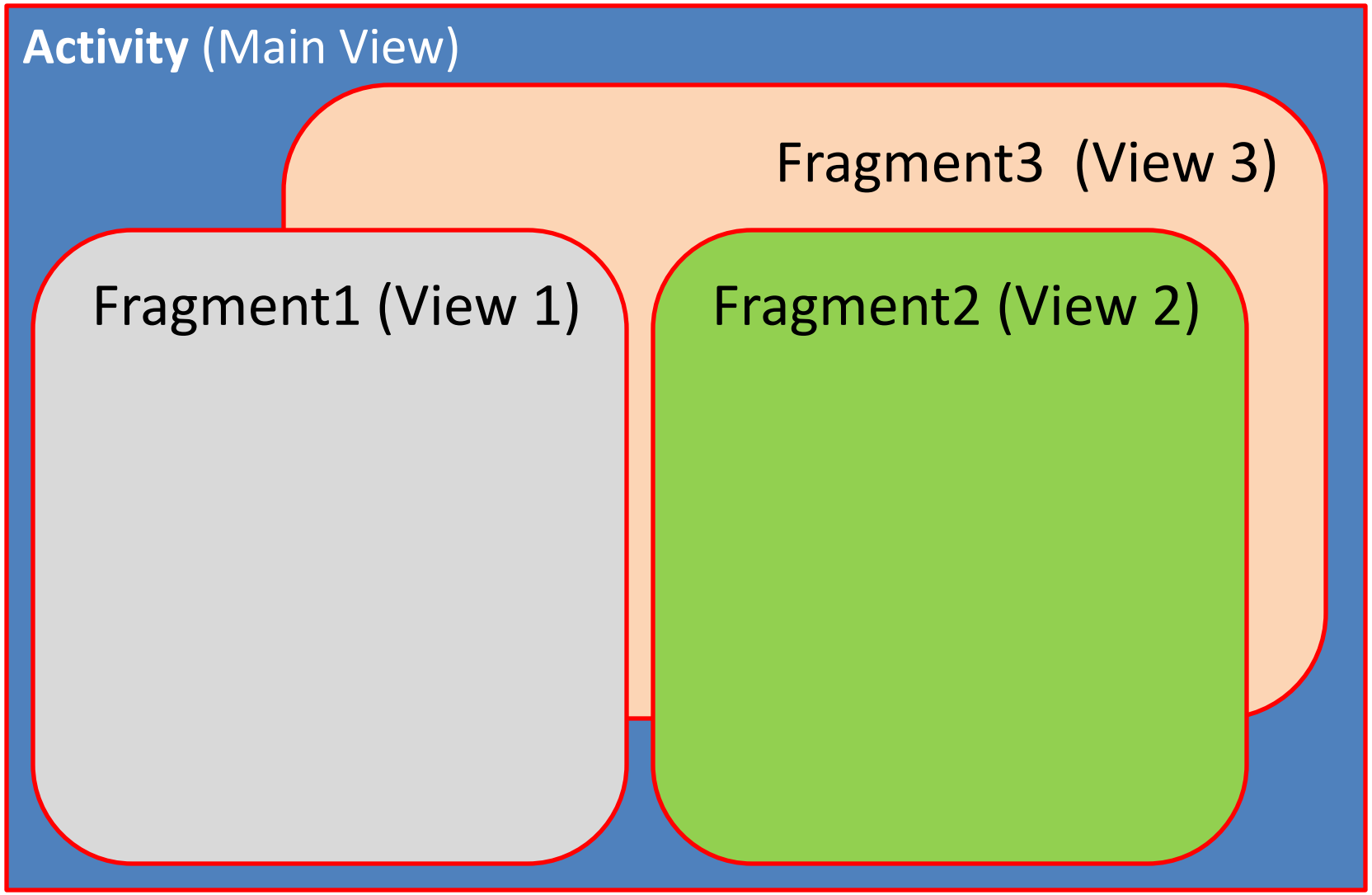
Fragments

- When you add a fragment as a part of your activity layout, it lives in a **ViewGroup** inside the activity's view hierarchy and the fragment defines its own view layout.

- 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**.

Reference:
http://developer.android.com/guide/components/fragments.html

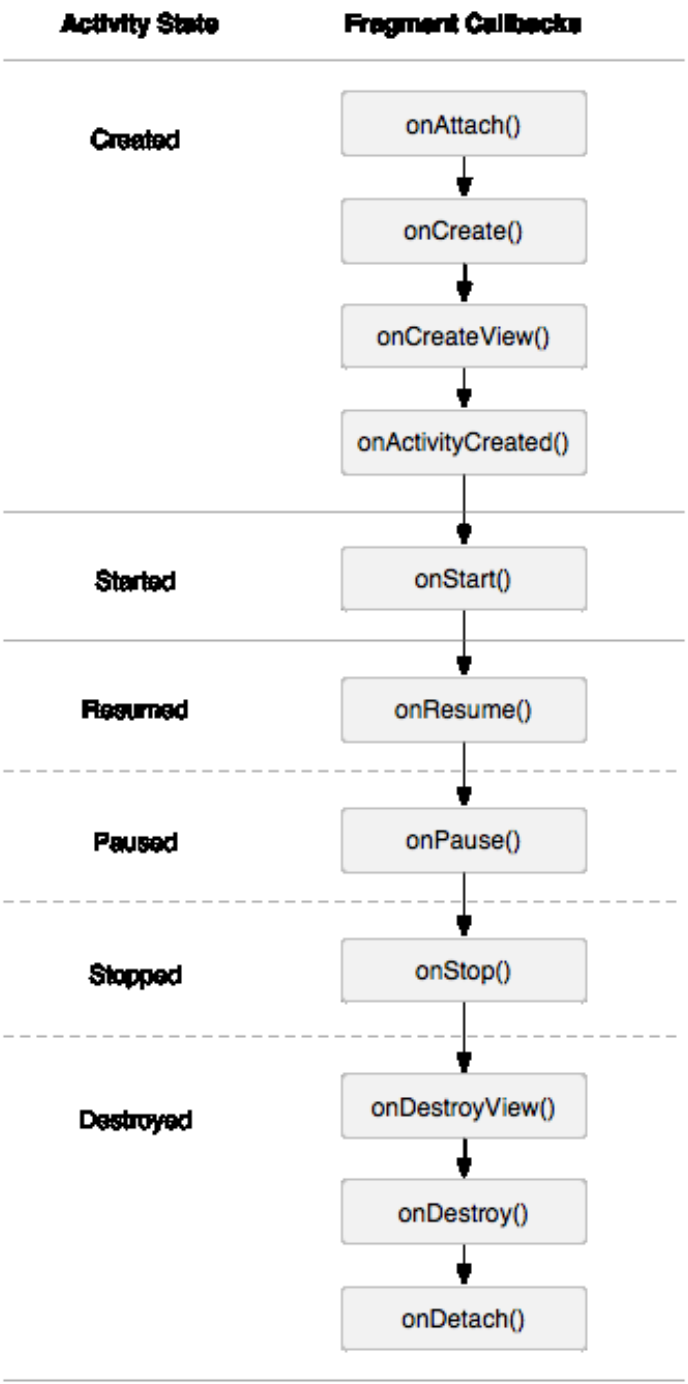
Fragments


Activity (Main View)
Fragment3 (View 3)
Fragment1 (View 1)
Fragment2 (View 2)

Fragment's Lifecycle

Reference:
http://developer.android.com/guide/components/fragments.htm

onAttach()Called when the fragment has been associated with the activity
onCreateView()Called to create the view hierarchy associated with the fragment.
onActivityCreated()Called when the activity's onCreate() method has returned.
onDestroyView()Called when the view hierarchy associated with the fragment is being removed.
onDetach()Called when the fragment is being disassociated from the activity.



Fragments

Inter-Fragment Communication

Reference:
http://developer.android.com/training/basics/fragments/communicating.html

- All Fragment-to-Fragment communication is done through the associated Activity.
- Two Fragments should *never* communicate directly.
- Activity and fragments interact through listeners and events (resp).

Fragments

Example3: Using Fragments and ActionBars

In this example an application shows a multi-tabbed UI. The 'look-&-feel' of the app is in line with the notion of standardization across devices /apps.

Individual tabs are implemented as Fragment objects. The screens operate as follows:

Tab1 Displays a list of picture names. When the fragment attaches to the main activity, a listener (in the main) is set to receive updates from the fragment's onItemSelected event. This strategy keeps the activity aware of selections made in fragment1.

Tab2 A GridView depicting all the images whose names were shown in fragment1 (TODO: keep activity informed of user's choices).

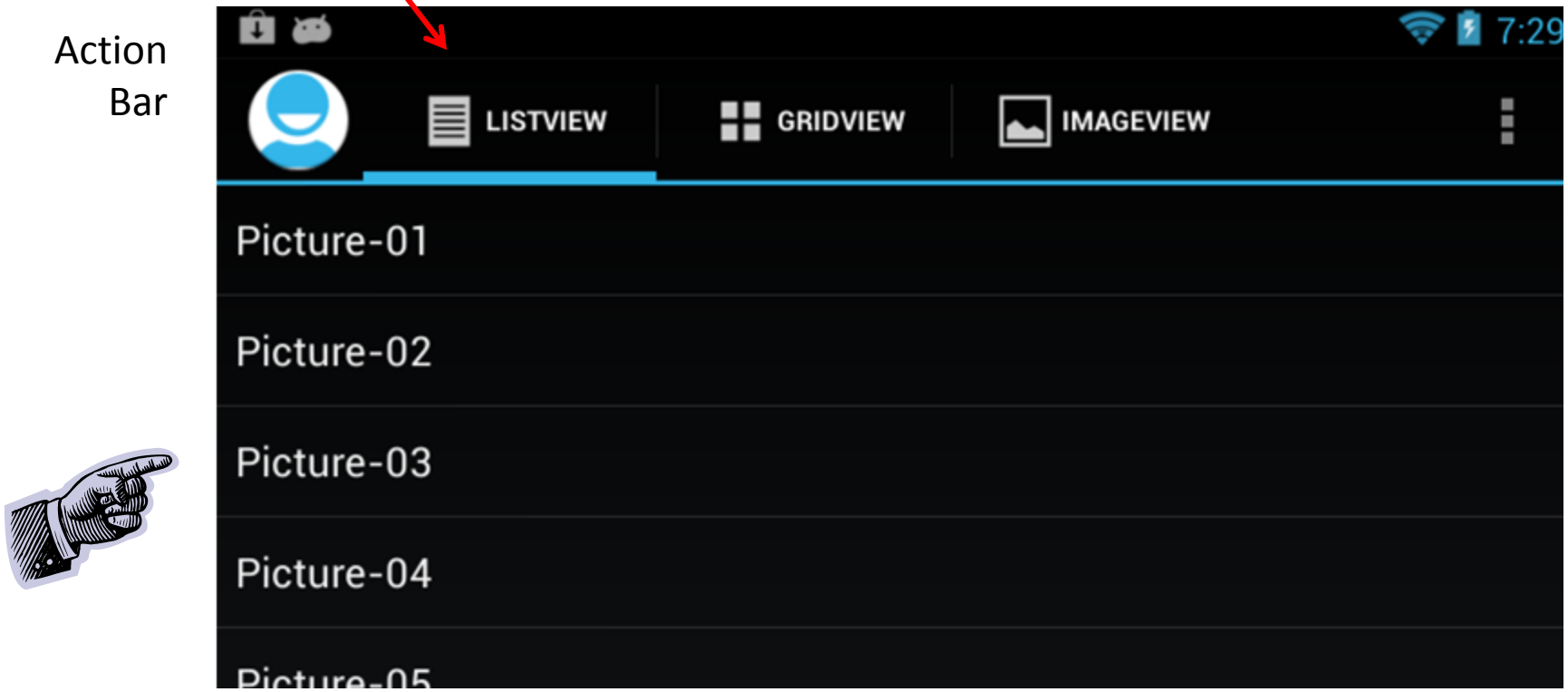
Tab3 Display a 'good quality' version of the picture selected by the user in fragment1.

Example3: Using Fragments and ActionBars

Fragment1

User makes a selection. Row position is sent back to main activity's listener

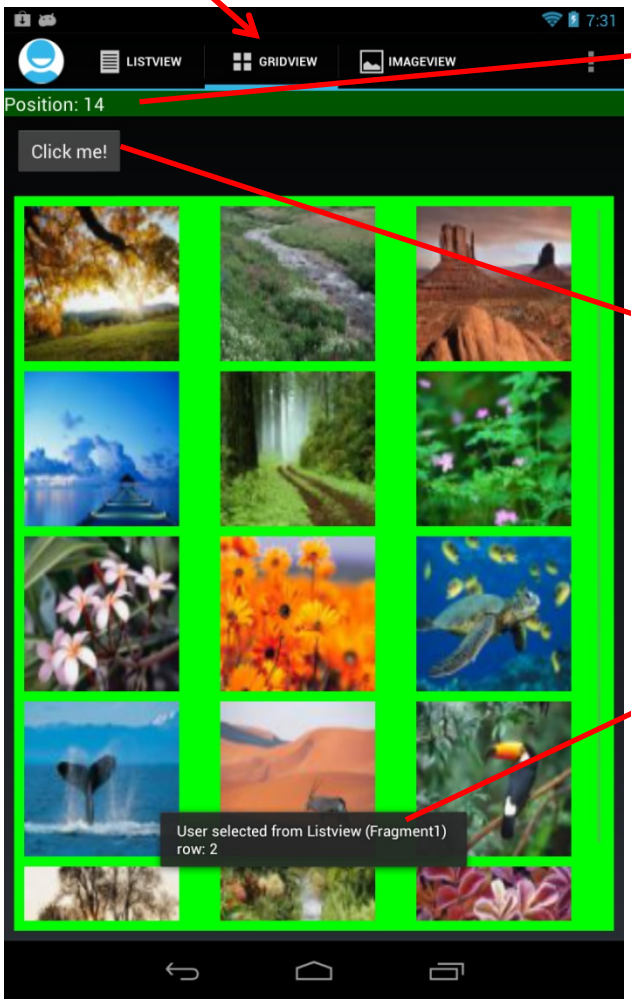
Action Bar



Example3: Using Fragments and ActionBars

Fragment2

User makes a selection. Row position is locally recognized (TODO: make main activity aware of it)



Here you tapped on image number …

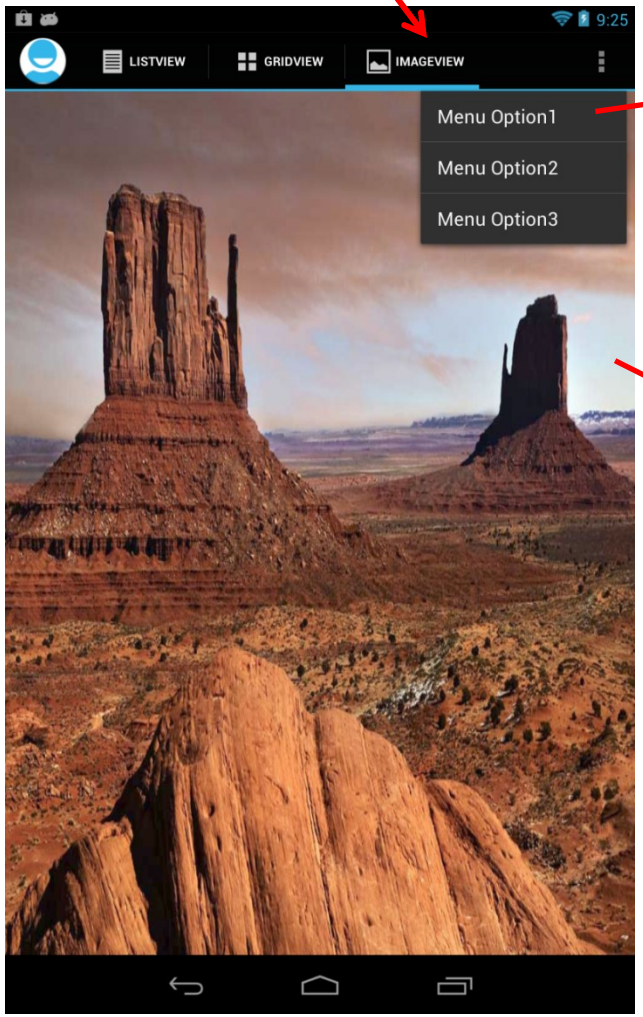
Button
Click to display current date/time

Notification of previous action:
User selected in fragment1 the row number ….

Example3: Using Fragments and ActionBars

Fragment3

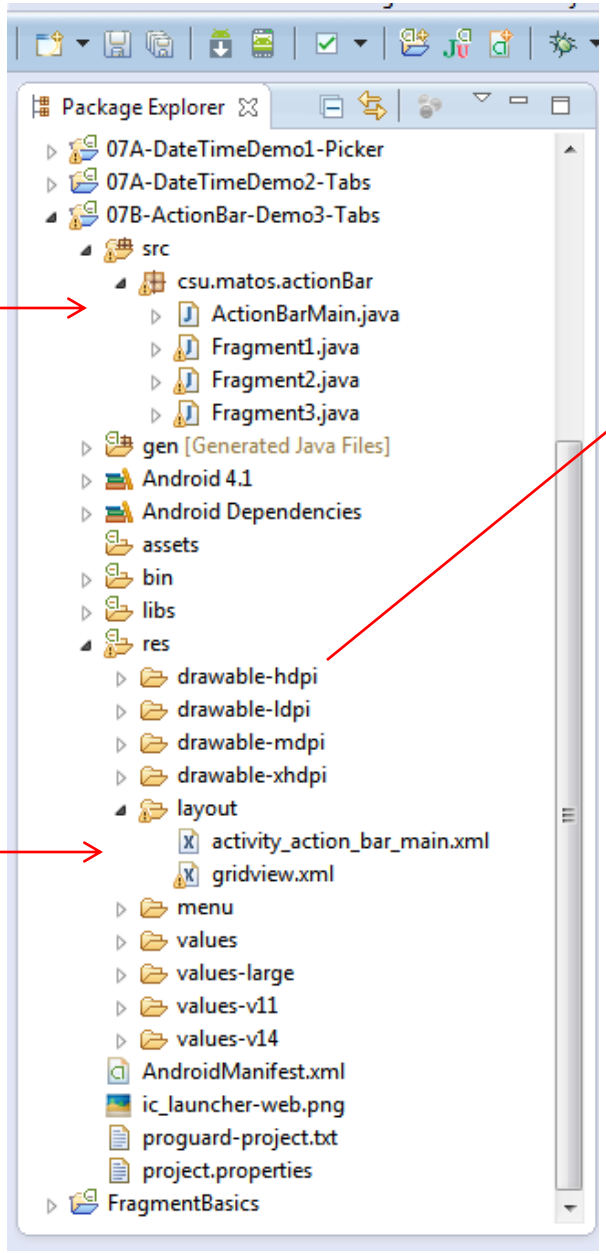
Shows high-quality image of picture selected in fragment1

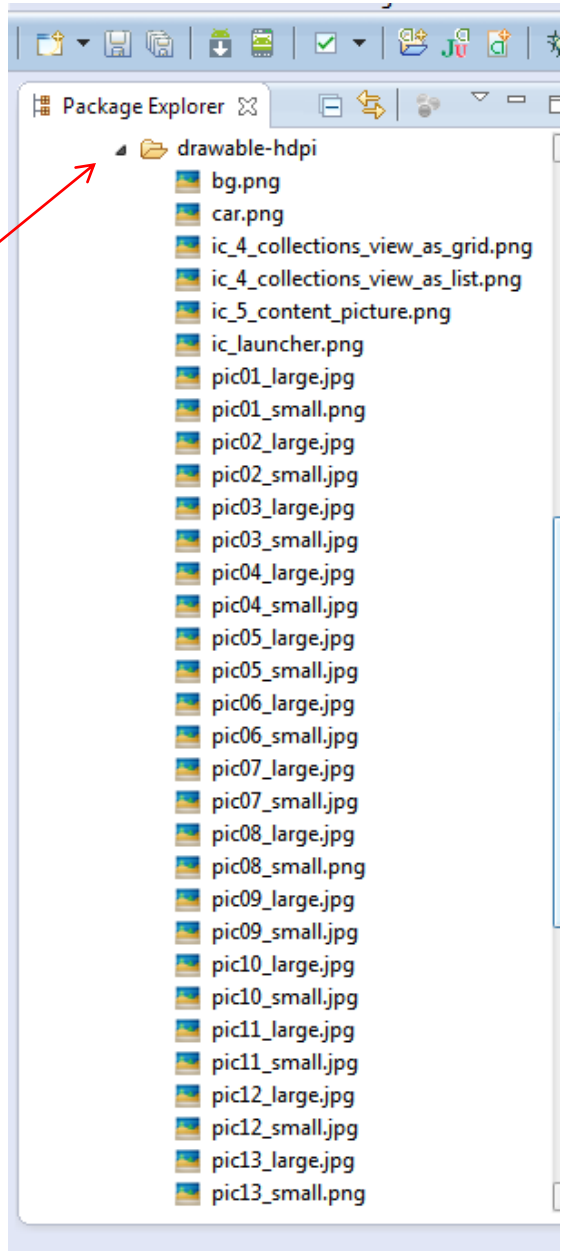


Menu options

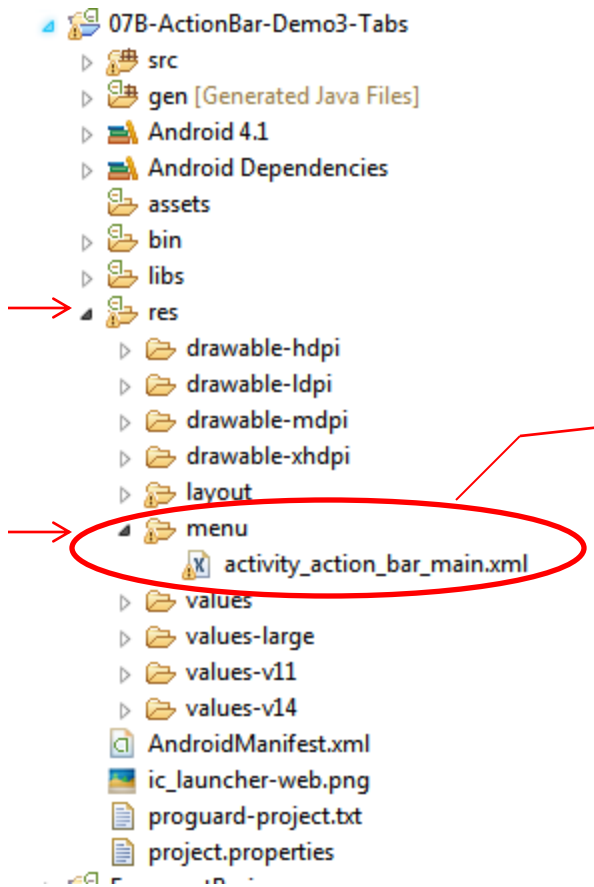
Image selected by the user in fragment1

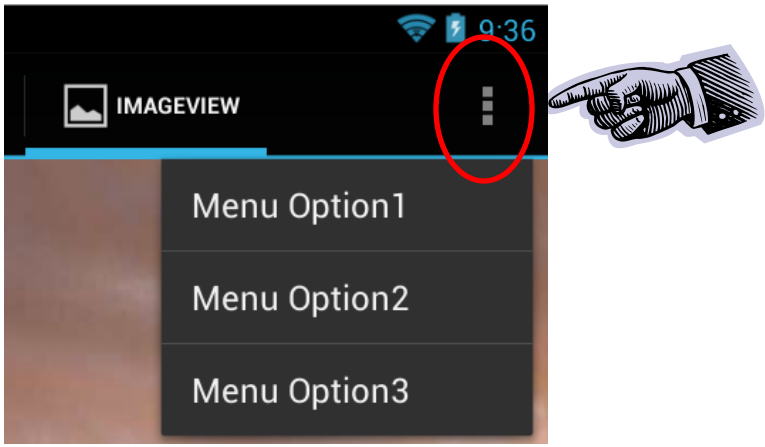
Example3: Using Fragments and ActionBars





Example3: Using Fragments and ActionBars






```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_settings"
        android:orderInCategory="100"
        android:title="Menu Option1" />
  <item
        android:id="@+id/menu_settings"
        android:orderInCategory="110"
        android:title="Menu Option2" />
  <item
        android:id="@+id/menu_settings"
        android:orderInCategory="120"
        android:title="Menu Option3" />
</menu>
```

Example3: Using Fragments and ActionBars

Main Activity Layout
(*activity_action_bar_main*)

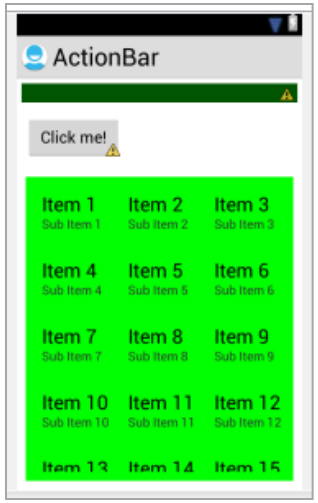
Main layout provides an empty space in which fragments will place their own UIs

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:id="@+id/mainLayout">

</RelativeLayout>
```

Example3: Using Fragments and ActionBars

GridView Layout

This layout will be inflated by fragment2




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

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_margin="10dp"
        android:background="#ff005500" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/editText1"
        android:layout_margin="10dp"
        android:text=" Click me! " />

    <GridView
        android:id="@+id/mainGrid"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/button1"
        android:layout_margin="10dip"
        android:background="#ff00ff00"
        android:horizontalSpacing="10dp"
        android:verticalSpacing="10dp"
        android:numColumns="3"
        android:padding="10dip"
        android:stretchMode="columnWidth" >
    </GridView>

</RelativeLayout>
```

Example3: Using Fragments and ActionBars

Main Activity: ActionBarMain


```
public class ActionBarMain extends Activity implements TabListener,
                                                    onPictureSelectedListener  {

   // data shared by fragments 1 & 2
   Integer selectedRow = 0;

   RelativeLayout mainLayout;

   FragmentTransaction fragTransactMgr = null;

   // tab's captions
   private final String CAPTION1 = "ListView";
   private final String CAPTION2 = "GridView";
   private final String CAPTION3 = "ImageView";

   // ////////////////////////////////////////////////////////

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_action_bar_main);
      try {
         mainLayout = (RelativeLayout) findViewById(R.id.mainLayout);
```

Example3: Using Fragments and ActionBars

Main Activity: ActionBarMain


```
        fragTransactMgr = getFragmentManager().beginTransaction();

        ActionBar bar = getActionBar();

        //create tabs adding caption and icon
        bar.addTab(bar.newTab().setText(CAPTION1)
                          .setIcon(R.drawable.ic_4_collections_view_as_list)
                          .setTabListener(this));

        bar.addTab(bar.newTab().setText(CAPTION2)
                          .setIcon(R.drawable.ic_4_collections_view_as_grid)
                          .setTabListener(this));

        bar.addTab(bar.newTab().setText(CAPTION3)
                            .setIcon(R.drawable.ic_5_content_picture)
                            .setTabListener(this));

        bar.setDisplayOptions(ActionBar.DISPLAY_SHOW_CUSTOM
              |               ActionBar.DISPLAY_USE_LOGO);
        bar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

        bar.setDisplayShowHomeEnabled(true);
        bar.setDisplayShowTitleEnabled(false);

        bar.show();

     } catch (Exception e) {
        //do something smart with the exception here
     }
  }//onCreate
```

Example3: Using Fragments and ActionBars

Main Activity: ActionBarMain


```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.activity_action_bar_main, menu);
   return true;
}
// ////////////////////////////////////////////////////////
@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
}
// ////////////////////////////////////////////////////////
@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {

   if (tab.getText().equals(CAPTION1)) {
      executeFragment( new Fragment1() );

   } else if (tab.getText().equals(CAPTION2)) {
      executeFragment( new Fragment2(selectedRow) );

   } else if (tab.getText().equals(CAPTION3)) {
      executeFragment( new Fragment3(selectedRow) );
   }
}
// ////////////////////////////////////////////////////////
@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
}
```

Example3: Using Fragments and ActionBars

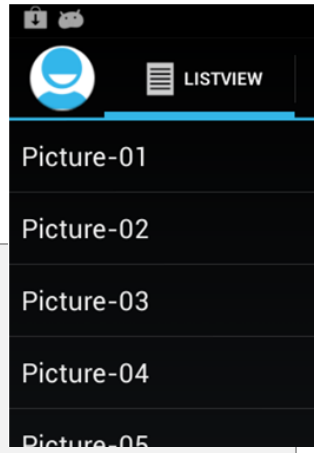
Main Activity: ActionBarMain


```
    public void executeFragment (Fragment fragment) {
        try {
            mainLayout.removeAllViews();
            fragTransactMgr.addToBackStack(null);
            fragTransactMgr = getFragmentManager().beginTransaction();
            fragTransactMgr.add(mainLayout.getId(), fragment);
            fragTransactMgr.commit();
        } catch (Exception e) {

        }
    }//executeFragment

    // ////////////////////////////////////////////////////////////
    // this method supports fragment-to-Activity communication. When
    // a row in Fragment1 is selected, this callBack is invoked. It
    // updates the valued of 'selectedRow' held in the main activity.

    @Override
    public void onPictureSelected(Integer selectedRow) {
        // as soon as the user picks a row in fragment1, its value
        // (position in the list) is saved here
        this.selectedRow = selectedRow;
    }

}//class
```

Example3: Using Fragments and ActionBars

Fragment1

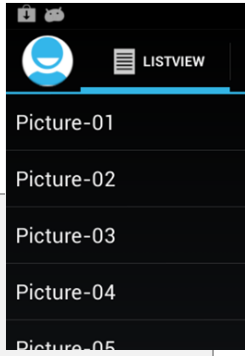
1 of 2




```
public class Fragment1 extends Fragment {

   onPictureSelectedListener mListener;

   private String items[] = {
         "Picture-01","Picture-02","Picture-03","Picture-04","Picture-05",
         "Picture-06","Picture-07","Picture-08","Picture-09","Picture-10",
         "Picture-11","Picture-12","Picture-13","Picture-14","Picture-15" };

   @Override
   public View onCreateView(LayoutInflater inflater,
                            ViewGroup container,
                            Bundle savedInstanceState) {

      // this view is dynamically created with code
      ListView listView = new ListView(getActivity());

      ArrayAdapter<String> array = new ArrayAdapter<String>(
                                          getActivity(),
                                          android.R.layout.simple_list_item_1,
                                          items);


      listView.setAdapter(array);
```

Example3: Using Fragments and ActionBars

Fragment1




```
        listView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent,
                                    View v, int position, long id) {
                Toast.makeText(getActivity(), " you picked: " + position, 1).show();
                // Send the event and clicked item's row ID to the host activity
                Listener.onPictureSelected(position);
            }
        });
        return listView;
    }//onCreateView

    // Main Activity must implement this interface
     public interface onPictureSelectedListener {
         public void onPictureSelected(Integer selectedRow);
     }

     @Override
     public void onAttach(Activity activity) {
         super.onAttach(activity);
         try {
             mListener = (onPictureSelectedListener) activity;
         } catch (ClassCastException e) {
             throw new ClassCastException(activity.toString()
                     + " must implement onPictureSelectedListener");
         }
     }//onAttach
}//class
```

Example3: Using Fragments and ActionBars

Fragment2

```
public class Fragment2 extends Fragment implements OnItemClickListener  {
   EditText txtMsg;
   Button btnGo;
   Integer[] smallImages = {
         R.drawable.pic01_small, R.drawable.pic02_small, R.drawable.pic03_small,
         R.drawable.pic04_small, R.drawable.pic05_small, R.drawable.pic06_small,
         R.drawable.pic07_small, R.drawable.pic08_small, R.drawable.pic09_small,
         R.drawable.pic10_small, R.drawable.pic11_small, R.drawable.pic12_small,
         R.drawable.pic13_small, R.drawable.pic14_small, R.drawable.pic15_small };
   Integer selectedRow;

   public Fragment2(Integer selectedRow) {
      super();
      this.selectedRow = selectedRow;
   }

   // this view is inflated using an XML layout file
   @Override
   public View onCreateView(LayoutInflater inflater,
                            ViewGroup container,
                            Bundle savedInstanceState) {

      View view = inflater.inflate(R.layout.gridview, null);
      GridView listView = (GridView) view.findViewById(R.id.mainGrid);

      txtMsg = (EditText) view.findViewById(R.id.editText1);
```

Example3: Using Fragments and ActionBars

Fragment2

```
    btnGo = (Button) view.findViewById(R.id.button1);
    btnGo.setOnClickListener(new OnClickListener() {

       @Override
       public void onClick(View v) {
          String text = new Date().toString();
          txtMsg.setText("NEW " + text);
       }
    });


    listView.setAdapter(new Adapter( getActivity() ));
    listView.setOnItemClickListener(this);

    // tell here what picture was already selected in fragment1
    String text = "User selected from Listview (Fragment1)\nrow: "
                + selectedRow;
    Toast.makeText(getActivity(), text, 1).show();

    return view;
}
```

Example3: Using Fragments and ActionBars

Fragment2


```
private class Adapter extends BaseAdapter {
   Context ctx;
   public Adapter(Context ctx){
      this.ctx = ctx;
   }
   @Override
   public int getCount() {
      return smallImages.length;
   }

   @Override
   public Object getItem(int position) {
      return null;
   }

   @Override
   public long getItemId(int position) {
      // TODO Auto-generated method stub
      return 0;
   }
```

Example3: Using Fragments and ActionBars

Fragment2

```
    @Override
    public View getView(int position,
                        View convertView,
                        ViewGroup parent) {
       ImageView image;

       if (convertView == null) {
          image = new ImageView(Fragment2.this.getActivity());
          image.setLayoutParams(new GridView.LayoutParams(50, 50));
          image.setScaleType(ScaleType.FIT_XY);
          convertView = image;

       } else {
          image = (ImageView) convertView;
       }

       txtMsg.setText("Position: " + position);
       image.setImageResource(smallImages[position]);

       return image;
    }

 }//ViewAdapter
```

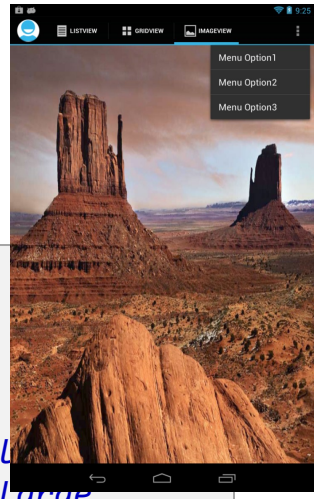
Example3: Using Fragments and ActionBars

Fragment2


```
   // ////////////////////////////////////////////////////////
   //TODO: repeat strategy used in fragment1, when user clicks
   //    on image let the callback method in main activity
   //    know what image (position) has been selected

   @Override
   public void onItemClick( AdapterView<?> parent, View v,
                            int position, long id) {
      txtMsg.setText("Position selected " + position);

   }//onItemClick

   // ////////////////////////////////////////////////////////
   public void updateItemFromList1(Integer selectedRow ){
      txtMsg.setText("User chose in LISTVIEW row#: " + selectedRow );
   }


}//Activity
```

Example3: Using Fragments and ActionBars

Fragment3




```
public class Fragment3 extends Fragment {

   private Integer selectedRow;

   Integer[] largeImages = {
         R.drawable.pic01_large, R.drawable.pic02_large, R.drawable.pic03_l
         R.drawable.pic04_large, R.drawable.pic05_large, R.drawable.pic06_large,
         R.drawable.pic07_large, R.drawable.pic08_large, R.drawable.pic09_large,
         R.drawable.pic10_large, R.drawable.pic11_large, R.drawable.pic12_large,
         R.drawable.pic13_large, R.drawable.pic14_large, R.drawable.pic15_large };

   public Fragment3(Integer selectedRow) {
      super();
      this.selectedRow = selectedRow;
   }

   // this UI is entirely created by code
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {

      ImageView image = new ImageView(getActivity());
      image.setLayoutParams(new RelativeLayout.LayoutParams(
                               LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
      image.setBackgroundResource( largeImages[selectedRow] );
      return image;
   }
}
```

Date/Time, TabHost, ActionBar

Questions ?