7A

Android
Date – Time - Tabs

Victor Matos
Cleveland State University

Notes are based on:

The Busy Coder's Guide to Android Development
by Mark L. Murphy

ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Date/Time Selection Widgets

Date

Android also supports widgets (**DatePicker**, **TimePicker**) and dialogs (**DatePickerDialog**, **TimePickerDialog**) for helping users enter dates and times.

The **DatePicker** and **DatePickerDialog** allow you to set the starting date for the selection, in the form of a ***year, month***, and ***day.***

Value of ***month*** runs from **0** for *January* through **11** for *December*.

Each widget provides a *callback* object (**OnDateChangedListener** or **OnDateSetListener**) where you are informed of a new date selected by the user.

Date/Time Selection Widgets

Time Selection

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

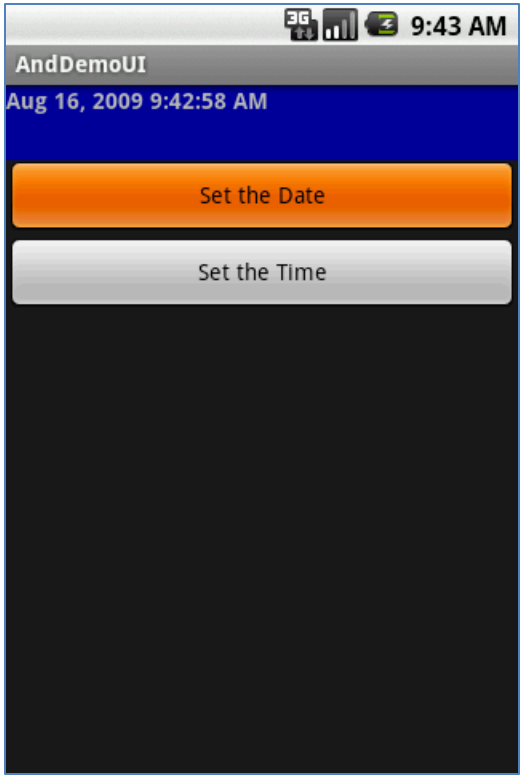
1. set the initial time the user can adjust, in the form of an ***hour*** (***0*** through ***23***) and a ***minute*** (***0*** through ***59***)

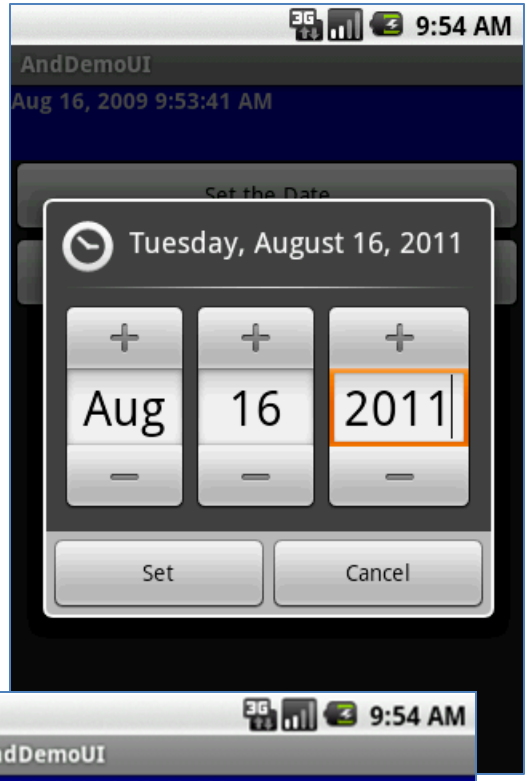
2. indicate if the selection should be in *12-hour mode* (with an AM/PM toggle), or in *24-hour mode*.

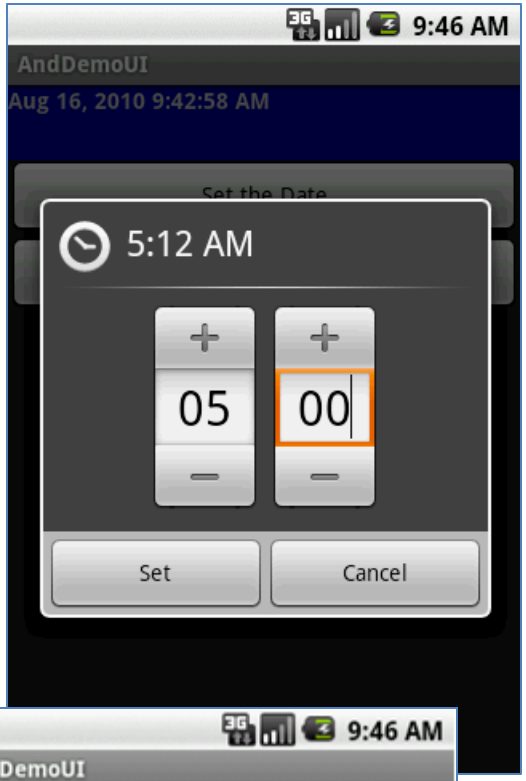
3. provide a callback object (**OnTimeChangedListener** or **OnTimeSetListener**) to be notified of when the user has chosen a new time (which is supplied to you in the form of an hour and minute)

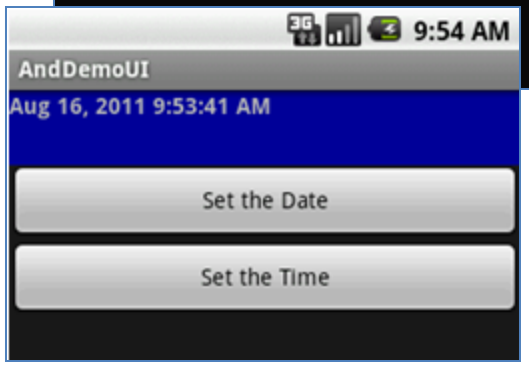
Date/Time Selection Widgets

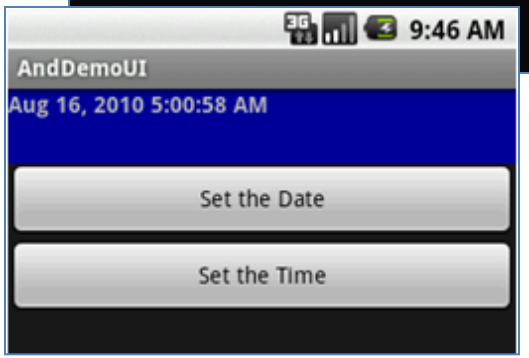
Example: Using Calendar Widgets







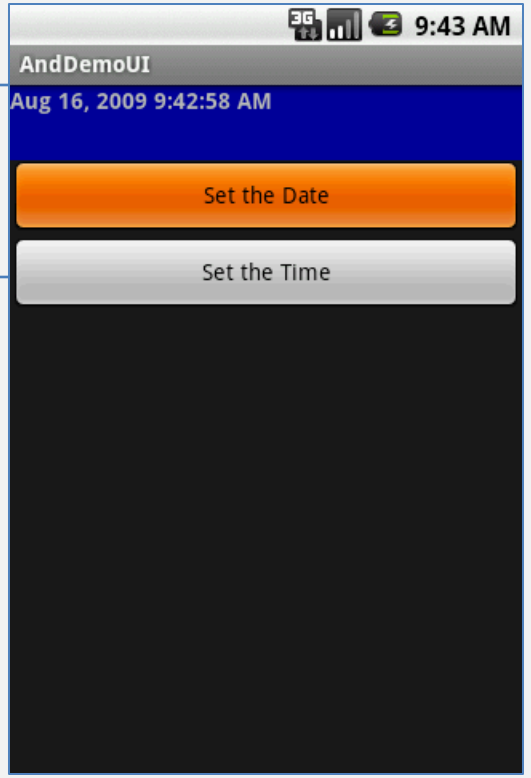




Date/Time Selection Widgets

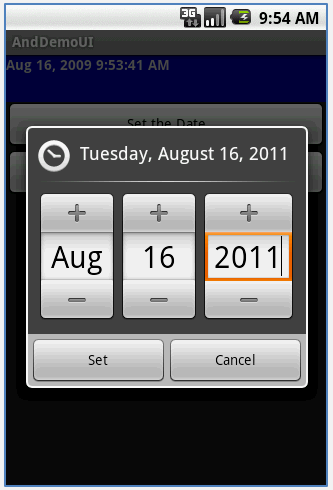
Example: Using Calendar Widgets

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/lblDateAndTime"
android:layout_width="fill_parent"
android:layout_height="47px"
android:background="#ff000099"
android:textStyle="bold"
>
</TextView>
<Button
android:id="@+id/btnDate"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Set the Date"
>
</Button>
<Button
android:id="@+id/btnTime"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Set the Time"
>
</Button>
</LinearLayout>
```



Date/Time Selection Widgets

```
package cis493.demoui;
import android.app.Activity;
import android.os.Bundle;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.TextView;
import java.text.DateFormat;
import java.util.Calendar;

public class AndDemoUI extends Activity {
     DateFormat fmtDateAndTime = DateFormat.getDateTimeInstance();
     TextView lblDateAndTime;
     Calendar myCalendar = Calendar.getInstance();
```

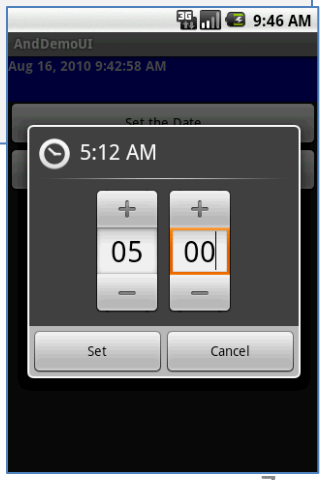



```
     DatePickerDialog.OnDateSetListener d = new DatePickerDialog.OnDateSetListener()
     {
          public void onDateSet(DatePicker view,
                                int year, int monthOfYear, int dayOfMonth) {
          myCalendar.set(Calendar.YEAR, year);
          myCalendar.set(Calendar.MONTH, monthOfYear);
          myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
          updateLabel();
          }
     };
```

Date/Time Selection Widgets

```
TimePickerDialog.OnTimeSetListener t = new TimePickerDialog.OnTimeSetListener()
{
     public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
          myCalendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
          myCalendar.set(Calendar.MINUTE, minute);
          updateLabel();
     }
};
```

```
private void updateLabel() {
     lblDateAndTime.setText(fmtDateAndTime.format(myCalendar.getTime()));
}
```



Date/Time Selection Widgets

```
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        lblDateAndTime = (TextView) findViewById(R.id.lblDateAndTime);
        Button btnDate = (Button) findViewById(R.id.btnDate);
        btnDate.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                new DatePickerDialog(AndDemoUI.this, d,
                        myCalendar.get(Calendar.YEAR),
                        myCalendar.get(Calendar.MONTH),
                        myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });

        Button btnTime = (Button) findViewById(R.id.btnTime);
        btnTime.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                new TimePickerDialog(AndDemoUI.this, t,
                        myCalendar.get(Calendar.HOUR_OF_DAY),
                        myCalendar.get(Calendar.MINUTE), true).show();
            }
        });

        updateLabel();
    }// onCreate
} //class
```

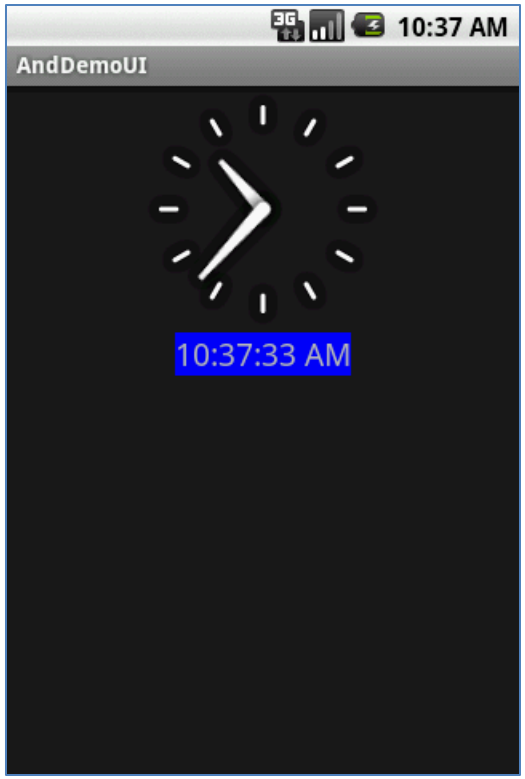
Date/Time Selection Widgets

Other Time Widgets

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

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

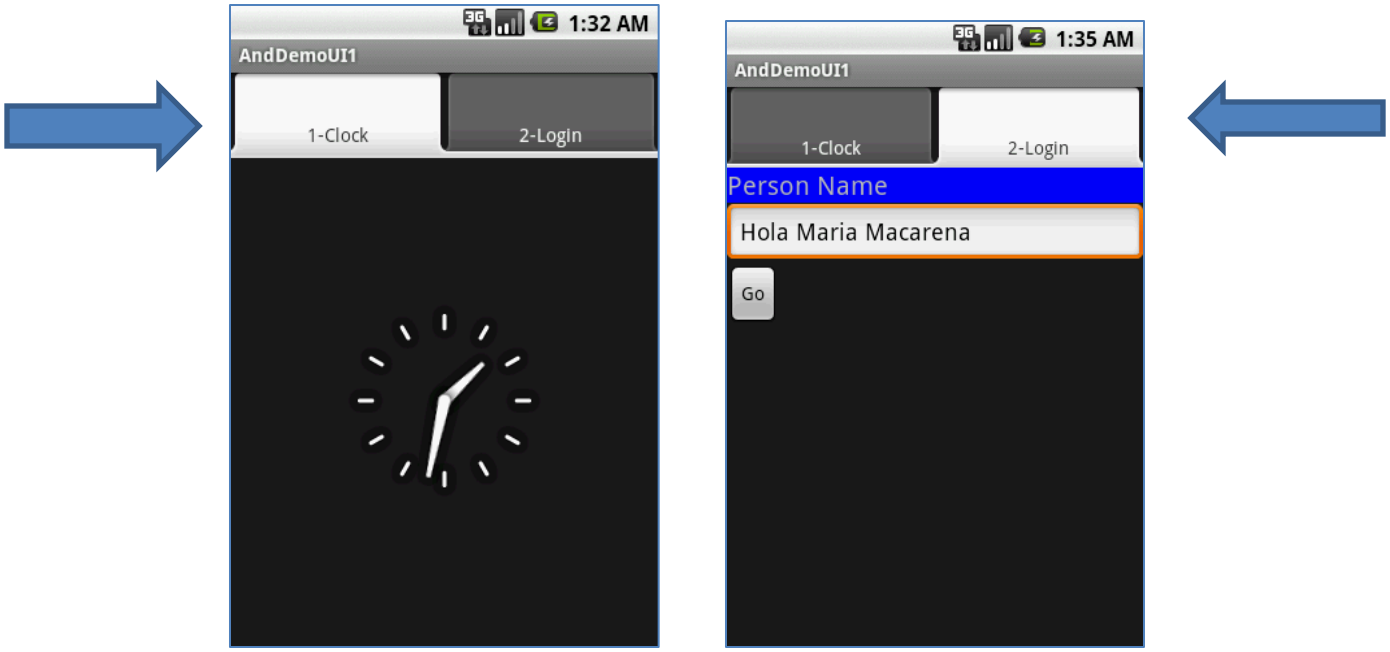
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
android:id="@+id/widget34"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<DigitalClock
android:id="@+id/digital"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:textSize="20px"
android:layout_below="@+id/analog"
android:layout_centerHorizontal="true"
>
</DigitalClock>
<AnalogClock
android:id="@+id/analog"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
>
</AnalogClock>
</RelativeLayout>
```



Tab Selection Widget

Tab Selector

1. Android UIs should be kept simple at all costs.
2. When many pieces of information must be displayed in a single app, the **Tab Widget** could be used to make the user aware of the pieces but show only a portion at the time.



Tab Selection Widget

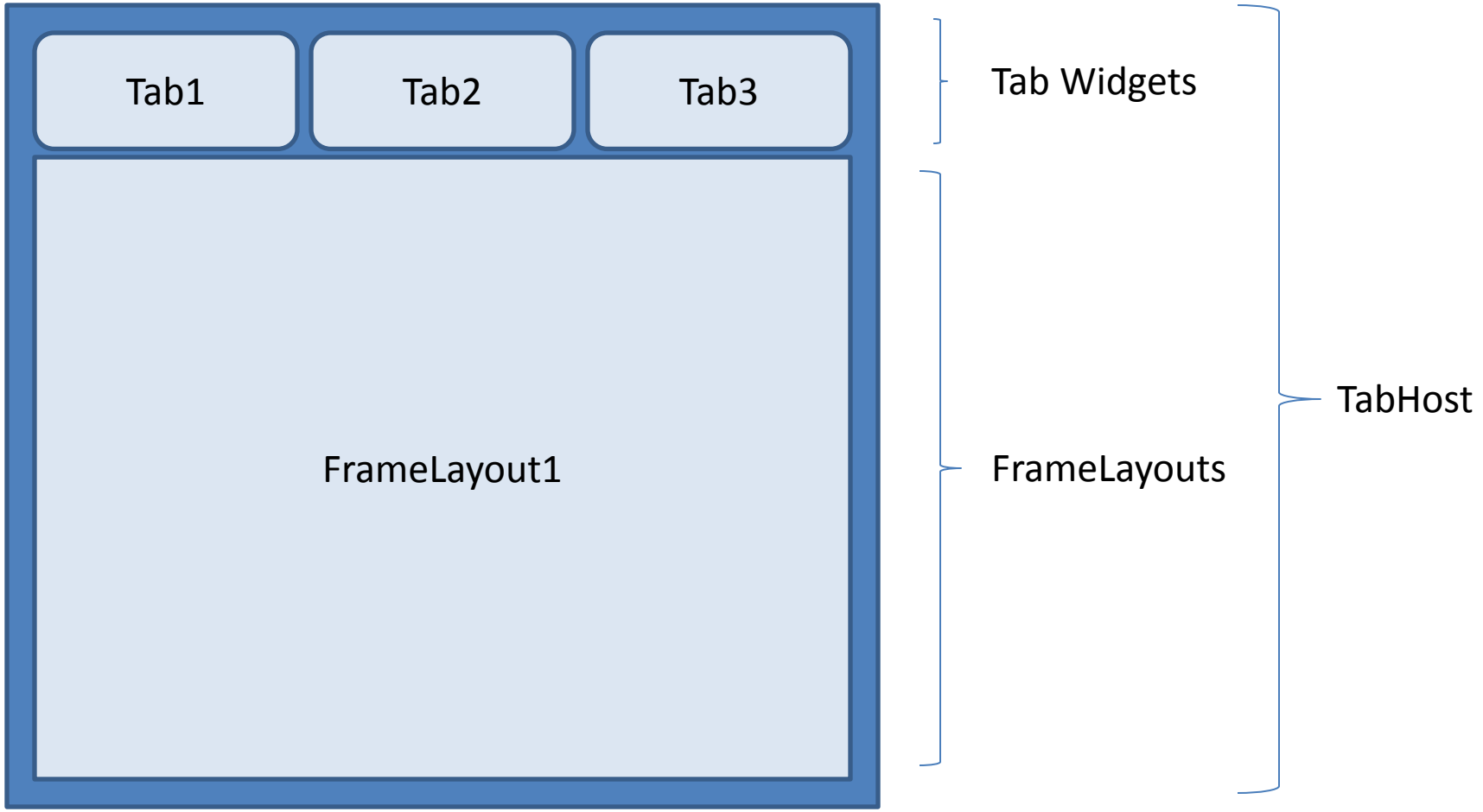
Tabs – Components

There are a few widgets and containers you need to use in order to set up a tabbed portion of a view:

1. **TabHost** is the main container for the tab buttons and tab contents
2. **TabWidget** 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

Components



Tab Selection Widget

CAUTION (Jan 22, 2011)
SDK 2.2 has an apparent bug on this issue. See link http://code.google.com/p/android/issues/detail?id=13092.
Temporal solution is to create app for SDK 1.6.

Example: Using Tabs

```
<?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">

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

        <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="62px">

            <!-- PUT HERE FrameLayout1 -->

            <!-- PUT HERE FrameLayout2  -->

        </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

Example: Using Tabs

This goes in place of *FrameLayout1*. It defines an analog clock

(optionally surround with a <FrameLayout > tag using the clause `android:id="@+id/tab1"` In that case apply a different **id** to the clock)

```
<AnalogClock
        android:id="@+id/tab1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerHorizontal="true"
 />
```

Tab Selection Widget

Example: Using Tabs

This is FrameLayout2. It defines a *LinearLayout* holding a *label*, a textBox, and a *button*.

```
<LinearLayout
android:id="@+id/tab2"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
       android:id="@+id/caption1"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:background="#ff0000ff"
       android:text="Person Name"
       android:textSize="20px"
>
</TextView>
<EditText
       android:id="@+id/txtPerson"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="txtPerson"
       android:textSize="18sp"
>
</EditText>
<Button
       android:id="@+id/btnGo"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Go"
>
</Button>
</LinearLayout>
```

Tab Selection Widget

Example: Using Tabs

In order to keep the **main.xml** design *as simple as possible*
ou may introduce **<include>** clauses as illustrated below

```
<!-- PUT HERE FrameLayout2  -->
<FrameLayout
   android:id="@+id/tab2"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent" >
      <include
         layout="@layout/screen2" />
</FrameLayout>
```

/res/layout/screen2.xml

```
<LinearLayout
android:id="@+id/tab2"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android
"
>
<TextView
            android:id="@+id/caption1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="#ff0000ff"
            android:text="Person Name"
            android:textSize="20px"
>
</TextView>
<EditText
            android:id="@+id/txtPerson"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="txtPerson"
            android:textSize="18sp"
>
</EditText>
<Button
            android:id="@+id/btnGo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Go"
>
</Button>
</LinearLayout>
```

Tab Selection Widget

Example: Using Tabs

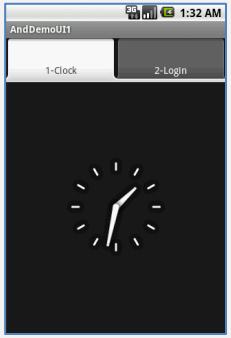
```
package cis493.selectionwidgets;

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.TabHost;


public class AndDemoUI1 extends Activity {
```

Tab Selection Widget

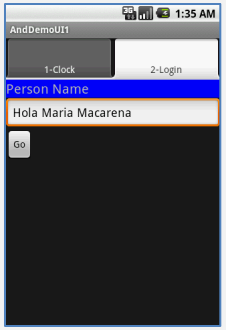
Example: Using Tabs

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

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

        tabs.setup();

        TabHost.TabSpec spec;

        spec =tabs.newTabSpec("tag1");
        spec.setContent(R.id.tab1);
        spec.setIndicator("1-Clock");
        tabs.addTab(spec);

        spec=tabs.newTabSpec("tag2");
        spec.setContent(R.id.tab2);
        spec.setIndicator("2-Login");
        tabs.addTab(spec);

        tabs.setCurrentTab(0);
```

Set Tab1



Set Tab2

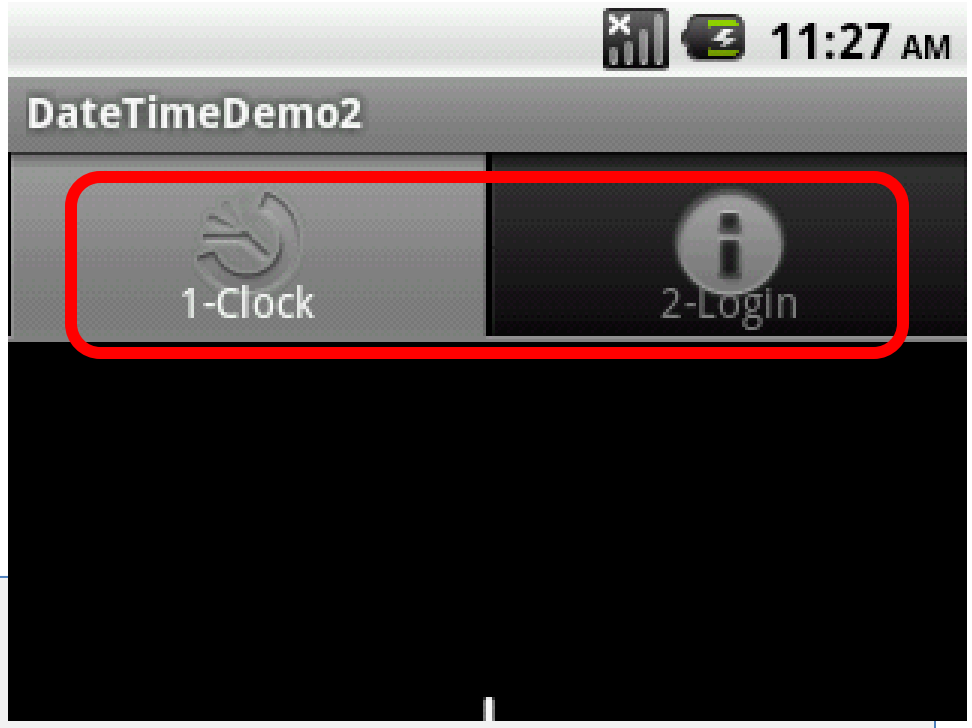


Tab Selection Widget

HINT

Example: Using Tabs

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



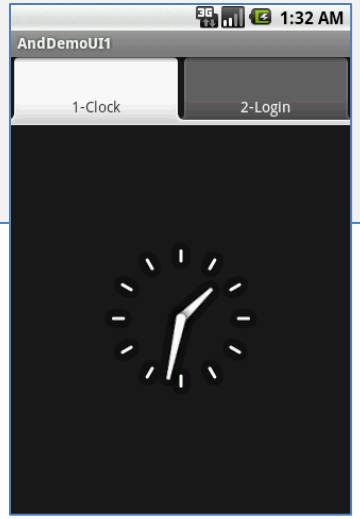

```
spec = tabs.newTabSpec("tag2");
spec.setContent(R.id.tab2);

spec.setIndicator("2-Login",
                  getResources().getDrawable(R.drawable.ic_menu_info_details));

tabs.addTab(spec);
```

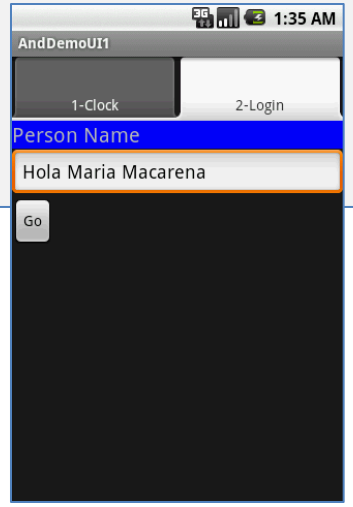
Note: Many icons available at SDKfolder\docs\images\icon-design

Tab Selection Widget

Example: Using Tabs

```
  Button btnGo = (Button)findViewById(R.id.btnGo);
  btnGo.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
                EditText txtPerson =
                        (EditText)findViewById(R.id.txtPerson);
                String theUser = txtPerson.getText().toString();
                txtPerson.setText("Hola " + theUser);
                }
        });
  }

}
```





Tab Selection Widget

Example: 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("tag1");

    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
```

SlidingDrawer Widget

SlidingDrawer

hides content out of the screen and allows the user to drag a **handle** to bring the **content** on screen.

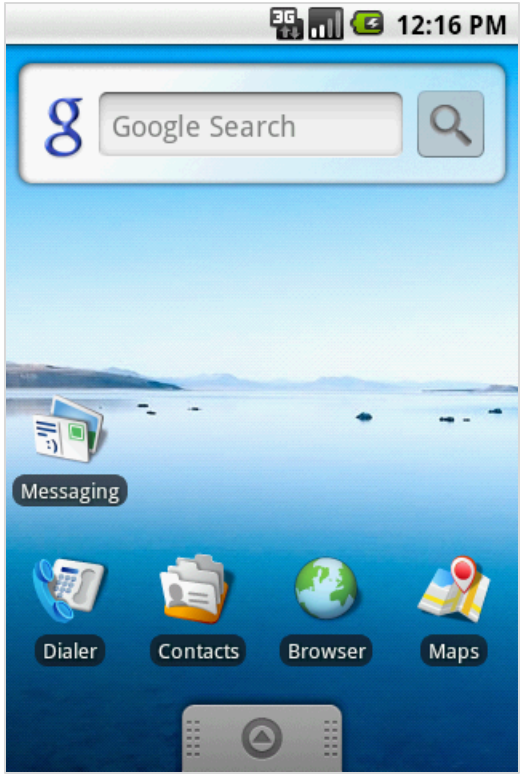
- SlidingDrawer can be used *vertically* or *horizontally*.
- SlidingDrawer should be used as an *overlay* inside layouts. This means SlidingDrawer should only be used inside of a **FrameLayout** or a **RelativeLayout** for instance.
- The size of the SlidingDrawer defines how much space the content will occupy once slid out so SlidingDrawer should usually use *fill_parent* for both its dimensions.

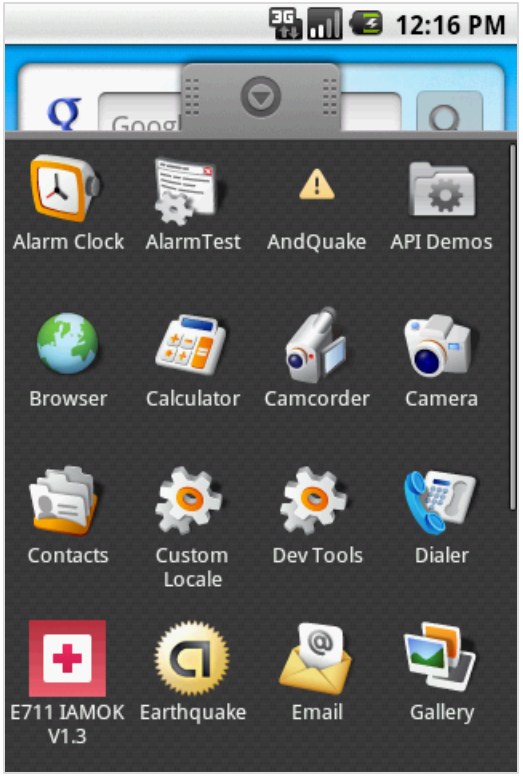
Taken from: http://developer.android.com/reference/android/widget/SlidingDrawer.html

SlidingDrawer Widget

Example:

This *SlidingDrawer* is used by the Android's interface to access applications installed in the device.





content

handle

SlidingDrawer Widget

Taken from: http://developer.android.com/reference/android/widget/SlidingDrawer.html

Example1:

Inside an XML layout, SlidingDrawer must define the **id** of the *handle* and the *content*:

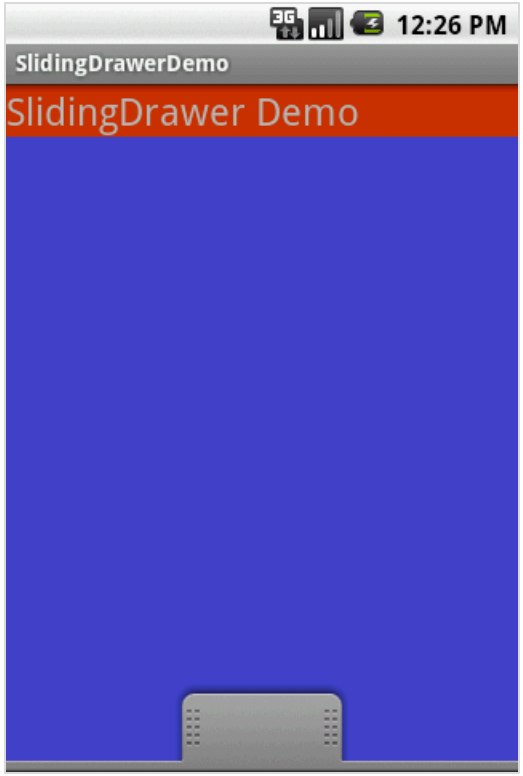
```
<SlidingDrawer
     android:id="@+id/drawer"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"

     android:handle="@+id/handle"
     android:content="@+id/content">

     <ImageView
          android:id="@id/handle"
          android:layout_width="88dip"
          android:layout_height="44dip" />

     <GridView
          android:id="@id/content"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent" />

 </SlidingDrawer>
```

handle is just a small graphic to visually indicate the opening/closing control

content is usually some type of container holding the desired UI held by the drawer

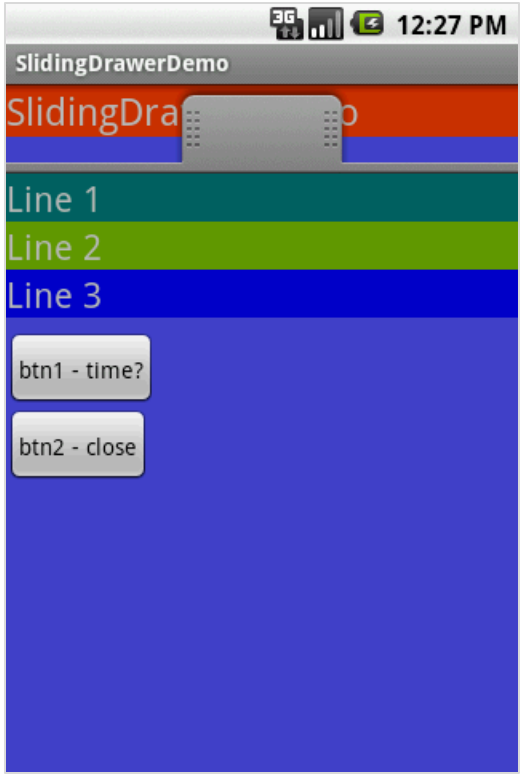
SlidingDrawer Widget

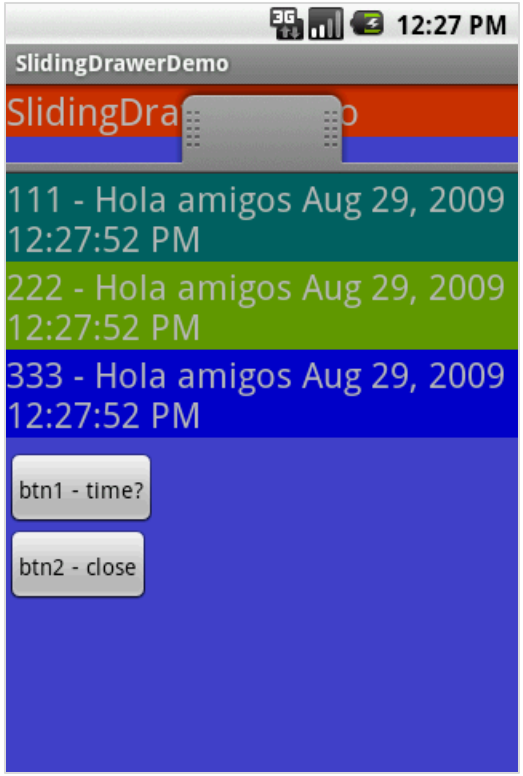
Example2. A more elaborated SlidingDrawer.



The red TextView simulates the main UI, the SlidingDrawer is an overlay, tapping the handle opens the new view

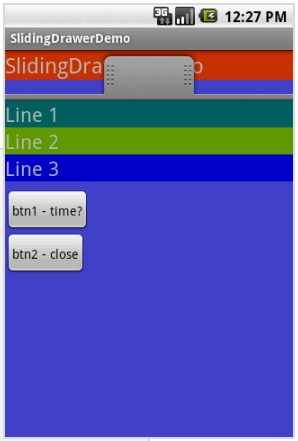
The background UI is overlapped by the contents of the drawer. Tapping the handle closes the drawer (but does not erase its data)





SlidingDrawer Widget

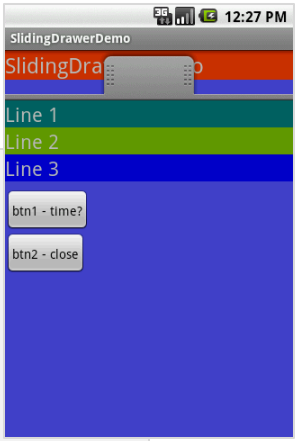
Example 2: SlidingDrawer XML layout (main UI)




```
<?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="#FF4444CC"
        >
  <TextView
    android:id="@+id/label0"
    android:layout_alignParentTop="true"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ffcc3300"
    android:text="SlidingDrawer Demo"
    android:textSize="24sp" />
```

SlidingDrawer Widget

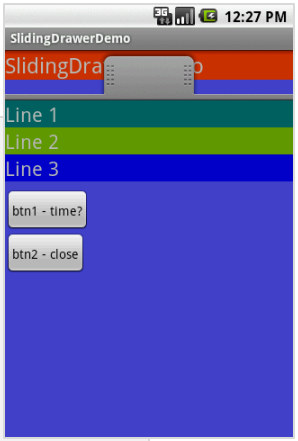
Example 2: SlidingDrawer XML layout (Drawer)




```
<SlidingDrawer
        android:id="@+id/drawer"
        android:layout_alignParentBottom="true"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:handle="@+id/handle"
        android:content="@+id/content" >

        <ImageView
            android:id="@id/handle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/tray_handle_normal"  />
```

SlidingDrawer Widget

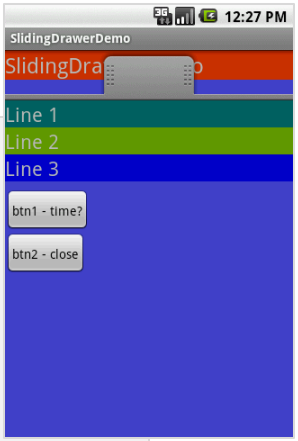
Example 2: SlidingDrawer XML layout (Drawer)




```
<LinearLayout
        android:id="@id/content"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >

        <TextView
        android:id="@+id/label1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff006666"
        android:text="Line 1"
        android:textSize="22sp"  />

        <TextView
        android:id="@+id/label2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff669900"
        android:text="Line 2"
        android:textSize="22sp"  />
```

SlidingDrawer Widget

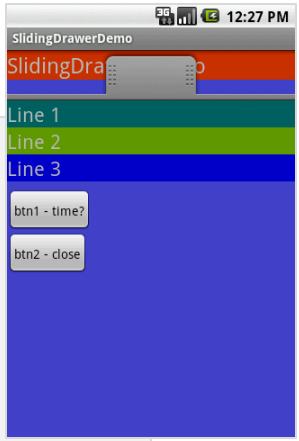
Example 2: SlidingDrawer XML layout (Drawer)




```
<TextView
android:id="@+id/label3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000cc"
android:text="Line 3"
android:textSize="22sp"   />

<TextView
android:id="@+id/filler1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="6sp"   />

<Button
android:id="@+id/btn1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="4px"
android:text=" btn1 - time? "   />
```

SlidingDrawer Widget

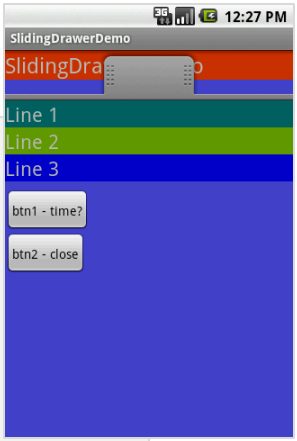
Example 2: SlidingDrawer XML layout (Drawer)




```
                <Button
                android:id="@+id/btn2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="4px"
                android:text=" btn2 - close "   />

        </LinearLayout>

    </SlidingDrawer>

</RelativeLayout>
```

SlidingDrawer Widget

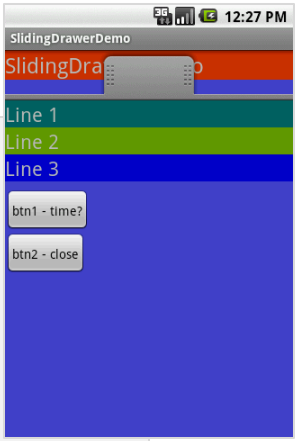
Example 2: SlidingDrawer. Android Activity




```
package cis493.slidingdreawerdemo;

import java.util.Date;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class SlidingDrawerDemo extends Activity
{
    Button btn1;
    Button btn2;
    TextView label1;
    TextView label2;
    TextView label3;
    SlidingDrawer myDrawer;
```

SlidingDrawer Widget

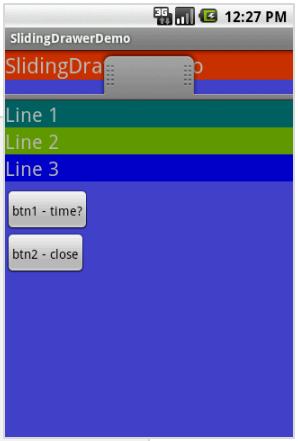
Example 2: SlidingDrawer. Android Activity




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

    myDrawer = (SlidingDrawer)findViewById(R.id.drawer);

    btn1 = (Button)findViewById(R.id.btn1);
    btn2 = (Button)findViewById(R.id.btn2);

    label1 = (TextView)findViewById(R.id.label1);
    label2 = (TextView)findViewById(R.id.label2);
    label3 = (TextView)findViewById(R.id.label3);
```

SlidingDrawer Widget

Example 2: SlidingDrawer. Android Activity




```
    btn1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
         Date dt = new Date();
         String now = dt.toLocaleString();
         label1.setText("111 - Hola amigos " + now);
         label2.setText("222 - Hola amigos " + now) ;
         label3.setText("333 - Hola amigos " + now);
    }
    });

    btn2.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
              myDrawer.animateClose();
         }
    });

   } //onCreate

} // class
```

UI Selection Widgets

Questions ?