



## Chương 3 :

# XỬ LÝ GIAO DIỆN NGƯỜI DÙNG



**TS. Huỳnh Hữu Nghĩa**

luckerrhuynhvn@gmail.com

# Nội dung:

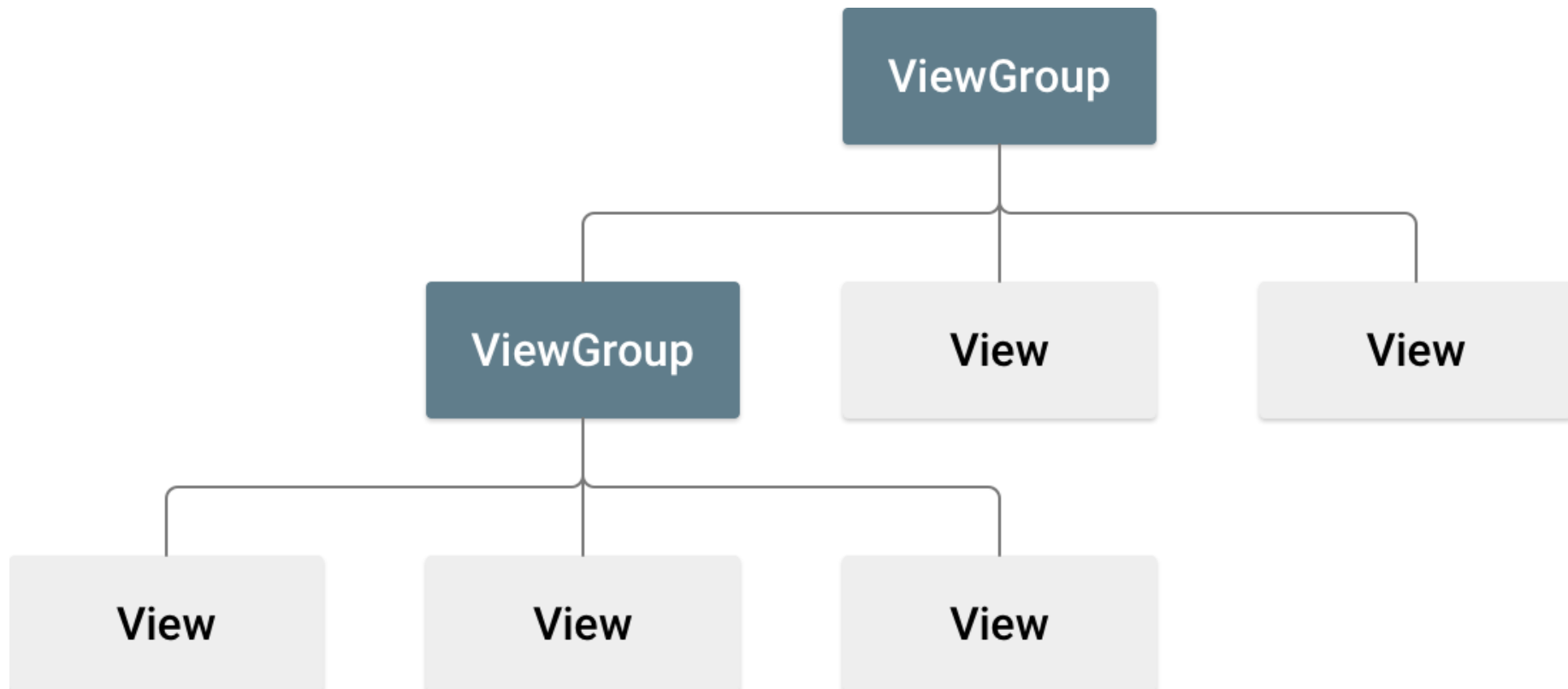
---

- Hiểu về các thành phần của giao diện
- Lập trình sự kiện trong Android
- Một số control thông dụng và nâng cao
- Webkit
- Cài đặt đa ngôn ngữ trên ứng dụng Android

# Các thành phần của giao diện

- ❖ Giao diện (UI) của ứng dụng là những cái gì mà người dùng có thể nhìn thấy và tương tác.
- ❖ Android cung cấp nhiều thành phần UI được xây dựng sẵn như: các đối tượng bố cục (layout) có cấu trúc và các điều khiển UI cho phép xây dựng UI đồ họa cho ứng dụng.
- ❖ Android cũng cung cấp nhiều module UI khác cho các giao diện đặc biệt như dialogs, notifications, và menus.

- Layout xác định cấu trúc hiển thị cho UI (trong Activity). Tất cả các thành phần trong layout được xây dựng kế thừa các đối tượng View và ViewGroup.
- View thường đưa ra một số cái người dùng có thể thấy và tương tác. Trong khi đó, ViewGroup là một nơi chứa không nhìn thấy xác định cấu trúc layout cho View và các đối tượng ViewGroup khác.



➤ Có 2 cách khai báo Layout:

- Khai báo các phần tử UI trong XML.
- Khởi tạo các phần tử layout vào thời điểm chạy.

## ➤ XML file.

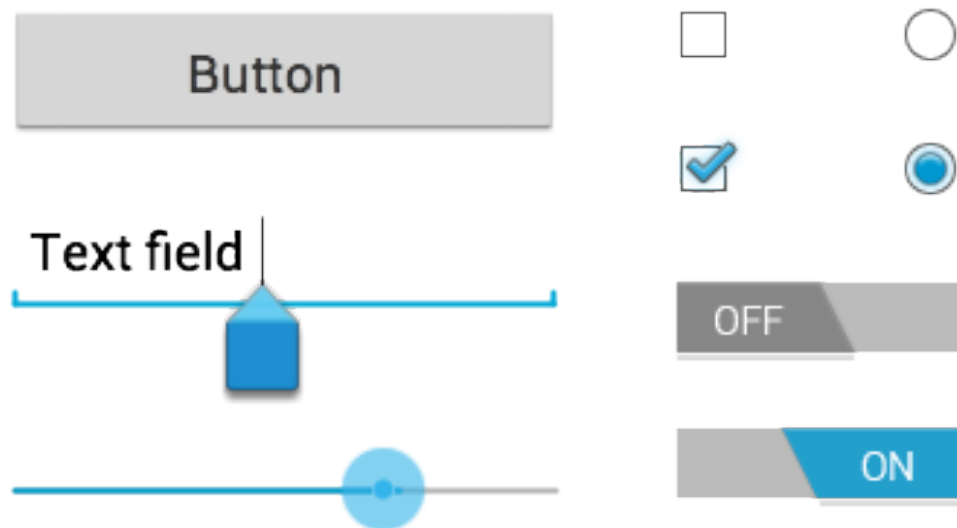
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

# Các kiểu Layout

- **LinearLayout** là một ViewGroup, sắp xếp các đối tượng bên trong theo một hướng duy nhất, chiều dọc hoặc ngang.
- **RelativeLayout** là một ViewGroup, canh các đối tượng theo các vị trí liên quan.
- **TableLayout** là một View, nhóm các View thành dòng và cột.
- **AbsoluteLayout** cho phép chỉ định chính xác của đối tượng.
- **ListView** là một ViewGroup hiển thị danh sách các mục..
- **GridView** là một ViewGroup hiển thị các mục trong một lưới 2 chiều.



- Controls đầu vào là các thành phần tương tác trong giao diện người dùng. Android cung cấp nhiều loại control khác nhau như: ***button***, ***text fields***, ***seek bars***, ***check box***, ***zoom buttons***, ***toggle button***, và nhiều hơn.



# Event Handling

- Sự kiện (events) là một cách hữu hiệu để thu thập dữ liệu về việc tương tác của người dùng với các thành phần tương tác của ứng dụng. Như bấm nút hoặc chạm vào màn hình.
- Có 3 khái niệm liên quan đến quản lý sự kiện Android:
  - Event Listeners.
  - Event Listeners Registration.
  - Event Handlers

- **Event Listeners:** là một giao diện trong lớp View, chứa một phương thức gọi lại duy nhất. Các phương thức sẽ được gọi bởi khung Android khi View listener đã đăng ký được kích hoạt bởi người dùng tương tác trên UI.
- **Event Listeners Registration:** Đăng ký Event là quá trình xử lý bởi trình xử lý sự kiện đăng ký với Event Listener để trình xử lý được gọi khi Event Listener kích hoạt sự kiện.

- **Event Handlers:** khi một sự kiện xảy ra và chúng được đăng ký việc lắng nghe sự kiện cho một sự kiện, việc lắng nghe sự kiện gọi là Event Handlers, nó là phương thức xử lý thực sự sự kiện.



# Event Handling Examples

```
hesoa = (EditText) findViewById(R.id.editText_hesoA);  
hesob = (EditText) findViewById(R.id.editText_hesoB);  
bt_tong = (Button) findViewById(R.id.button_Tong);  
bt_thuong = (Button) findViewById(R.id.button_Thuong);  
ketqua = (TextView) findViewById(R.id.textView_ketqua);
```

```
a = Integer.parseInt(hesoa.getText().toString());  
b = Integer.parseInt(hesob.getText().toString());
```

```
bt_tong.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        ketqua.setText("Kết quả: " + (a + b));  
    }  
});  
bt_thuong.setOnClickListener((view) → {  
    if(b != 0)  
        ketqua.setText("Kết quả: " + a/b*1.0f);  
    else  
        ketqua.setText("Mẫu số bằng 0, nên không chia được.");  
});
```

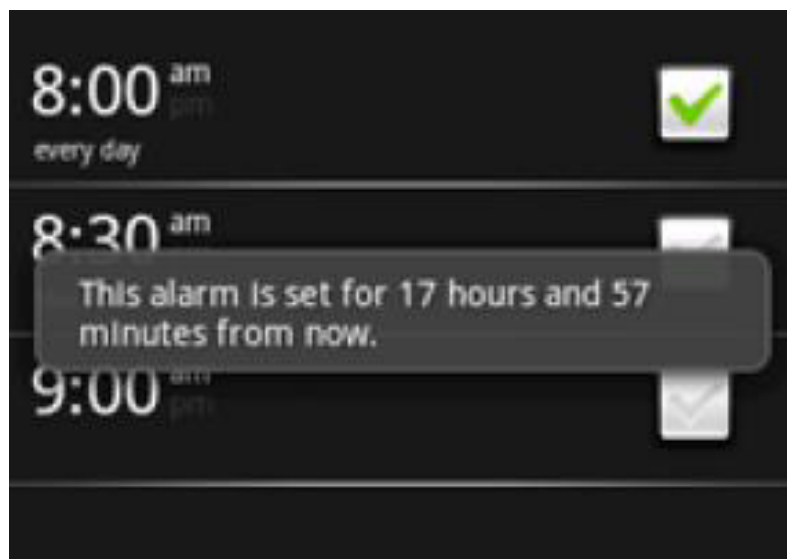


# Toast and Alert Dialog

- Toast notification
- Alert Dialog

# Toast notification

- A message that pops up on the surface of window.
  - It only fills the amount of space required for the message.
  - The notification automatically fades in and out, and does not accept interaction events.
  - can be created and displayed from an Activity or Service.





# Toast notification

```
Toast toast=Toast.makeText(StylesActivity.this, "text",  
    Toast.LENGTH_SHORT);  
toast.setGravity(Gravity.CENTER, 0, 0);  
toast.show();
```

## ► Short form

```
Toast.makeText(context, text, duration).show();
```

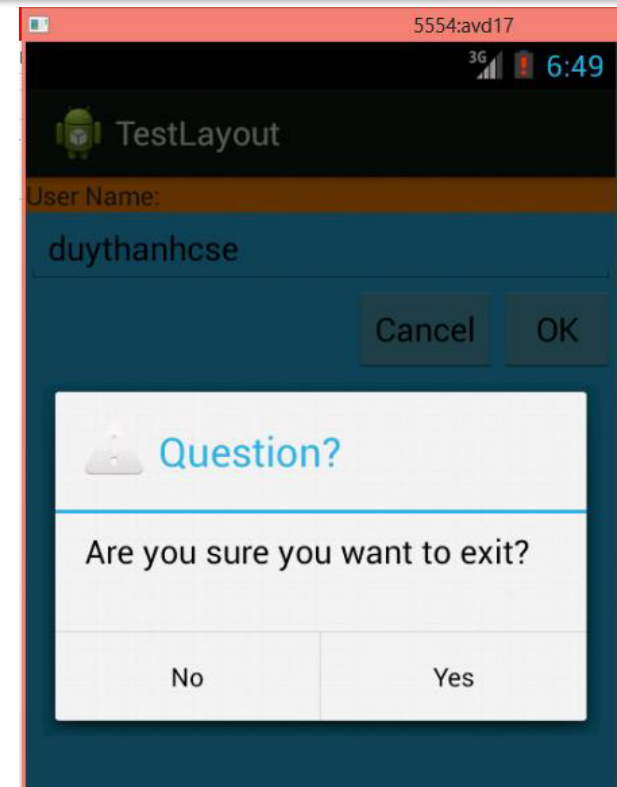
## ► Use Application Context or Activity context

► 2 values for duration: **Toast.LENGTH\_SHORT** to display for a short duration (2 seconds) or **Toast.LENGTH\_LONG** for longer duration (3.5 seconds)



# Alert Dialog

- show critical messages to the user
- information about our application
- Confirm
  - Yes/No message dialog
  - Yes/No Long Message Dialog
- Pick One from a List Dialog
- Pick a number of items from a larger set
- Progress Dialog
- Single choice from a set of choices dialog
- A prompt dialog
- Custom dialog



# Alert Dialog

- create an instance of **AlertDialog.Builder**.
  - ✓ activity context
- setTitle → Sets the title of the pop-up. Just a String
- setMessage → We can add a message. A String
- setIcon: passing a Drawable object
  - ✓ R.drawable.icon
- setCancelable (true/false)

# Alert Dialog

- `setNegativeButton` → add a simple button (cancel button)
- `setPositiveButton` → add a simple button. (OK button)
- `setNeutralButton` → button to perform another functionality other than ok or cancel
  - no restrictions on the use of the three buttons, cause the Alert dialog to dismiss
  - they can perform the same functionality the difference is just in logical meaning.
- `setOnCancelListener`

# Alert Dialog

- Add the following statements to the **MainActivity.java** file:

```
public class MainActivity extends AppCompatActivity {
    TextView tv_ketqua;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        AlertDialog.Builder mydialog = new AlertDialog.Builder(context: this);
        mydialog.setTitle("Message");
        tv_ketqua = (TextView) findViewById(R.id.textView_ketqua);
        mydialog.setPositiveButton(text: "Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                tv_ketqua.setText("You choice yes.");
            }
        });
        AlertDialog alertDialog = mydialog.create();
        alertDialog.show();
    }
}
```

# Alert Dialog

- Add the following statements to the **MainActivity.java** file:

```
public class MainActivity extends AppCompatActivity {
    TextView tv_ketqua;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        AlertDialog.Builder mydialog = new AlertDialog.Builder(context: this);
        mydialog.setTitle("Message");
        tv_ketqua = (TextView)findViewById(R.id.textView_ketqua);
        final CharSequence[] items = {"Đỏ", "Vàng", "Cam"};
        mydialog.setItems(items, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                tv_ketqua.setText(items[i].toString());
            }
        });
        AlertDialog alertDialog = mydialog.create();
        alertDialog.show();
    }
}
```

# Alert Dialog

## ➤ Radiobutton:

```
mydialog.setSingleChoiceItems(items, checkedItem: -1,
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            tv_ketqua.setText(items[i].toString());
        }
    });
```

## ➤ Checkbox:

```
final CharSequence[] items = {"Đỏ", "Vàng", "Cam"};
boolean[] arraycheck = {true, false, true};
mydialog.setMultiChoiceItems(items, arraycheck,
    new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i, boolean b) {
            // xử lý
        }
    });
```

# Alert Dialog

➤ Add the following statements to the **MainActivity.java** file:

```
final CharSequence[] items = {"Đỏ", "Vàng", "Cam"};
final boolean[] arraycheck = {false, false, false};
mydialog.setMultiChoiceItems(items, arraycheck,
    new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i, boolean b) {
            arraycheck[i] = b;
        }
    });
mydialog.setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        String st = "";
        for(int j = 0; j < items.length; j++)
            if(arraycheck[j])
                st += items[j].toString() + "\n";
        tv_ketqua.setText(st);
    }
});
```

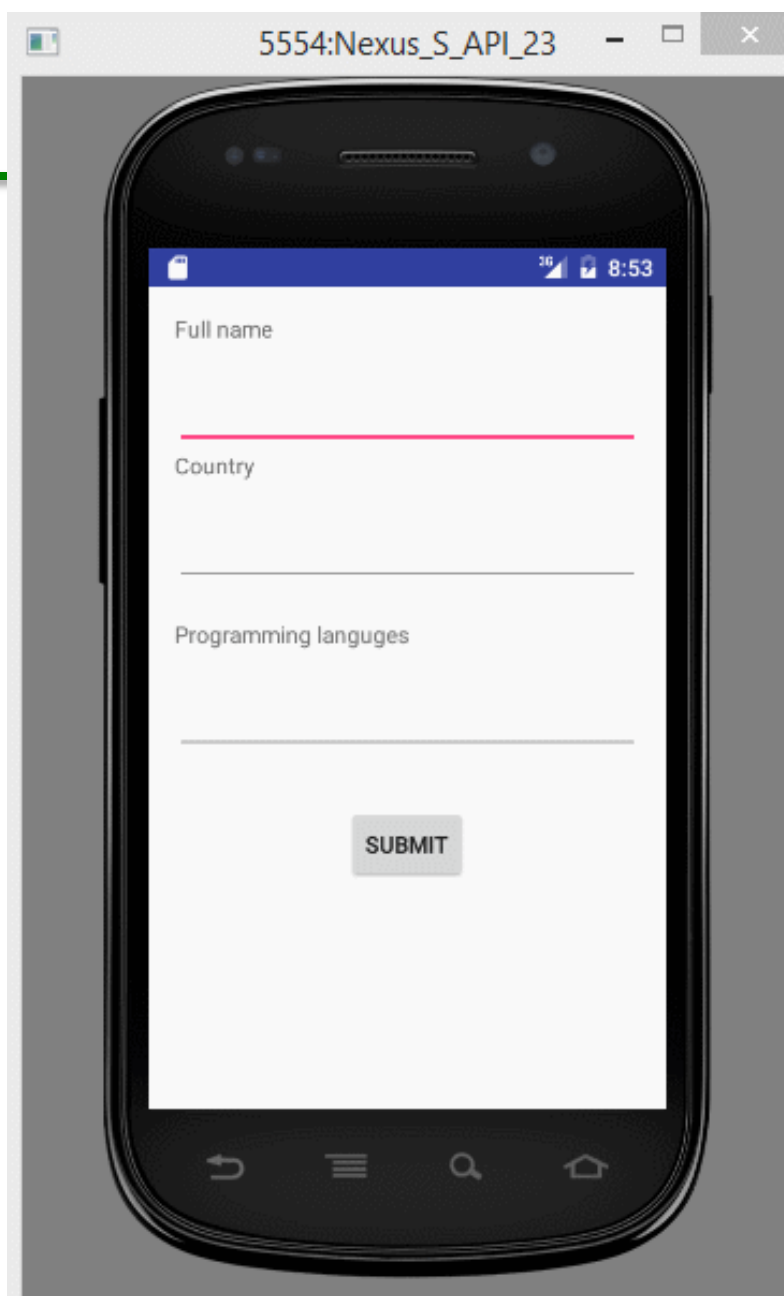
# Common controls

- View
- TextView, EditView
- Button, Checkbox, RadioButton
- ImageView
- ScrollView control

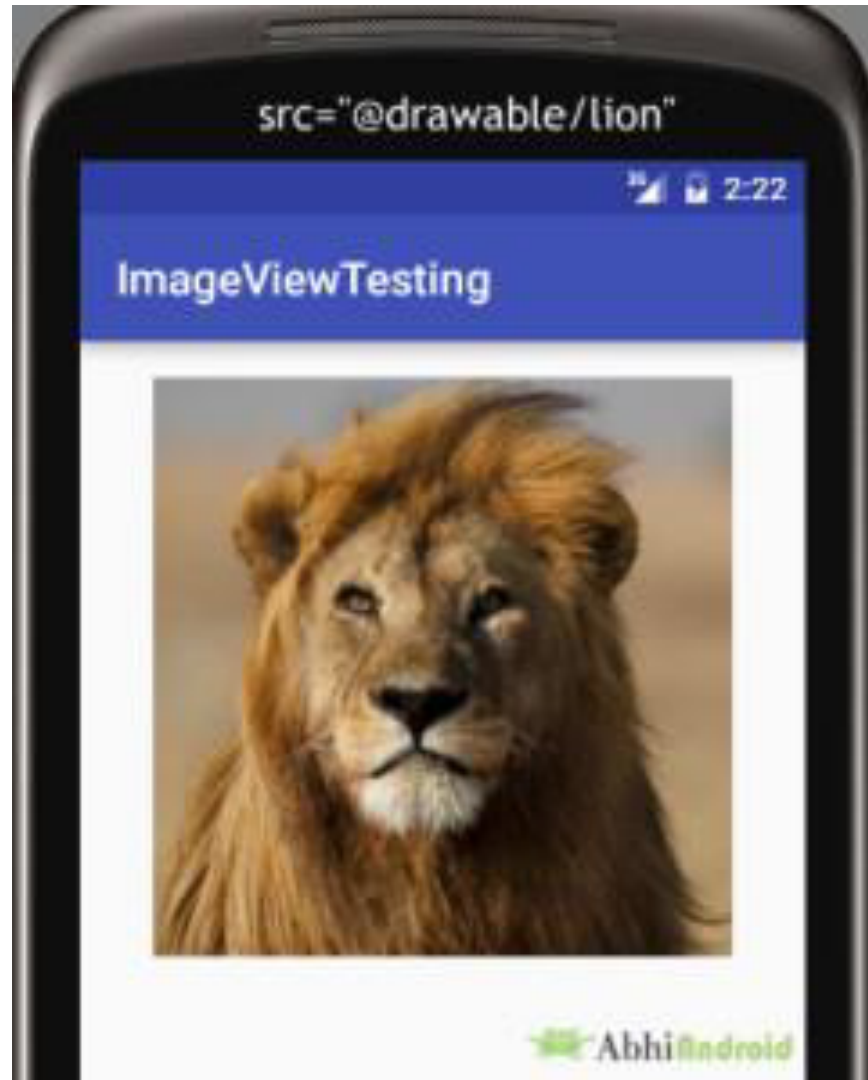


# Basic Controls

## ➤ AutoCompleteTextView

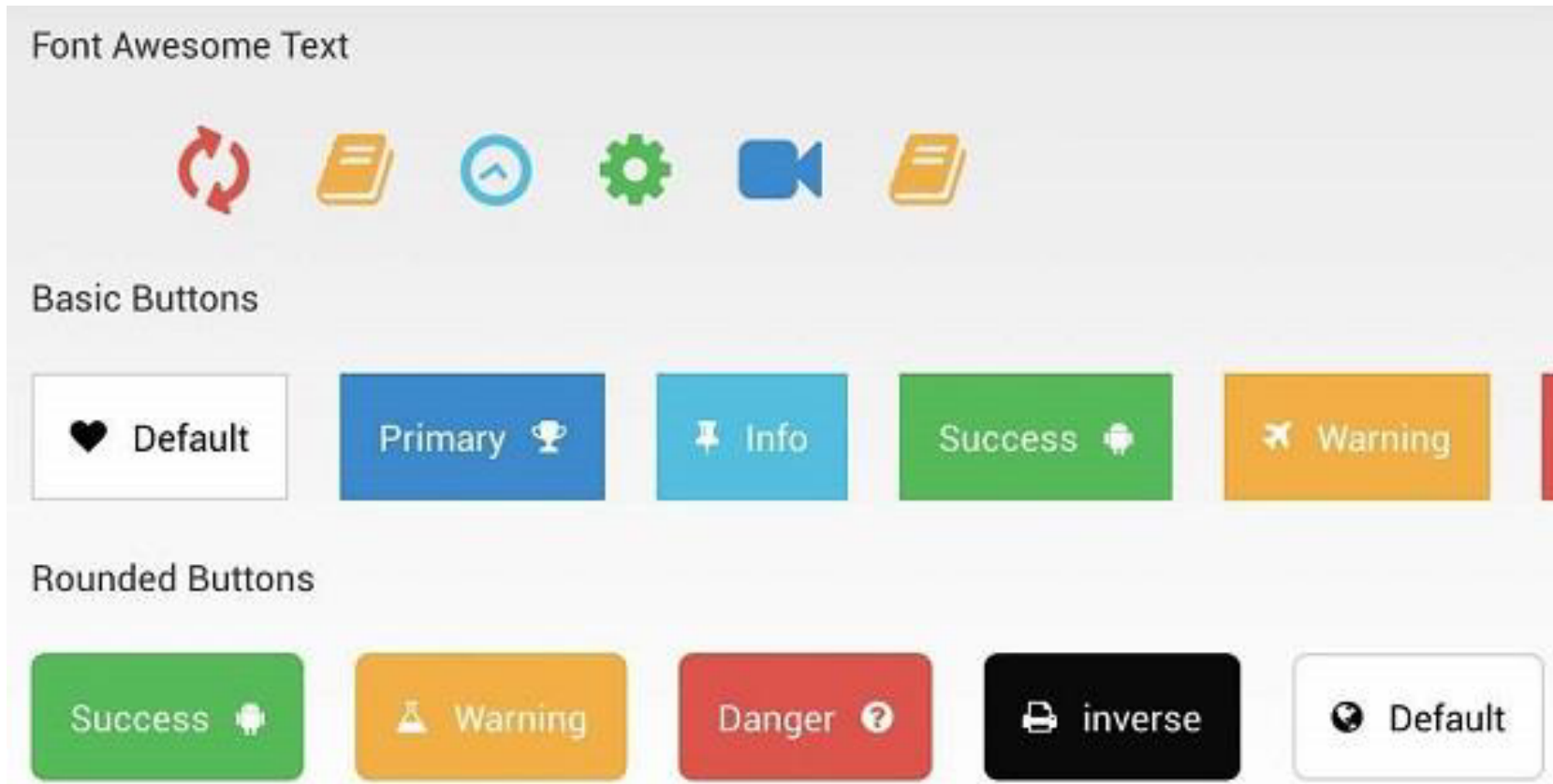


## ➤ ImageView



# Basic Controls

## ➤ ImageButton

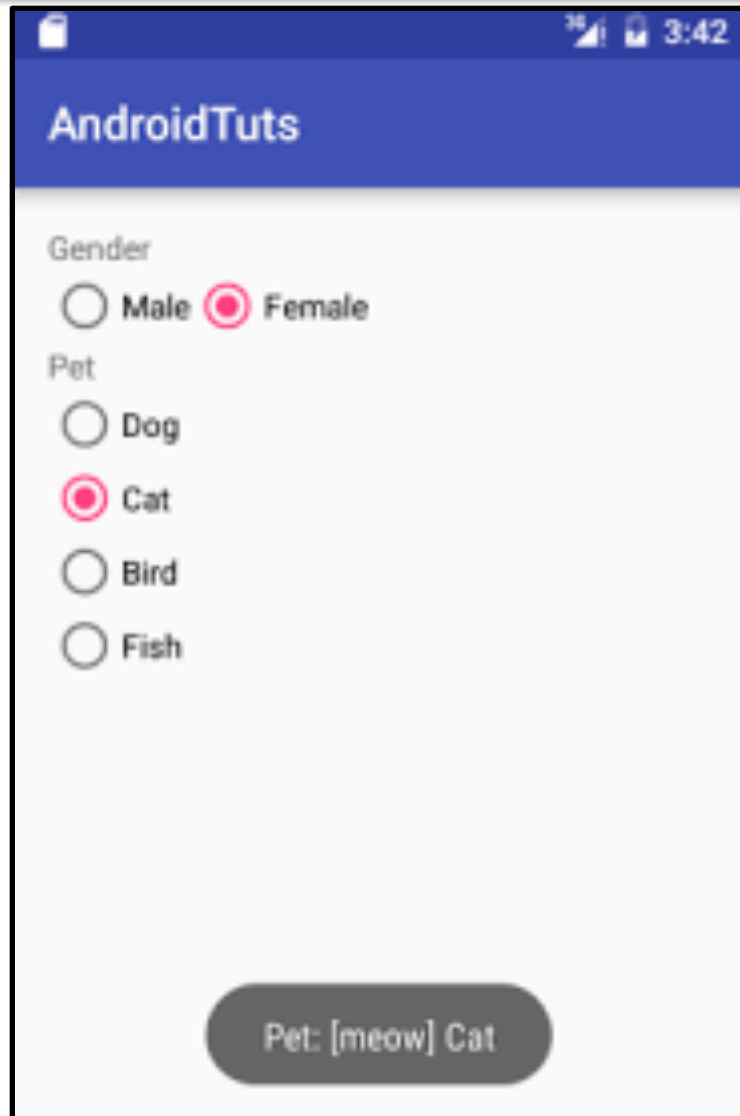


## ➤ ToggleButton



# Basic Controls

## ➤ RadioButton



The screenshot shows an Android application interface with a blue header bar labeled "AndroidTuts". Below the header, there are two sections: "Gender" and "Pet".

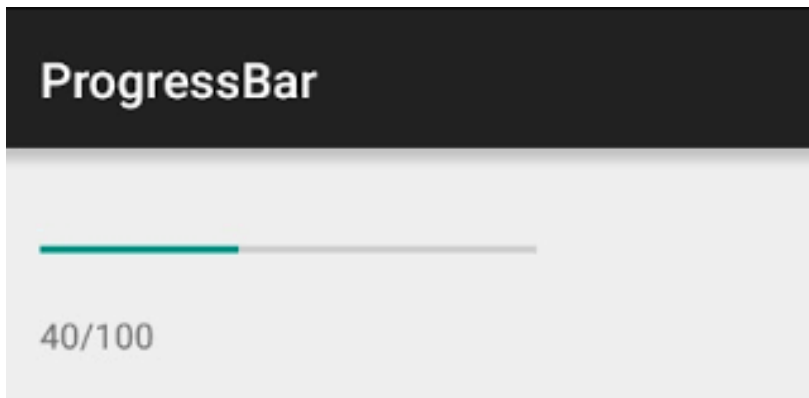
**Gender:** Two radio buttons are displayed. The "Female" option is selected, indicated by a pink dot in the center of the radio button.

**Pet:** Four radio buttons are displayed. The "Cat" option is selected, indicated by a pink dot in the center of the radio button.

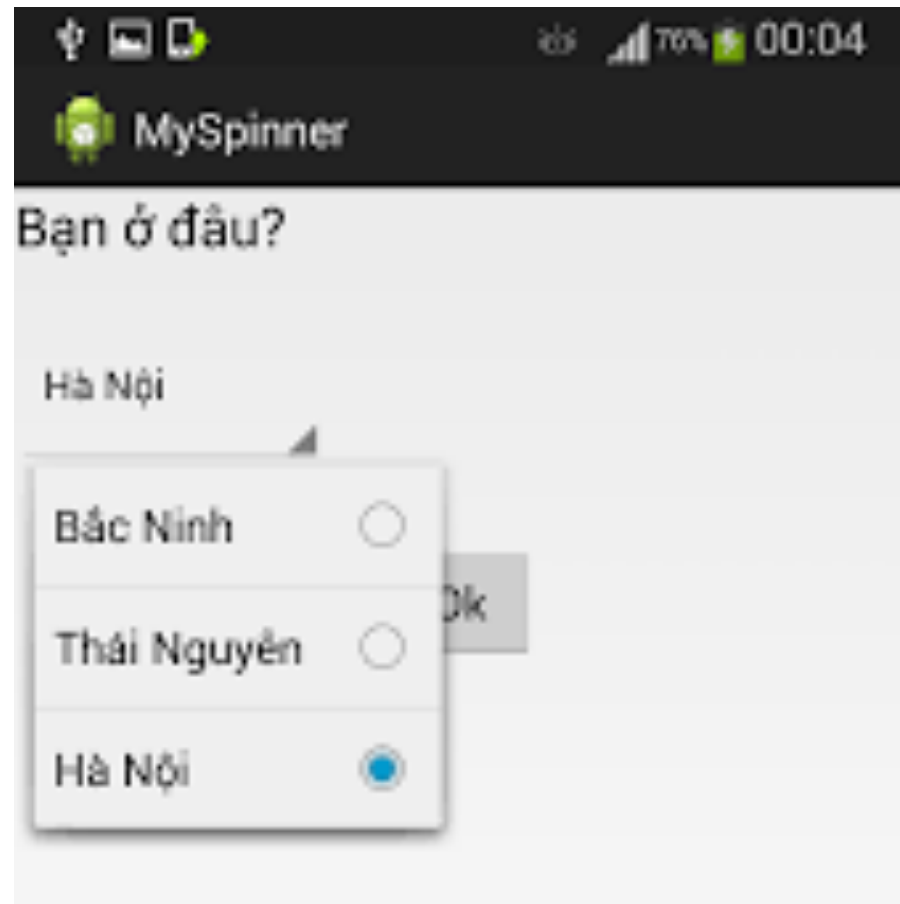
At the bottom of the screen, there is a grey button with the text "Pet: [meow] Cat".

# Basic Controls

## ➤ ProgressBar

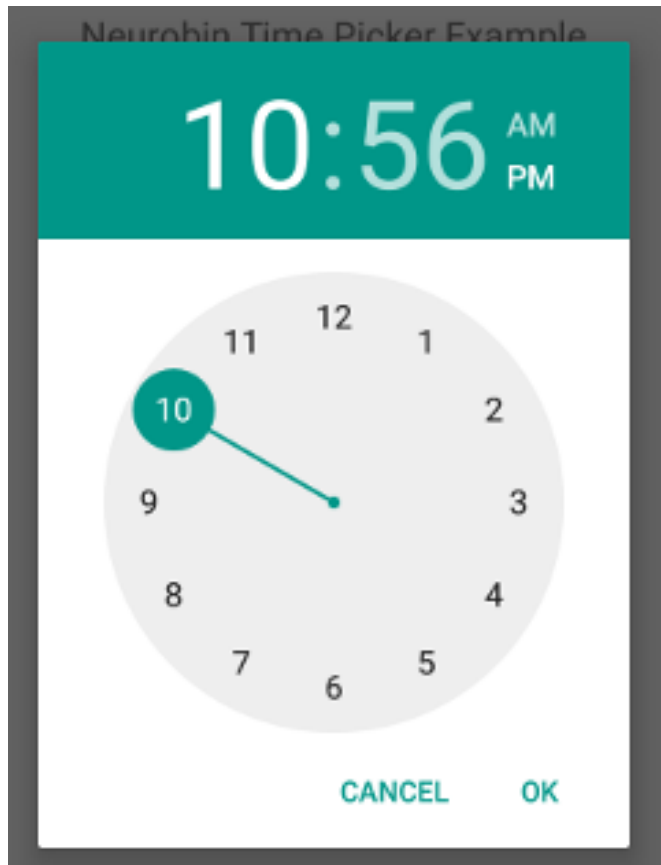


## Spinner



# Basic Controls

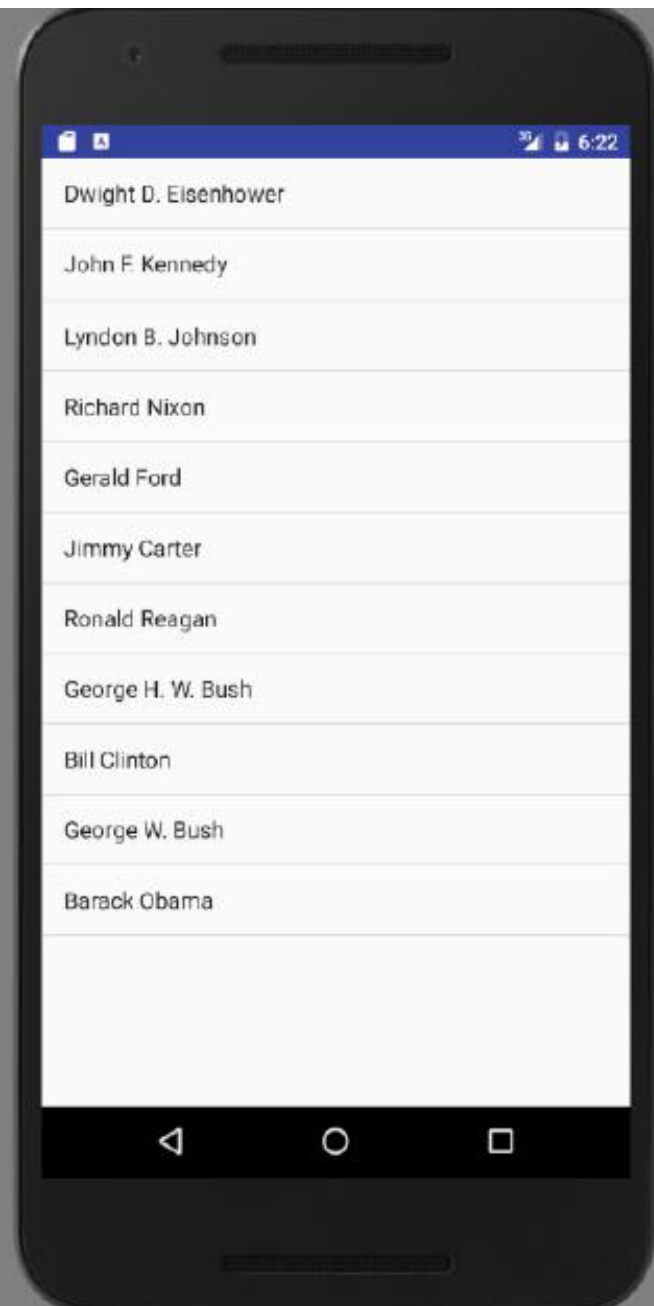
## ➤ TimePicker, DatePicker



- ListView, Spinner
- SlidingDrawer
- AutocompleteTextView
- GridView
- Time Selection, Picture Gallery
- Tab selector, Menu



- ListView are views that enable you to display a long list of items. In Android, there are two types of list views: ListView and SpinnerView.
- The ListView display a list of items in a vertically scrolling list.



```
String[] presidents = {  
    "Dwight D. Eisenhower",  
    "John F. Kennedy",  
    "Lyndon B. Johnson",  
    "Richard Nixon",  
    "Gerald Ford",  
    "Jimmy Carter",  
    "Ronald Reagan",  
    "George H. W. Bush",  
    "Bill Clinton",  
    "George W. Bush",  
    "Barack Obama"  
};
```

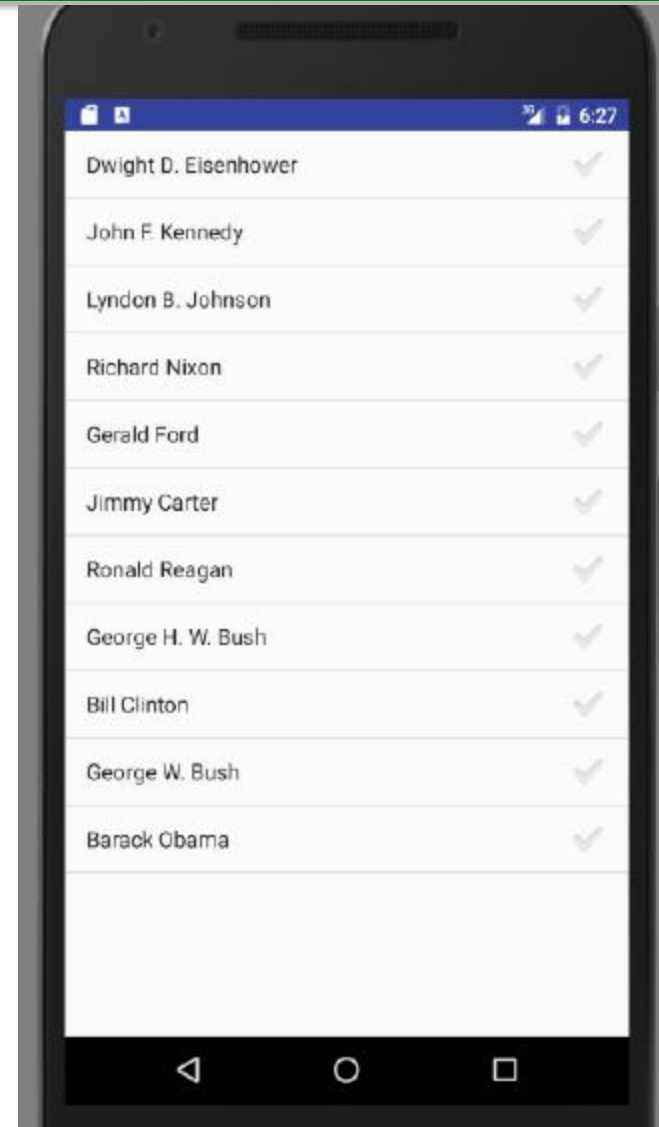
```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, presidents));
}

public void onListItemClick(
    ListView parent, View v, int position, long id)
{
    Toast.makeText(this,
        "You have selected " + presidents[position],
        Toast.LENGTH_SHORT).show();
}
```

# Customizing the ListView

- The ListView is a versatile view that you can further customize.



# Customizing the ListView

- Add the following to the **strings.xml**

```
<string-array name="presidents_array">
    <item>Dwight D. Eisenhower</item>
    <item>John F. Kennedy</item>
    <item>Lyndon B. Johnson</item>
    <item>Richard Nixon</item>
    <item>Gerald Ford</item>
    <item>Jimmy Carter</item>
    <item>Ronald Reagan</item>
    <item>George H. W. Bush</item>
    <item>Bill Clinton</item>
    <item>George W. Bush</item>
    <item>Barack Obama</item>
</string-array>
```

# Customizing the ListView

## ➤ Modify the MainActivity.java

```
String[] presidents;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    listView.setTextFilterEnabled(true);
    presidents =
        getResources().getStringArray(R.array.presidents_array);
    setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_checked, presidents));
}

public void onItemClick(
    ListView parent, View v, int position, long id)
{
    Toast.makeText(this,
        "You have selected " + presidents[position],
        Toast.LENGTH_SHORT).show();
}
```

# Checking which items are selected

- Add the following to the **activity\_main.xml** file:

```
<?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" >

    <Button
        android:id="@+id/btn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Show selected items"
        android:onClick="onClick"/>

    <ListView
        android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

# Checking which items are selected

- Add the following bolded lines to the **MainActivity.java** file:

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

    setContentView(R.layout.activity_main);
    ListView listView = getListView();
    listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    listView.setTextFilterEnabled(true);
    presidents =
        getResources().getStringArray(R.array.presidents_array);
    setListAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_checked, presidents));
}
```



# Checking which items are selected

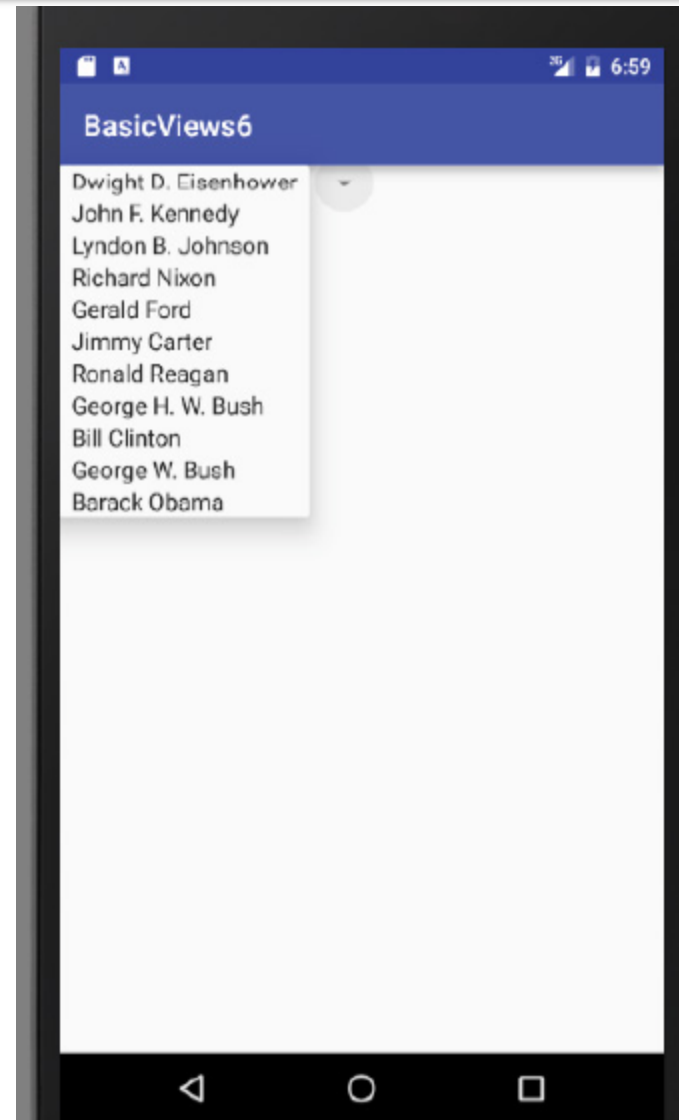
- Add the following bolded lines to the **MainActivity.java** file:

```
public void onItemClick(  
    ListView parent, View v, int position, long id)  
{  
    Toast.makeText(this,  
        "You have selected " + presidents[position],  
        Toast.LENGTH_SHORT).show();  
}  
  
public void onClick(View view) {  
    ListView listView = getListView();  
  
    String itemsSelected = "Selected items: \n";  
    for (int i=0; i<listView.getCount(); i++) {  
        if (listView.isItemChecked(i)) {  
            itemsSelected += listView.getItemAtPosition(i) + "\n";  
        }  
    }  
    Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();  
}
```



# Using the Spinner View

- The ListView displays a long list of items in an activity, but you might want the user interface to display other views, meaning you do not have the additional space for a full-screen view, such as the ListView.
- In such cases, you should use the SpinnerView.





# Using the Spinner View

- Create an Android project and modify the `activity_main.xml`

```
<?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" >
    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true" />
</LinearLayout>
```



# Using the Spinner View

- Add the following lines to the **strings.xml** file:

```
<resources>
    <string name="hello">Hello World, BasicViews6Activity!</string>
    <string name="app_name">BasicViews6</string>
    <string-array name="presidents_array">
        <item>Dwight D. Eisenhower</item>
        <item>John F. Kennedy</item>
        <item>Lyndon B. Johnson</item>
        <item>Richard Nixon</item>
        <item>Gerald Ford</item>
        <item>Jimmy Carter</item>
        <item>Ronald Reagan</item>
        <item>George H. W. Bush</item>
        <item>Bill Clinton</item>
        <item>George W. Bush</item>
        <item>Barack Obama</item>
    </string-array>
</resources>
```



# Using the Spinner View

- Add the following statements to the **MainActivity.java** file:

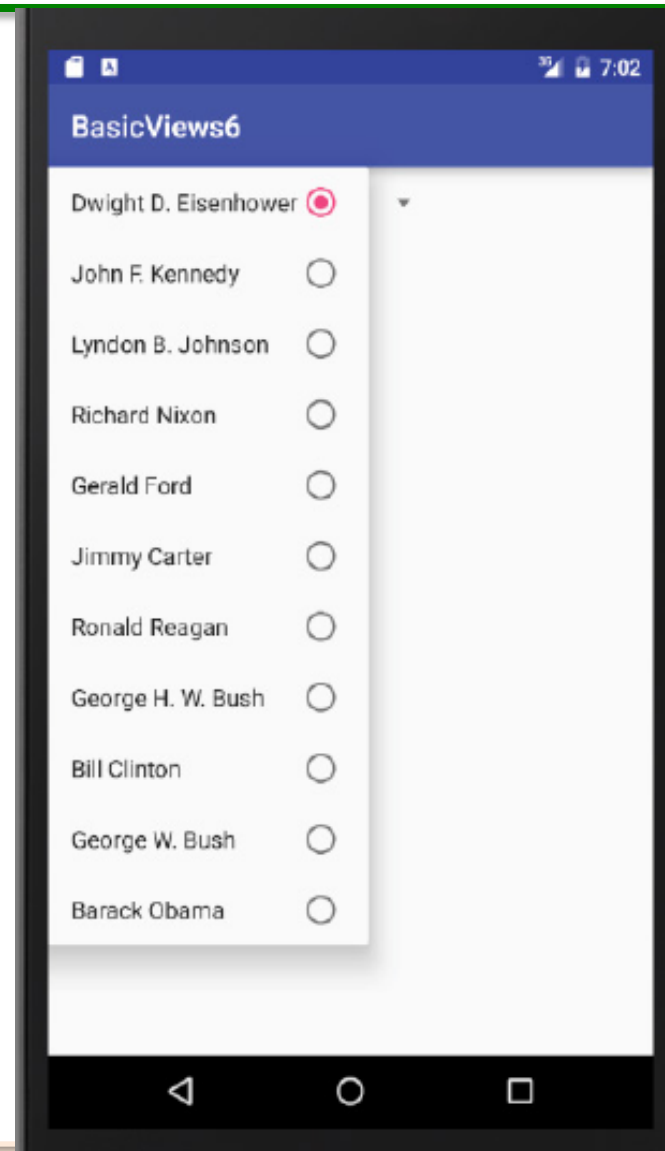
```
public class MainActivity extends AppCompatActivity {
    String[] presidents;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        presidents =
            getResources().getStringArray(R.array.presidents_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, presidents);
        s1.setAdapter(adapter);
        s1.setOnItemClickListener(new AdapterView.OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> arg0,
                View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getApplicationContext(),
                    "You have selected item : " + presidents[index],
                    Toast.LENGTH_SHORT).show();
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) { }
        });
    }
}
```



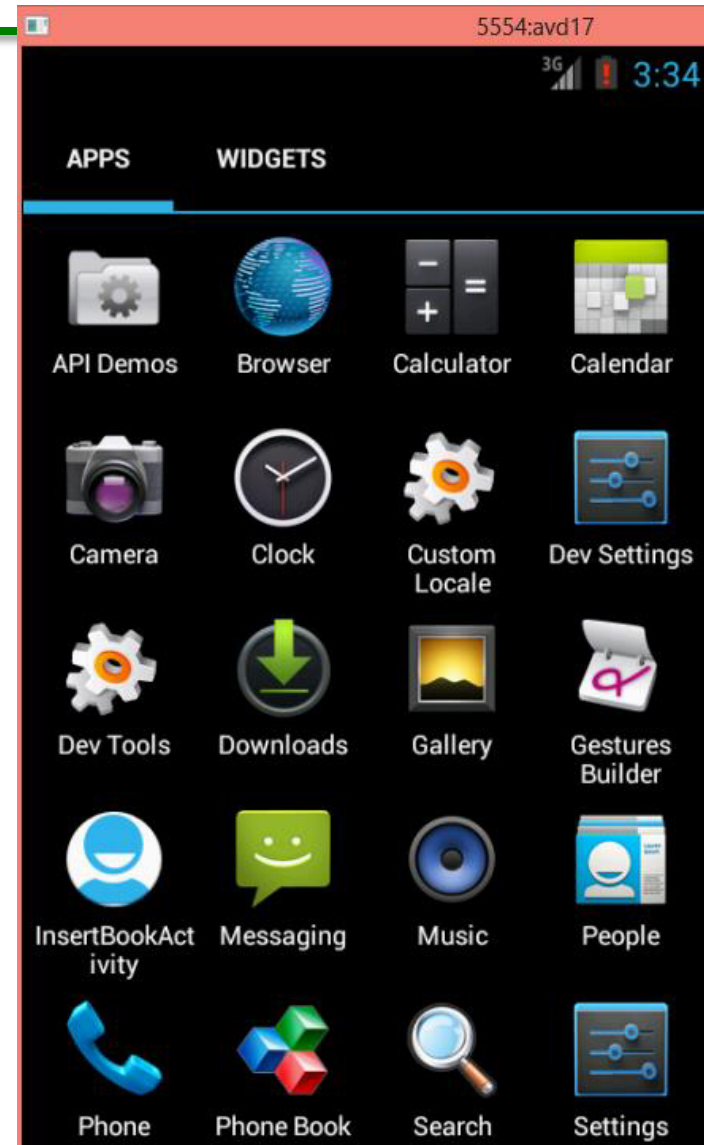
# Using the Spinner View

- Add the following statements to the **MainActivity.java** file:

```
ArrayAdapter<String> adapter = new  
    ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_sing  
        le_choice, presidents);
```



- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.
- The grid items are automatically inserted to the layout using a ListAdapter.



- GridView Some properties used to determine the number of columns and their sizes:
  - **android:numColumns** spells out how many columns there are, or, if you supply a value of `auto_fit`, Android will compute the number of columns based on available space and the properties listed below.
  - **android:verticalSpacing** and its counterpart **android:horizontalSpacing** indicate how much whitespace there should be between items in the grid.

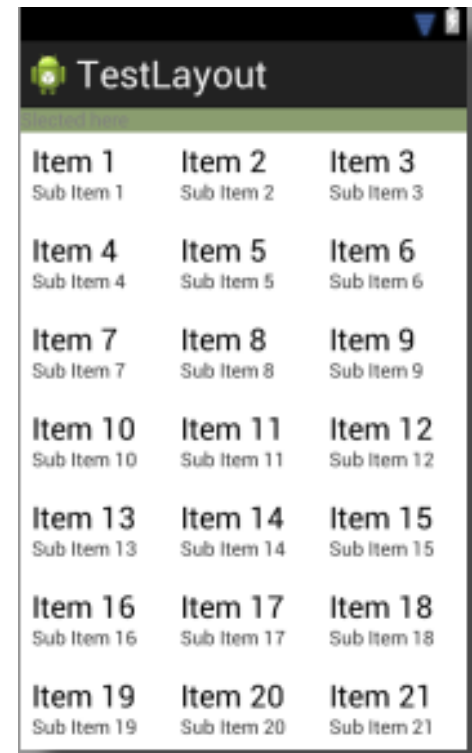


- GridView Some properties used to determine the number of columns and their sizes:
  - **android:columnWidth** indicates how many pixels wide each column should be.
  - **android:stretchMode** indicates, for grids with `auto_fit` for **android:numColumns**, what should happen for any available space not taken up by columns or spacing .

# GridView

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".GridViewActivity" >
    <TextView
        android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#8A9D6F"
        android:hint="Selected here" />
    <GridView
        android:id="@+id/gridView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:numColumns="3" >

    </GridView>
</LinearLayout>
```



# GridView

```
public class GridviewActivity extends Activity {
    String arr[]={ "Ipad", "Iphone", "New Ipad",
        "SamSung", "Nokia", "Sony Ericson",
        "LG", "Q-Mobile", "HTC", "Blackberry",
        "G Phone", "FPT - Phone", "HK Phone"
    };

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gridview);
        final TextView selection=(TextView)
            findViewById(R.id.selection);
        final GridView gv=(GridView) findViewById(R.id.gridView1);
        ArrayAdapter<String>da=new ArrayAdapter<String>
            (this, android.R.layout.simple_list_item_1, arr);
        gv.setAdapter(da);
        gv.setOnItemClickListener(new AdapterView
            .OnItemClickListener() {
                public void onItemClick(AdapterView<?> arg0,
                    View arg1, int arg2,
                    long arg3) {
                    selection.setText(arr[arg2]);
                }
            });
    }
}
```

Q-Mobile		
Ipad	Iphone	New Ipad
SamSung	Nokia	Sony Ericson
LG	Q-Mobile	HTC
Blackberry	G Phone	FPT - Phone
HK Phone		

# Time Selection

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

# Time Selection

- The widgets `TimePicker` and `TimePickerDialog`:
  - ✓ Set the initial **time** the user can adjust, in the form of an **hour** (**0** through **23**) and a **minute** (**0** through **59**).
  - ✓ Indicate if the selection should be in **12-hour** mode (with an AM/PM toggle), or in **24-hour** mode.
  - ✓ Provide a callback object (**`OnTimeChangeListener`** 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)



# Time Selection

```

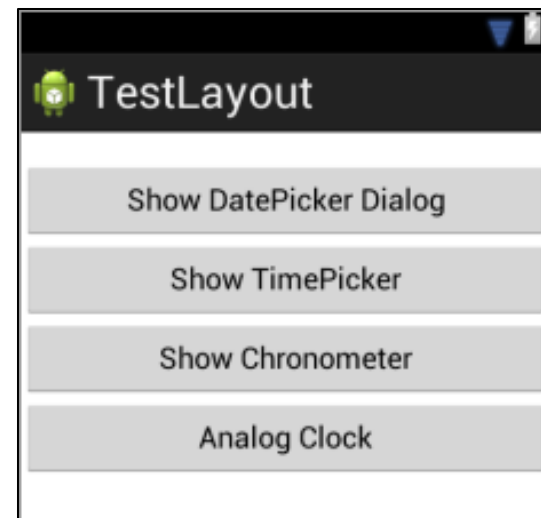
<LinearLayout xmlns:android="http://schemas.android
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/LinearLayout1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".TimeSelectionActivity" >
<TextView
    android:id="@+id/txtdate"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="" />
<Button
    android:id="@+id/btnDatePickerDialog"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Show DatePicker Dialog" />
<Button
    android:id="@+id/btnTimePicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Show TimePicker" />
<Button
    android:id="@+id/btnChronometer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Show Chronometer" />
<Button
    android:id="@+id/btnAnalogClock"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Analog Clock" />

```

```

<LinearLayout
    android:id="@+id/mylayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
</LinearLayout>
</LinearLayout>

```





# Time Selection

```
public class TimeSelectionActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_time_selection);
        doWork();
    }
    public void doWork()
    {
        Button b1= (Button) findViewById(R.id.btnAnalogClock);
        b1.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                AnalogClock analog=new AnalogClock
                    (TimeSelectionActivity.this);
                ((LinearLayout)findViewById(R.id.mylayout))
                    .addView(analog);
            }
        });
        Button b2= (Button) findViewById(R.id.btnChronometer);
        b2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Chronometer cro=new Chronometer
                    (TimeSelectionActivity.this);
                ((LinearLayout)findViewById(R.id.mylayout))
                    .addView(cro);
            }
        });
    }
}
```



# Time Selection

```
Button b3= (Button) findViewById(R.id.btnTimePicker);
b3.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        OnTimeSetListener callback=new OnTimeSetListener() {
            public void onTimeSet(TimePicker view,
                int hourOfDay, int minute) {
                ((TextView)findViewById(R.id.txtdate))
                .setText(hourOfDay + " - " +
                    minute + "###" +
                    view.getCurrentHour() + " - " +
                    view.getCurrentMinute());
            }
        };
        TimePickerDialog time=new TimePickerDialog(
            TimeSelectionActivity.this,
            callback, 11, 30, true);
        time.show();
    }
});
```



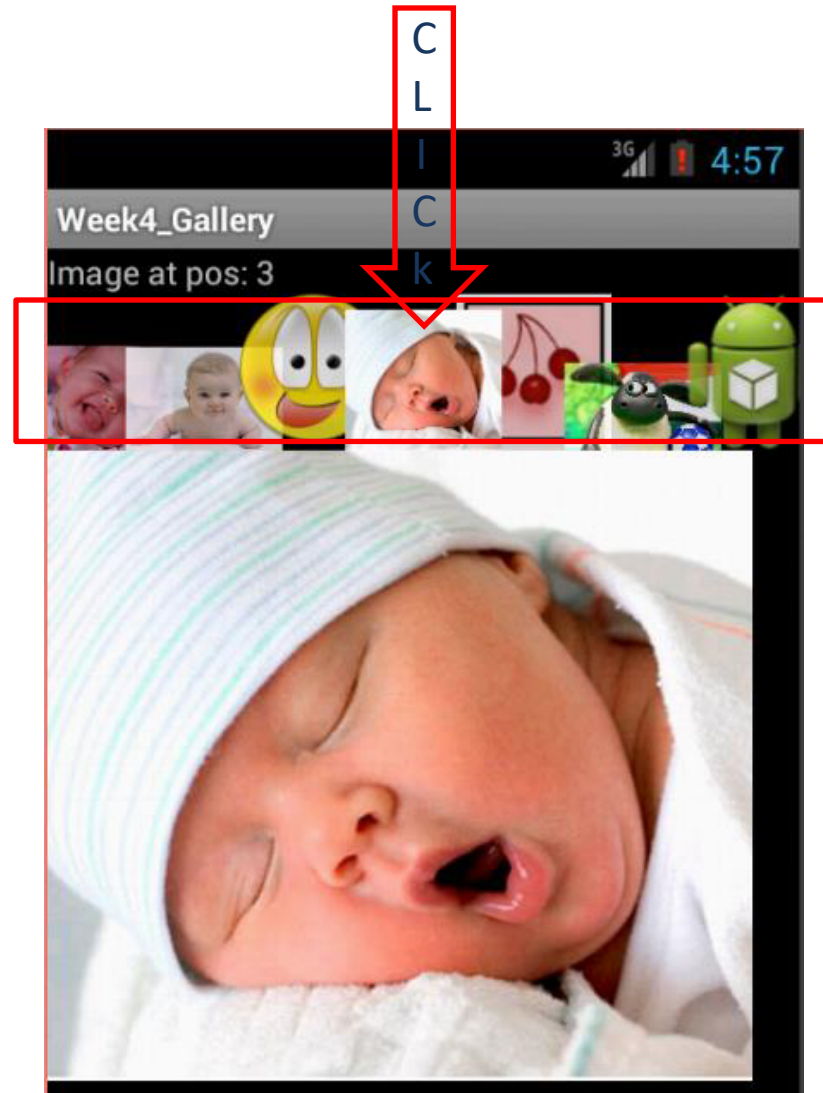


# Time Selection

```
Button b4= (Button) findViewById(R.id.btnDatePickerDialog);
b4.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        OnDateSetListener callback=new OnDateSetListener() {
            public void onDateSet(DatePicker view, int year,
                int monthOfYear,
                int dayOfMonth) {
                ((TextView)findViewById(R.id.txtdate))
                    .setText((dayOfMonth+1) + "/" +
                        (monthOfYear+1) + "/" + year);
            }
        };
        DatePickerDialog pic=new DatePickerDialog(
            TimeSelectionActivity.this,
            callback, 2012, 11, 30);
        pic.setTitle("My Datetime picker");
        pic.show();
    }
});
```

## ➤ Gallery Widget:

- The Gallery widget provides a set of options depicted as images.
- Image choices are offered on a contiguous horizontal mode, you may scroll across the image-set





# Picture Gallery

```
<?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" >
    <TextView
        android:id="@+id/txtselection"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
    {
    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
        />
    }
    <ImageView
        android:id="@+id/imageView1"
        android:contentDescription="view image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```



# Picture Gallery

```
public class Week4_GalleryActivity extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        doWork();  
    }  
    public void doWork()  
    {  
        final Gallery gallery=(Gallery) findViewById(R.id.gallery1);  
        gallery.setAdapter(new MyImageAdapter(this));  
        gallery.setOnItemClickListener(new OnItemSelectedListener() {  
            public void onItemClick(AdapterView<?> arg0, View arg1,  
                int arg2, long arg3) {  
                ((TextView)findViewById(R.id.txtselection)).  
                    setText("Image at pos: "+arg2);  
                int id=(Integer) gallery.getAdapter().getItem(arg2);  
                ((ImageView)findViewById(R.id.imageView1)).  
                    setImageResource(id);  
            }  
            public void onNothingSelected(AdapterView<?> arg0) {  
                ((TextView)findViewById(R.id.txtselection)).setText("nonono");  
            }  
        });  
    }  
}
```



# Picture Gallery

```
public class MyImageAdapter extends BaseAdapter {
    private Context context;
    private int []arr={R.drawable.image1,R.drawable.image2,
                        R.drawable.image3,R.drawable.image4,
                        R.drawable.image5,R.drawable.image6,
                        R.drawable.ic_launcher};

    public MyImageAdapter(Context context) {
        super();
        this.context = context;
    }
    public int getCount() {
        return arr.length;
    }
    public Object getItem(int arg0) {
        return arr[arg0];
    }
    public long getItemId(int arg0) {
        return arg0;
    }
    public View getView(int arg0, View arg1, ViewGroup arg2) {
        ImageView view=new ImageView(context);
        view.setImageResource(arr[arg0]);
        view.setScaleType(ImageView.ScaleType.FIT_END);
        view.setLayoutParams(new Gallery.LayoutParams(100, 100));
        return view;
    }
}
```



# Menus – Context menu

- Menus usually increase the functionality of an app by providing additional operations on a small overlapping panel. Android provides two types of menu known as: options menu and context menu.
  - The options menu is triggered by pressing the hardware menu button on the device, while
  - the context menu is raised by a tap-and-hold on the widget associated to the menu.



# Menus – Context menu

- Options menu and context menu may include:
  - ✓ Text.
  - ✓ Icons
  - ✓ RadioButton
  - ✓ CheckBox
  - ✓ Sub-menu

- Webkit Browser
- Permission to access internet
- Browser commands
- HTML + Javascript + android
- Demo find location



# Webkit Browser

- In Android you can embed the built-in Web browser as a widget in your own activities, for displaying HTML material or perform Internet browsing.
- The Android browser is based on Webkit, the same engine that powers Apple's Safari Web browser.
- Android uses the WebView widget to host the browser's pages.
- Applications using the WebView component must request INTERNET permission.

# Multi language in Android

- Why multi language?
- Setup multi language

