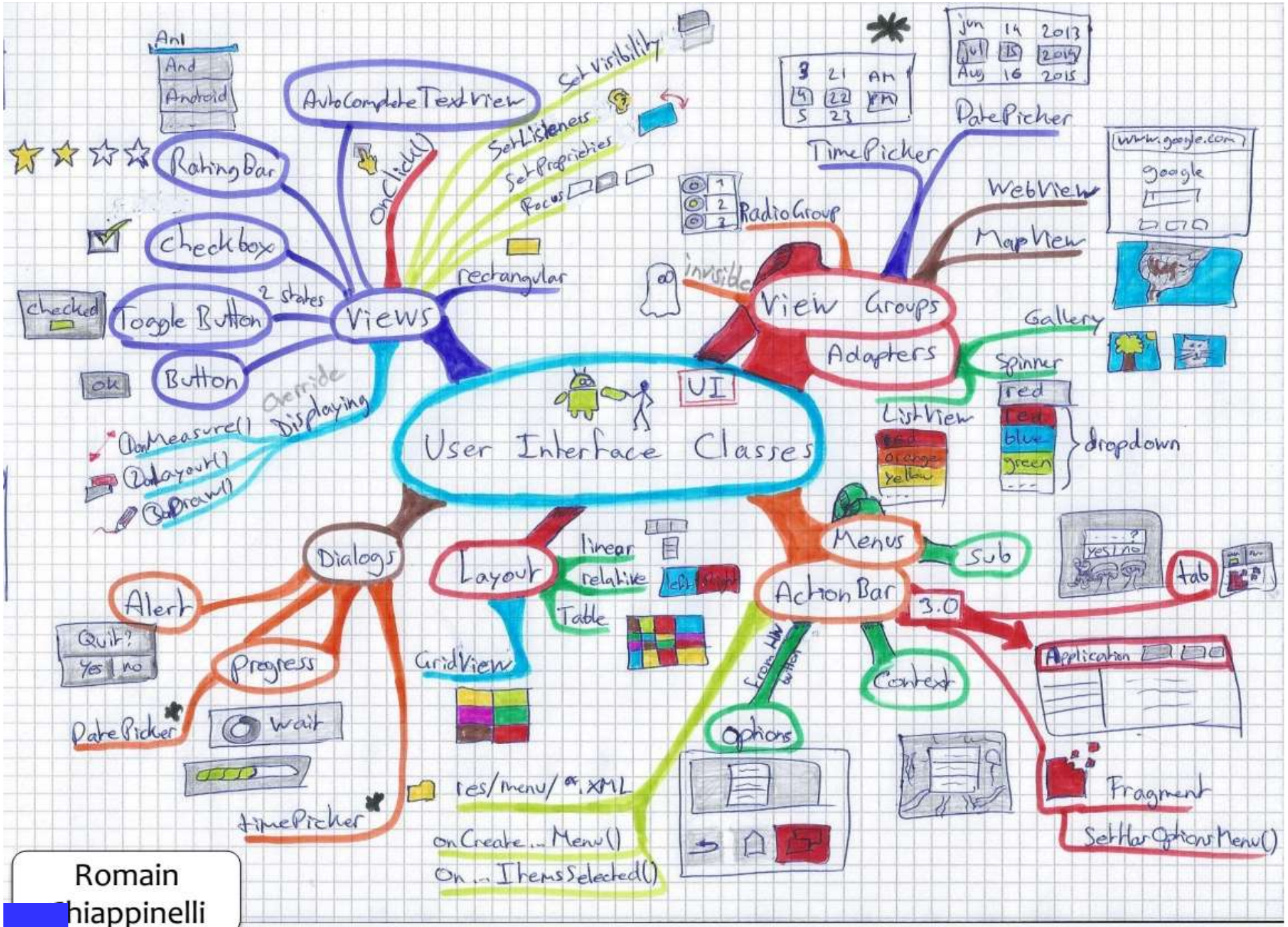
CHAPITRE 6
LES CLASSES « UI »

• Mr. MEGHAZI

2016-2017

Cours pour les Master II - GL



Romain Chiappinelli

Contenu

- Views et View Events
- View Groups, AdapterViews et Layouts
- Menus et ActionBar
- Dialogs

Android User Interface

- C'est l'endroit et le moyen par lequel un utilisateur et une application interagissent/échangent des infos .
- Une activité généralement affiche une interface utilisateur.
- Android fournit plusieurs classes pour la construction des interfaces utilisateurs.

Modern UI : sonsors: audio, led light, ondes radio ...

View (1)

- Un élément clé pour la construction des blocs des composants d'une UI.
- Occupe un espace rectangulaire dans l'écran.
- Responsable pour se dessiner et prendre en charge les évènements

Les vues (View) prédéfinie (2)

- Button
- ToggleButton
- Checkbox
- RatingBar
- AutoCompleteTextView

Les Boutons (3) - 1

- C'est une vue qui permet de faire des cliques là-dessus pour exécuter une action.

UIBUTTON

Les Boutons (3) - 2

UIButton

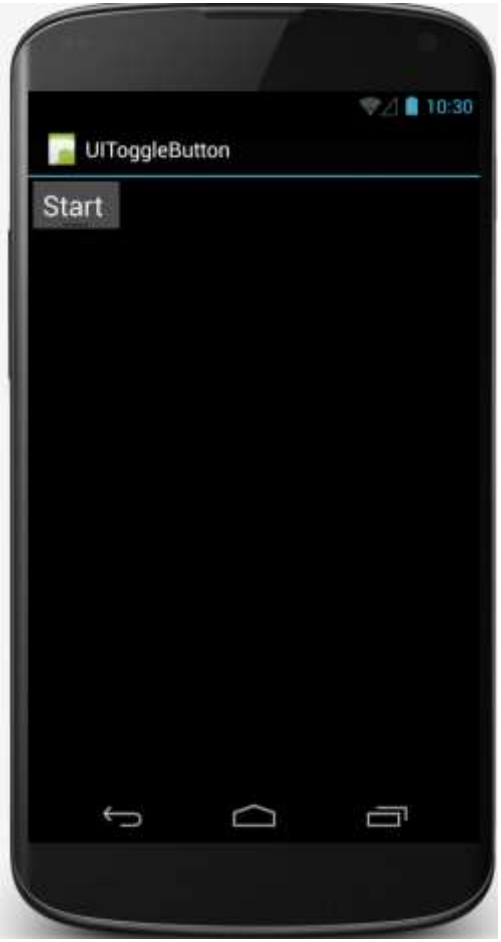
```
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="10dip"
    android:text="@string/press_me_string" >
</Button>
```

```
// Get a reference to the Press Me Button
final Button button = (Button) findViewById(R.id.button);

// Set an OnClickListener on this Button
// Called each time the user clicks the Button
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        // Maintain a count of user presses
        // Display count as text on the Button
        button.setText("Got Pressed:" + ++count);

    }
});
```

ToggleButton (4) - 1

- Un bouton qui a deux états, checked/not-checked.
- Voyant lumineux indiquant l'état.

UIToggleButton



ToggleButton (4) - 2

UIToggleButton

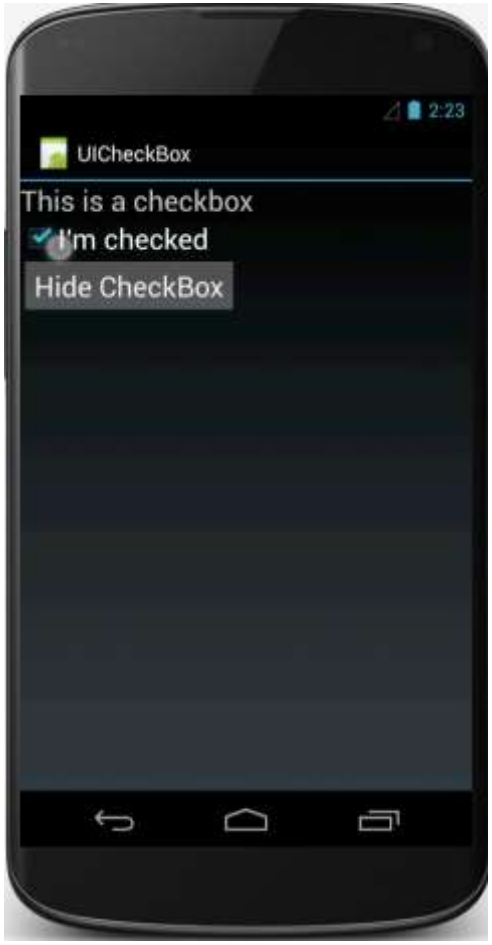
```
<ToggleButton
    android:id="@+id/togglebutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="@string/start_string"
    android:textOn="@string/stop_string"
    android:textSize="24sp" />
```

```
// Get a reference to the ToggleButton
final ToggleButton button = (ToggleButton) findViewById(R.id.togglebutton);

// Set an OnClickListener on the ToggleButton
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        // Toggle the Background color between a light and dark color
        if (button.isChecked()) {
            bg.setBackgroundColor(0xFFF3F3F3);
        } else {
            bg.setBackgroundColor(0xFF000000);
        }
    }
});
```

CheckBox (5) - 1

- Un autre type de bouton qui a deux états, checked/not-checked.

UICheckBox



CheckBox (5) - 2

UICheckBox

```
<CheckBox
    android:id="@+id/checkbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/im_not_checked_string"
    android:textSize="24sp"/>
```

```
// Get a reference to the CheckBox
final CheckBox checkbox = (CheckBox) findViewById(R.id.checkbox);

// Set an OnClickListener on the CheckBox
checkbox.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        // Check whether CheckBox is currently checked
        // Set CheckBox text accordingly
        if (checkbox.isChecked()) {
            checkbox.setText("I'm checked");
        } else {
            checkbox.setText("I'm not checked");
        }
    }
});
```

RatingBar (6) - 1

- Une vue qui contient une ligne d'étoiles.
- L'utilisateur peut cliquer ou faire glisser les étoiles pour sélectionner un certain nombre d'étoiles.

UIRatingBar

RatingBar (6) - 2

UIRatingBar

```
<RatingBar
    android:id="@+id/ratingbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="4"
    android:stepSize="1.0" >
</RatingBar>
```

```
final RatingBar bar = (RatingBar) findViewById(R.id.ratingbar);

bar.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {

    // Called when the user swipes the RatingBar
    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {
        tv.setText("Rating:" + rating);
    }
});
```

AutoCompletTextView (7) - 1

- Un champ de texte modifiable qui fournit des suggestions que les utilisateurs saisissent dans la zone de texte.

UIAutoComplete TextView

AutoCompletTextView (7) - 2

UIAutoComplete TextView

```
<AutoCompleteTextView
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:textSize="24sp"/>
```

```
// Get a reference to the AutoCompleteTextView
AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete_country);

// Create an ArrayAdapter containing country names
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        R.layout.list_item, COUNTRIES);

// Set the adapter for the AutoCompleteTextView
textView.setAdapter(adapter);
```

Opérations communes aux Vues

- **Set visibility**: Affiche ou cache une vue
- **Set Checked state**
- **Set Listeners**: code à exécuter quand un évènement bien précis se produit
- **Set properties**: opacity, background, rotation ....
- **Manage input focus**: allow view to take focus, request focus

Les sources des évènements d'une vue

- Interaction avec l'user
  - Touché
  - Keyboard/trackball/D-pad
- System control
  - Les changements du cycle de vie

Prise en charge des évènements sur une vue

- Dans la majeure partie des cas la prise en charge d'un évènement se fait par les écouteurs (listeners).
- Il existe plusieurs écouteurs définies par la classe « View ».

Interfaces des écouteurs de « View » - (1)

- OnClickListener.onClick()
  - La vue vient de recevoir un clique
- OnLongClickListener.onLongClick()
  - Quand une vue reçoit un évènement qui a pour effet d'appuyer sur cette vue et maintenir cette action pour un moment.

Interfaces des écouteurs de « View » - (2)

- OnFocusChangeListener.onFocusChange()
  - View has received or lost focus
- OnKeyListener.onKey()
  - View about to receive a hardware key press

Affichage des vues

- Les vues sont organisé sous forme d'une arborescence .
- L' affichage a plusieurs étapes:
  - **Measure** - obtenir les dimensions de chaque vue.
  - **Layout** – positionner chaque vue
  - **Draw** – Dessiner chaque vue

Prise en charge des évènement de « View » (1)

- onMeasure()
  - Détermine les dimensions de la vue et ses sous vues
- onLayout()
  - La vue doit attribuer une taille et une position à ses sous vues
- onDraw()
  - La vue doit afficher son contenu.

Prise en charge des évènement de « View » (2)

- onFocusChanged()
  - View's focus state has changed
- onKeyUp(), onKeyDown()
  - A hardware key event has occurred
- onWindowVisibilityChanged()
  - Window containing view has changed its visibility status

ViewGroup

Les « ViewGroup »

- Une vue invisible qui contient d'autres vues
- Utilisé pour le regroupement et l'organisation d'un ensemble de vues
- Classe de base pour les conteneurs de vues et les Layouts

Exemple:

sélectionner une tranche d'âge

Quelques « ViewGroups » prédéfinis

- RadioGroup
- TimePicker
- DatePicker
- WebView
- MapView
- Gallery
- Spinner

RadioGroup (1) - 1

- Une « ViewGroup » qui contient un ensemble de Radio Buttons (CheckBoxes).
- Seulement un seul bouton peut être sélectionné à un moment donné

UIRadioGroup

RadioGroup (1) - 2

UIRadioGroup

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >

    <RadioButton
        android:id="@+id/choice1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_1_string"
        android:textSize="24sp"/>

    <RadioButton
        android:id="@+id/choice2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_2_string"
        android:textSize="24sp"/>

    <RadioButton
        android:id="@+id/choice3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_3_string"
        android:textSize="24sp" />
</RadioGroup>
```

RadioGroup (1) - 3

UIRadioGroup

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

    final TextView tv = (TextView) findViewById(R.id.textView);

    // Define a generic listener for all three RadioButtons in the RadioGroup
    final OnClickListener radioListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            RadioButton rb = (RadioButton) v;
            tv.setText(rb.getText() + " chosen");
        }
    };

    final RadioButton choice1 = (RadioButton) findViewById(R.id.choice1);
    // Called when RadioButton choice1 is clicked
    choice1.setOnClickListener(radioListener);

    final RadioButton choice2 = (RadioButton) findViewById(R.id.choice2);
    // Called when RadioButton choice2 is clicked
    choice2.setOnClickListener(radioListener);

    final RadioButton choice3 = (RadioButton) findViewById(R.id.choice3);
    // Called when RadioButton choice3 is clicked
    choice3.setOnClickListener(radioListener);

}
```

TimePicker (2) - 1

- Une « ViewGroup » qui permet à un utilisateur de choisir une heure.

UITIMEPICKER

TimePicker (2) - 2

UITimePicker

```
<TextView
    android:id="@+id/timeDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="24sp" />

<Button
    android:id="@+id/pickTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/change_the_time_string"
    android:textSize="24sp" />
```

TimePicker (2) - 3

UITimePicker

```
// The callback received when the user "sets" the time in the dialog
private TimePickerDialog.OnTimeSetListener mTimeSetListener = new TimePickerDialog.OnTimeSetListener() {
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        mHour = hourOfDay;
        mMinute = minute;
        updateDisplay();
    }
};
```

```
@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case TIME_DIALOG_ID:
        return new TimePickerDialog(this, mTimeSetListener, mHour, mMinute,
                false);
    }
    return null;
}
```

DatePicker (3) - 1

- Une « ViewGroup » qui permet à un utilisateur de choisir une date.

UIDatePicker

DatePicker (3) - 2

UIDatePicker

```
<TextView
    android:id="@+id/dateDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="24sp" />

<Button
    android:id="@+id/pickDate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/change_the_date_string"
    android:textSize="24sp" />
```

DatePicker (3) - 3

UIDatePicker

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

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

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

WebView (4) - 1

- Une « ViewGroup » qui affiche une page Web

UIWebView

WebView (4) - 2

UIWebView

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```

WebView (4) - 3

UIWebView

```
mWebView = (WebView) findViewById(R.id.webview);

// Set a kind of listener on the WebView so the WebView can intercept
// URL loading requests if it wants to

mWebView.setWebViewClient(new HelloWebViewClient());

mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.loadUrl("http://www.google.com");
```

```
private class HelloWebViewClient extends WebViewClient {
    private static final String TAG = "HelloWebViewClient";;

    // Give application a chance to catch additional URL loading requests
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        Log.i(TAG, "About to load:" + url);
        view.loadUrl(url);
        return true;
    }
}
```

MapView (5) - 1

- Une « ViewGroup » qui affiche une carte

UIGoogleMaps

MapView (5) - 2

UIGoogleMaps

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>
```

```
// The GoogleMap instance underlying the GoogleMapFragment defined in main.xml
mMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map))
        .getMap();

if (mMap != null) {

    // Set the map position
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(29,
            -88), 0));

    // Add a marker on Washington, DC, USA
    mMap.addMarker(new MarkerOptions().position(
            new LatLng(38.8895, -77.0352)).title(
            getString(R.string.in_washington_string)));

    // Add a marker on Mexico City, Mexico
    mMap.addMarker(new MarkerOptions().position(
            new LatLng(19.13, -99.4)).title(
            getString(R.string.in_mexico_string)));
}
```

Adapters et AdapterViews

Adapters et AdapterViews

- « AdapterViews » sont des vues où leurs instances et leurs données sont gérés par des « Adapters »
- Les « Adapters » gèrent les données et fournissent des données de la vue à un « AdapterView »
- « AdapterView » affiche la vue des données.

ListView (6) - 1

- Un « **AdapterView** » affichant une liste avec une barre de défilement pour des éléments qui sont gérés par un « **ListAdapter** »
- « **ListView** » peut filtrer la liste des éléments en se basant sur ce qui est écrit.

UILISTVIEW

ListView (6) - 2

UIListView

```
// Create a new Adapter containing a list of colors
// Set the adapter on this ListActivity's built-in ListView
setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item,
        getResources().getStringArray(R.array.colors)));

ListView lv = getListView();

// Enable filtering when the user types in the virtual keyboard
lv.setTextFilterEnabled(true);

// Set an setOnItemClickListener on the ListView
lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {

        // Display a Toast message indicting the selected item
        Toast.makeText(getApplicationContext(),
                ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
    }
});
```

Spinner (7) - 1

- Un « **AdapterView** » qui fournie une liste défilante d'éléments
- L'User peut sélectionner un seul élément de la liste

UISPINNER

Spinner (7) - 2

UISpinner

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp" />
```

```
// Get a reference to the Spinner
Spinner spinner = (Spinner) findViewById(R.id.spinner);

// Create an Adapter that holds a list of colors
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
        this, R.array.colors, R.layout.dropdown_item);

// Set the Adapter for the spinner
spinner.setAdapter(adapter);

// Set an setOnItemSelectedListener on the spinner
spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
```

Gallery (8) - 1

- Une « ViewGroup » affichant horizontalement une liste défilante.
- Les éléments sont gérés par un « SpinnerAdapter »

UIGALLERY

Gallery (8) - 2

UIGallery

```
<Gallery xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

```
Gallery g = (Gallery) findViewById(R.id.gallery);

// Create a new ImageAdapter and set in as the Adapter for the Gallery
g.setAdapter(new ImageAdapter(this));

// Set an setOnItemClickListener on the Gallery
g.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {
        // Display a Toast message indicate the selected item
        Toast.makeText(GalleryActivity.this, "" + position,
                Toast.LENGTH_SHORT).show();
    }
});
```

Gallery (8) - 3

UIGallery

```
// The Adapter class used with the Gallery
public class ImageAdapter extends BaseAdapter {

    private static final int IMAGE_DIM = 800;

    private int mGalleryItemBackground;
    private Context mContext;

    // List of IDs corresponding to the images
    private Integer[] mImageIds = { R.drawable.sample_1,
            R.drawable.sample_2, R.drawable.sample_3, R.drawable.sample_4,
            R.drawable.sample_5, R.drawable.sample_6, R.drawable.sample_7 };

    public ImageAdapter(Context c) {
        mContext = c;
        TypedArray a = obtainStyledAttributes(R.styleable.GalleryActivity);
        mGalleryItemBackground = a.getResourceId(
                R.styleable.GalleryActivity_android_galleryItemBackground,
                0);
        a.recycle();
    }

    public int getCount() {
        return mImageIds.length;
    }

    public Object getItem(int position) {
        return mImageIds[position];
    }

    public long getItemId(int position) {
        return position;
    }
```

Gallery (8) - 4

UIGallery

```
public View getView(int position, View convertView, ViewGroup parent) {

    ImageView imageView = (ImageView) convertView;

    // If convertView is not recycled set it up now
    if (null == imageView) {
        imageView = new ImageView(mContext);

        imageView.setLayoutParams(new Gallery.LayoutParams(IMAGE_DIM,
                IMAGE_DIM));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setBackgroundResource(mGalleryItemBackground);

    }

    // Set the image for the imageView
    imageView.setImageResource(mImageIds[position]);

    return imageView;
}
```

Les Layouts

Layouts

- Un « ViewGroup » générique qui **définie** et **structure** pour les vues leurs **Contenus**

LinearLayout (1)

- Les vues filles sont rangées dans une seule ligne, **Horizontalement** ou **verticalement**.

UILINEARLAYOUT

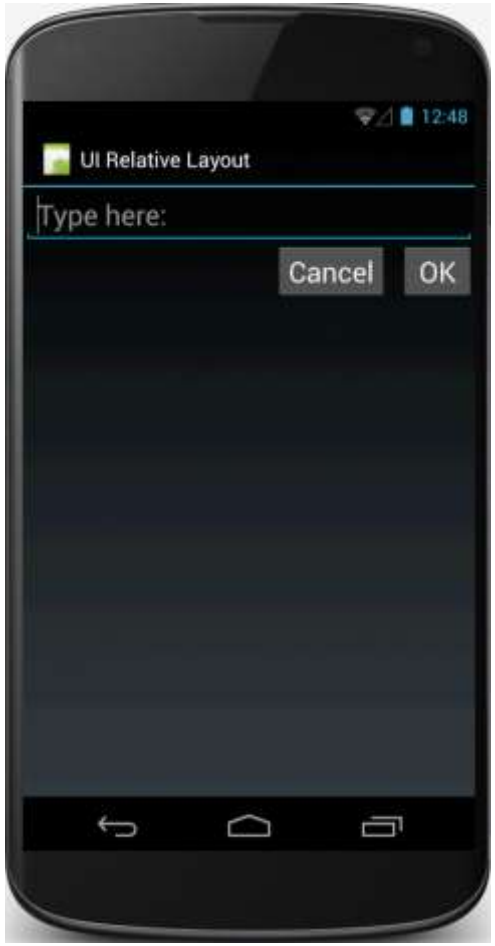
LinearLayout (2)

UILinearLayout

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal" >

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:orientation="vertical" >
```

RelativeLayout (1)

- Les vues filles sont positionnées relativement entre-elles et par rapport à la vue parent



UIRelativeLayout

RelativeLayout (2)

UIRelativeLayout

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

    <!-- Note the use of android:hint to put explanatory text in the EditText  -->
    <EditText
        android:id="@+id/entry"
        android:layout width="match parent"
```

TableLayout (1)

- Les vues filles sont rangées dans des lignes et des colonnes.

UITableLayout

TableLayout (2)

UITableLayout

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1" >

    <!-- First row -->
    <TableRow>

        <!-- start in column 1 -->
        <TextView
            android:layout_column="1"
            android:padding="3dip"
            android:text="@string/open_string"
            android:textSize="24sp" />

        <TextView
            android:gravity="right"
            android:padding="3dip"
            android:text="@string/ctrl_o_string"
            android:textSize="24sp" />
    </TableRow>
```

GridLayout (1)

- Les vues filles sont rangées dans une grille 2D défilante.

UIGridView

GridLayout (2)

UIGridView

```
<GridView
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="90dp"
    android:gravity="center"
    android:horizontalSpacing="10dp"
    android:numColumns="auto_fit"
    android:stretchMode="columnWidth"
    android:verticalSpacing="10dp" />
```

```
GridView gridview = (GridView) findViewById(R.id.gridview);

// Create a new ImageAdapter and set it as the Adapter for this GridView
gridview.setAdapter(new ImageAdapter(this, mThumbIdsFlowers));

// Set an setOnItemClickListener on the GridView
gridview.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {
```

Les Menus et les ActionBars

- Les activités peuvent utiliser des Menus
- Les Activités peuvent:
  - Ajouter un élément à un menu
  - Prendre en charge les cliques sur un élément d'un menu

Les types de menus

- **Options**
  - Menus qui sont affichés quand un utilisateur appuie le bouton « Menu »
- **Context**
  - Spécifique à une vue, un menu est affiché quand un utilisateur touche maintient la vue
- **SousMenu (SubMenu)**
  - Un menu est activer quand un utilisateur touche un élément visible d'un menu

Création des menus

- Gonfler une ressource de type « Menu » en utilisant un menu
- Le gonfleur (inflater) est invoqué en utilisant des méthodes «onCreate...Menu()»
- Prise en charge de la sélection d'un élément dans une des méthodes appropriées «on...ItemsSelected()»

Exemple sur les menus (1)

HelloAndroid WithMenus

Exemple sur les menus (2)

```
// Create Options Menu
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.top_menu, menu);
    return true;
}

// Process clicks on Options Menu items
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.help:
        Toast.makeText(getApplicationContext(), "you've been helped",
                Toast.LENGTH_SHORT).show();
        return true;
    case R.id.more_help:
        Toast.makeText(getApplicationContext(), "you've been helped more",
                Toast.LENGTH_SHORT).show();
        return true;
    case R.id.even_more_help:
        return true;
    default:
        return false;
    }
}
```

On veut afficher ce menu maintenant

Exemple sur les menus (3)

```
// Create Context Menu
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}

// Process clicks on Context Menu Items
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.help_guide:
        Toast.makeText(getApplicationContext(), "ContextMenu Shown",
                Toast.LENGTH_SHORT).show();
        return true;
    default:
        return false;
    }
}
```

Un + sur les menus

- Beaucoup d'autres fonctionnalités sont prises en charge:
  - Grouper des éléments d'un menu
  - Lier des raccourcis à des éléments d'un menu
  - Lier des « intents » aux éléments d'un menu

Les « ActionBar »

- Similaire à des « Application Bar » dans +sieurs applications de bureau
- Permettent un accès rapide à certaines fonctionnalités

Les « ActionBar » - Exemple

FragmentDynamiqueLayoutWithActionBar

Les boites de dialogues

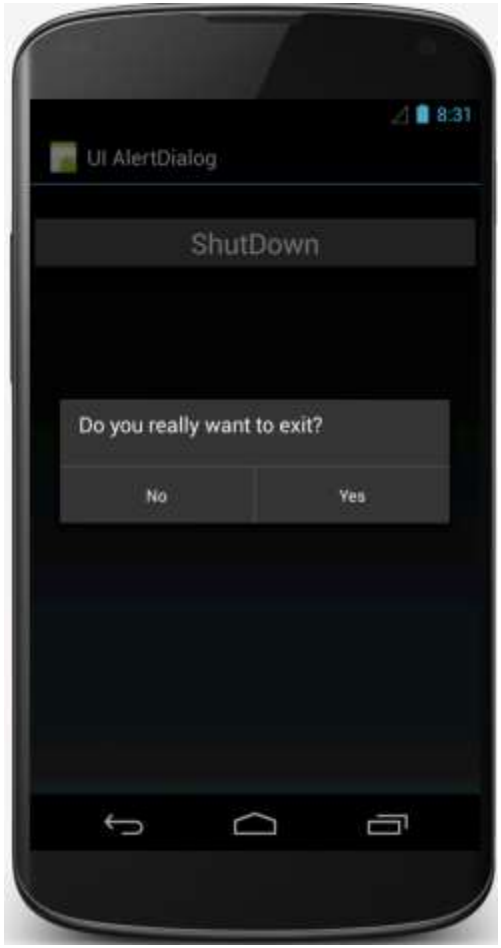
- Des sous-fenêtres utilisées par les activités pour communiquer avec l'utilisateur

- Sous Classes de « **Dialog** » :
  - **AlertDialog**
  - **ProgressDialog**
  - **DatePickerDialog**
  - **TimePickerDialog**

Exemple sur les Dialogs (1)

UIAlertDialog

ProgressDialog



Exemple sur les Dialogs (2)

UIAlertDialog

```
// Class that creates the AlertDialog
public static class AlertDialogFragment extends DialogFragment {

    public static AlertDialogFragment newInstance() {
        return new AlertDialogFragment();
    }

    // Build AlertDialog using AlertDialog.Builder
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return new AlertDialog.Builder(getActivity())
                .setMessage("Do you really want to exit?")

                // User cannot dismiss dialog by hitting back button
                .setCancelable(false)

                // Set up No Button
                .setNegativeButton("No",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                    int id) {
                                ((AlertDialogActivity) getActivity())
                                        .continueShutdown(false);
                            }
                        })

                // Set up Yes Button
                .setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {
                            public void onClick(
                                    final DialogInterface dialog, int id) {
                                ((AlertDialogActivity) getActivity())
                                        .continueShutdown(true);
                            }
                        }).create();
    }
}
```