P3 : Les interfaces utilisateur

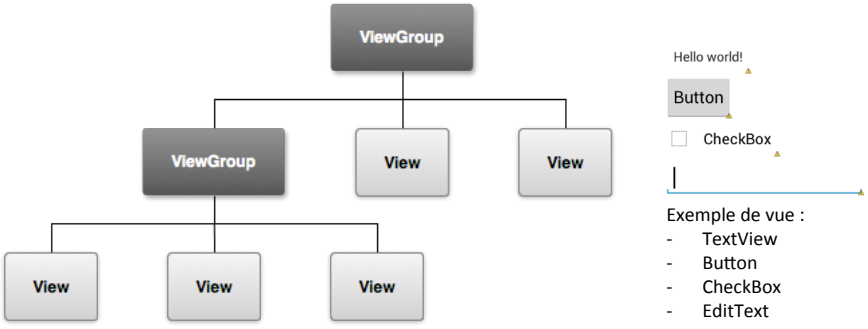
1. Les layouts (gabarit) et les Composants (vue)
2. Les évènements
3. Les listes
4. Les menus
5. Styles et thèmes
6. La gestion des langues
7. TP: Développement d'une application simple

1 – Layouts (gabarit) et Composants (vue)

La classe centrale pour réaliser une interface graphique sous Android est la classe « View ». Chaque partie de votre interface est une vue ou un groupe de vues qui hérite de cette classe.

Une ou plusieurs vues peuvent être regroupées dans ce que l'on appelle « ViewGroup » .

Les vues peuvent être un champ de texte, bouton, checkbox ...



Exemple de vue :
- TextView
- Button
- CheckBox
- EditText

1 – Layouts (gabarit) et Composants (vue)

2 méthodes possibles pour afficher un layout :

- méthode procédurale en Java
- méthode déclarative en XML

Privilégier autant que possible la méthode déclarative.

- interface et code séparé
- aucune perte de performance car le XML est compilé.

Utiliser la méthode procédurale pour effectuer des modifications dynamique de l'interface graphique.

1 – Layouts (gabarit) et Composants (vue)

Valeurs des attributs « width » et « height » :

WRAP_CONTENT :

- le compostant utilisera l'espace nécessaire.

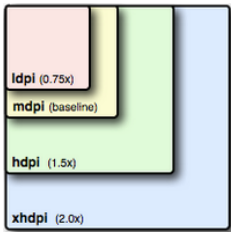
MATCH_PARENT (ou FILL_PARENT) :

- le composant utilisera tout l'espace disponible.

2 – Les unités de mesure

Android gère de nombreuses unités de mesure dont voici les principales :

- Pixels (px) : correspond à un pixel à l'écran
- Pixels à densité indépendante (dp ou dip) : unité basée sur la densité de l'écran (par défaut 160 dpi pour 1 dp).
- Pixels à taille indépendante (sp) : identique au dip mais prend en charge les préférences utilisateur (utilisé principalement sur les textes).



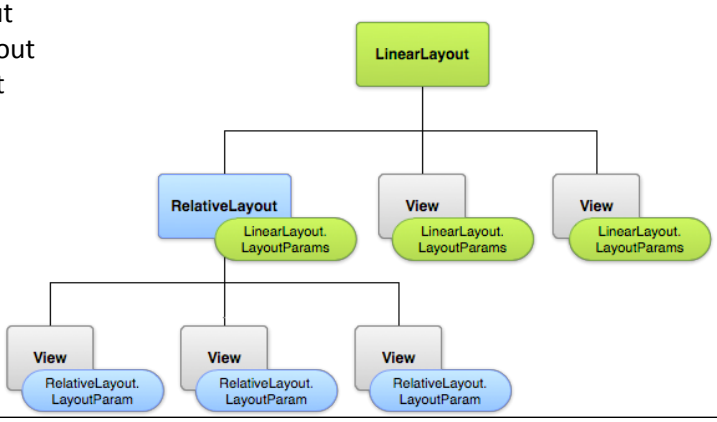
1 – Layouts (gabarit) et Composants (vue)

Les layout permettent de contenir des vues et de les positionner.

Ils sont représentés sous forme de fichier XML ou créés directement dans le code.

Il existe plusieurs types de Layout :

- LinearLayout
- RelativeLayout
- TableLayout
- GridView
- ScrollView
- etc ...



1 – Layouts (gabarit) et Composants (vue)

Plusieurs type de layout :

FrameLayout : empile les éléments les uns au-dessus des autres (calques).

LinearLayout : aligne les éléments dans une direction unique (horizontal ou vertical).

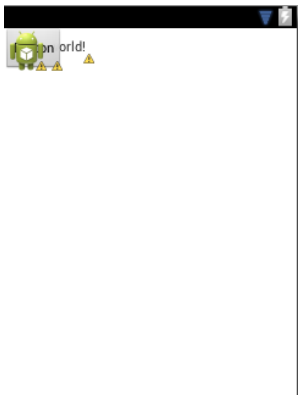
RelativeLayout : positionne les éléments fils par rapport au parent (exemple: A à droite de B et C en dessous de B).

TableLayout : organise les éléments en ligne et en colonne (équivalent tableau html).

1 – Layouts (gabarit) et Composants (vue)

FrameLayout : empile les éléments les uns au-dessus des autres (calques).

```
<FrameLayout 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"
                          >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dip"
        android:text="@string/hello_world" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

</FrameLayout>
```



1 – Layouts (gabarit) et Composants (vue)

LinearLayout : aligne les éléments dans une direction unique (horizontal ou vertical).

```
<LinearLayout 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:orientation="vertical">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:padding="10dip" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />
</LinearLayout>
```

1 – Layouts (gabarit) et Composants (vue)

RelativeLayout : positionne les éléments fils par rapport au parent (exemple: A à droite de B et C en dessous de B).

```
<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"
                        >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:padding="10dip"
        android:text="@string/hello_world" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/imageView1"
        android:text="Button" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />
</RelativeLayout>
```

1 – Layouts (gabarit) et Composants (vue)

TableLayout : organise les éléments en ligne et en colonne (équivalent tableau html).

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tableLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dip" >
        <Button
            android:id="@+id/button1"
            android:text="Column 1" />
        <Button
            android:id="@+id/button2"
            android:text="Column 2" />
        <Button
            android:id="@+id/button3"
            android:text="Column 3" />
    </TableRow>

    <TableRow
        android:id="@+id/tableRow2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dip" >
        <Button
            android:id="@+id/button1"
            android:text="Column 1" />
```

1 – Layouts (gabarit) et Composants (vue)

Il existe de nombreux composants graphiques (vues) fournis de base sur Android :

- EditText
- TextView
- Button
- ImageView
- ListView
- Etc ...

1 – Layouts (gabarit) et Composants (vue)

EditText :

Permet l'ajout de texte.

- new EditText(Context context)
- setText() / getText().toString()

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
                                        >
</EditText>
```

1 – Layouts (gabarit) et Composants (vue)

TextView :

Création d'un champs de texte.

- new TextView(Context context)
- setText() pour ajouter un texte

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    />
```

1 – Layouts (gabarit) et Composants (vue)

Button :

Un simple bouton cliquable.

- new Button(Context context)
- setText() pour ajouter un texte

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button" />
```

1 – Layouts (gabarit) et Composants (vue)

ImageView :

Affiche une image.

- new ImageView(Context context)
- setImageRessource() pour modifier l'image

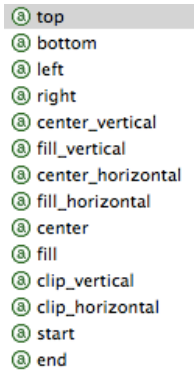
```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />
```

1 – Layouts (gabarit) et Composants (vue)

« gravity » :

Cette attribut permet de positionner une vue.

Options disponibles :



android:gravity="center"

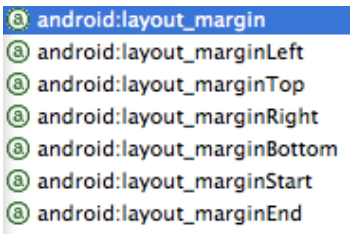
```
android:gravity="center"
```

1 – Layouts (gabarit) et Composants (vue)

« margin » :

Cette attribut permet d'ajouter une marge.

Options disponibles :



android:layout_ma
rgin="50dp"

```
android:layout_margin="50dp"
```

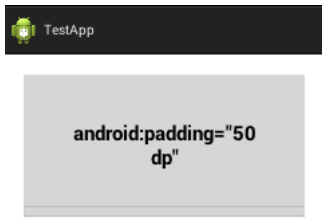
1 – Layouts (gabarit) et Composants (vue)

« padding » :

Cette attribut permet d'ajouter une marge intérieur.

Options disponibles :

- android:padding
- android:paddingLeft
- android:paddingTop
- android:paddingRight
- android:paddingBottom
- android:paddingStart
- android:paddingEnd




```
android:padding="50dp"
```

2 – Les événements

L'attribut « @+id/ » permet de récupérer la vue dans le code Java.

XML :

```
<TextView
    android:id="@+id/textViewTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Java :

```
TextView textViewTitle = (TextView) findViewById(R.id.textViewTitle);
```

2 – Les événements

Certains éléments graphiques peuvent être activés par l'utilisateur. Pour ce faire, il suffit d'utiliser un « listener ».

```
public class MainActivity extends Activity {

    private Button button1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button1 = (Button) findViewById(R.id.button1);

        button1.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

            }
        });
    }
}
```

2 – Les événements

Depuis 1.6 l'attribut « android:onClick » permet de définir le nom de la méthode de l'activité à appeler.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Home"
    android:onClick="goToHome" />
```

```
public void goToHome(View v) {

}
```

3 – Les listes

ListView:

Permet d'afficher des informations en liste.

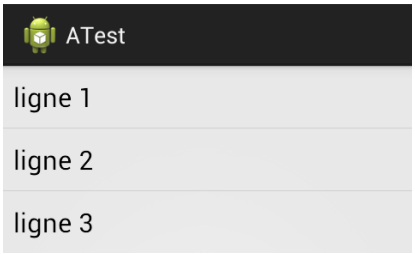
Il faut utiliser une classe Adapter pour remplir les champs.

```
<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    >
</ListView>
```

3 – Les listes

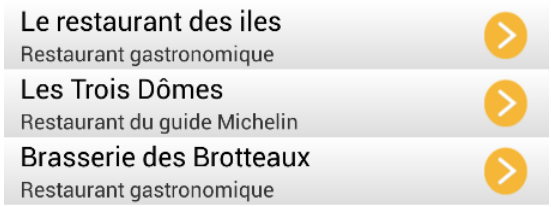
ListView (ArrayAdapter):

```
ListView l = (ListView) findViewById(R.id.listView1);
String[] items={"ligne 1", "ligne 2", "ligne 3"};

l.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, items));
```



3 – Les listes

ListView personnalisé :




```
AdapterRestaurants mAdapter = new AdapterRestaurants(this, R.layout.row_restaurants, restaurants);

getListView().setAdapter(mAdapter);
```

```
getListView().setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int position,
            long arg3) {
        // Lancer action
    }
});
```

3 – Les listes

```
public class AdapterRestaurants extends ArrayAdapter<Restaurant>  {
    private Restaurant item;
    private LayoutInflater mInflater;
    private int mLayoutId;

    public AdapterRestaurants(Context c, int textViewResourceId, List<Restaurant> objects) {
        super(c, textViewResourceId, objects);

        mInflater = LayoutInflater.from(c);
        mLayoutId = textViewResourceId;
    }

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

        final ViewHolder holder;

        // si le convertView est null on declare les variables sinon on le réutilise
        if (convertView == null) {
            convertView = mInflater.inflate(mLayoutId, null);

            holder = new ViewHolder();
            holder.textViewName         = (TextView) convertView.findViewById(R.id.textViewName);
            holder.textViewDescription  = (TextView) convertView.findViewById(R.id.textViewDescription);

            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }

        item = this.getItem(position);

        holder.textViewName.setText(item.getName());
        holder.textViewDescription.setText(item.getDescription());

        return convertView;
    }

    static class ViewHolder {
        TextView textViewName;
        TextView textViewDescription;
    }
}
```

4 – Les menus

Les menus permettent d'accéder à des fonctionnalités sans pour autant prendre de la place sur l'interface.

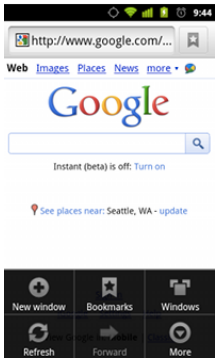
Il existe 3 type de menus :

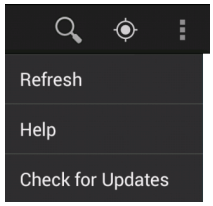
- Options Menu
- Contextual Menu
- Popup Menu

4 – Les menus

Options Menu :

- Afficher quand l'utilisateur presse le bouton menu du téléphone
- Les 6 premiers menus sont affichés. S'il y en a d'autres, un icône nous propose de les afficher.
- Il est possible d'ajouter des sous-menus.
- Les menus sont créés à partir d'une activité dans une fonction « onCreateOptionMenu ».





4 – Les menus

Exemple « Options Menu » :

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

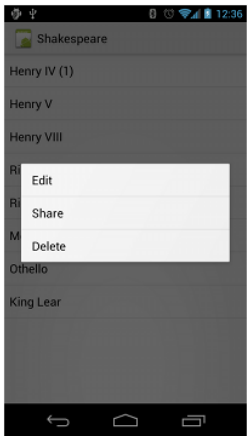
```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.itemAbout:

            return true;
        case R.id.itemQuit:
            finish();
            return true;
    }

    return false;
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/itemAbout"
        android:icon="@android:drawable/ic_menu_help"
        android:title="@string/menu_about">
    </item>
    <item
        android:id="@+id/itemQuit"
        android:icon="@android:drawable/ic_menu_close_clear_cancel"
        android:title="@string/menu_quit">
    </item>

</menu>
```

4 – Les menus

Contextual Menu:

- Il s'affiche sur une boite de dialogue.
- On l'associe directement à un composant graphique avec « registerForContextMenu »
- L'utilisation de ce menu est identique au menu classique « Option Menu »



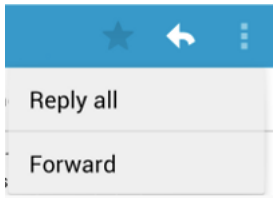
4 – Les menus

Popup Menu:

- Il est attaché à une vue.

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:contentDescription="@string/descr_overflow_button"
    android:onClick="showPopup"
    android:src="@drawable/ic_overflow_holo_dark" />
```

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.activity_main, popup.getMenu());
    popup.show();
}
```



5 – Styles et thèmes

Il est possible de définir un style pour un composant (TextView, Button, EditText ...).

Il faut éditer le fichier « res/values/styles.xml »

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:android="http://schemas.android.com/apk/res/android">

    <style name="aboutText">
        <item name="android:textStyle">normal</item>
        <item name="android:textSize">13dip</item>
        <item name="android:textColor">@color/black</item>
        <item name="android:background">@android:color/transparent</item>
    </style>

</resources>
```

```
<WebView
    android:id="@+id/webViewAbout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginBottom="10dip"
    android:layout_marginLeft="10dip"
    android:layout_marginRight="10dip"
    android:layout_marginTop="10dip"
    android:text="@string/about_text"
    style="@style/aboutText" />
```

(layout xml)

5 – Styles et thèmes

Il est possible de définir un thème pour une activité ou une boite de dialogue (fonctionnement similaire au style).

Il faut éditer le fichier « res/values/themes.xml »

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:android="http://schemas.android.com/apk/res/android">

    <style name="StarAfricaTheme" parent="android:style/Theme.Light.NoTitleBar">
        <item name="android:windowTitleBackgroundStyle">@android:color/white</item>
        <item name="android:background">@android:color/white</item>
    </style>

</resources>
```

```
<application
    android:hardwareAccelerated="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/StarAfricaTheme" >
```

(thème appliqué dans le manifest)

6 – La gestion des langues

Les langues sont récupérées par défaut depuis le fichier :

- « res/values/strings.xml » (en francais)

Pour différencier la langue anglaise ou espagnol, il faut créer les arborescences suivantes :

- « res/values-en/strings.xml »
- « res/values-es/strings.xml »

Il est possible de différencier l'anglais parlé au Etats-Unis ou en Angleterre avec le prefix « r » :

- « res/values-en-rGB/strings.xml »

7 – TP: développement d'une application …

TP « Température » :

- Créer une nouvelle application avec une interface graphique simple pour convertir les degrés Celsius et Fahrenheit
- Application disponible en Français / Anglais
- Ajouter un bouton « sauvegarder ».
- Utiliser les listes avec la possibilité de supprimer le contenu d'une ligne.
- Utiliser le menu pour effacer le contenu de la liste.

7 – TP: développement d'une application …

TP « Temperature » :

- [°C] = 5/9 x ([°F] – 32)
- [°F] = (9/5 * °C) + 32