Android Composants, Layout & Menu

Rémi Forax

Les composants graphiques et gestion d'évènement

Android.widget.View

Classe de base de tous les composants graphiques d'Android

regroupe l'affichage ainsi que la gestion des évènements liés à la zone d'affichage en un seul objet

View en Java

id calculé à partir d'un nom dans R.java

Depuis une activité :

- Activity.setContentView(int)

  demande l'affichage de la hierarchie à partir d'un layout/foo.xml
- View context.findViewById(int)

  demande d'un composant **après** création hiérarchie

```
public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    EditText editText = (EditText)findByView(R.layout.name);
    ...
}
```

Listener d'évènements

Il existe 3 façons de récupérer un évènement suite à une actions d'un utilisateur

- Enregistrement d'un listener avec setOn*XXX*Listener(*XXX*Listener)
- Dans le XML, propriété android:onClick dans layout/foo_activity.xml

- Mais en aucun cas, on ne redéfinie la méthode on*XXX*() du composant (abus d'héritage !)

Listener en code

On s'enregistre pour être prévénu ultérieurement

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

    Button button = (Button)findViewById(R.id.signon);
    button.setOnClickListener(new ButtonOnClickListener(button));
}

static class ButtonOnClickListener implements View.OnClickListener {
    private final Button button;

    ButtonOnClickListener(Button button) { this.button = button; }

    @Override
    public void onClick(View v) {
        Log.e(MAIN_ACTIVITY, button.getId()+" is clicked");
    }
}
```

Interface interne

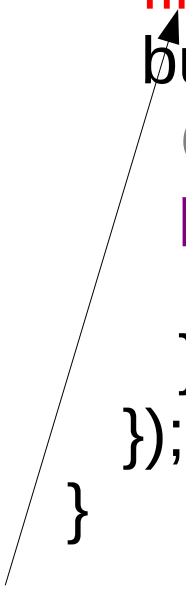
android.util.Log, apparait dans le logCat

Listener en code

On utilise une classe annonyme

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

    final Button button = (Button)findViewById(R.id.signon);
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Log.e(MAIN_ACTIVITY, button.getId()+" is clicked");
      }
   });
}
```



On doit déclarer la variable local final car on est pas en 1.8 :(

Activité et champ

Ce code est **idiot**, il n'est pas nécessaire déclarer les composants graphiques en tant que champ de l'activité !

```
public class DemoActivity extends Activity {
    private Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = (Button)findViewById(R.id.signon);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.e(MAIN_ACTIVITY, button.getId()+" is clicked");
            }
        });
    }
}
```

devrait être dans une variable locale

Listener en XML

On utilise l'attribut android:onClick, l'appel se fait par réflection

```
<Button
    android:id="@+id/signon"
    ...
    android:onClick="signOnClicked" />


public class MainActivity extends Activity {
  ...
  public void signOnClicked(View view) {
    Button button = (Button)view;
    Log.e(MAIN_ACTIVITY, button.getId()+" is clicked");
  }
}
```

Interception global d'évènements

Pratique pour débugger

- Activity.dispatch*XXX*Event(*XXX*Event) :
  dispatch de l'évènement de l'activité vers la vue concernée (*XXX*={GenericMotion | Key | KeyShortcut |PopulateAccessiblity |Touch | Trackball})
- ViewGroup.onIntercept*XXX*Event(*XXX*Event) et ViewParent.requestDisallowInterceptXXXEvent(MotionEvent) :
  vol d'évènement par la vue parent (*XXX*={Touch | Hover})

View basique

Éléments de formulaire

- TextView : affiche une chaîne
- EditText : permet la saisie d'une chaîne (propriété inputType pour le type d'entrée attendu)
- Button : bouton cliquable, variante de type interrupteur avec ToggleButton
- CheckBox : case à cocher
- RadioButton : bouton radio regroupable dans un RadioGroup
- CheckedTextView : chaîne cochable (implante Checkable)
- ProgressBar : barre de progression (horizontale, circulaire), variante avec étoiles de notation avec RatingBar
- SeekBar : barre de réglage
- SearchView : champ de recherche avec proposition de suggestions

Éléments multimédias

- ImageView : affichage d'une ressource image
- ImageButton : bouton avec image
- VideoView : affichage contrôlable de vidéo

Boutons

Bouton simple:

```
<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button1_text"
    ... />
```

Bouton avec une image à gauche (drawableLeft)

```
<Button android:id="@+id/my_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/button1_text"
  android:drawableLeft="@drawable/button1_icon"
   ... />
```

CheckBox

- Case à cocher

```
<CheckBox android:id="@+id/my_checkbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/a_text"/>
```

RadioButton

On regroupe les RadioButton dans un ButtonGroup (layout)

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
  <RadioButton android:id="@+id/radio1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/rhino"
   <RadioButton android:id="@+id/radio2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/elephant"
</RadioGroup>
```

RadioButton et Evènements

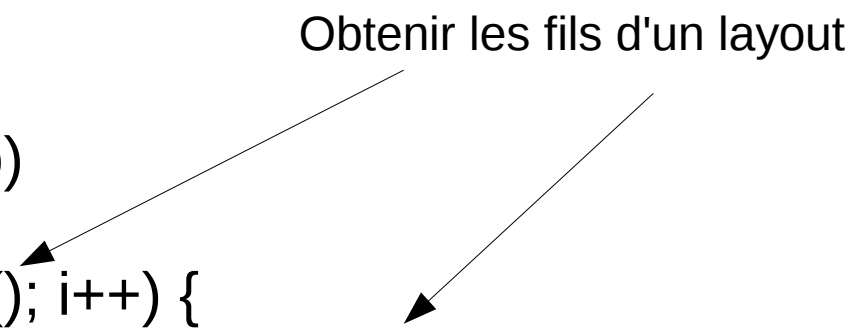
Comme il est rare d'avoir un seul RadioButton, créer autant de OnClickListener est idiot

- Soit on créé 1 seul listener que l'on enregistre plusieurs fois
- Soit on utilise onClick avec le même nom de méthode pour tous les RadioButton

```
<RadioGroup ...>
  <RadioButton android:id="@+id/radio1"
     onClick="radioClicked"/>
   <RadioButton android:id="@+id/radio2"
     onClick="radioClicked"
</RadioGroup>
```

1 seul OnCLickListener

```
public class Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    OnClickListener listener = new OnClickListener() {
      public void onClick(View view) {
        // ...
      }
    };
    RadioGroup group = (RadioGroup)
        findViewById(R.id.radioGroup);
    for(int i=0; i<group.getChildCount(); i++) {
      RadioButton radio = (RadioButton)group.getChildAt(i);
      radio.addOnClickListener(listener);
    }
  }
}
```



1 méthode onClick dans le XML

```
<RadioGroup ...>
  <RadioButton android:id="@+id/radio1"
      onClick="radioClicked"/>
...
public class Activity {
  ...
  public void radioClicked(View view) {
    RadioButton radio = (RadioButton)view;
    Log.e("my activity", radio.getId()+" is clicked");
  }
}
```

EditText

Champs texte editable par l'utilisateur

attributs XML les + fréquents:

- android:ems, taille en caractère (adapté à la fonte)
- android:text, texte
- android:hint, conseil qui sera affiché si pas de texte
- android:inputType, type valeur attendu, textPassword, textEmailAddress, date, time, phone

```
<EditText
  android:id="@+id/password"
  android:ems="16"
  android:hint="password"
  android:inputType="textPassword"/>
```

Spinner

Appelé aussi combo-box

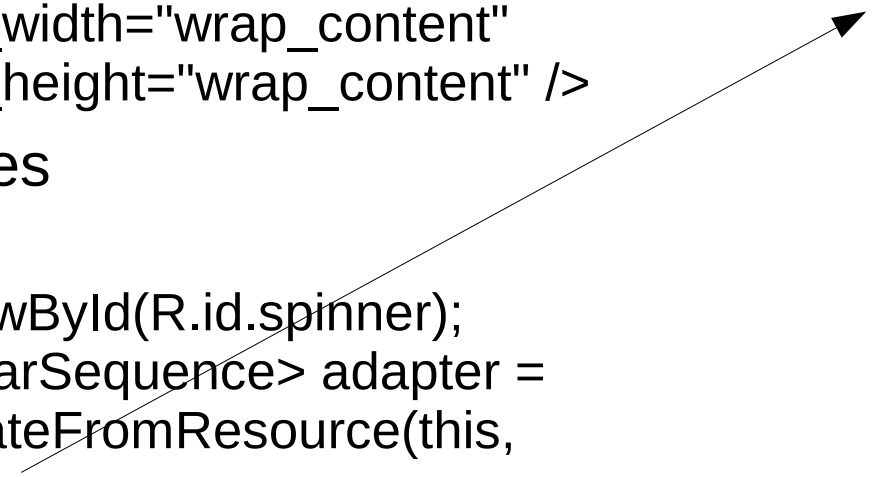
- Déclarer

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

- Ajouter les données

```
Spinner spinner =
  (Spinner)findViewById(R.id.spinner);
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
   R.array.my_array,
   android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
   android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

```
<resources>
   <string-array name="my_array">
      <item>elem1</item>
      <item>elem2</item>
   </string-array>
</resources>
```



WebView

View permettant d'afficher du HTML en utilisant le moteur de rendu Webkit

(utilisé par ~~Chromium~~, Safari, Opera)

Configurable par WebView.getSettings()

- setJavaScriptEnabled() pour activer/désactivé le JavaScript
- setPluginState(PluginState.OFF) pour désactiver les plugins
- setAllowContentAccess(false) pour désactiver le chargement d'URL
- setAllowFileAccess(false) pour l'accès au fichiers locaux

Utilisation

- loadData(String data, String mimeType, String encoding) affiche le contenu
- loadUrl(String url) poaur charger le contenu à une URL spécifique

WebView

Interception des clics sur les liens hypertextes en redéfinissant

- shouldOverrideUrlLoading(WebView, String Url)

Possibilité de communication d'un objet Java à un script JavaScript

- addJavaScriptInterface(Object object, String name)

Les Layouts

ViewGroup

Container de vues enfants, gère leur placement

Deux types de ViewGroup prédéfinis

- Agencement statique (*Layout)
  - Il est conseillé d'utilisé un resource layout XML
  - Manipulation des enfants possibles à l'exécution
    - Ajout avec ViewGroup.addView(view), suppression avec removeView(view)
    - Parcourt de la liste des enfants avec getChildCount() et getChildAt(index)
- Agencement dynamique, basé sur un Adapter (modèle du MVC) efficace pour afficher un grand nombre d'élements en recyclant les vues

Les layouts

Vues (LinearLayout, RelativeLayout, etc.) héritant de ViewGroup qui gére le placement de vues filles à l'intérieur du ViewGroup

Chaque layout permet d'associer à une View (fille du Layout) un ensemble de contraintes de placement

Arbre de vues statique doit être définie en XML dans le répertoire ressource layout

res/**layout**/activity_x.xml

Fichier XML décrivant l'interface graphique d'une activité

```
<RelativeLayout
   xmlns:android=”http://schemas.android.com/apk/res/android“
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin">

  <EditText
     android:id="@+id/name"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />
</RelativeLayout>
```

génère moi un id SVP

Les layouts

FrameLayout

place les composants les un au dessus des autres
(un seul est visible à la fois)

LinearLayout

place les composants verticalement (ou horizontalement) les uns derrières les autres

RelativeLayout

place les composants relativement les uns par rapport aux autres

TableLayout

place les composants dans une table (comme en HTML)

GridLayout

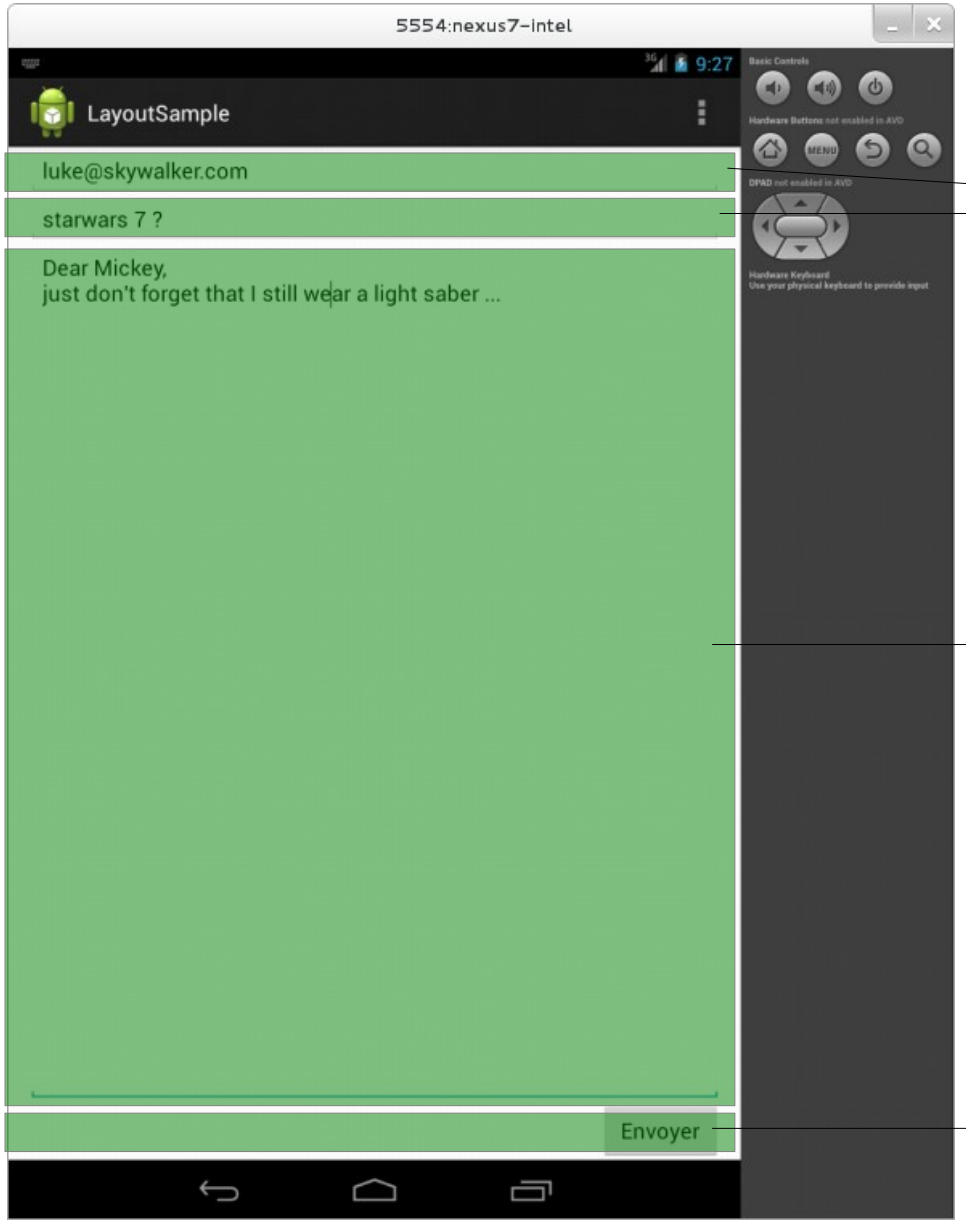
place les composants sur une grille flexible

FrameLayout

Affichage d'une pile de vues avec gestion basique du positionnement

- Paramètre de positionnement FrameLayout.LayoutParams(width, height, gravity)
- Gravity définie l'emplacement de la vue enfant (top, bottom, left, right, fill...)

LinearLayout



layout_width = fill_parent
layout_height = wrap_content

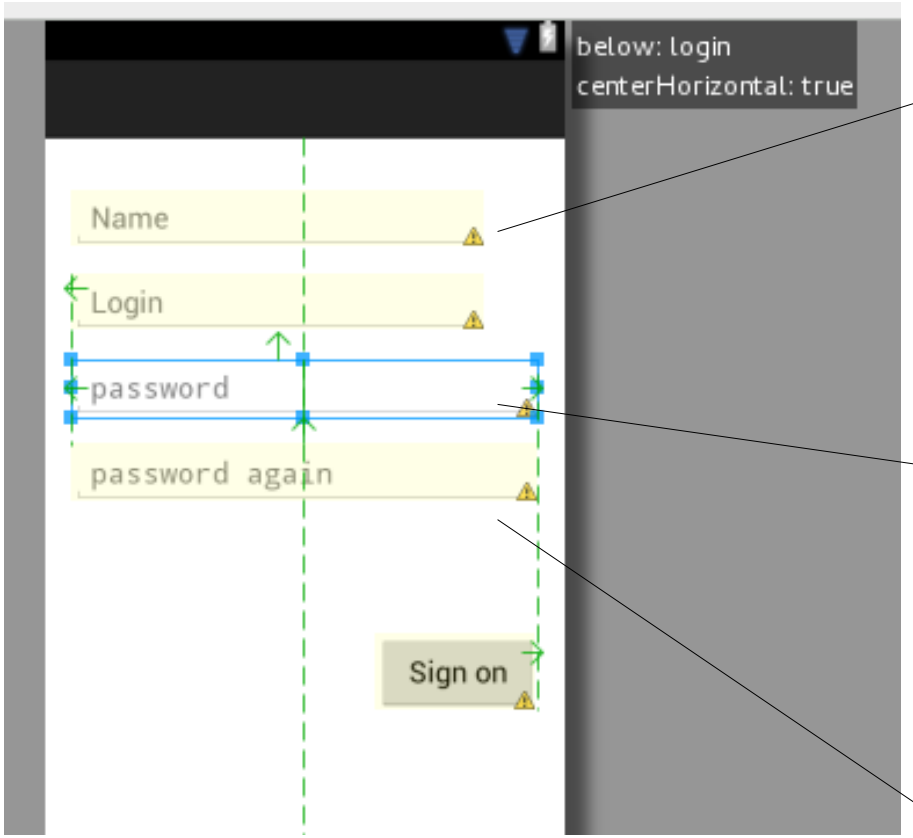
layout_width = fill_parent
layout_height = 0dp
layout_weight = 1
layout_gravity = top

layout_width = 100dp
layout_height = wrap_content
layout_gravity = right

Contraintes du LinearLayout

- width/height: controle l'occupation de la case
  - fill_parent: prend la taille du parent
  - wrap_content: prend la taille du contenu
- weight: poid pour le dimensionnement
- gravity: placement du contenu dans sa case
  - left, top, bottom, right

RelativeLayout



layout_width = wrap_content
layout_height = wrap_content
layout_below = @+id/name
layout_marginTop = 16dp
layout_alignLeft = @+id/passwd

layout_width = wrap_content
layout_height = wrap_content
layout_below = @+id/login
layout_marginTop = 16dp
layout_centerHorizontal = true

layout_width = 100dp
layout_height = wrap_content
layout_below = “@+id/passwd2
layout_marginTop = 80dp
layout_alignRight = @+id/passwd

Contraintes du RelativeLayout

- width/height: controle l'occupation de la case
- below, above: placement relatif à un autre composant (en dessous/au dessus)
- alignLeft, alignRight: alignement relatif entre composants
- marginTop, marginLeft, marginBottom, marginRight: marges autour du composant

TableLayout

Tableau de positionnement des vues en ligne de TableRow

(similaire au <table> <tr> <td> de HTML)

TableRow hérite de LinearLayout avec alignement automatique des colonnes sur chaque ligne

- Propriétés de TableRow.LayoutParams
  - layout_column: indice de départ de la colonne (à partir de 0)
  - layout_span: nombre de colonnes occupées

GridLayout

- Positionnement sur une grille rectangulaire de N colonnes
  - contrairement au TableLayout les vues sont ajouté rectement avec leur paramètres de positionnement
  - Propriétés de GridLayout.LayoutParams
    - layout_column/layout_columnSpan: colonne de départ, nombre de colonnes occupées
    - layout_gravity: emplacement de la vue enfant dans sa zone

Créer son ViewGroup

MyViewGroup

- hérite de ViewGroup
- à une classe interne MyViewGroup.LayoutParams pour les paramètres de placement de chaque vues filles (optionel)
- redéfinie onLayout()

    2 parcours en profondeur

    - Calcul des dimensions souhaitées de chaque vue
    - Calcul de positionnement de chaque vue dans sa zone définie par le ViewGroup

Algorithme de placement

- Calcul des dimensions
  - Appel récursif de View.measure(width,height) pour que chaque vue fille indique sa dimension souhaitée
    - La vue peut redéfinir onMeasure(width,height) et doit appeler setMeasuredDimension(width,height) avec les dimensions souhaitée par la vue
    - Un viewGroup peut appeler plusieurs fois child.measure avec différente taille
- Placement des vues dans leur zone
  - Appel récursif de View.layout(left,top,right,bottom) pour indiquer la zone de placement
    - Un ViewGroup peut redéfinir onLayout(changed,left,top,right,bottom) et placer chaque enfant avec child.set[Left|Top|Right|Bottom](int) en s'aidant de child.getMesured[Width|Height]()

La boucle d'évènement et gestion d'évènements

La boucle d'évènement

La thread principale

- Traite les évèvements clavier, touch, sourie, ...
  - Appel le/les listeners correspondant
- Affiche les composants
  - Recalcul l'agencement des Views et dessine les portion nécessaire de chaque View
- Traite les demandes utilisateurs
  - requestLayout()
    - Les limites de la vue doivent être changées
  - invalidate()
    - L'apparence de la vue doit être changée

Thread & interface graphique

Une **seule** thread gère l'affichage graphique

- Ne pas la bloquer avec de longs calculs ou communications réseau

  sinon **A**pplication **N**ot **R**esponding

- Les autres threads créer par l'utilisateur n'ont pas le droit de modifier l'interface graphique
  (cf cours sur les traitements long)

Evènements courants

void **OnClickListener**.**onClick**(View)

clic tactile, par trackball ou validation

boolean **OnLongClickListener.onLongClick**(View)

clic long (1s)

void **OnFocusChangeListener**.**onFocusChange**(View v, boolean hasFocus)

gain ou perte de focus

boolean **OnKeyListener**.**onKey**(View v, int keyCode, KeyEvent e)

appui sur une touche matérielle

boolean **OnTouchListener**.**onTouch**(View v, MotionEvent e)

dispatch d'un événement de touché (appelé avant le transfert de l'évènement à la vue enfant concernée). Les gestures peuvent être composées de plusieurs MotionEvent.

Valeur de retour boolean : permet d'indiquer si l'évènement a été consommé, i.e. s'il ne doit plus être communiqué à d'autres listeners (de vues parents)

Gestion du focus

Le focus indique la View qui va recevoir les évènements clavier

- Les Views comme EditText sont focusable (isFocusable() renvoie vrai)

  isFocusableInTouchMode() pour les composants non focusable en mode Touch (comme les boutons)

- Ordre du focus est définie avec les propriétés XML nextFocus[Down|Up|Left|Right]

  permet de passer d'un champs de texte à l'autre après validation

- Demande dynamique de focus, View.requestFocus(), View.requestFocusFromTouch()

- Touver le prochain composant

  View.focusSearch(View.FOCUS_[UP¦DOWN|LEFT|RIGHT])

OnTouchListener

- Un seul listener pour un click de souris, touch avec **plusieurs** doigts, etc.

  onTouch(View view, MotionEvent event)

  - MotionEvent.ACTION_DOWN
    - Premier doigt posé
  - MotionEvent.ACTION_POINTER_DOWN
    - Autre doigt posé
  - MotionEvent.ACTION_MOVE
    - Au moins 1 doigt bouge
  - MotionEvent.ACTION_POINTER_UP
    - Un doigt est retiré
  - MotionEvent.ACTION_UP
    - Le dernier doigt est retiré

1 doigt -> 1 pointer

L'écran est capable de “tracker” plusieurs doigts en même temps (1 à 6 au moins)

- MotionEvent contient un tableau avec les coordonnées de chaque doigt
  - event.getPointerCount() indique le nombre de doigts
  - float getX(index), float getY(index)
  - Les doigts sont **pas** toujours dans le **même ordre**
- Chaque doigt posséde le même PointerID tant qu'il est en contact avec la surface
  - event.getPointerId(index) / event.getPointerIndex(pointerId)
- Pour POINTER_[DOWN|UP] e.getActionIndex() donne l'index du doigt ajouté/supprimé

  Attention, de temps en temps, le framework loupe des UP/DOWN

  Performance, **éviter** les **allocations** en cas de MOVE

Gestures

On créé un détecteur de Gesture que l'appel dans onTouch()

gesture simple: GestureDetector, appel onGestureListener

- onDown, onDoubleTap, onLongPress, onFling, onScroll, onShowPress, onSingleTapConfirmed, onSingleTapUp

gesture de zoom: ScaleGestureDetector

- onScaleBegin, onScale, onScaleEnd

Exemple avec onFling

Reconnaitre une swipe gauche à doite ou droite à gauche

```
private static final int SWIPE_MIN_DIFF_X = 100;
private static final int SWIPE_MAX_DIFF_Y = 100;
private static final int SWIPE_MIN_VELOCITY = 70;

@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float
velocityY) {
  if (Math.abs(e1.getY() - e2.getY()) > SWIPE_MAX_DIFF_Y) {
    return false;
  }
  int diffX = e1.getX() - e2.getX() ;
  int velocity =  Math.abs(velocityX);
  if (Math.abs(diffX) > SWIPE_MIN_DIFF_X && velocity > SWIPE_MIN_VELOCITY) {
    // if diff <0 SWIPE right to left, if diff>0 SWIPE left to right
  }
  return false;
}
```

Gestures complexes

Pour reconnaitre des gestes plus complexe comme des lettres de l'alphabet par ex.

- GestureOverlayView est un FrameLayout transparent qui se met devant une autre View pour trapper les Touch events
- OnGesturePerformedListener.onGesturePerformed(GestureOverlayView view, Gesture gesture) est appelée à la fin d'une gesture
- Une GestureLibrary permettre d'avoir un score de confiance par rapport à des Gestures pré-enregistrées

Exemple

```
private final GestureLibrary library = ...

@Override
public void onGesturePerformed(GestureOverlayView overlay,
                               Gesture gesture) {
  List<Prediction> predictions = library.recognize(gesture);
  if (!predictions.isEmpty()) {
    return;
  }

  // take the first one
  Prediction prediction = predictions.get(0);
  if (prediction.score < 1.0) {
    return;
  }

  // do something useful here
}
```

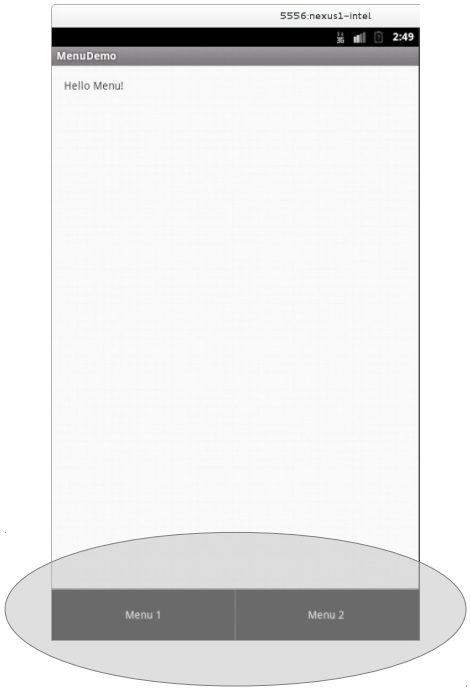
Les Menus

Il y a 3 types de menu

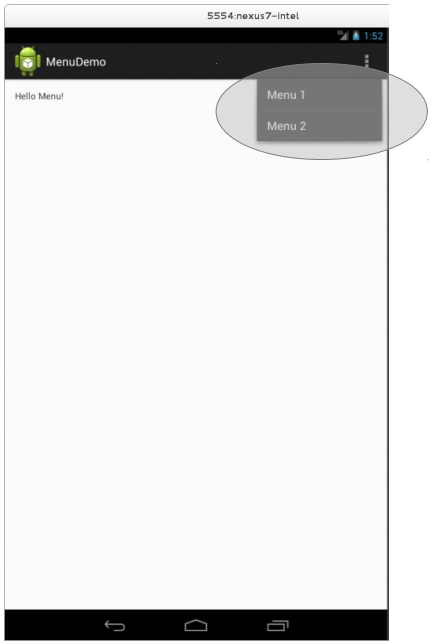
- l'Option Menu
  - pour la navigation dans l'application
- les popups menus
  - pour les actions secondaires
- les menus contextuelles
  - en fonction du contenu touché

L'option menu

Le menu de navigation ou Option Menu est placé à des endroits différents en fonction des versions d'Android



Android 3.2.2 version<11



Android 4.0.1 version>11

Déclaration en XML

Dans le fichier XML res/menu/main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_one"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="Menu 1"/>

    <item
        android:id="@+id/action_two"
        android:orderInCategory="101"
        android:showAsAction="never"
        android:title="Menu 2"/>

</menu>
```

Dans l'activité

Activity.onCreateOptionMenu est appelée avec un Menu, un objet MenuInflater permet de lire le fichier XML et d'ajouter les items au menu

```
public class MainActivity extends Activity {
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Ressource correspondant au fichier res/menu/main.xml

Evènement de sélection d'un Item

Il y a deux façons d'attraper la sélection d'un item

- Redéfinir Activiy.onOptionsItemSelected(MenuItem item)
  - Mais cela oblige à faire un switch item.getItemId() et de comparer avec les ids R.id.xxx
- Récupérer les menu item dans onCreateOptionsMenu et ajouter un

Evènement de sélection d'un Item

Il y a deux façons d'attraper la sélection d'un item

- Redéfinir Activiy.onOptionsItemSelected(MenuItem item)
  - Mais cela oblige à faire un switch item.getItemId() et de comparer avec les ids R.id.xxx
- Récupérer les menu item dans onCreateOptionsMenu et ajouter un OnMenuItemClickListener
  - Mais cela oblige à créer autant de listeners que de menu items

dans onCreateOptionMenu ...

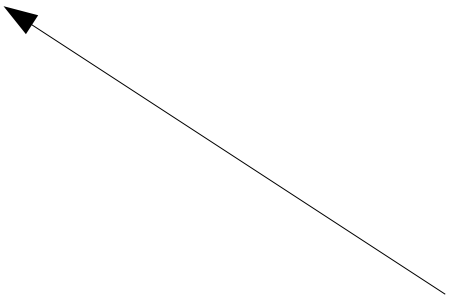
```
public class MainActivity extends Activity {

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
       getMenuInflater().inflate(R.menu.main, menu);

       MenuItem item1 = menu.findItem(R.id.action_one);
       item1.setOnMenuItemClickListener(
         new OnMenuItemClickListener() {
           @Override
           public boolean onMenuItemClick(MenuItem item) {
              // action 1
              return true;
           }
       });
       return true;
    }
```

dans onOptionsItemSelected ...

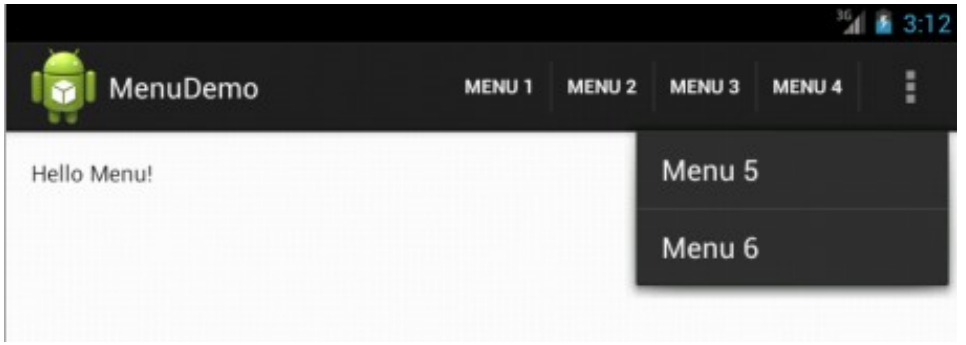
```
public class MainActivity extends Activity {
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
            case R.id.action_two:
                // action 2
                break;
        }
        return false;
    }
...
```



Même si c'est clairement pas la façon la plus jolie d'écrire du code, c'est la façon qui est préféré par Google car cela créé moins d'objets

ActionBar

Présente depuis version 11 (Android 3.0), il est possible de sortir certain menu items pour les rendres directement accessible si il y a de la place (avec showAsAction=ifRoom)



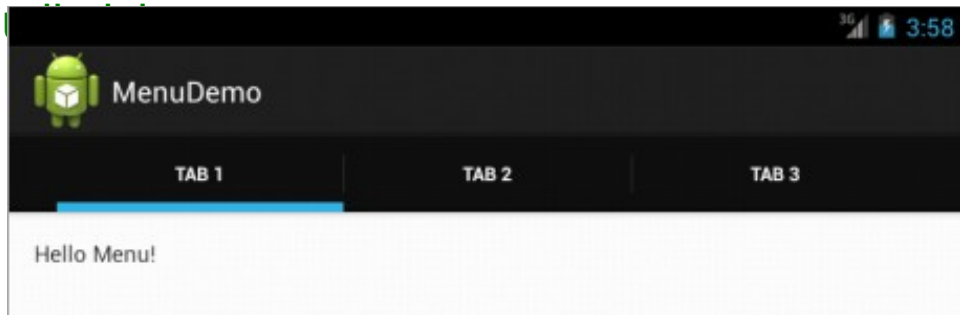

```
<item
    ...
    android:showAsAction="ifRoom"
    ...
    />
```

ActionBar & Onglet

La menuBar permet aussi de définir des onglets (Tab) si on utilise le mode NAVIGATION_MODE_TABS

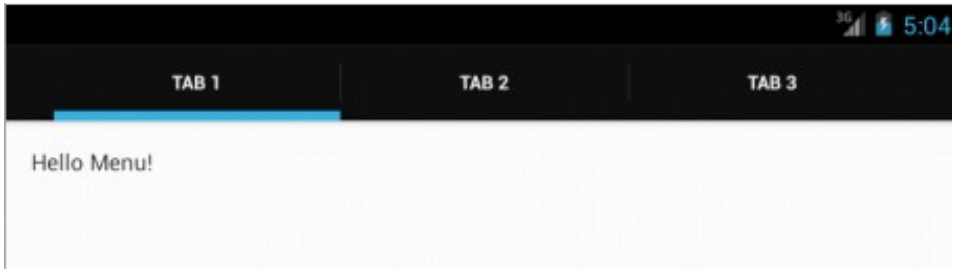
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(NAVIGATION_MODE_TABS);
    ActionBar.Tab tab = actionBar.newTab();
    tab.setText("tab " + i);
    tab.setTabListener(new TabListener() {
        @Override
        public void onTabSelected(Tab tab, FragmentTransaction
ft) {
            // faire quelque chose d'
        }
        ...
    });
    actionBar.addTab(tab);
```



Onglets sans ActionBar

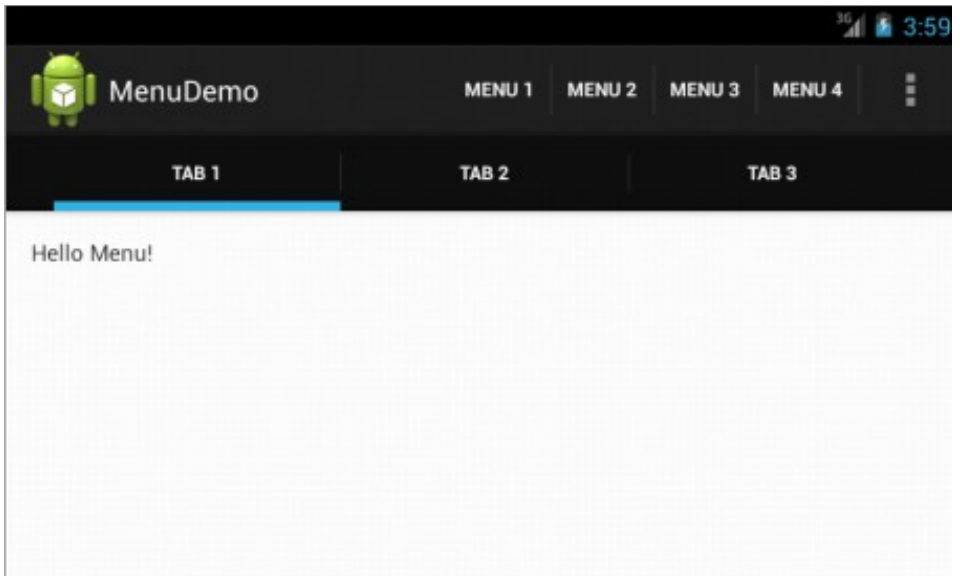
Il est aussi possible de supprimer le titre de l'application (setDisplayShowTitleEnabled) ou l'icône de l'application (setDisplayShowHomeEnabled)




```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(NAVIGATION_MODE_TABS);
    actionBar.setDisplayShowTitleEnabled(false);
    actionBar.setDisplayShowHomeEnabled(false);
    ...
```

Onglets + Menu

On peut aussi mélanger les onglets et les menus



Popup Menu

Un popup menu (version >= 11) est un menu que l'on peut associe à une vue et qui apparait en dessous de celle-ci

On crée un PopupMenu(context, view), puis on utilise la méthode show() pour l'afficher

Il est possible d'utiliser un MenuInflater pour créer les menu items à partir d'un fichier XML

Exemple de Popup Menu


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

    final Button secondaryButton = (Button)findViewById(R.id.secondary);
    secondaryButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PopupMenu popup = new PopupMenu(
                    MainActivity.this, secondaryButton);
            MenuInflater inflater = popup.getMenuInflater();
            inflater.inflate(R.menu.secondary_actions, popup.getMenu());
            popup.show();
        }
    });
```

Menu Contextuel

Deux façons de gérer les menus contextuels

- Floating Context Menu

  Avant la version 11 (Android 3.0), il est lié à l'activité courante et donc global pour toutes les views d'une activité

  - Utiliser pour les listes d'éléments (cf cours suivant)

- Contextual Action Mode

  Version 11 ou plus, il est possible d'avoir plusieurs action mode par activité mais on doit se souvenir si un action mode est en cours d'exécution ou pas

Menu Contextuel

Propose des action typiquement affiché lors d'un clic long

Création en 3 temps

- Ajout d'un champ actionMode dans l'activité
- Création d'un ActionMode.CallBack
  - boolean onCreationActionMode(ActionMode mode, Menu menu)
    on rempli avec un MenuInflater par exemple
  - boolean onPrepareActionMode(ActionMode mode, Menu menu)
    appelé juste avant l'affichage
  - boolean onActionItemClicked(ActionMode mode, MenuItem item)
    il est possible de sortir du mode menu avec mode.finish()
  - boolean onDestroyActionMode(ActionMode mode)
    lorsque l'on sort du menu, on **met** actionMode à **null**
- Enregisterement d'un listener sur un click long

```
myView.setOnLongClickListener(new OnLongClickListener() {
    public boolean onLongClick(View view) {
        if (actionMode != null) return false;
        actionMode = MainActivity.this.startActionMode(callback);
        return true;
    }
}
```