**Les vues**

**Les boîtes de dialogue**

#### Généralités

Les boîtes de dialogue héritent de la classe [Dialog](http://developer.android.com/reference/android/app/Dialog.html) et se trouvent dans le package android.app.Dialog.

Une boîte de dialogue conserve le focus jusqu'à ce que l'utilisateur la ferme.

Dialog est la classe de base pour les dialogues. En général, on utilise ses classes dérivées :

* AlertDialog,
* ProgressDialog,
* DatePickerDialog
* TimePickerDialog.

***AlertDialog***

Il s'agit de la boîte de dialogue la plus polyvalente. Package : android.app.AlertDialog.

Pour construire une AlertDialog on utilise :

* Le constructeur de la classe AlertDialog est la classe AlertDialog.Builder qui permet de simplifier la construction. Ce constructeur prend en argument un Context.

builder = **new** AlertDialog.Builder(**this**);

Un objet de type AlertDialog.Builder possède les méthodes suivantes :

* setCancelable (boolean cancelable) : si le paramètre cancelable vaut true, alors on pourra sortir de la boîte avec le bouton **retour de l'appareil**.
* setIcon (int ressource) ou setIcon (Drawable icon) : Permet d'ajouter une icône à la boîte de dialogue.
* setMessage (int ressource) ou setMessage (String message).
* setTitle (int ressource) ou setTitle (String title).
* setView (View view) ou setView (int ressource): il s'agit de l'équivalent de setContentView pour un objet de type Context.
* setPositiveButton(), setNeutralButton() et setNegativeButton()

Ces méthodes permettent d’indiquer les boutons qui apparaîtront en bas de la boîte de dialogue, leur emplacement latéral (à gauche, au centre ou à droite), leur texte et le code qui sera appelé lorsqu’on clique sur un bouton.

**Exemple**

**public** **class** MainActivity **extends** Activity {

Button b;

Builder builder;

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

b = (Button) findViewById(R.id.*b*);

b.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

builder = **new** AlertDialog.Builder(**this**);

builder.setTitle("Contact Information");

builder.setMessage("Le telephone est : ");

builder.setPositiveButton("OK", **null**);

builder.show();

}

});

}

On peut ensuite ajouter des boutons avec les méthodes suivantes :

* setPositiveButton (text, DialogInterface.OnClickListener listener), avec text une ressource de type String ou une String, et listener définit que faire en cas de clic.
* setNegativeButton (text, DialogInterface.OnClickListener listener). Ce bouton se trouvera tout à droite.
* setNeutralButton (text, DialogInterface.OnClickListener listener). Ce bouton se trouvera entre les deux autres boutons.

**Exemple**

**public** **class** MainActivity **extends** Activity {

Button b;

**private** **int** compteur = 0;

TextView tv;

AlertDialog.Builder builder;

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

tv=(TextView)findViewById(R.id.*text*);

b = (Button) findViewById(R.id.*b1*);

b.setOnClickListener(**new** OnClickListener() {

**public** **void** onClick(View v) {

openAlert(v);

}});

}

**private** **void** openAlert(View view) {

AlertDialog.Builder ad = **new** AlertDialog.Builder(MainActivity.**this**);

ad.setTitle(**this**.getTitle()+ " decision");

ad.setMessage("Are you sure?");

ad.setPositiveButton("Yes",**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,**int** id) {

Toast.*makeText*(getApplicationContext(),"You chose a positive answer", Toast.*LENGTH\_LONG*).show();

} });

ad.setNegativeButton("No",**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,**int** id) {

dialog.cancel(); //finish();

Toast.*makeText*(getApplicationContext(), "You chose a negative answer",

Toast.*LENGTH\_LONG*).show();

} });

ad.setNeutralButton("Exit the app",**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,**int** id) {

MainActivity.**this**.finish();

} });

AlertDialog alertDialog = ad.create();

alertDialog.show();

}}

**Choix de date et d’heure : calendar**

Pour utiliser les dates, il suffit de récupérer un objet de type [Calendar](http://developer.android.com/reference/java/util/Calendar.html) à l'aide de la méthode Calendar.getInstance(). Cette méthode retourne un Calendar qui contient les informations sur la date et l'heure.

Ensuite on peut récupérer des informations avec la méthode int get(int champ) avec champ pouvant prendre des valeurs telle que :

* Calendar.YEAR pour l'année ;
* Calendar.MONTH pour le mois. Attention, le premier mois est de rang 0, alors que le premier jour du mois est de rang 1 !
* Calendar.DAY\_OF\_MONTH pour le jour dans le mois ;
* Calendar.HOUR\_OF\_DAY pour l'heure ;
* Calendar.MINUTE pour les minutes ;
* Calendar.SECOND pour les secondes.

Calendar calendrier = Calendar.getInstance();// Contient la date et l'heure au moment de sa création

int mois = calendrier.get(Calendar.MONTH);

**Calendar view**

Le widget calendar view est utilisé pour afficher et sélectionner des dates.

La plage de dates prise en charge de ce calendrier est configurable. L'utilisateur peut sélectionner une date en cliquant dessus et peut également faire défiler et trouver le calendrier à une date souhaitée. On peut également définir les dates minimale et maximale.

<CalendarView

android:id="@+id/simpleCalendarView"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" />

#### ****Important Methods Of Calendar View:****

**1. getDate():** Cette méthode permet d’obtenir la date sélectionnée de CalendarView en millisecondes depuis le 1er janvier 1970 à 00:00:00, Cette méthode retourne une valeur de type long pour la date sélectionnée.

CalendarView scv = (CalendarView) findViewById(R.id.simpleCalendarView);

long d = scv.getDate(); // get selected date in milliseconds

**2.setDate(long date):** cette méthode est utilisée pour mettre la date sélectionnée en milliseconds depuis January 1, 1970 00:00:00.

setDate(1463918226920L); // set selected date 22 May 2016 in milliseconds

**3. setFirstDayOfWeek(int firstDayOfWeek):** cette methode est utilisée pour fixer le premier jour de la semaine.

**4. getFirstDayOfWeek():**

**5. setMaxDate(long maxDate):**

**6. setMinDate(long minDate):**

**Exemple**

Calendar calendrier = Calendar.getInstance();

calendrier.set(Calendar.DAY\_OF\_MONTH, 12);

calendrier.set(Calendar.YEAR, 2016);

calendrier.set(Calendar.MONTH, Calendar.MARCH);

calendarView.setDate(calendrier.getTimeInMillis(), false, false);

En xml **android:maxDate**et**android:minDate sont** utilisés pour définir une plage personnalisée sur le calendrier. Les dates spécifiées sont du format MM/dd/yyyy.

On peut utiliser les attributs **android:dateTextAppearance**et**android:weekTextAppearance** pour changer de style.

L'écouteur est **calendarView.setOnDateChangeListener**

calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {

@Override

public void onSelectedDayChange (CalendarView view, int year, int month, int dayOfMonth) {

….

} });

}

Cela se déclenche chaque fois que la date est modifiée par l'utilisateur.

**Exemple**

<CalendarView

android:id="@+id/calendarView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

public class MainActivity extends AppCompatActivity {

CalendarView calendarView;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

calendarView = findViewById(R.id.calendarView);

calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {

@Override

public void onSelectedDayChange(CalendarView view, int year, int month, int dayOfMonth) {

Toast.makeText(getApplicationContext(), dayOfMonth + "/" + month + "/" + year, Toast.LENGTH\_LONG).show();

} }); }

**• DatePicker : **

**Propriétés**

android:startYear : Pour définir l'année de départ du calendrier affiché

android:endYear : Pour définir l'année de fin du calendrier affiché

android:minDate : Pour définir la date affichée de départ du calendrier sous la forme mm/jj/aaaa

android:maxDate : Pour définir la date affichée de fin du calendrier sous la forme mm/jj/aaaa

|  |  |  |  |
| --- | --- | --- | --- |
| **Evenement** | **Interface** | **Méthode** | **Méthodes à redéfinir** |
| Choix | DatePicker.OnDateChangedListener | setOnDateChangedListener | onDateChanged(DatePicker,  int, int, int)  • Elément concerné  • Année  • Mois  • Jour |
| Choix | TimePicker.OnTimeChangedListener | setOnTimeChangedListener | onTimeChanged(TimePicker, int, int)  • Elément concerné  • Heure  • Minutes |

**La méthode init :**

Pour initialiser le widget on utilise la méthode init.

void init(int annee, int mois, int jour\_dans\_le\_mois, DatePicker.OnDateChangedListener listener\_en\_cas\_de\_change\_de\_date)

public void init (int year, int monthOfYear, int dayOfMonth, [DatePicker.OnDateChangedListener](http://developer.android.com/reference/android/widget/DatePicker.OnDateChangedListener.html) onDateChangedListener)

##### Parameters

|  |  |
| --- | --- |
| *year* | The initial year. |
| *monthOfYear* | The initial month **starting from zero**. |
| *dayOfMonth* | The initial day of the month. |
| *onDateChangedListener* | Context, peut être null. |

On peut récupérer l'année avec int getYear(), le mois avec int getMonth() et le jour avec int getDayOfMonth().

**Exemple**

<DatePicker

android:id="@+id/datePicker1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:startYear="2012"

android:endYear="2032" />

public void onClick(View v) {

simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker);

String day = "Day = " + simpleDatePicker.getDayOfMonth();

String month = "Month = " + (simpleDatePicker.getMonth() + 1);

String year = "Year = " + simpleDatePicker.getYear();

Toast.makeText(getApplicationContext(), day + "\n" + month + "\n" + year, Toast.LENGTH\_LONG).show();

}

**code java**

**public** **class** MainActivity **extends** Activity **implements** OnDateChangedListener{

TextView tv;

DatePicker dp;

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

tv=(TextView)findViewById(R.id.*tv1*);

dp=(DatePicker)findViewById(R.id. datePicker1);

dp.init(dp.getYear(), dp.getMonth(), dp.getDayOfMonth(),**this**);

}

**public** **void** onDateChanged(DatePicker arg0, **int** arg1, **int** arg2, **int** arg3) {

tv.setText(arg3+"/"+arg2+"/"+arg1);

}

**• TimePicker : **

Il est possible de définir l'heure avec void setCurrentHour(Integer hour), de la récupérer avec Integer getCurrentHour(), et de définir les minutes avec void setCurrentMinute(Integer minute), puis de les récupérer avec Integer getCurrentMinute().

Pour activer 24h on utilise la méthode

void setIs24HourView(Boolean mettre\_en\_format\_24h).

**Code xml**

<TimePicker

android:id="@+id/timePicker1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true" />

**code java**

**public** **class** MainActivity **extends** Activity **implements** OnTimeChangedListener{

TimePicker t;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

t = (TimePicker) findViewById(R.id.*timePicker1*);

t.setIs24HourView(**true**);

t.setOnTimeChangedListener(**this**);

}

**public** **void** onTimeChanged(TimePicker view, **int** hourOfDay, **int** minute) {

Toast.*makeText*(**this**, " il est " + t.getCurrentHour() + ":" + t.getCurrentMinute(), Toast.*LENGTH\_SHORT*).show();

}

**• DatePickerDialog : **

Les classes DatePickerDialog et TimePickerDialog ont respectivement les méthodes de callback onDateSetListener() et onTimeSetListener(). Ces méthodes sont appelées lorsque l'utilisateur a fini de renseigner la date ou l'heure.

public DatePickerDialog ([Context](https://developer.android.com/reference/android/content/Context.html) context)

Crée un nouveau sélecteur de date pour la date courante en utilisant le contexte parent.

public DatePickerDialog ([Context](https://developer.android.com/reference/android/content/Context.html) context,

[DatePickerDialog.OnDateSetListener](https://developer.android.com/reference/android/app/DatePickerDialog.OnDateSetListener.html) listener,

int year,

int month,

int dayOfMonth)

DatePickerDialog.OnDateSetListener: l'écouteur à appeler lorsque l'utilisateur définit la date

**Exemple**

Button btnDatePicker;

EditText txtDate, txtTime;

private int mYear, mMonth, mDay, mHour, mMinute;

@Override

public void onClick(View v) {

// Get Current Date

final Calendar c = Calendar.getInstance();

mYear = c.get(Calendar.YEAR);

mMonth = c.get(Calendar.MONTH);

mDay = c.get(Calendar.DAY\_OF\_MONTH);

DatePickerDialog datePickerDialog = new DatePickerDialog(this,

new DatePickerDialog.OnDateSetListener() {

@Override

public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

txtDate.setText(dayOfMonth + "-" + (monthOfYear + 1) + "-" + year);

}

}, mYear, mMonth, mDay);

datePickerDialog.show();

}

##### **Horloges et Chronomètres :**

• AnalogClock, DigitalClock, et Chronometer

**Exemple**

<AnalogClock

android:id=*"@+id/analogClock1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentTop=*"true"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginTop=*"22dp"* />

<DigitalClock

android:id=*"@+id/digitalClock1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@+id/analogClock1"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginTop=*"81dp"*

android:text=*"DigitalClock"* />

|  |  |
| --- | --- |
| * **ProgressBar** |  |

Les **ProgressBar** sont utilisées pour la progression des taches.

• Deux comportements selon que l’on connaît ou pas la valeur maximale

android:indeterminate : Pour définir le type de progressBar (true=indéterminé, false=déterminé).

**• Animation (si indéterminé)**

android:indeterminateBehavior="i" (où i peut être : repeat ou cycle) : définit le comportement de l'animation pour le type circulaire (repeat=recommence l'animation, cycle=changer le sens de l'animation)

**• Dimensions**

android:maxHeight="unité"

android:minHeight="unité"

android:maxWidth="unité"

android:minWidth="unité"

**• Valeurs (si déterminé)**

android:max Pour définir la valeur maximale

android:progress Pour définir la valeur initiale

android:secondaryProgress Pour définir une valeur secondaire (par exemple celle d'un buffer comme on le voit sur des vidéos en streaming.

**Forme des ProgressBar**

• En l’absence du paramètre style, la forme est circulaire

• Pour obtenir d’autres formes on doit utiliser le paramètre style :

style="?android:attr/s" où s peut être :

• progressBarStyleHorizontal

• progressBarStyleSmall

• progressBarStyleLarge

**Quelques méthodes**

* setProgress(int progress) : donne le niveau de progress bar
* setSecondaryProgress(int progress)
* setVisibility (int v), ([VISIBLE](http://developer.android.com/reference/android/view/View.html#VISIBLE), [INVISIBLE](http://developer.android.com/reference/android/view/View.html#INVISIBLE), [GONE](http://developer.android.com/reference/android/view/View.html#GONE))

**Exemple**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<ProgressBar android:id="@+id/progress"

style="?android:attr/progressBarStyleHorizontal"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

**Code java**

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

setContentView(R.layout.main);

bar=(ProgressBar)findViewById(R.id.progress);

Bar.setVisibility(View.*VISIBLE*);

Bar.setMax(150);

}

Suite au clic d'un bouton :

**int i=0;**

**public** **void** onClick(View arg0) {

**if** ( i< Bar.getMax() ) {

Bar.setProgress(i);

Bar.setSecondaryProgress(i + 10);

}

i = i + 10;

}

**SeekBar : **

C'est un ProgressBar sous forme de barre horizontale dotée d'un curseur permettant de modifier la valeur si on a choisi android:indeterminate="false" sinon le curseur ne marche pas et la barre bouge sans arrêt.

**RatingBar : **

**• Paramètres :**

android:isIndicator : indique si l'utilisateur peut modifier ou non la valeur (true= non modifiable)

android:numStars : définit le nombre d'étoiles affichées

android:rating : Pour définir la position initiale

android:stepSize Pour définir le pas de progression.

**Les événements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Evenement** | **Interface** | **Méthode** | **méthodes** |
| Curseur  déplacé | SeekBar.OnSeekBarChangeListener | setOnSeekBarChangeListener | onProgressChanged(  SeekBar, int, boolean)  • Elément concerné  • Position du curseur  • Action de l’utilisateur |
| Début de  déplacement | onStartTrackingTouch(  SeekBar)  •Elément concerné |
| Fin de  déplacement | onStopTrackingTouch(  SeekBar)  •Elément concerné |
| **RatingBar**  Valeur  modifiée | RatingBar.OnRatingBarChangeListener | setOnRatingBarChangeListener | onRatingChanged(RatingBar,  float, boolean)  • Elément concerné  • Valeur choisie  • Action de l’utilisateur |
| **Chronometer**  Incrémentation | Chronometer.OnChronometerTickListener | setOnChronometerTickListener | onChronometerTick(  Chronometer)  • Elément concerné |

**Les onglets**

Pour mettre en place des onglets dans une vue, on a besoin des widgets et des conteneurs suivants :

- TabHost : est le conteneur général pour les boutons et les contenus des onglets.

- TabWidget : implémente la ligne des boutons des onglets, qui contient les labels et les icônes.

- FrameLayout : est le conteneur des contenus des onglets : chaque contenu d’onglet est un fils du FrameLayout.

* L’identifiant android:id de TabWidget doit être @android:id/tabs.
* Pour utiliser TabActivity, l’identifiant android:id du TabHost doit être @android:id/tabhost.

**Exemple**

<TabHost android:id=*"@+id/tabhost"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<TabWidget android:id=*"@android:id/tabs"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

/>

<FrameLayout android:id=*"@android:id/tabcontent"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:paddingTop=*"62px"*>

<AnalogClock android:id=*"@+id/tab1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:layout\_centerHorizontal=*"true"*

/>

<Button android:id=*"@+id/tab2"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:text=*"Bouton semi-aleatoire"*

/>

</FrameLayout>

</TabHost>

</LinearLayout>

**Code Java**

Le code Java doit indiquer au TabHost quelles sont les vues qui représentent les contenus des onglets et à quoi doivent ressembler les boutons de ces onglets.

Tout est encapsulé dans des objets TabSpec.

On récupère une instance de TabSpec via la méthode **newTabSpec()** de TabHost, on la rempli puis on l’ajoute au TabHost dans le bon ordre.

### Méthodes TabHost.

* **newTabSpec (String tabName)** : crée un nouvel onglet de TabSpec.
* **addTab (TabSpec tabSpec)** : Ajouter un onglet dans TabHost.
* **setOnTabChangedListener (écouteur OnTabChangeListener)** :  **On Change** : écoutera l'événement de changement d'onglet.
* setup() : est appelée avant de configurer les objets TabSpec.

Les deux méthodes essentielles de TabSpec sont les suivantes :

* setContent(), qui permet d’indiquer le contenu de cet onglet. Généralement, il s’agit de l’identifiant android:id de la vue que l’on veut montrer lorsque l’onglet est choisi.
* setIndicator(), qui permet de fournir le titre du bouton de l’onglet. Cette méthode est surchargée pour permettre de fournir également un objet Drawable représentant l’icône de l’onglet.

tabSpec.setIndicator ("Tab1", getResources().getDrawable(R.type.image));

* setCurrentTab(): Pour choisir l’onglet qui s’affichera initialement

**public** **class** MainActivity **extends** Activity {

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

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

// TabHost tabs=getTabHost();

tabs.setup();

TabHost.TabSpec spec=tabs.newTabSpec("tag1");

spec.setContent(R.id.*tab1*);

spec.setIndicator("Heure");

tabs.addTab(spec);

spec=tabs.newTabSpec("tag2");

spec.setContent(R.id.*tab2*);

spec.setIndicator("Bouton");

tabs.addTab(spec);

tabs.setCurrentTab(0);

}}

# Ecouteur

# android.widget.TabHost.OnTabChangeListener

# public void setOnTabChangedListener ([TabHost.OnTabChangeListener](http://developer.android.com/reference/android/widget/TabHost.OnTabChangeListener.html) )

# void onTabChanged ([String](http://developer.android.com/reference/java/lang/String.html) tabId)

public void onTabChanged(String tabId) {

tabHost.getCurrentTabView().setBackgroundColor(color);

tabHost.getCurrentTabView().setBackgroundDrawable(drawable);

tabHost.getCurrentTabView().setBackgroundResource(resid);

if("tag1".equals(tabId)) {

…

}

if("tag2".equals(tabId)) {

…

}

**Les Menus**

• Deux types

– Menu général de l’activité : apparaît par appui de la touche Menu

– Menu contextuel associé à un élément d’interface

• Contiennent des rubriques sous forme texte et/ou image

• peuvent être décrits par un fichier XML placé dans res/menu (répertoire à créer) de la forme :

**Menu.xml**

<?xml version="1.0" encoding="utf-8"?>

<menu

<item

android:id="@+id/nom\_du\_choix\_1"

android:icon="@drawable/image\_du\_choix\_1"

android:title="@string/texte\_du\_choix\_1" /> …

<item … />

…

</menu>

**Sous menus**

• Chaque élément d’un menu peut proposer des sous menus

• Décrits dans le fichier XML sous la forme :

<item

android:id="@+id/nom\_du\_choix\_N"

android:icon="@drawable/image\_du\_choix\_N"

android:title="@string/texte\_du\_choix\_N">

**<menu>**

**android:id="@+id/nom\_du\_sous\_choix\_1"**

**android:title="texte\_du\_sous\_choix\_1" />**

**…**

**</menu>**

</item>

**Menu général**

• Apparaît par appui de la touche Menu

• Création dans la méthode onCreateOptionsMenu de l'activité à partir du fichier xml de description du menu sous la forme :

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.*nom\_du\_fichier\_xml\_du\_menu*, menu);

//menu.add("menu1");

//menu.add("menu2");

return true;

}

**Réactions aux choix**

• Dans la méthode onOptionsItemSelected de l'activité qui est appelée lorsque l'utilisateur fait un choix dans un menu ou un sous menu général :

public boolean onOptionsItemSelected(MenuItem item) {

 // Toast.makeText(this, item.getTitle(), Toast.LENGTH\_SHORT).show();

switch (item.getItemId()) {

case R.id.nom\_du\_choix\_1:

// traitement du choix 1

return true;

… case R.id.nom\_du\_sous\_choix\_1:

// traitement du sous choix 1

return true;

… default: return super.onOptionsItemSelected(item);

}}

**Menu contextuel**

• Apparaît par appui long sur l’élément d’interface

• Associé à l’élément d’interface par la méthode : registerForContextMenu(element\_associe\_au\_menu\_contextuel);

• Création dans la méthode onCreateContextMenu de l'activité :

public void onCreateContextMenu(ContextMenu menu, View element, ContextMenuInfo info) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.*nom\_du\_fichier\_xml\_du\_menu*, menu);

}

**Réactions aux choix**

• Dans la méthode onContextItemSelected de l'activité qui est appelée lorsque l'utilisateur fait un choix dans un menu ou un sous menu contextuel :

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.*nom\_du\_choix\_1*:

// traitement du choix 1

return true;

… case R.id.*nom\_du\_sous\_choix\_1*:

// traitement du sous choix 1

return true;

… default: return super.onContextItemSelected(item);

}

}

Pour ajouter les items par java :

**public** **void** **onCreateContextMenu**(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

**super**.onCreateContextMenu(menu, v, menuInfo);

menu.setHeaderTitle("Context Menu");

menu.add(**0**, v.getId(), **0**, "Action 1");

menu.add(**0**, v.getId(), **0**, "Action 2");

menu.add(**0**, v.getId(), **0**, "Action 3");

}

**public** **boolean** **onContextItemSelected**(MenuItem item) {

**if** (item.getTitle() == "Action 1") {

Toast.makeText(**this**, "Action 1 invoked", Toast.LENGTH\_SHORT).show();

}

**else** **if** (item.getTitle() == "Action 2") {

oast.makeText(**this**, "Action 2 invoked", Toast.LENGTH\_SHORT).show();

}

**else** **if** (item.getTitle() == "Action 3") {

Toast.makeText(**this**, "Action 3 invoked", Toast.LENGTH\_SHORT).show();

}

**else** {

**return** **false**;

}

**return** **true**;

}

Le mode d'action contextuelle est une implémentation système d'ActionMode qui concentre l'interaction de l'utilisateur sur l'exécution d'actions contextuelles. Lorsqu'un utilisateur active ce mode en sélectionnant un élément, une barre d'actions contextuelles apparaît en haut de l'écran pour présenter les actions que l'utilisateur peut effectuer sur les éléments actuellement sélectionnés. Lorsque ce mode est activé, l'utilisateur peut sélectionner plusieurs éléments (si vous le permettez), désélectionner des éléments et continuer à naviguer dans l'activité (autant que vous le souhaitez). Le mode d'action est désactivé et la barre d'action contextuelle disparaît lorsque l'utilisateur désélectionne tous les éléments, appuie sur le bouton RETOUR ou sélectionne l'action Terminé sur le côté gauche de la barre.

Popup menu

Menu par xml :

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/one"

android:title="One"/>

<item

android:id="@+id/two"

android:title="Two"/>

<item

android:id="@+id/three"

android:title="Three"/>

</menu>

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) {

//Creating the instance of PopupMenu

PopupMenu popup = new PopupMenu(MainActivity.this, button1);

//Inflating the Popup using xml file

popup.getMenuInflater().inflate(R.menu.popup\_menu, popup.getMenu());

//registering popup with OnMenuItemClickListener

popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {

public boolean onMenuItemClick(MenuItem item) {

Toast.makeText(

MainActivity.this,

"You Clicked : " + item.getTitle(),

Toast.LENGTH\_SHORT

).show();

return true;

}

});

popup.show(); //showing popup menu

} });

}}

Pour ajouter les items par java :

PopupMenu menu = new PopupMenu(this, view);

menu.getMenu().add("One");

menu.getMenu().add("Two");

menu.getMenu().add("Three");

menu.show();