MCours.com

Cours 2 : programmation Android

Plan

- Général
  - Log
  - Listeners
- Démarrer une activité
  - Intent
  - startActivity, startActivityForResult
- InIterface graphiques avancées
  - ListView
  - Menu
- Programmation réseau
  - WebView
  - Socket

Les log

- Class Log
- Affiche les messages dans le logcat de façon structurée
  - Id, estampille, tag personnalisé, message
- Niveaux de verbosité
  - Verbose, debug, info, …
- Methodes statiques de la class Log
  - Log.XXX
  - XXX=i,e,d,…
- Exemples
  - Log.d("MyActivity" , "démarrage de l'activité")
  - Log.e("MyActivity" , "Erreur")

Les listeners personnalisés

- On a déjà une méthode pour réagir à un événement en spécifiant le nom de la méthode à appeler dans l'attribut onClick de la View correspondante
  - onClick="afficheResultat"
- Il faut implémenter la méthode dans l'activité qui possède la vue correspondante
  - public void afficheResultat(View v){…}
- Filtrage par nom de méthode
  - Noms de méthodes unique dans l'activité associée
- Il existe une autre méthode pour associer un listener à une vue
  - Directement dans le code java

InputEvent et Listener

- Une autre approche est de d'abonner l'activité à des évéments spécifiques.
  - Listener = Observer design pattern (conception OO)
  - http://www.vogella.com/articles/DesignPatternObserver/article.html
- Méthodes de call-back pour InputEvent
  1. Etendre l'interface OnXXXListener correspondante par la classe qui intercepte l'évt
  2. La classe doit donc surcharger la méthode correspondante onXXX(…)
  3. La méthode est appelée par le framework Android quand une action correspondante se produit sur l'objet.
- Exemple
  - Quand une vue est touchée par l'utilisateur, la méthode onTouchEvent() method est appelée sur cet objet.

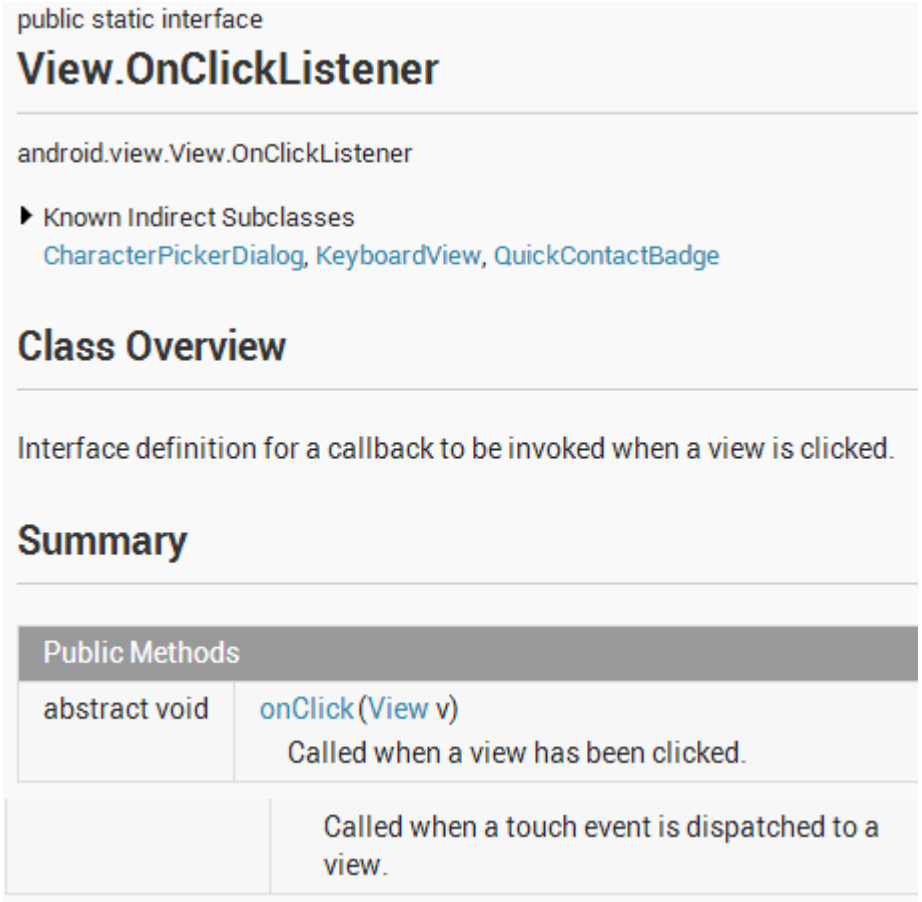
Liste des interfaces de Listener graphique

- http://developer.android.com/reference/android/view/View.html

interface	View.OnAttachStateChangeListener	Interface definition for a callback to be invoked when this view is attached or detached from its window.
interface	View.OnClickListener	Interface definition for a callback to be invoked when a view is clicked.
interface	View.OnCreateContextMenuListener	Interface definition for a callback to be invoked when the context menu for this view is being built.
interface	View.OnDragListener	Interface definition for a callback to be invoked when a drag is being dispatched to this view.
interface	View.OnFocusChangeListener	Interface definition for a callback to be invoked when the focus state of a view changed.
interface	View.OnGenericMotionListener	Interface definition for a callback to be invoked when a generic motion event is dispatched to this view.
interface	View.OnHoverListener	Interface definition for a callback to be invoked when a hover event is dispatched to this view.
interface	View.OnKeyListener	Interface definition for a callback to be invoked when a hardware key event is dispatched to this view.
interface	View.OnLayoutChangeListener	Interface definition for a callback to be invoked when the layout bounds of a view changes due to layout processing.
interface	View.OnLongClickListener	Interface definition for a callback to be invoked when a view has been clicked and held.
interface	View.OnSystemUiVisibilityChangeListener	Interface definition for a callback to be invoked when the status bar changes visibility.
interface	View.OnTouchListener	Interface definition for a callback to be invoked when a touch event is dispatched to this view.

Exemple d'interfaces

- http://developer.android.com/reference/android/view/View.OnClickListener.html
- http://developer.android.com/reference/android/view/View.OnTouchListener.html



public static interface

View.OnClickListener

android.view.View.OnClickListener

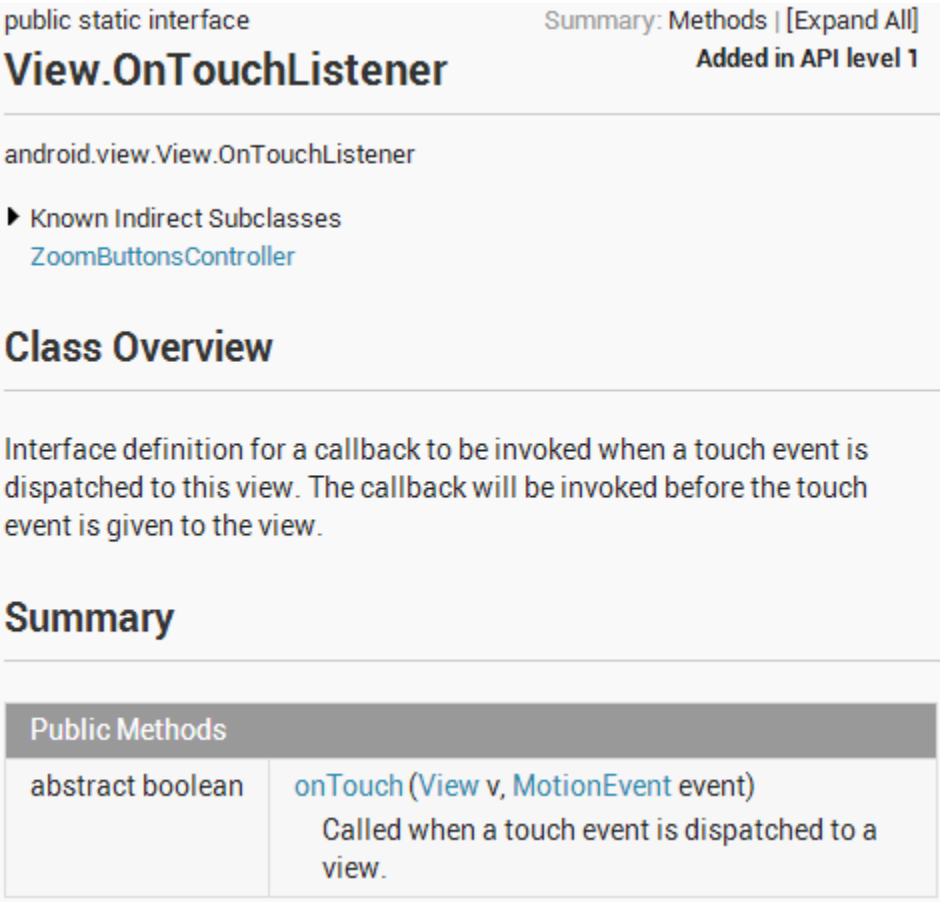
▸ Known Indirect Subclasses
CharacterPickerDialog, KeyboardView, QuickContactBadge

Class Overview

Interface definition for a callback to be invoked when a view is clicked.

Summary

Public Methods	
abstract void	onClick (View v) Called when a view has been clicked.
	Called when a touch event is dispatched to a view.



public static interface

Summary: Methods | [Expand All]

Added in API level 1

View.OnTouchListener

android.view.View.OnTouchListener

▸ Known Indirect Subclasses
ZoomButtonsController

Class Overview

Interface definition for a callback to be invoked when a touch event is dispatched to this view. The callback will be invoked before the touch event is given to the view.

Summary

Public Methods	
abstract boolean	onTouch (View v, MotionEvent event) Called when a touch event is dispatched to a view.

Les listeners communs

```
Class myActivity implementes OnClickListener{
// contient 2 boutons b1 et b2
// une seule méthode pour tous les OnClickEvent

  @Override
  public void onClick(View v) {
    Log.i("quelqu'un a cliqué quelquepart");
    if( b1.getId() == ((Button)v).getId() ){
       // it was the first button
    } else if( b2.getId() == ((Button)v).getId() ){
       // it was the second button
    }
   }
}
```

View.getid() pour identifier la source

Les listeners dédiées à une View

- Une autre approche est de d'abonner la vue à des évéments spécifiques via une classe anonyme.
- Chaque View construit son listener personnel.

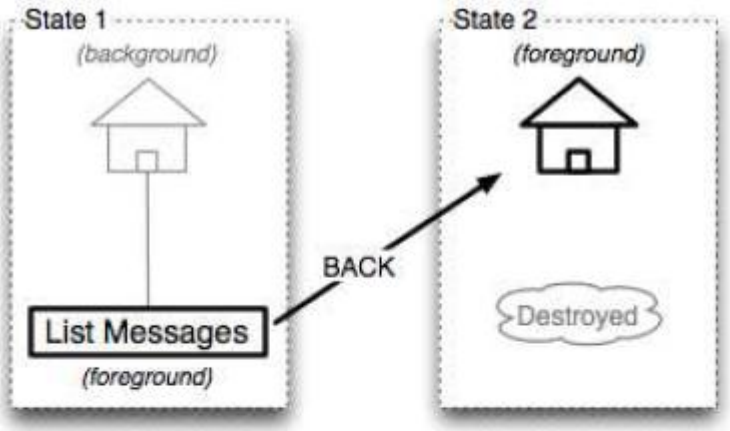
```
Button b = (Button)findViewById(R.id.Button01);
b.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(this, "on m'a cliqué dessus!",
                       Toast.LENGTH_LONG).show();
     }
});
```

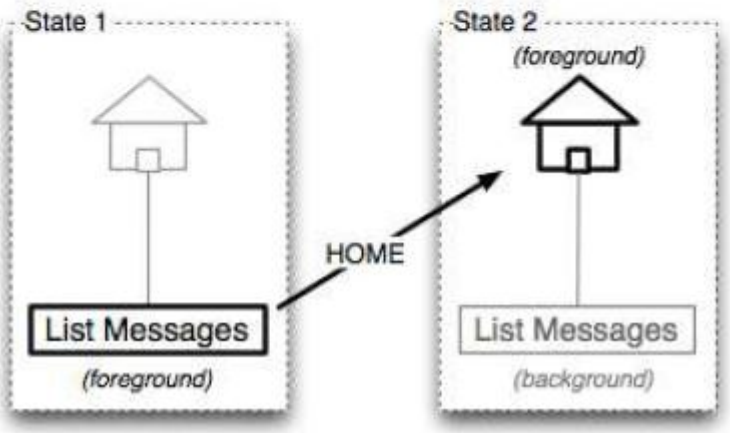
MCours.com

Les activités

La pile des activités

- Les activités sont empilées/dépilées
- Empilée quand une activité démarre
- Dépilée (i.e. détruite) quand on presse le bouton 'BACK'
- Une pression sur le bouton 'HOME' ne
- dépile pas l'activité.
  - Elle passe simplement en arrière plan



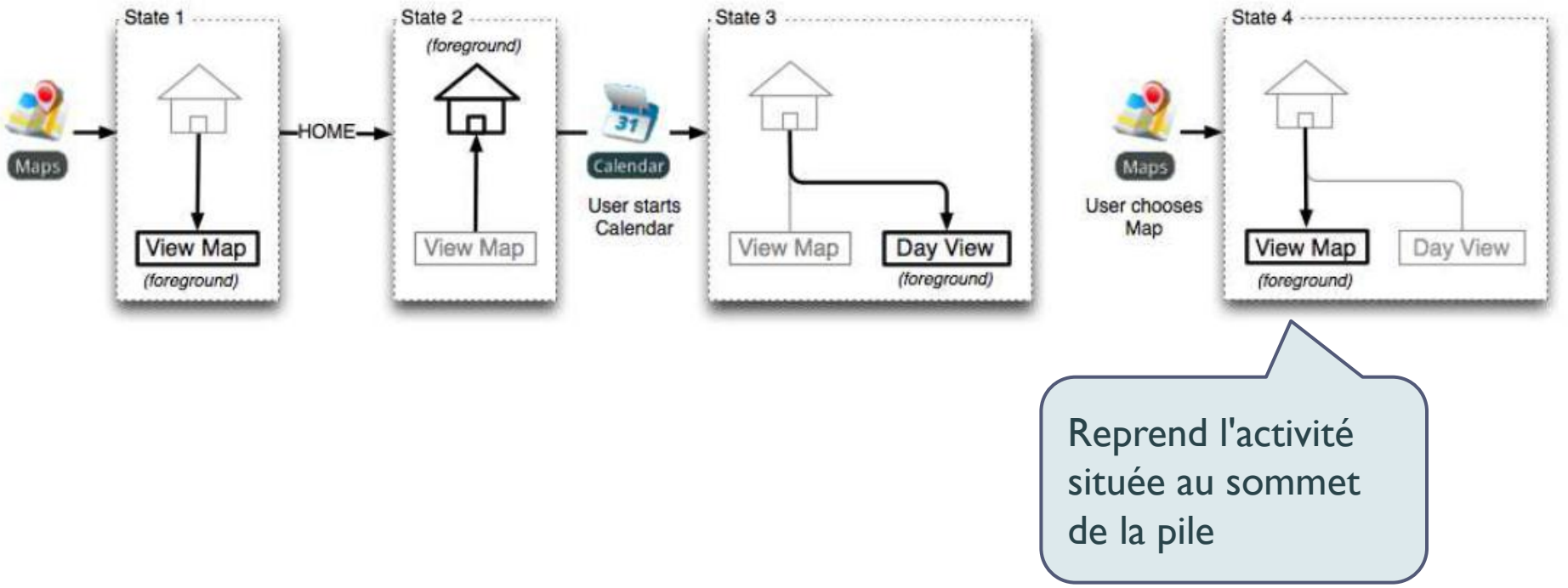


Méthodes de call-back d'une activité

- On a déjà vu que le changement d'état d'une activité provoque le déclenchement de la méthode de call-back correspondante.
- Il est indispensable d'implémenter les méthodes suivantes sous peine de comportement instable
  - En particulier lorsque l'application est constituée de plusieurs activités
  - onCreate(…): alloue les ressources
  - onStop() : sauvegarde si nécessaire
  - onDestroy() : désalloue les ressources
- Attention à toujours créer ces méthodes, et à appeler la méthode correspondante sur super
  - Dans onStop(), appel à super.stop()

Multi-tâches

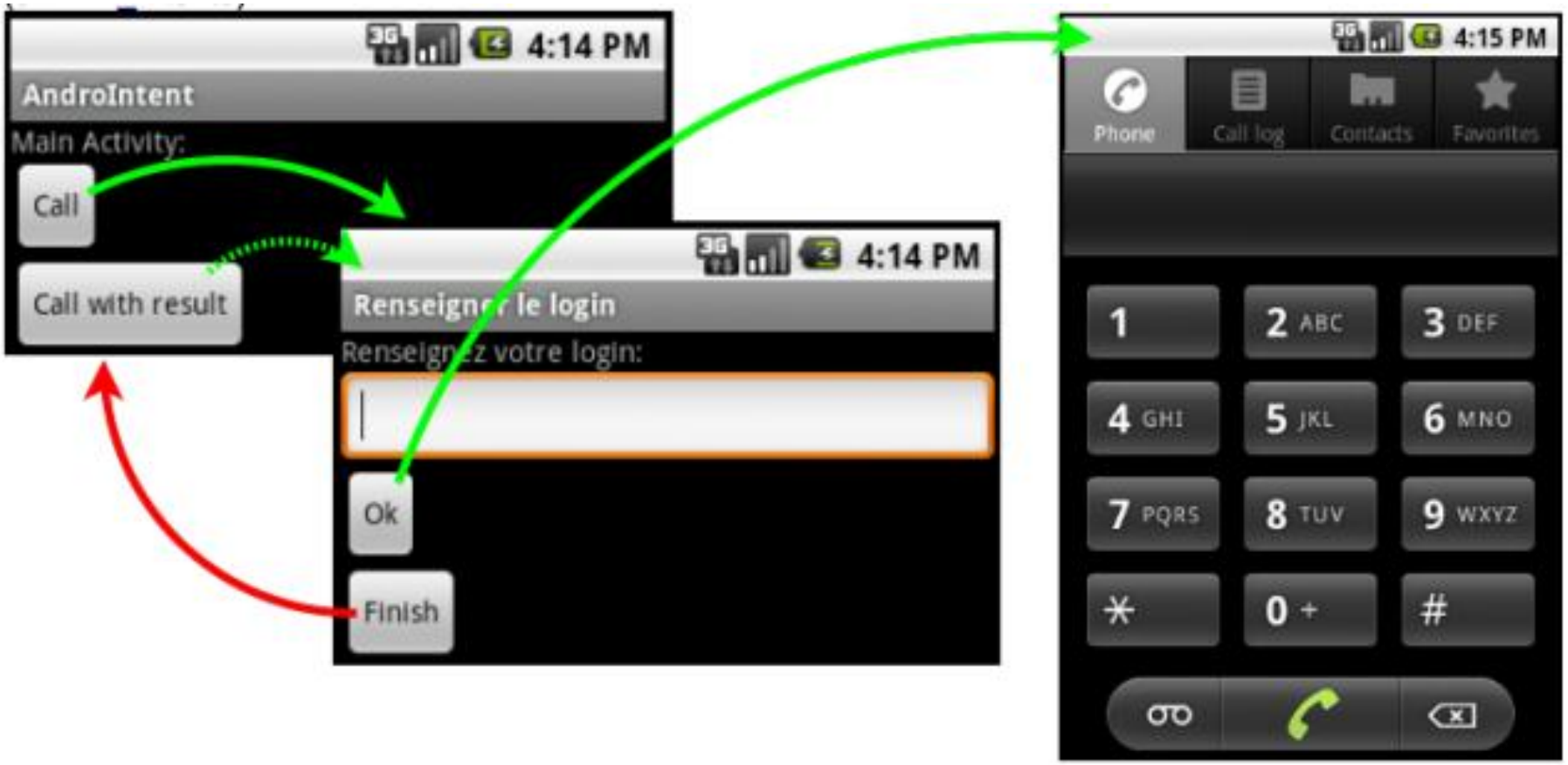
- Plusieurs piles d'activités peuvent co-exister avec Android
  - L'utilisateur passe de l'une à l'autre



Démarrer une activité

- Pour pouvoir être lancée, toute activité doit être préalablement déclarée dans le manifeste
- Une activité est désignée comme activité initiale de l'application
  - Ceci est indiqué dans le fichier AndroidManifest.xml
- Lancer une activité
  - Méthode startActivity(…)
- Lancer une activité en vue d'obtenir un résultat en retour
  - Méthode startActivityForResult(Intent)

Exemple



http://www.univ-orleans.fr/lifo/Members/Jean-Francois.Lalande/enseignement/android/cours-android.pdf

Intent

- Les Intents permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications.
  - Le but des Intents est de déléguer une action à un autre composant, une autre application ou une autre activité de l'application courante.
- Classification des Intent
  - Intent explicite : indique l'identifiant du composant destinataire
  - Intent implicite : indique le type d'action demandée et optionnellement l'URI correspondante
- URI : Uniform Resource Identifier
  - Généralise la notion d'URL
  - Schéma générique = `scheme://host:port/path`
  - schemes usuels : http, mailto, tel, mms, geo, file, ...

Intent

- Un objet Intent contient les information suivantes:
  - le nom du composant ciblé (facultatif)
  - l'action à réaliser, sous forme de chaine de caractères
  - les données: contenu MIME et URI
  - des données supplémentaires sous forme de paires de clef/valeur
  - une catégorie pour cibler un type d'application
  - des drapeaux (information supplémentaire)

Intent Explicite

- Intent explicite : indique l'identifiant du composant destinataire
- Demande la création d'une nouvelle activité en lui passant le nom de la classe en paramètre. Le système démarre une nouvelle instance de la classe
- startActivity(this, ActivityTwo.class);

```
Intent login = new Intent(this, GiveLogin.class);
startActivity(login);
```

Intent Implicite

- Intent implicite : n'indique pas directement l'identifiant du composant destinataire; indique le type d'action demandée et optionnellement l'URI correspondante
- Demande la création d'une activité capable d'effectuer l'action demandée.
  - Le système cherche une application ayant la capacité à effectuer l'action demandée
  - Le système démarre une nouvelle instance de la classe

```
Intent intent=new Intent(ACTION, [uri]);
startActivity(intent);
```

Intent Implicite

```
Button b = (Button)findViewById(R.id.Button01);
b.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
  Uri telnumber = Uri.parse("tel:0248484000");
  Intent call = new Intent(Intent.ACTION_DIAL,
telnumber);
  startActivity(call);
}
});
```

Demande l'ouverture d'un service capable de composer un numéro de téléphone et lui passe l'URI correspondant au schema reconnu tel:<numerodetel>

Liste des actions possibles

- Plusieurs actions natives existent par défaut sur Android.
- La plus courante est l'action `Intent.ACTION_VIEW` qui permet d'appeler une application pour visualiser un contenu dont on donne l'URI.
- Les autres actions sont:
  - ACTION_CALL (ANSWER, DIAL): passer/réceptionner/afficher un appel
  - ACTION_EDIT (DELETE): éditer/supprimer une donnée
  - ACTION_SEND: envoyer des données par SMS ou E-mail
  - ACTION_WEB_SEARCH: rechercher sur internet

Intent Implicite

```
Intent emailIntent = new
Intent(android.content.Intent.ACTION_SEND);
String[] recipients = new String[]{"my@email.com", "",};
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
                     recipients);
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
                     "Test");
emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
                     "Message");
emailIntent.setType("text/plain");
startActivity(Intent.createChooser(emailIntent, "Send
mail..."));
finish();
```

L'activité peut forcer le système à la dépiler avec finish()

Retour d'une activité

- Lorsque le bouton retour est pressé, l'activité courante prend fin et revient à l'activité précédente.
  - Cela permet par exemple de terminer son appel téléphonique et de revenir à l'interface ayant initié l'appel.
- Au sein d'une application, une activité peut vouloir récupérer un code de retour de l'activité "enfant".
- On utilise pour cela la méthode `startActivityForResult(Intent i,int requestCode)` qui envoie un code de retour à l'activité enfant.
- Lorsque l'activité parent reprend la main, il devient possible de filtrer le code de retour dans la méthode `onActivityResul(int requestCode, int resultCode, Intent data)` pour savoir si l'on revient ou pas de l'activité enfant.

Retour d'une activité

- Code d'appel dans l'activité appelante

```
Intent login = new Intent(this,
                          GivePhoneNumber.class);
startActivityForResult(login,48);
```

- Code de traitement par l'activité appelante

```
protected void onActivityResult(int requestCode, int
resultCode, Intent data){
  if (requestCode == 48){
    Toast.makeText(this,
                   "Coup de tel terminé",
                    Toast.LENGTH_LONG).show();
  }
}
```

Filtre sur le numero de requete

Code exécuté au retour de l'activité appelée

Transfert de données

- Les Intent peuvent aussi transporter des données qu'une application veut transmettre à une autre application
- Activité appelante : `intent.putExtra(key,value);`

```
Intent login = new Intent(this, GivePhoneNumber.class);
login.putExtra("nom", "toto");
login.putExtra("pwd", "$$toto31");
startActivityForResult(login,48);
```

- Activité appelée: `intent.getExtras().getXXX(key);`

```
Bundle extra = intent.getExtras();
String nom = extra.getString("nom");
String pwd = extra.getString("pwd");
```

Transfert de données

- Les types de base (+array) sont gérés
  - putExtra(String key String val)
  - `putStringArrayListExtra(String key, ArrayList<String> value)`
- Les types complexes (c-a-d les classes) doivent implémenter Parcelable, ou Serializable

- Récupération du données via `Bundle bundle = getIntent().getExtras();`
- Récupération par la méthode `bundle.getXXX()` propre à chaque type
  - `bundle.getDouble()`
  - `bundle.getString()`
  - `bundle.getArray()`
  - ...

Retour d'une activité

- L'activité appelée peut :
  - transmettre un code de retour via `setResult(codeRetour)`
  - transmettre des données supplémentaires
    1. Construire un nouvel Intent via `new Intent()`
    2. Ajouter des données dans l'Intent via `putExtra(String)`
    3. Envoyer le resultat via `setResult (int resultCode, Intent data)`
    4. Le code de la requête est ajoutée automatiquement
    5. Le nouvel Intent est envoyé à l'application appelante

Retour d'une activité

- L'activité appelée fabrique un nouvel Intent

```
Intent intent = new Intent();
intent.putExtra("resultat", ""+result);
setResult(Activity.RESULT_OK, resultIntent);
```

RESULT_OK = constante prédéfinie

- L'activité appelante récupère les données

```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  switch(requestCode) {
    case (48) : {
      if (resultCode == Activity.RESULT_OK) {
        // TODO Extract the data } break; }
  }
}
```

Retour d'une activité

- L'activité appelée peut forcer le retour en appelant `finish()`

```
// méthode onCreate de l'activite pour implémenter
// un bouton qui « referme » l'activité
Button finish = (Button)findViewById(R.id.finish);
  finish.setOnClickListener(new OnClickListener() {
  @Override
public void onClick(View v) {
    setResult(50);
    finish();
  }});
```

Code différent de RESULT_OK pour signaler un cas particulier

Interfaces graphiques avancées

ListView

- Certains composants graphiques permettent d'afficher beaucoup d'items par des mécanismes de défilement.
  - les Gallery (liste centrée défilement horizontal)
  - les Spinner (boutons de défilement)
  - les ListView (liste défilantes)
- Pour fournir les items à ces composants graphiques, on utilise des AdapterView
- Les AdapterView gèrent aussi les événements (i.e. les interactions) sur le composant graphique avec lequel ils sont associés

Le composant ListView

- Il permet d'afficher une "énorme" liste d'items,
  - accessible par défilement
  - accessible par filtre : l'utilisateur tape les premières lettres, le ListView n'affiche plus que les items qui commencent par ces lettres
- En général les items proviennent d'un accès éventuellement distant à des données

MenuItem

- On commence par indiquer comment seront afficher chaque item du ListView dans un fichier XML à part du répertoire /res/layout
- Ici chaque item est un TextView.
- On construit le fichier res/layout/list_item.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:padding="10dp"
android:textSize="16sp" >
</TextView>
```


http://developer.android.com/resources/tutorials/views/hellolistview.html

ListView : ihm

- On construit l'interface graphique de la ListView dans un fichier XML du répertoire /res/layout

```
<?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">

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

</LinearLayout>
```

L'activité ListActivity

- Une ListView occupe tout l'écran. Elle peut être construite à partir d'une android.app.ListActivity (qui hérite de android.app.Activity)
- Lorsque on construit une classe qui hérite de ListActivity, on peut alors utiliser plusieurs méthodes de la classe ListActivity :
  - public ListView getListView() qui retourne le composant graphique ListView associé à cette activité
  - public void setListAdapter(ListAdapter adapter) positionne le ListAdapter associé à cette activité
- On spécialise la méthode onCreate()

Etendre la classe ListActivity

- Dans le code Java, on récupère cette ListView (par la méthode findViewById()) et on l'alimente d'items :

```
ListView laListe = (ListView) findViewById(R.id.listMenu);
Resources res = getResources();
String [] lesItems = new String[] {
    res.getString(R.string.country),
    res.getString(R.string.society)
};
laListe.setAdapter(new ArrayAdapter<String>(this,
R.layout.list_item, lesItems));
```

Définis dans /res/strings.xml

Affiche la liste des items qui ne sont pas cliquables

ListView : gestion des événements

- La gestion des événements est assuré par le listener

```
public void setOnItemClickListener
(AdapterView.OnItemClickListener listener)
```

- Il faut implémenter la méthode

```
public void onItemClick(AdapterView<?> parent,
View view, int position, long id)
```

- La méthode est appelée lorsque l'utilisateur sélectionne un item de la ListView
  - `parent` est la ListView qui a été utilisée lors de l'événement
  - `view` est la View (c'est à dire l'item a l'intérieur de la ListView) qui a été utilisé
  - `position` est le numéro de l'item dans cette ListView
  - `id` est l'identificateur de l'item sélectionné

ListView : gestion des événements

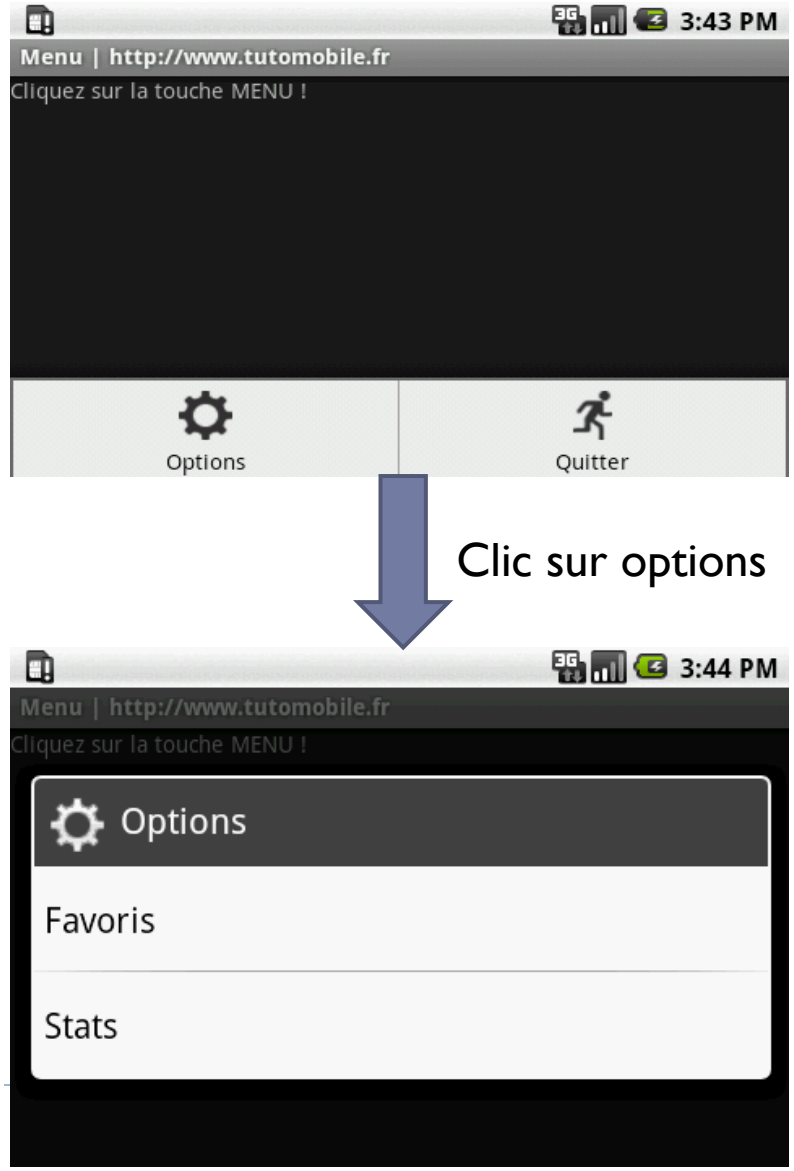
```
Public class HelloListViewActivity extends ListActivity {
  // les items a afficher sont dans un tableau
  static final String[] COUNTRIES = new String[] { ... }

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setListAdapter(new ArrayAdapter<String>(this,
                   R.layout.list_item, COUNTRIES));
    ListView lv = getListView();
    lv.setTextFilterEnabled(true);
    lv.setOnItemClickListener(new OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent, View view,
                              int position, long id) {
        // le code lancé lors de la sélection d'un item
      }
    });
  }
}
```

getListView seulement dans ListActivity

setTextFilterEnabled permet d'activer/de désactiver la sélection par préfixe

Menu

- S'affiche quand on accède au bouton « menu » du mobile
- Le principe est le même que celui des ListView
- Il faut mettre le fichier menu.xml dans /res/menu
- On peut avoir des sous-menus
- source du tutorial sur les menus



Menu : IHM

```
<!-- fichier /res/menu/menu.xm -->
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/option" android:title="Options" >
     <menu android:id="@+id/sousmenu">
        <item android:id="@+id/favoris"
              android:title="Favoris" />
        <item android:id="@+id/stats"
              android:title="Stats" />
     </menu>
  </item>
  <item android:id="@+id/quitter"/>
</menu>
```

```
public class MonMenu extends Activity {
@Override
 public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.menu.menu);
}
  //Méthode qui se déclenchera lorsque vous appuierez sur le bouton menu du téléphone
public boolean onCreateOptionsMenu(Menu menu) {…}

//Méthode qui se déclenchera au clic sur un item
public boolean onOptionsItemSelected(MenuItem item) { /
 //On regarde quel item a été cliqué grâce à son id et on déclenche une action
  switch (item.getItemId()) {
    case R.id.option: Toast.makeText(this, "Option", Toast.LENGTH_SHORT).show();
                      return true;
    case R.id.favoris: Toast.makeText(this, "Favoris", Toast.LENGTH_SHORT).show();
                      return true;
    case R.id.stats: Toast.makeText(this, "Stats", Toast.LENGTH_SHORT).show();
                      return true;
    case R.id.quitter: finish(); return true;
 }
  return false;
}   }
```

MCours.com

Programmation réseau

WebView

- Les WebView permettent d'obtenir une page Web à partir d'une application Android
  - A l'intérieur du layout de l'activité.
  - N'inclue pas toutes les fonctionnalités d'un niavigateur
- Usages
  - Afficher un document hébergé sur un serveur Web
  - Ouvrir une application Web

Webview : IHM, manifeste et javascript

- Fichier /res/layout/webview.xml

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

- Ajout des permissions dans le manifest

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Chargement d'une page web avec `loadURL(String url)`
- Autoriser javascript avec `webSettings.setJavaScriptEnabled(true);`

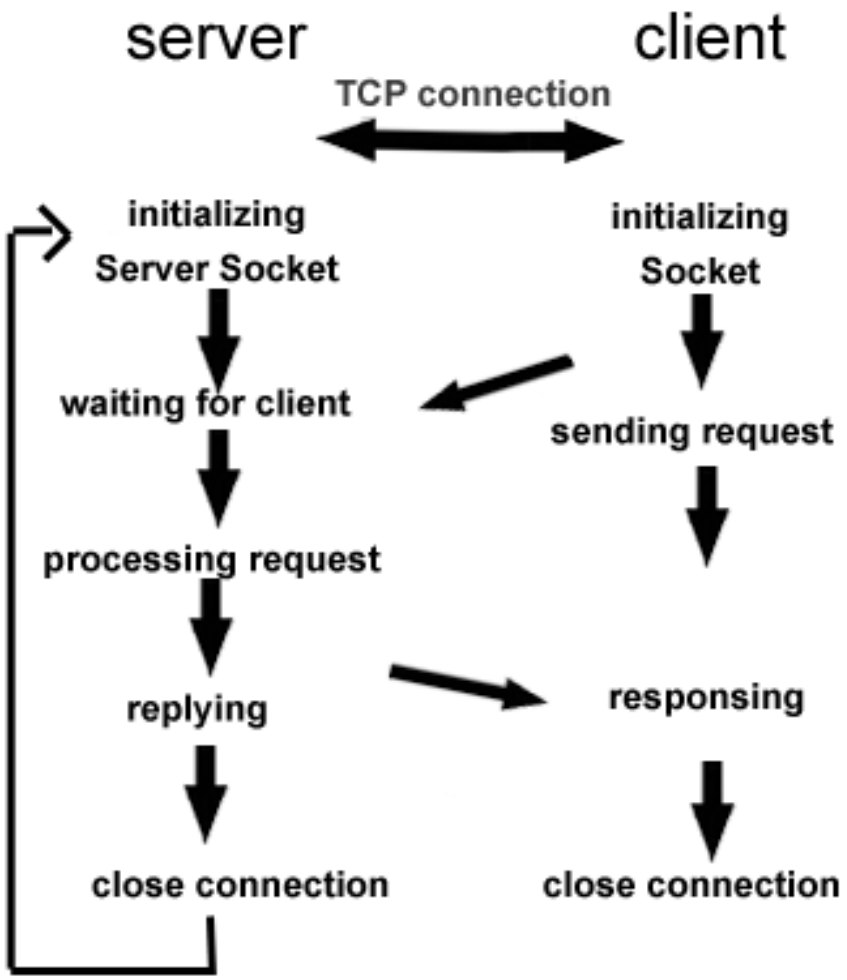
Webview : Activity

```
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class HelloWebView extends Activity {
    WebView webview;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webview = (WebView) findViewById(R.id.webview);
        webview.getSettings().setJavaScriptEnabled(true);
        webview.loadUrl("http://www.google.com");
    }
```

Socket

- Utiliser les Socket java.net avec Android comme avec une application Java classique filaire.
- Masque la couche 2 (Wifi, Ethernet)
- Nécessite la permission INTERNET

```
<uses-permission android:name=
"android.permission.INTERNET" />
```



TCPServer

```
try {
        Boolean end = false;
        ServerSocket ss = new ServerSocket(12345); //port TCP
        while(!end){
                //Server is waiting for client here, if needed
                Socket s = ss.accept();
                BufferedReader input = new BufferedReader(
                        newInputStreamReader(s.getInputStream()));
                PrintWriter output = newPrintWriter(
                             s.getOutputStream(),true); //Autoflush
                String st = input.readLine();
                Log.d("TcpServer",st);
                output.println("salut");
                s.close();
        }
         ss.close();
}catch(Exception e){e.printStackTrace();}
```

TCPClient

```
try {
        Socket s = new Socket("localhost",12345);

        //Ecrire dans la socket
        OutputStream out = s.getOutputStream();
        PrintWriter output = new PrintWriter(out);
        output.println("Hello Android!");

        //Lire dans la socket
        BufferedReader input = new BufferedReader(
               newInputStreamReader(s.getInputStream()));
        String st = input.readLine();

        //Referme la connexion
        s.close();

} catch (UnknownHostException e) {    e.printStackTrace();}
```