Mobilité: Programmation Android

Erick STATTNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-antilles.fr

www.erickstattner.com

Description de l'enseignement

Objectifs pédagogiques:

- Se familiariser à la Programmation d'applications pour mobile
- Maitriser les principes autour des applications Android
- Concevoir des applications graphiques sous Android
- Mettre en place la persistance des données

Organisation:

- 30h
- 1 CC + 1 CT

Sommaire

Chapitre V.
Persistance et Interfaces avancées

1. Les listes
2. Clics sur une liste
3. Les popus
4. Pour aller plus loin sur les interfaces
5. Persistance des données

V. Persistance et Interfaces avancées

Les listes

Une liste

- Ensemble d'éléments affichés les uns à la suite des autres
- Chaque élément sur 1 ou 3 lignes maximum
- Chaque ligne est personnalisable
  - *TextView*, *Button*, *ImageView*, etc.
- Classe ***ListView***

V. Persistance et Interfaces avancées

Les listes

Quatre étapes

1. Placer un **ListView** dans le layout de l'activité
   Par exemple en utilisant l'éditeur graphique
2. Créer un **adapter** (adaptateur)
   qui permet de remplir une liste à partir d'un tableau ou d'une collection
3. Lier l'**adapter** qui contient les données au **ListView**
4. Maintenir la liste

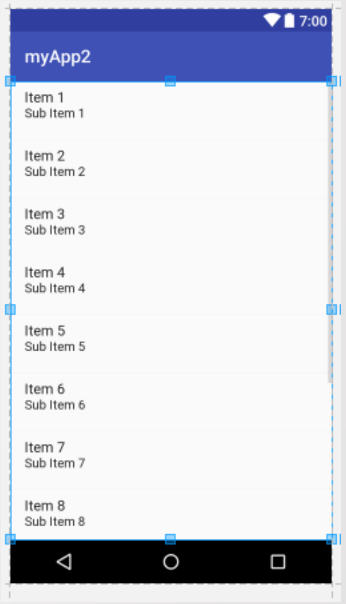
V. Persistance et Interfaces avancées

Les listes

1. Placer un ListView dans le layout de l'activité

- La liste est vide et ne contient rien

```
<ListView
    android:id="@+id/maliste"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
```



V. Persistance et Interfaces avancées

Les listes

2. Créer un adapter

- Les données de la liste sont chargées à partir d'un tableau ou d'une collection

```
String[] data = {"Personne 1", "Personne 2", "Personne 3", "Personne 4", "Personne 5"};
```

OU

```
ArrayList<String> data2 = new ArrayList<String>();
for(int i = 0; i < 20; i++){
    data2.add("Personne "+(i+1));
}
```

- Les données sont utilisées pour générer l'adapter

```
ArrayAdapter matching = new ArrayAdapter(this, android.R.layout.simple_list_item_1, data2);
```

V. Persistance et Interfaces avancées

Les listes

2. Créer un adapter

- Les données de la liste sont chargées à partir d'un tableau ou d'une collection

```
String[] data = {"Personne 1", "Personne 2", "Personne 3", "Personne 4", "Personne 5"};
```

OU

```
ArrayList<String> data2 = new ArrayList<String>();
for(int i = 0; i < 20; i++){
    data2.add("Personne "+(i+1));
}
```

- Les données sont utilisées pour générer l'adapter

```
ArrayAdapter matching = new ArrayAdapter(this, android.R.layout.simple_list_item_1, data2);
```

Activité courante

Style graphique des items

V. Persistance et Interfaces avancées

Les listes

3. Lier Adapter et ListView

- Récupérer une référence vers la liste
- Attribuer l'adaptateur à la liste

```
liste = (ListView)findViewById(R.id.maliste);
liste.setAdapter(matching);
```

V. Persistance et Interfaces avancées

Les listes

4. Maintenir la liste

- Ajouter un élément

```
data2.add("nouvelle personne");
```

- Notifier la mise à jour

```
matching.notifyDataSetChanged();
```

V. Persistance et Interfaces avancées

Les listes

4. Maintenir la liste

- Supprimer un élément

```
data2.remove(0);
```

- Notifier la mise à jour

```
matching.notifyDataSetChanged();
```

V. Persistance et Interfaces avancées

Clics sur une liste

Clic sur un élément de la liste

- Rendre les items d'une liste cliquables

```
liste.setOnItemClickListener(this);
```

ou

```
liste.setOnItemLongClickListener(this);
```

- Du coté de l'écouteur
  - Implements OnItemClickListener et OnItemLongClickListener

Parameters	
AdapterView: parent	L'adapteur sur lequel est survenu
View: view	La vue sur laquelle le clic est survenu
int: position	La position de la vue sur l'adapteur
int: id	L'identifiant de la ligne

V. Persistance et Interfaces avancées

Clics sur une liste

Clic sur un élément de la liste

- *Exemple*

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

    Log.v("TEST", "Vous avez cliqué sur l'élément"+position+"de la liste");

}
```

V. Persistance et Interfaces avancées

Les popus

Les popus

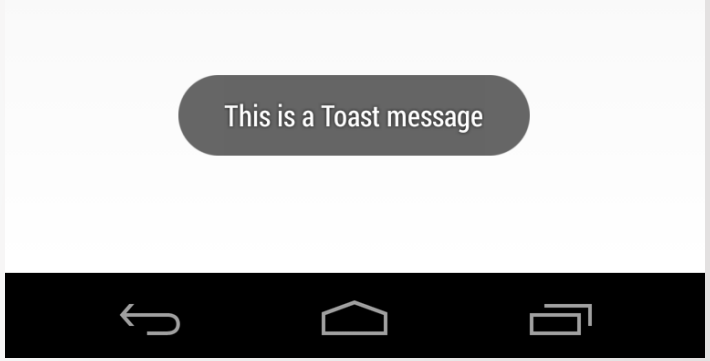
- Boite permettant de dialoguer avec l'utilisateur
- Objectif:
  - Indiquer un message court
  - Poser une question
  - Afficher une progression
- Différents types
  - Toasts
  - AlertDialog
  - ProgressDialog

V. Persistance et Interfaces avancées

Les popus

Toast

- Afficher un message court
- Pas d'interaction avec l'utilisateur
- Syntaxe
  Toast.makeText(Context, Message, Duree)
  - *Context*: Activité appelante
  - *Message*: Le texte du message à afficher
  - *Duree*: La durée de l'affichage
    Toast.LENGTH_SHORT **ou** **Toast.LENGTH_LONG**
  - Appeler la méthode *show* pour lancer l'affichage le toast
    Toast.makeText(this, "This is a Toast message", Toast.LENGTH_SHORT).show();

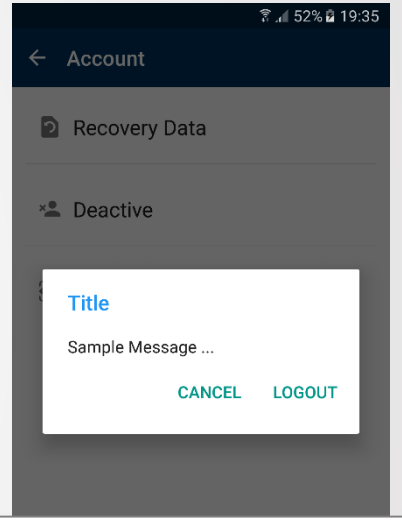


V. Persistance et Interfaces avancées

Les popus

AlertDialog

- Afficher un message
- Créer une interaction avec l'utilisateur
- Composé de trois parties
  - Un contenu
  - Un ou plusieurs boutons




```
Builder builder = new AlertDialog.Builder(this);
builder.setMessage("This will end the activity");

builder.setPositiveButton("I agree", new OkOnClickListener());
builder.setNegativeButton("No, no", new CancelOnClickListener());

AlertDialog dialog = builder.create();
dialog.show();
```

V. Persistance et Interfaces avancées

Les popus

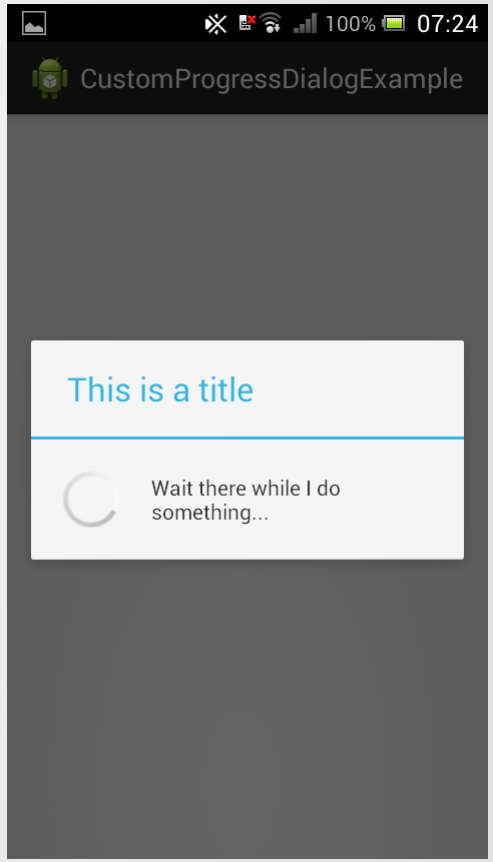
ProgressDialog

- Barre de progression dans l'application
- Afficher avant un traitement long

```
ProgressDialog progressDoalog = new ProgressDialog(this);
progressDialog.setMessage("Its loading....");
progressDialog.setTitle("ProgressDialog bar example");
progressDialog.show();
```

- Apres le traitement, appeler méthode *dismiss()*

```
progressDialog.dismiss();
```



V. Persistance et Interfaces avancées

Les popus

Pour aller plus loin

- **Bonnes pratiques**
  - Etre indépendant de la résolution et de la taille de l'écran
  - Privilégier les tailles relatives (wrap content / match parent) sinon taille en **dp** ou **sp**
  - Layout/Drawable pour les différentes tailles d'écrans **(small, normal, large)**
  - Layout pour les différentes positions **(port, land)**
  - Attention au API disponibles parfois uniquement sous certaines versions d'android
- **Application plus riches**
  - ActionBar, Onglets, Fragments, WebView, Notifications
  - Animation

V. Persistance et Interfaces avancées

Persistance des données

Persistance des données

- Stocker les données nécessaires au bon fonctionnement
- Conserver données entre deux exécutions
- Plusieurs possibilités de stocker des données
  - SharedPreferences
  - **Fichiers**
  - **Base de données SQL Lite**
  - Passer par des services WEB

V. Persistance et Interfaces avancées

Persistance des données

Stockage dans fichiers internes

- Sauvegarder des données dans des fichiers directement sur mémoire interne
- Deux étapes à mettre en place
  1. Ecriture dans le fichier
  2. Lecture dans le fichier

V. Persistance et Interfaces avancées

Persistance des données

Ecriture dans un fichier

- en 3 phases
  1. Appeler la méthode *openFileOutput*
     FileOutputStream out = openFileOutput("monFichier.txt", Context.MODE_PRIVATE);
  2. Utiliser la méthode *write* pour écrire
     String t = "mon message" ;
     out.write(t.getBytes());
  3. Fermer le fichier avec *close*
     out.close();

Type d'ouverture
EX. Context.MODE_APPEND

Attention aux erreurs qui peuvent survenir: problème écriture, droit, etc.

V. Persistance et Interfaces avancées

Persistance des données

Lecture dans un fichier

- en 3 phases
  1. Appeler la méthode *openFileInput*
     FileInputStream in = openFileInput("monFichier.txt")
  2. Utiliser la méthode *read* pour remplir au fur et à mesure un buffer
     ```
     byte[] buffer = new byte[1024];
     StringBuilder mess = new StringBuilder();
     while(in.read(buffer) != -1){
             mess.append(new String(buffer));
     }
     ```
  3. Fermer le fichier avec *close*
     in.close();

Attention aux erreurs qui peuvent survenir: problème lecture, fichier introuvable, etc.

V. Persistance et Interfaces avancées

Persistance des données

Adaptation à des fichiers textes

- **Ecriture**

```
FileOutputStream fout = openFileOutput("monFichier.txt", Context.MODE_APPEND);
OutputStreamWriter out = new OutputStreamWriter(fout);
out.write(t+"\n");
out.close();
```

- **Lecture**

```
FileInputStream fin = openFileInput("monFichier.txt");
InputStreamReader in = new InputStreamReader(fin);
BufferedReader buffer = new BufferedReader(in);
String ligne = "";
String t = "";
while ( (ligne = buffer.readLine()) != null ) {
    t += ligne;
}
fin.close();
```