

Android

Développer des applications pour mobiles

Plan

- Présentation 3
- Architecture 11
- Interfaces graphiques 34
- Gestion événementielle 44
- Interfaces avancées 48
- Applications multi-écrans 91
- Fragment et activité fille 103
- Réseaux et services web 139
- Persistance des données 164
- SQLite 178
- API et fonctionnalités multimédia 194
- GoogleMap et GPS 211
- Déploiement des packages 225

Présentation

- ▶ Historique
- ▶ Modèle de développement
- ▶ Contexte et enjeux
- ▶ Outils et SDK
- ▶ Java version Android



fabien.brissonneau@gmail.com

3

Historique

- ▶ Android est un système développé par Open Handset Alliance, un regroupement de 84 sociétés
- ▶ Ce système d'exploitation tourne sur plusieurs matériels
- ▶ Le Software Development Kit permet de construire les applications qui s'exécutent sur le système

fabien.brissonneau@gmail.com

4

Modèle de développement

- ▶ Environnement Java : disposer d'un JDK
- ▶ L'IDE à utiliser : Android Studio
- ▶ Le SDK Android est inclus dans la livraison de Android Studio
- ▶ Utilisation de Gradle, pour automatiser la construction et la gestion du projet

fabien.brissonneau@gmail.com

5

SDK Android

- ▶ Outils nécessaires à la construction d'une application Android
- ▶ Android Asset Packaging Tool (aapt) : créer et analyse les fichiers .apk
- ▶ Android Debug Bridge (adb) : établir les connexions vers un téléphone ou un émulateur
- ▶ Dalvik Debug Monitor Service (ddms) : pour déboguer
- ▶ SDK Manager

fabien.brissonneau@gmail.com

6

Le gestionnaire de SDK

- ▶ Tools : SDK Tools, SDK Platform-tools, Android SDK Build Tools
- ▶ Android : Documentation, SDK Platform, Sample, Google APIs, Source
- ▶ Extras : Android Support Library, Google Repository, Google USB Driver Package, Google Play Services

fabien.brissonneau@gmail.com

7

Emulateur

- ▶ Permet de simuler un téléphone ou une tablette
- ▶ Utilise un AVD
- ▶ Android Virtual Device
 - un matériel
 - version de l'API



fabien.brissonneau@gmail.com

8

Versions

- Android 1.0 en 2008, 1.5 (API 3) en 2009, baptisée CupCake
- Donut, v1.6 (API 4) en septembre 2009
- Eclair, v2 (API 7) en octobre 2009
- Froyo v2.2 (API 8) en mai 2010
- Gingerbread v2.3 (API 10) en décembre 2010
- Honeycomb v3 (API 11, 12, 13) en février 2011
- Ice Cream Sandwich v4 (API 14, 15) en octobre 2011
- JellyBean v4.1 à 4.3.1 (API 16,17,18) en juin 2012
- KitKat v4.4 (API 19) en octobre 2013
- Lollipop v5 (API 21) en octobre 2014
- Marshmallow v6 (API 23) en mai 2015
- Nougat v7 (API 24, 25) en août 2016
- Oreo v8 (API 26) en septembre 2017
- Pie v9 (API 28) en août 2018

fabien.brissonneau@gmail.com

9

Exercice 0

fabien.brissonneau@gmail.com

10

Architecture d'une application

- ▶ Fichier manifest
- ▶ Activités
- ▶ Ressources
- ▶ Receivers
- ▶ Services
- ▶ Les fournisseurs de contenu

fabien.brissonneau@gmail.com

11

Architecture Android

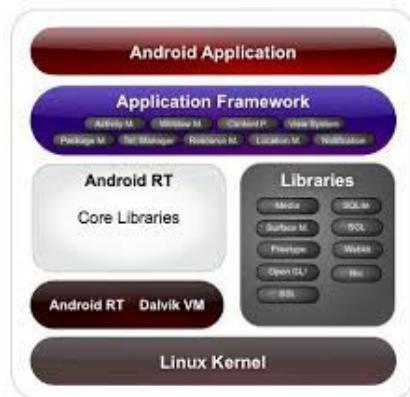
- ▶ Orientée maîtrise des ressources et de la consommation
- ▶ Les applications Android s'exécutent dans un environnement contraint
- ▶ Porter attention à : la création des objets, l'utilisation du processeur, de la RAM, du stockage... la consommation, la diversité des tailles, des matériels et des versions

fabien.brissonneau@gmail.com

12

Architecture Android

- ▶ 5 parties distinctes : application, framework, bibliothèques, android runtime, le noyau linux
- ▶ Une application est un user linux



fabien.brissonneau@gmail.com

13

Android RunTime

- ▶ Machine virtuelle, utilisant AOT Compilation (lors de l'installation de l'application)
- ▶ Apparue en version 4.4, seule utilisée en version 5
- ▶ Elle remplace Dalvik, une machine JIT

fabien.brissonneau@gmail.com

14

Native Development Kit

- ▶ Ecrire des applications en C
- ▶ Appelables en Java via JNI
- ▶ Utilisé pour des parties que l'on veut performantes
- ▶ Ou réutilisation de code C existant

fabien.brissonneau@gmail.com

15

Android Package APK

- ▶ Fichier binaire livré sur le téléphone
- ▶ Contient
 - Le fichier .dex de l'application compilée
 - Les ressources
 - Les « assets »
 - Les certificats
 - Le manifeste
- ▶ Le package est généré par aapt

fabien.brissonneau@gmail.com

16

Signer un package

- ▶ Créer ou utiliser un fichier de clés (keystore)
- ▶ Mettre un mot de passe sur le fichier
- ▶ Créer une clé en donnant un alias et un mot de passe
- ▶ ...
- ▶ Générer le package en indiquant le keystore
- ▶ Ne pas perdre le keystore, nécessaire pour les mises à jour
- ▶ En ligne de commande
 - keytool permet de générer la clé de signature
 - jarsigner permet de signer le package a posteriori

fabien.brissonneau@gmail.com

17

Les composantes

- ▶ Activité
- ▶ Fragment
- ▶ Service
- ▶ Broadcast Receiver
- ▶ Content Provider
- ▶ Intent
- ▶ Intent filter
- ▶ PendingIntent
- ▶ Application

fabien.brissonneau@gmail.com

18

L'activité

- ▶ Composante principale d'une application, est l'implémentation des interfaces
- ▶ Une activité est liée à un fichier de description de la vue XML
- ▶ Le framework Android peut tuer une activité si celle-ci n'est plus en premier plan et s'il a besoin de ressources

Le fragment

- ▶ Notion qui permet d'adapter une interface à la taille et à la forme du matériel
- ▶ Une activité peut être composée d'un ou plusieurs fragments
- ▶ Par exemple,
 - En fonction du matériel ou de la disposition, les fragments peuvent être dans une activité ou dans deux activités

Le service

- ▶ Exécution d'une tâche de fond (longue)
- ▶ Ne possède pas d'interface
- ▶ Un service s'arrête lorsqu'il est interrompu ou terminé

fabien.brissonneau@gmail.com

21

Le broadcast receiver

- ▶ Récepteur d'événement système
- ▶ Peut capter : réception d'un SMS, démarrage du téléphone, écran verrouillé...
- ▶ Pas d'interface et doivent être des tâches légères
- ▶ Peut afficher une notification, lancer une activité, lancer un service...

fabien.brissonneau@gmail.com

22

Le content provider

- ▶ Fournisseur de contenu, accessible via des URI
 - Doit être publié dans le manifeste
- ▶ Donne accès à des données, en base SQLite, en fichiers, sur le web...
- ▶ Le système propose des content providers pour les contacts, l'agenda, ...

fabien.brissonneau@gmail.com

23

L'Intent

- ▶ Message système qui permet de communiquer entre composantes du système
- ▶ Les applications peuvent créer leur propres Intent, et sont à l'écoute d'Intent
- ▶ Les Intents explicites servent à lancer une application bien précise
- ▶ Les Intents implicites laissent le système trouver l'application capable de traiter une tâche

fabien.brissonneau@gmail.com

24

L'Intent-filter

- ▶ Les filtres d'intention servent à déclarer quelle action une activité, un broadcast receiver ou un service peut traiter
- ▶ Les filtres d'intention sont typiquement déclarés dans le manifeste (AndroidManifest.xml)
- ▶ Les filtres déclarent une action et une catégorie
- ▶ L'activité principale d'une application contient : MAIN et LAUNCHER
- ▶ Il est possible aussi d'utiliser la classe IntentFilter

Le PendingIntent

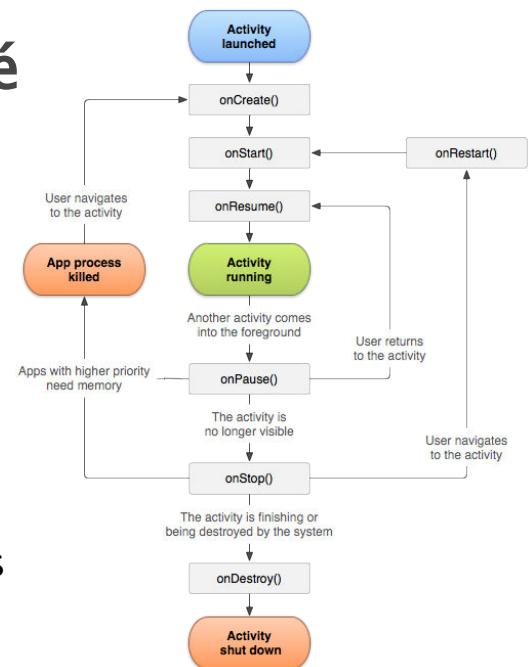
- ▶ Permet de décrire un Intent, utilisé plus tard
- ▶ Crée avec getActivity, getBroadcast ou getService pour lancer respectivement Activity, BroadcastReceiver et Service
- ▶ Permet à un tiers de déclencher un Intent dans l'application elle-même
- ▶ Exemple classique : pour qu'une notification redonne la main à votre application

L'Application

- ▶ La classe Application existe pour chaque application Android
- ▶ Il est possible de remplacer cette classe
- ▶ Intérêt : gérer des ressources globales à votre application
- ▶ Doit être alors déclarée dans le manifeste
- ▶ La classe Application est instanciée lors du lancement de l'application Android

Cycle de vie d'une activité

- ▶ onCreate : initialiser la vue
 - Récupérer l'état sauvé
- ▶ onStart
- ▶ onResume : abonner les listeners
- ▶ onPause: désabonner les listeners
 - Sauver l'état
- ▶ onStop : tout arrêter
- ▶ La back stack est la pile des activités



La classe Activity

- ▶ Une sous-classe redéfinit ou pas les méthodes

```
public class Activity extends ApplicationContext {  
    protected void onCreate(Bundle savedInstanceState);  
  
    protected void onStart();  
  
    protected void onRestart();  
  
    protected void onResume();  
  
    protected void onPause();  
  
    protected void onStop();  
  
    protected void onDestroy();  
}
```

fabien.brissonneau@gmail.com

29

Les permissions

- ▶ Droit d'accès à une fonctionnalité ou à une donnée
- ▶ Utiliser une permission : <uses-permission>
- ▶ Déclarer une permission
 - Forcer une application à déclarer une permission pour utiliser un service
- ▶ Android 6 : les permissions se demandent plutôt à l'exécution

fabien.brissonneau@gmail.com

30

Les log

- ▶ Le logcat est l'interface pour voir les messages
- ▶ Les méthodes : d(), e(), i(), v(), w()
- ▶ Passer un identifiant (TAG) et un message

```
    }
    catch(IOException io) {
        Log.v("MIC", "Probleme..." + io);
    }
    enregistreur.start();

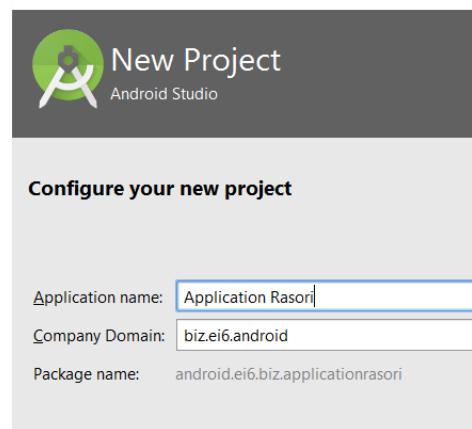
    Log.v("MIC", "En cours d'enregistrement");
```

fabien.brissonneau@gmail.com

31

Organisation du projet Android Studio

- ▶ Création de projet
- ▶ Architecture du projet
 - Manifest
 - Ressources : drawable, layout, values,
 - Le fichier généré R.java
 - Fichiers sources
 - Les fichiers de configuration



fabien.brissonneau@gmail.com

32

Exercice 00

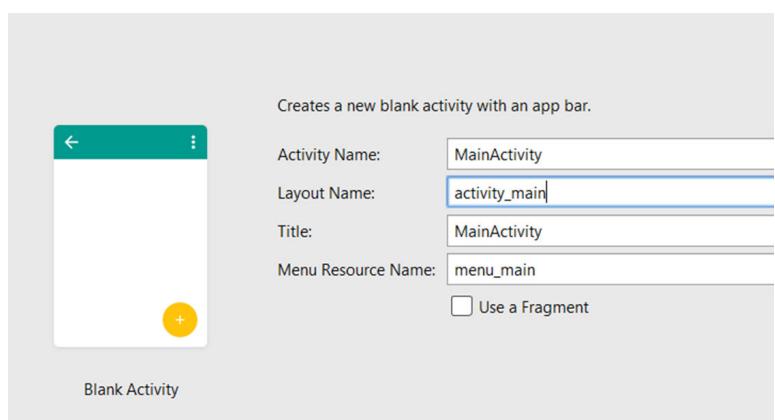
- ▶ Démonstration Hello World

fabien.brissonneau@gmail.com

33

Interfaces graphiques

- ▶ Les ressources
- ▶ Les layouts
- ▶ Les strings
- ▶ Les drawable



fabien.brissonneau@gmail.com

34

Les ressources

- Les ressources sont utilisées de façon conventionnelle
- Dans le répertoire « res »
- Dans un sous répertoire selon le type de ressource : layout, drawable, values, menu, etc
- Selon le type de la ressource, le nom du fichier peut être significatif

Les layout

Les vues

- ▶ Dans un fichier XML et dans une activité
- ▶ Il faut un identifiant pour chaque élément

```
<TextView  
      android:id="@+id/identifiant"
```

- ▶ Cet élément dans le code Java

```
    dViewById(R.id.identifiant);
```

- ▶ Ou dans un fichier XML

```
    android:id="@+id/textView"  
    android:layout_below="@+id/identifiant"
```

fabien.brissonneau@gmail.com

37

Lien avec les activités

- ▶ La classe dérivée de Activity
- ▶ Implémente au minimum onCreate
- ▶ S'associe avec la vue grâce à setContentView(...)

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

fabien.brissonneau@gmail.com

38

Les layouts

- ▶ Sont des conteneurs intermédiaires
- ▶ Dérivés de ViewGroup
- ▶ Il existe plusieurs classes de Layout
- ▶ Cas du ConstraintLayout
 - Les composants doivent avoir un id
 - Un composant peut être attaché sur le parent
 - Un composant peut être attaché sur un autre



fabien.brissonneau@gmail.com

39

ConstraintLayout

Attachement au parent ou id d'un composant

```
app:layout_constraintStart_toStartOf="parent"  
app:layout_constraintEnd_toEndOf="parent"
```

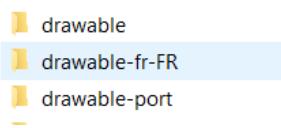
```
app:layout_constraintTop_toBottomOf="@+id/takePhotoBtn"
```

Attachement de Start, End, Left, Right, Top, Bottom
Vers toStartOf, toEndOf ... etc



Les ressources

- ▶ Les ressources sont spécifiques au code du pays, la langue, largeur minimum, largeur, hauteur, taille, orientation, version



- ▶ mipmap-hdpi
- ▶ mipmap-mdpi
- ▶ mipmap-xhdpi
- ▶ mipmap-xxhdpi
- ▶ mipmap-xxxhdpi
- ▶ values
- ▶ values-v21
- ▶ values-w820dp

fabien.brissonneau@gmail.com

41

Les valeurs

- ▶ Les constantes dans le projet
- ▶ Les chaînes de caractères

```
<string name="app_name">Application Rasori</string>
<string name="action_settings">Settings</string>
```

```
setTitle(R.string.app_name);
```

```
        android:text="@string/app_name"
```

- ▶ Gestion possible des pluriels avec getQuantityString(id,qté)
- ▶ Gestion des formats avec res.getString(id,params...)
- ▶ Gestion des langues en créant les sous-répertoires values-xx
- ▶ Stockage possible de tableaux de chaînes
- ▶ Pour le formatage, penser à String.format(....)

fabien.brissonneau@gmail.com

42

Tableaux de chaînes et pluriels

```
<string-array name="pays">
    <item>France</item>
    <item>Allemagne</item>
    <item>Italie</item>
</string-array>
```

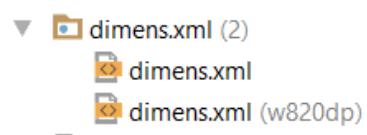
`resources.getStringArray(R.array.pays)`

```
<resources>
    <plurals name="texteExemple">
        <item quantity="one">%d cheval</item>
        <item quantity="many">%d chevaux</item>
    </plurals>
</resources>
```

`resources.getQuantityString(R.plurals.texteExemple, quantity: 1)`

Les dimensions

- ▶ Il est possible de définir les dimensions récurrentes
- ▶ Utiliser `@dimens/id` pour les utiliser
- ▶ Les unités sont
 - `dp` pour la mise en page
 - `sp` pour les polices



Champ texte, bouton et case à cocher

- ▶ TextView
 - Utiliser android:text pour positionner la chaîne
- ▶ EditText
 - Utiliser android:hint pour proposer une aide
 - Utiliser android:inputType pour cibler le clavier
- ▶ Button
 - Utiliser android:Text pour positionner le texte
- ▶ CheckBox
 - Utiliser android:checked pour définir l'état
 - Utiliser android:text pour lui associer un texte

Un TextView

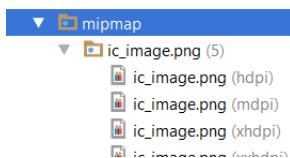
UN BOUTON

Un EditText

Une CheckBox

Champ image

- ▶ ImageView
 - Utiliser android:src pour positionner une image
 - Utiliser android:contentDescription pour décrire cette image



```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/imageView"  
    android:src="@mipmap/ic_image"
```

Exercice 1

fabien.brissonneau@gmail.com

47

La gestion événementielle

- ▶ Basée sur la notion de listeners, autre nom pour observers
- ▶ Implémentable par classes indépendantes, classe internes, classes anonymes et expressions lambda
 - La syntaxe Java8 est paramétrable

fabien.brissonneau@gmail.com

48

Réagir sur une CheckBox

- ▶ Récupérer une référence sur l'objet
- ▶ Utiliser l'identifiant
- ▶ Positionner un listener

```
CheckBox chkb = (CheckBox) findViewById(R.id.checkBox);
chkb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        //blah blah
    }
});
```

- ▶ Typiquement une classe anonyme
- ▶ Récupération de la source d'événement et de l'état

fabien.brissonneau@gmail.com

49

Réagir sur un Button

- ▶ Utiliser des classes anonymes
 - Pas besoin de rechercher la source d'événement
 - Correspond aux interfaces fonctionnelles de Java8
- ▶ Mettre la callback dans l'activité
 - Une seule méthode pour tous
 - Besoin de connaître la source d'événement pour réagir
 - Utiliser getId() sur le paramètre de type View

```
Button leBouton = (Button) findViewById(R.id.button);
leBouton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //blah blah
    }
});
```

fabien.brissonneau@gmail.com

50

Exercice 2

fabien.brissonneau@gmail.com

51

L'interface graphique

- ▶ Vues
- ▶ Contrôles
- ▶ Layouts
- ▶ Boîtes de dialogues
- ▶ Thèmes
- ▶ ListView

fabien.brissonneau@gmail.com

52

Vues

- ▶ Les éléments des vues doivent avoir une taille spécifiée
 - Avec android:layout_width et android:layout_height
 - Valeurs prédéfinies match_parent et wrap_content
 - Ou bien taille spécifique en dp
- ▶ Possibilité d'inclure une vue dans une autre

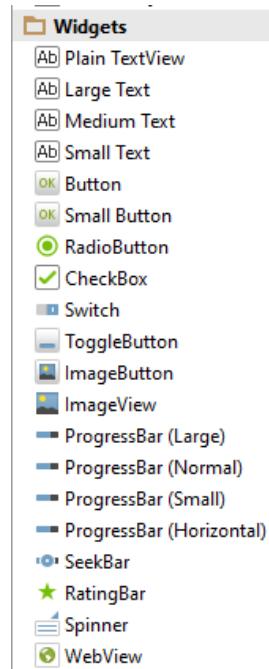
```
<include layout="@layout/content_main" />
```

Contrôles

- ▶ Diversité
- ▶ Etats
- ▶ Autres

Diversité des contrôles

- ▶ La palette présente les contrôles
- ▶ Le nom du contrôle est
 - Le nom de la balise XML
 - Le nom de la classe Java



fabien.brissonneau@gmail.com

55

Etats d'un contrôle

- ▶ Définir les changements dans un fichier de drawable

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/vert" android:state_pressed="true"/>
    <item android:drawable="@color/bleu" android:state_focused="true"/>
```

- ▶ Appliquer sur un contrôle

```
<Button
    android:background="@drawable/etats"
    android:layout_width="wrap_content"
```

```
schemas.android.com/apk/res/android">
or/vert" android:state_pressed="true"/>
or/rouge" android:state_focused="true"/>
or/bleu" android:state_hovered="true"/>
or/jaune" android:state_selected="true"/>
or/orange" android:state_checkable="true"/>
or/violet" android:state_checked="true"/>
or/gris" android:state_enabled="true"/>
or/marron" android:state_activated="true"/>
```

fabien.brissonneau@gmail.com

56

Autres éléments

- ▶ Créer un gradient de couleur en fond
 - Drawable de type shape avec gradient
- ▶ Placer une élévation android:elevation
- ▶ Effet d'ondulation
 - Drawable de type ripple
- ▶ Police de caractères
 - Attribut android:textSize = 12sp, 14sp, 18sp, 22sp
 - Attribut android:typeface = monospace, normal, sans, serif
 - Attribut android:fontFamily : sous-type
 - Attribut android:textStyle = bold, italic, normal...
- ▶ Il est possible de charger un fichier de police ttf

Layouts

- ▶ FrameLayout
- ▶ LinearLayout
- ▶ TableLayout
- ▶ RelativeLayout
- ▶ GridLayout
- ▶ ScrollView

FrameLayout

- ▶ Affiche un objet à la fois
- ▶ Les objets sont ou non visibles
- ▶ La position des éléments est variable grâce à android:gravity

LinearLayout

- ▶ Les éléments sont placés en ligne, suivant une orientation
 - Utiliser android:orientation, par défaut horizontale
- ▶ Les éléments supportent les attributs
 - Utiliser android:layout_gravity pour l'ancrage dans le conteneur
 - Utiliser android:gravity pour l'ancrage du contenu de l'élément
 - Utiliser android:layout_weight pour donner un poids relatif
 - La taille doit être 0dp (layout_height ou layout_width)
 - Le layout possède lui android:weightSum

TableLayout

- ▶ Positionnement en tableau régulier
- ▶ Cellules vides possible
- ▶ Utilise une balise <TableRow>

fabien.brissonneau@gmail.com

61

RelativeLayout

- ▶ Positionnement relatif des éléments, comme déjà vu

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="android.ei6.biz.appliphoto.MainActivity">
    <Button
        android:id="@+id/enregistre"
        android:onClick="enregistrer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enregistre !" />
    <Button
        android:id="@+id/stop"
        android:layout_below="@+id/enregistre"
        android:onClick="stopper"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop !" />
</RelativeLayout>
```

fabien.brissonneau@gmail.com

62

GridLayout

- ▶ Utilise des lignes et des colonnes
 - Avec android:columnCount et android:rowCount
- ▶ Cellules vides possible, mais aussi étendues sur plusieurs lignes et colonnes
- ▶ Des éléments d'espaces peuvent être insérés
- ▶ Utiliser les attributs android:layout_row et layout_column
- ▶ Possible d'utiliser android:layout_gravity

ScrollView

- ▶ Un décorateur qui permet de rajouter des barres de scroll
- ▶ Ne contient qu'un seul layout, qui devient scrollable
- ▶ Incompatible avec ListView

Boîtes de dialogue

- ▶ Toast
- ▶ Snackbar
- ▶ AlertDialog
- ▶ ProgressDialog
- ▶ Boîtes personnalisées

fabien.brissonneau@gmail.com

65

Toast

- ▶ Pour des messages courts qui disparaissent seuls, sans action

```
Toast.makeText(MainActivity.this, "Voici un toast", Toast.LENGTH_LONG).show();
```



fabien.brissonneau@gmail.com

66

Snackbar

- ▶ Comme un Toast, mais peut référencer une action et peut être éliminé de l'écran par l'utilisateur
- ▶ Fait partie de « Android Design Support Library »
 - Cf fichier Gradle

```
Snackbar.make(v, "Snackbar", Snackbar.LENGTH_LONG).show();  
  
Snackbar.make(v, "Snackbar", Snackbar.LENGTH_LONG)  
    .setAction("Ouvrir", new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            //blah  
        }  
    }).show();
```

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    testCompile 'junit:junit:4.12'  
    compile 'com.android.support:appcompat-v7:23.2.1'  
    compile 'com.android.support:design:23.2.1'  
}
```



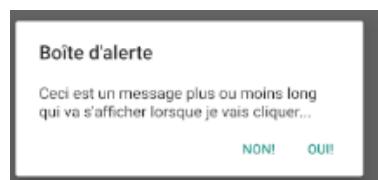
fabien.brissonneau@gmail.com

67

AlertDialog

- ▶ Une boîte de dialogue avec ou sans boutons

```
new AlertDialog.Builder(MainActivity.this)  
    .setMessage("Boîte d'alerte")  
    .show();
```



```
new AlertDialog.Builder(MainActivity.this)  
    .setMessage("Ceci est un message plus ou moins long qui va s'afficher")  
    .setTitle("Boîte d'alerte")  
    .setPositiveButton("Oui!", new DialogInterface.OnClickListener() {  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            //blah  
        }  
    })  
    .setNegativeButton("Non!", new DialogInterface.OnClickListener() {  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            //blah  
        }  
    })  
    .show();
```

fabien.brissonneau@gmail.com

68

ProgressDialog

- ▶ 2 styles possibles
- ▶ Donner le max : setMax
- ▶ Donner le progrès : setProgress
- ▶ Arrêter la boîte : dismiss()



```
ProgressDialog pd = new ProgressDialog(MainActivity.this);
pd.setTitle("Un titre pour la progression");
pd.setMessage("Ca avance ...");
pd.setMax(300);
pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
pd.show();
```

fabien.brissonneau@gmail.com

69

Boîtes personnalisées

- ▶ Classe android.app.Dialog
- ▶ Utilise un fichier layout

```
Dialog dlg = new Dialog(MainActivity.this);
dlg.setContentView(R.layout.maboite);
dlg.show();
```



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Boîte perso !"/>
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_joli"/>
</LinearLayout>
```

fabien.brissonneau@gmail.com

70

Thèmes

- ▶ Le thème par défaut de Android 5 est Material Design
- ▶ Environnement Google, théoriquement en 3D
- ▶ Il y a 3 thèmes fournis, paramétrables
 - Dark Material (sombre) soit android:Theme.Material
 - Light Material soit android:Theme.Material.Light
 - Light Material avec ActionBar sombre
 android:Theme.Material.Light.DarkActionBar
- ▶ Pour choisir le thème, modifier l'attribut android:theme de l'application
- ▶ Le thème Material est disponible à partir de API v21

fabien.brissonneau@gmail.com

71

Thèmes

- ▶ Choix du thème dans Manifest.xml
- ▶ Ou via un style

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Rasori"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity
        android:name=".MainActivity">
```



```
        <!-- Base application theme. -->
        <style name="AppTheme" parent="android:Theme.Material">
            <!-- Customize your theme here. -->
            <item name="colorPrimary">@color/colorPrimary</item>
```

fabien.brissonneau@gmail.com

72

ListView

- ▶ Un ensemble d'éléments les uns après les autres
 - De 1 à 3 lignes, chacune avec une disposition propre
- ▶ Les données sont chargées grâce à un adaptateur
 - ArrayAdapter, SimpleCursorAdapter, ...
- ▶ L'activité conteneur peut être une sous-classe de
 - Activity
 - ListActivity : ne comprend qu'une liste

fabien.brissonneau@gmail.com

73

ListActivity

- ▶ Ce layout correspond à une ListActivity
- ▶ Pour remplir la liste, créer un adaptateur

```
ArrayAdapter adaptateur =
    new ArrayAdapter(this, android.R.layout.simple_list_item_1, systemes);
```
- ▶ Un adaptateur a besoin
 - Du contexte
 - Du layout de chaque ligne
 - Du tableau des données
- ▶ Positionner l'adaptateur sur la liste

```
setListAdapter(adaptateur);
```

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/list" />
```

Identifiant remarquable

fabien.brissonneau@gmail.com

74

Utiliser ListActivity

- ▶ Identifiant obligatoire
- ▶ Code minimaliste

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    String[] donnees = {"Marshmallow", "Lollipop", "Kitkat", "Jellybean"};  
  
    ArrayAdapter<String> adaptateur =  
        new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, donnees);  
  
    setListAdapter(adaptateur);  
}
```

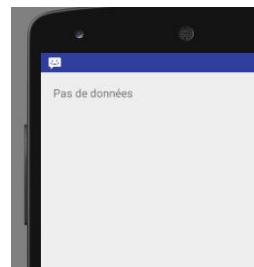
fabien.brissonneau@gmail.com

75

Possibilités supplémentaires

- ▶ Identifiant utilisé lorsque la liste est vide

```
<ListView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/list" />  
  
<TextView  
    android:id="@+id/empty"  
    android:text="Pas de données"  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent" />
```



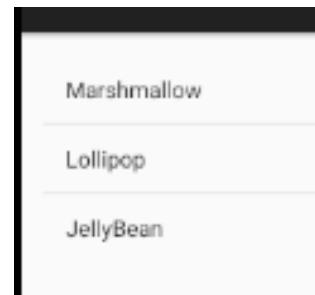
fabien.brissonneau@gmail.com

76

Avec Activity

- ▶ ListView est alors un contrôle comme un autre
- ▶ Récupérer une référence grâce à findViewById(...)
- ▶ Positionner l'adaptateur sur la liste

```
ListView liste = (ListView) findViewById(R.id.liste);  
  
String[] tab = {"Marshmallow", "Lollipop", "JellyBean"};  
  
ArrayAdapter<String> adaptateur =  
    new ArrayAdapter<String>(this, android.R.layout.s  
  
liste.setAdapter(adaptateur);
```



fabien.brissonneau@gmail.com

77

Items spécialisés

- ▶ Une liste peut afficher des items complexes
 - Un layout pour décrire l'item
 - Une classe représentant la donnée à afficher
 - Un adaptateur spécialisé, par dérivation

```
<LinearLayout  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
    <TextView  
        android:id="@+id/nom"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
  
    <TextView  
        android:id="@+id/prenom"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
  
</LinearLayout>
```

fabien.brissonneau@gmail.com

78

Adaptateur spécialisé

- Hérite de ArrayAdapter, redéfinit getView et getCount

```
public class MonAdaptateur extends ArrayAdapter<Personne>{  
  
    private ArrayList<Personne> donnees;  
    private Context contexte;  
  
    private int viewRes;  
    private Resources res;  
  
    public MonAdaptateur(Context context, int resource, ArrayList<Personne> tab) {  
        super(context, resource,tab);  
  
        donnees = tab;  
        contexte = context;  
        viewRes = resource;  
        res = context.getResources();  
    }  
}
```

fabien.brissonneau@gmail.com

79

Méthode getView

- Génération d'une vue à partir des données

```
@Override  
public View getView(int position, View cv, ViewGroup parent) {  
    View view = cv;  
  
    if(view == null) {  
        LayoutInflator layoutInflater =  
            (LayoutInflator) contexte.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        view = layoutInflater.inflate(viewRes, parent, false);  
        Log.v("ADAPTATEUR", "inflate");  
    }  
  
    final Personne p = donnees.get(position);  
    if(p != null) {  
        final TextView nom = (TextView) view.findViewById(R.id.nom);  
        final TextView prenom = (TextView) view.findViewById(R.id.prenom);  
        nom.setText(p.getNom());  
        prenom.setText(p.getPrenom());  
        Log.v("ADAPTATEUR", "findview");  
    }  
  
    return view;  
}
```

fabien.brissonneau@gmail.com

80

Utilisation de l'adaptateur spécialisé

- ▶ Connecter les données à la liste

```
ArrayList<Personne> personnes = new ArrayList<Personne>();
personnes.add(new Personne("Fabien", "Brissonneau"));
personnes.add(new Personne("Daniel", "Martin"));

MonAdaptateur ma = new MonAdaptateur(this, R.layout.itemdeliste, personnes);

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



fabien.brissonneau@gmail.com

81

Événement sur une liste

- ▶ Choisir le listener en fonction de l'événement
- ▶ Pour la sélection : onItemClick

```
liste.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Personne p = (Personne) parent.getItemAtPosition(position);
        Log.v("LISTE", p.getNom() + " " + p.getPrenom());
    }
});
```

fabien.brissonneau@gmail.com

82

Mise à jour de la liste

- ▶ Modifier l'adaptateur et notifier du changement

```
MonAdaptateur ma = new MonAdaptateur(this, R.layout.itemdeliste, personnes);  
  
ListView liste = (ListView) findViewById(R.id.liste);  
liste.setAdapter(ma);  
  
ma.add(new Personne("Mickey", "Mouse"));  
  
ma.notifyDataSetChanged();
```



fabien.brissonneau@gmail.com

83

Performances

- ▶ Dans le code précédent
 - Inflate est réalisé une seule fois, à la construction
 - findViewById est réalisée à chaque visualisation des items
- ▶ Donc il faut essayer de minimiser les appels à findViewById

```
class ViewHolder {  
    TextView nom;  
    TextView prenom;  
}  
ViewHolder holder;  
@Override  
public View getView(int position, View cv, ViewGroup parent) {  
    View view = cv;  
    if(view == null) {  
        LayoutInflator layoutInflater =  
            (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        view = layoutInflater.inflate(viewRes, parent, false);  
        Log.v("ADAPTATEUR", "inflate");  
        holder = new ViewHolder();  
    }  
    holder.nom.setText(personnes.get(position).getNom());  
    holder.prenom.setText(personnes.get(position).getPrenom());  
    return view;  
}
```

fabien.brissonneau@gmail.com

84

Code modifié

```
public View getView(int position, View cv, ViewGroup parent) {
    View view = cv;
    if(view == null) {
        LayoutInflator layoutInflater =
            (LayoutInflator) contexte.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        view = layoutInflater.inflate(viewRes, parent, false);
        Log.v("ADAPTATEUR", "inflate");
        holder = new ViewHolder();
        holder.nom = (TextView) view.findViewById(R.id.nom);
        holder.prenom = (TextView) view.findViewById(R.id.prenom);
        view.setTag(holder);
        Log.v("ADAPTATEUR", "findview");
    }
    final Personne p = donnees.get(position);
    if(p != null) {

        holder.nom.setText(p.getNom());
        holder.prenom.setText(p.getPrenom());
        Log.v("ADAPTATEUR", "setter");
    }
    return view;
}
```

fabien.brissonneau@gmail.com

85

RecyclerView

- ▶ Existe depuis la version 5
- ▶ Propose 2 positionnements
 - LinearLayoutManager
 - GridLayoutManager
- ▶ Nécessite des dépendances de compilation

```
compile 'com.android.support:appcompat-v7:23.2.1'
compile 'com.android.support:recyclerview-v7:23.2.1'
```

- ▶ Placer dans le layout

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mon_recycler"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
/>
```

fabien.brissonneau@gmail.com

86

Créer l'adaptateur

- ▶ Un viewholder déjà prévu

```
public static class ViewHolder extends RecyclerView.ViewHolder {  
    public TextView nom;  
    public TextView prenom;  
  
    public ViewHolder(View v) {  
        super(v);  
        nom = (TextView) v.findViewById(R.id.nom);  
        prenom = (TextView) v.findViewById(R.id.prenom);  
    }  
}
```

fabien.brissonneau@gmail.com

87

Une classe RecyclerViewAdapter

- ▶ Reste à implémenter ...

```
public class RecyclerAdaptateur  
    extends RecyclerView.Adapter<RecyclerAdaptateur.ViewHolder>{  
  
    @Override  
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        return null;  
    }  
  
    @Override  
    public void onBindViewHolder(ViewHolder holder, int position) {  
    }  
  
    @Override  
    public int getItemCount() {  
        return 0;  
    }  
}
```

fabien.brissonneau@gmail.com

88

Implémentation

- ▶ Prévoir le stockage des infos

```
ArrayList<Personne> donnees;
public RecyclerAdaptateur(ArrayList<Personne> data) {
    donnees=data;
}
@Override
public int getItemCount() {
    return donnees.size();
}
```

fabien.brissonneau@gmail.com

89

Implémentation

- ▶ Séparer création et rafraîchissement

```
@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    View v = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.itemdeliste,parent,false);
    Viewholder vh = new Viewholder(v);
    return vh;
}
@Override
public void onBindViewHolder(Viewholder holder, int position) {
    holder.nom.setText(donnees.get(position).getNom());
    holder.prenom.setText(donnees.get(position).getPrenom());
}
```

fabien.brissonneau@gmail.com

90

Utiliser le RecyclerView

- ▶ Déclaration d'une référence sur le composant
- ▶ Création de l'adaptateur
- ▶ Création d'un layoutmanager

```
RecyclerView rcw = (RecyclerView) findViewById(R.id.mon_recycler);

RecyclerAdaptateur ma = new RecyclerAdaptateur(personnes);
rcw.setAdapter(ma);

RecyclerView.LayoutManager llm = new LinearLayoutManager(this);
rcw.setLayoutManager(llm);
```

fabien.brissonneau@gmail.com

91

CardView

- ▶ Utilisée dans un ListView ou un RecyclerView
- ▶ Représente un item d'information
- ▶ Nécessite une bibliothèque

```
compile 'com.android.support:design:23.2.1'
compile 'com.android.support:cardview-v7:23.2.1'
}
```

fabien.brissonneau@gmail.com

92

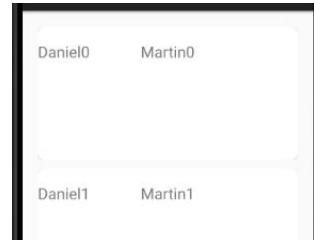
Utiliser CardView

- ▶ Une aide pour le layout de l'item de la liste

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:orientation="vertical"
    android:padding="6dp">

    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        card_view:cardCornerRadius="10dp">

        <TextView
            android:id="@+id/nom"
            ...>
    
```



fabien.brissonneau@gmail.com

93

Exercice 3

fabien.brissonneau@gmail.com

94

Application multi-écrans

- ▶ Sous-activités
- ▶ Fragments

fabien.brissonneau@gmail.com

95

Les fragments

- ▶ Une partie d'une activité
- ▶ Un fragment a un layout et un cycle de vie
- ▶ Une activité peut être composée de plusieurs fragments
- ▶ Un fragment peut être ajouté ou enlevé dynamiquement
- ▶ Les ajouts et enlèvements se font dans des transactions
- ▶ L'utilisation des fragments est soit statique soit dynamique

fabien.brissonneau@gmail.com

96

Cycle de vie du fragment

- ▶ onAttach : fragment attaché à la vue
- ▶ onCreate : initialisations essentielles
- ▶ onCreateView : chargement de la vue
- ▶ onActivityCreated : fin du chargement
- ▶ Puis onStart/onResume/onPause/onStop
- ▶ onDestroyView
- ▶ onDestroy
- ▶ onDetach

fabien.brissonneau@gmail.com

97

Dériver la classe Fragment

- ▶ Implémenter les méthodes essentielles

```
public class MonFragment extends Fragment {  
    @Override  
    public void onCreate(Bundle sis) {  
        super.onCreate(sis);  
    }  
  
    @Override  
    public View onCreateView( LayoutInflater inflater, ViewGroup parent, Bundle sis) {  
        View v = inflater.inflate(R.layout.details, parent, false);  
        return v;  
    }  
  
    @Override  
    public void onActivityCreated(Bundle sis) {  
        super.onActivityCreated(sis);  
    }  
}
```

fabien.brissonneau@gmail.com

98

Utilisation statique des fragments

- ▶ Suivant l'orientation
 - Mode portrait
 - 2 activités
 - Mode paysage
 - 1 activité



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

fabien.brissonneau@gmail.com

99

Orientation de l'activité principale

- ▶ Mode paysage ou portrait

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/laliste"  
        android:layout_width="150dp"  
        android:layout_height="match_parent"  
        class="android.e16.biz.applifragments.Liste">  
    </fragment>  
    <fragment android:id="@+id/details"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        class="android.e16.biz.applifragments.Details">  
    </fragment>  
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/laliste"  
        android:layout_width="150dp"  
        android:layout_height="match_parent"  
        class="android.e16.biz.applifragments.Liste">  
    </fragment>  
</LinearLayout>
```

fabien.brissonneau@gmail.com

100

Une activité pour 2 fragments

```
public class Liste extends ListFragment {  
  
    private String[] donnees = {"Marshmallow", "Lollipop", "KitKat"};  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
  
    @Override  
    public void onActivityCreated(Bundle savedInstanceState) {  
        super.onActivityCreated(savedInstanceState);  
  
        ArrayAdapter<String> adapteur =  
            new ArrayAdapter<String>(getActivity(), android.R.layout.  
                android.R.layout.simple_list_item_1, donnees);  
  
        setListAdapter(adaptateur);  
    }  
  
    @Override  
    public void onListItemClick(ListView l, View v, int position, long id) {  
        String item = (String) getListAdapter().getItem(position);  
        Details fragment = (Details)getFragmentManager().findFragmentById(R.id.details);  
        if(fragment != null && fragment.isInLayout()) {  
            fragment.setText(item);  
        }  
        else {  
    
```

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <TextView  
        android:id="@+id/detailsText"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" />  
</LinearLayout>
```

```
    }  
    @Override  
    public void onActivityCreated(Bundle savedInstanceState) {  
        super.onActivityCreated(savedInstanceState);  
    }  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {  
        View view = inflater.inflate(R.layout.details, parent, false);  
        return view;  
    }  
    public void setText(String item) {  
        TextView view = (TextView) getView().findViewById(R.id.detailsText);  
        view.setText(item);  
    }  
}
```

fabien.brissonneau@gmail.com

101

En mode portrait, 2 activités

► Démarrer l'activité de détails

```
@Override  
public void onListItemClick(ListView l, View v, int position, long id) {  
    String item = (String) getListAdapter().getItem(position);  
    Details fragment = (Details)getFragmentManager().findFragmentById(R.id.details);  
    if(fragment != null && fragment.isInLayout()) {  
        fragment.setText(item);  
    }  
    else {  
        Intent intent = new Intent(getActivity(), DetailsActivity.class);  
        intent.putExtra("donnee", item);  
        startActivity(intent);  
    }  
}
```

```
public class DetailsActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.details);  
        Bundle extras = getIntent().getExtras();  
        if(extras != null) {  
            String s = extras.getString("donnee");  
            TextView view = (TextView) findViewById(R.id.detailsText);  
            view.setText(s);  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

102

Présenter les 2 activités dans le manifeste

- ▶ Les deux activités sont présentées, mais une seule est principale

```
<manifest xmlns="http://schemas.android.com/apk/res/android">  
    <application>  
        <activity  
            android:name=".MainActivity"  
            android:label="AppliFragments"  
            android:theme="@style/AppTheme.NoActionBar">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <activity  
            android:name=".DetailsActivity" />  
    </application>
```

fabien.brissonneau@gmail.com

103

Utilisation dynamique des fragments

- ▶ Les fragments peuvent être ajoutés dynamiquement sur un FrameLayout

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

fabien.brissonneau@gmail.com

104

Placer le fragment dans le FrameLayout

- ▶ Dans l'activité, pour une classe de fragment « MonFragment »

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    getFragmentManager().beginTransaction()  
        .add(R.id.conteneur,new MonFragment()).commit();  
}  
  
public class MonFragment extends Fragment {  
  
    public MonFragment() {}  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {  
        View vue = inflater.inflate(R.layout.fragment,parent,false);  
        return vue;  
    }  
}
```

fabien.brissonneau@gmail.com

105

Exercice 4

fabien.brissonneau@gmail.com

106

Fragment et activité fille

- ▶ Le fragment n'est pas une activité
 - Pas de retour sur l'ajout d'un fragment dans le cas de BackStack
 - Possibilité de le mettre dedans : transaction.addToBackStack
- ▶ Une activité fille est une activité sur laquelle on peut revenir
 - Déclarer cette parenté dans le manifest
 - Visible dans l'ActionBar

Lancer une activité

- ▶ Déclarer les relations de parenté éventuelles

```
</activity>
<activity android:name=".ActiviteFille" android:parentActivityName=".MainActivity">
</activity>
```

- ▶ Créer un Intent et lancer ...

```
public void creerLaFille(View v) {
    Intent intent = new Intent(this,ActiviteFille.class);
    startActivity(intent);
}
```

Passer des arguments

- ▶ Lors du lancement

```
String valeur="Fabien";
Intent intent = new Intent(this,ActiviteFille.class);
intent.putExtra("nom",valeur);
startActivity(intent);
```

- ▶ Dans l'activité nouvelle

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

    Intent intent = getIntent();
    String nom = intent.getStringExtra("nom");

    TextView txt = (TextView) findViewById(R.id.nom);
    txt.setText(nom);
}
```

- ▶ Limité aux types de base

fabien.brissonneau@gmail.com

109

Les arguments « parcelable »

- ▶ Pour des objets complexes, qui seront sérialisés
- ▶ Suppose l'implémentation de Parcelable
 - Et les méthodes de sérialisation/désérialisation

```
public class Personne implements Parcelable{
    private String nom;
    public String getNom() { return nom; }
    public void setNom(String n) { nom=n; }

    public Personne(String nom) { this.nom=nom; }
    public Personne(Parcel source) {this.nom=source.readString();}

    @Override
    public int describeContents() {
        //pas d'objet "spécial"
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(nom);
    }
}
```

```
public static final Parcelable.Creator<Personne> CREATOR
    = new Parcelable.Creator<Personne>() {
        @Override
        public Personne createFromParcel(Parcel source) {
            return new Personne(source);
        }
        @Override
        public Personne[] newArray(int size) {
            return new Personne[size];
        }
    };
}
```

fabien.brissonneau@gmail.com

110

Passage de Parcelables

- ▶ Code directement avec le type utilisateur

```
public void creerLaFille(View v) {  
  
    Personne fabien = new Personne("fabien");  
    Intent intent = new Intent(this,ActiviteFille.class);  
    intent.putExtra("nom",fabien);  
    startActivity(intent);  
  
}  
  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activite_fille);  
  
    Intent intent = getIntent();  
    Personne p = intent.getParcelableExtra("nom");  
  
    TextView txt = (TextView)findViewById(R.id.nom);  
    txt.setText(p.getNom());  
}
```

fabien.brissonneau@gmail.com

111

Attendre un résultat

- ▶ Lancement
- ▶ Prévoir la callback
- ▶ Des codes préexistant : RESULT_OK, RESULT_CANCELED

```
Personne fabien = new Personne("fabien");  
Intent intent = new Intent(this,ActiviteFille.class);  
intent.putExtra("nom", fabien);  
startActivityForResult(intent, CODE_REQUETE);  
  
@Override  
public void onActivityResult(int codeRequete, int codeResultat, Intent donnees) {  
    super.onActivityResult(codeRequete,codeResultat,donnees);  
  
    if(codeRequete == CODE_REQUETE) {  
        if(codeResultat == RESULT_OK){  
            Log.v("FILLE", "Le nom est " + donnees.getStringExtra("NOM"));  
        }  
        else {  
            Log.v("FILLE", "Pas bon résultat");  
        }  
    }  
    else {  
        Log.v("FILLE", "Pas bon code");  
    }  
}
```

fabien.brissonneau@gmail.com

112

La fille retourne

- ▶ Lorsque le résultat doit être retourné
 - Les codes résultats existent déjà
 - RESULT_OK, RESULT_CANCELED
- ▶ D'autres valeurs de codes peuvent être utilisées
- ▶ L'activité fille fournit la valeur adaptée

```
public void envoyerResultat(View v) {  
    Intent intent = new Intent();  
    intent.putExtra("NOM","brissonneau");  
    setResult(RESULT_OK,intent);  
    finish();  
}
```

fabien.brissonneau@gmail.com

113

Déclencher une autre application

- ▶ Soit en connaissant le package de l'application
 - Il faut utiliser PackageManager.getLaunchIntentForPackage
- ▶ Soit en déclenchant une action
 - Intent.ACTION_VIEW : ouvrir un fichier
 - Intent.ACTION_DIAL : lancer un appel
 - Intent.WEB_SEARCH : lancer le navigateur
 - ...

fabien.brissonneau@gmail.com

114

Utiliser le nom de package

- ▶ Connaître le nom du package !
 - com.android.settings
 - com.android.camera
 - ...
- ▶ Retourne null si aucun Intent

```
public void lancerSettings(View v) {  
    PackageManager pm = getPackageManager();  
    Intent intent = pm.getLaunchIntentForPackage("com.android.settings");  
  
    if(intent != null) {  
        startActivity(intent);  
    }  
}
```

fabien.brissonneau@gmail.com

115

Déclencher une action

- ▶ Fournir une action
- ▶ Vérifier qu'une activité existe pour supporter cette action

```
public void lancerAction(View v) {  
    PackageManager pm = getPackageManager();  
    Intent intent = new Intent(Intent.ACTION_VIEW);  
    ComponentName cp = intent.resolveActivity(pm);  
  
    if(cp != null) {  
        startActivity(intent);  
    }  
}
```

fabien.brissonneau@gmail.com

116

Passer des paramètres à une activité

- ▶ En spécifiant les données et le type
 - Soit plusieurs applications
 - Soit un seule, qui sera ouverte

```
PackageManager pm = getPackageManager();
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(data, "image/*");
ComponentName cp = intent.resolveActivity(pm);

if(cp != null) {
    Log.v("ACTION", "OK composant");
    startActivity(intent);
}
else {
    Log.v("ACTION", "Pas de composant");
```

fabien.brissonneau@gmail.com

117

Contrôles avancés

- ▶ WebView
- ▶ ActionBar
- ▶ NavigationDrawer
- ▶ TabHost
- ▶ ViewPager

fabien.brissonneau@gmail.com

118

WebView

- ▶ Un contrôle pour inclure des pages Web
- ▶ Possibilité de lire des pages locales, distantes...
- ▶ Interactions possible entre JavaScript et les contrôles

```
<WebView  
    android:id="@+id/webview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
/>
```

fabien.brissonneau@gmail.com

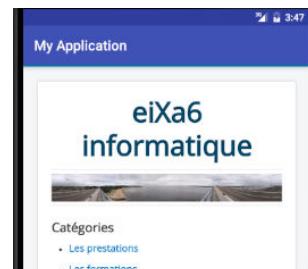
119

Utilisation

- ▶ loadUrl
 - Nécessite d'avoir la permission d'accès à Internet

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

```
-----  
WebView webview = (WebView) view.findViewById(R.id.webview);  
webview.loadUrl("http://www.eixa6.fr");
```



fabien.brissonneau@gmail.com

120

Utilisation de WebView

- ▶ Paramètres
 - Autoriser le zoom
 - Permettre les interactions JavaScript

```
WebSettings ws = webview.getSettings();  
ws.setBuiltInZoomControls(true);  
ws.setJavaScriptEnabled(true);
```

- Retour en arrière

```
@Override  
public boolean onKeyDown(int keyCode, KeyEvent event) {  
  
    if(keyCode == KeyEvent.KEYCODE_BACK && webview.canGoBack()) {  
        webview.goBack();  
        return true;  
    }  
    return super.onKeyDown(keyCode, event);  
}
```



fabien.brissonneau@gmail.com

121

App Bar

- ▶ Une barre placée en haut d'écran
- ▶ API 11 minimum
- ▶ Rassemble les actions importantes de l'application
- ▶ Facilite la navigation
- ▶ Peut comporter :
 - Icône de l'application
 - Navigation
 - Boutons d'actions
 - Actions supplémentaires

fabien.brissonneau@gmail.com

122

Déclaration du menu de l'AppBar

- ▶ Déclarer des items de menu
- ▶ Donner un identifiant
- ▶ Spécifier l'ordre
- ▶ Associer image et texte

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools">
    <item android:id="@+id/action_parametres"
          android:icon="@android:drawable/ic_menu_manage"
          android:orderInCategory="1"
          android:showAsAction="ifRoom"
          android:title="@string/action_parametres"/>
    <item android:id="@+id/action_partage"
          android:icon="@android:drawable/ic_menu_share"
          android:orderInCategory="2"
          android:showAsAction="ifRoom"/>
```

fabien.brissonneau@gmail.com

123

Comportement du menu

- ▶ L'attribut android:showAction
 - ifRoom : item visible seulement s'il y a de la place
 - withText : le texte est placé dans la barre
 - never : ne pas mettre dans la barre
 - always : toujours dans la barre
 - collapseActionView : une action rétractable
- ▶ L'AppBar s'appelait ActionBar
- ▶ Un mauvais affichage peut être dû à des incompatibilités
 - Thème @android:style:Theme.Holo.Light avec showAsAction
 - Ou bien utiliser ToolBar

fabien.brissonneau@gmail.com

124

Mise en place de l'AppBar

- ▶ Redéfinir la méthode onCreateOptionsMenu de l'activité

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



fabien.brissonneau@gmail.com

125

Réagir sur l'AppBar

- ▶ Redéfinir la méthode onOptionsItemSelected

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    int id= item.getGroupId();  
    switch(id) {  
        case R.id.action_cherche :  
            //blah  
            return true;  
        case R.id.action_information :  
            //blah  
            return true;  
        case R.id.action_parametres :  
            //blah  
            return true;  
        case R.id.action_partage :  
            //blah  
            return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

fabien.brissonneau@gmail.com

126

La compatibilité

- ▶ Disposer de la bibliothèque de compatibilité v7
 - Android Support Library dans les Extras du SDK
 - Dépendance de compilation
- ▶ Dériver l'Activity de AppCompatActivity
- ▶ Utiliser le thème AppCompat
 - Theme.AppCompat.Light.DarkActionBar

fabien.brissonneau@gmail.com

127

La navigation et les vues

- ▶ ActionBar permet de naviguer entre vues
- ▶ Suppose une parenté entre les activités

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"
        <category android:name="android.intent.category.LI
    </intent-filter>
</activity>
<activity android:name=".ActiviteEnfant"
    android:parentActivityName=".MainActivity">
</activity>
```

fabien.brissonneau@gmail.com

128

NavigationDrawer

- ▶ Le menu de navigation est situé sur le côté de l'écran
 - Déployé à partir de l'ActionBar
 - En glissant sur le côté
- ▶ Etablir les dépendances

```
compile 'com.android.support:appcompat-v7:23.2.1'  
compile 'com.android.support:support-v4:23.2.1'
```

- ▶ Il faut
 - Une description (layout) pour le menu
 - Une description adaptée pour l'activité

fabien.brissonneau@gmail.com

129

Disposition de l'activité

- ▶ Un espace pour le contenu
 - Sera un fragment
- ▶ Et une liste d'items
 - Items à droite
 - Largeur fixe
 - Choix simple

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.v4.widget.DrawerLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/drawer_layout"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <FrameLayout android:id="@+id/contenu"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"/>  
    <ListView android:id="@+id/list_drawer"  
        android:layout_height="match_parent"  
        android:layout_width="240dp"  
        android:layout_gravity="start"  
        android:choiceMode="singleChoice"  
        android:divider="@android:color/transparent"  
        android:dividerHeight="0dp"  
        android:background="#FFF"/>  
</android.support.v4.widget.DrawerLayout>
```

fabien.brissonneau@gmail.com

130

Constituer le menu

- ▶ Un tableau de chaînes de caractères

```
String[] lesItems = getResources().getStringArray(R.array.menu);
ListView dll = (ListView) findViewById(R.id.list_drawer);
dll.setAdapter(new ArrayAdapter<String>(this, R.layout.drawer, lesItems));
```

- ▶ Disposer chaque item dans la liste

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:textColor="#07F"
    android:fontFamily="sans-serif"
    android:gravity="center_vertical"
    android:textSize="22sp"/>
```

fabien.brissonneau@gmail.com

131

Cohérence

- ▶ Mettre en cohérence l'ActionBar avec le menu
 - Changer le titre de la barre et enlever les actions

```
DrawerLayout dl = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle mDrawerToggle = new ActionBarDrawerToggle(this, dl,
    R.string.app_name, // nav draver open - description for accessibility
    R.string.app_name // nav draver close - description for accessibility
) {
    public void onDrawerClosed(View view) {
        getActionBar().setTitle("Menu global");
        // calling onPrepareOptionsMenu() to show action bar icons
        invalidateOptionsMenu();
    }

    public void onDrawerOpened(View drawerView) {
        getActionBar().setTitle("Menu navigation");
        // calling onPrepareOptionsMenu() to hide action bar icons
        invalidateOptionsMenu();
    }
};

dl.addDrawerListener(mDrawerToggle);
```

fabien.brissonneau@gmail.com

132

Réagir

- ▶ Réagir sur la liste
- ▶ Fermer le tiroir

```
    dll.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Log.v("CLIC", "clic sur "+position);
            DrawerLayout dl = (DrawerLayout) findViewById(R.id.drawer_layout);
            ListView dll = (ListView) findViewById(R.id.list_drawer);
            dl.closeDrawer(dll);
        }
    });
}
```

fabien.brissonneau@gmail.com

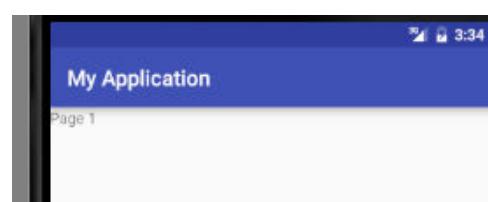
133

ViewPager

- ▶ Un composant graphique de la bibliothèque v4

```
compile 'com.android.support:appcompat-v7:23.2.1'
compile 'com.android.support:support-v4:23.2.1'

<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:id="@+id/viewpager"/>
```



fabien.brissonneau@gmail.com

134

Créer un adaptateur pour alimenter

▶ Dériver FragmentPagerAdapter

```
public class MonAdaptateur extends FragmentPagerAdapter {  
  
    public MonAdaptateur(FragmentManager fm) {  
        super(fm);  
    }  
  
    @Override  
    public Fragment getItem(int position) {  
        return null;  
    }  
  
    @Override  
    public int getCount() {  
        return 0;  
    }  
}
```

fabien.brissonneau@gmail.com

135

Implémenter l'adaptateur, l'utiliser

```
public class MonAdaptateur extends FragmentPagerAdapter {  
  
    String[] pages = {"Page 1", "Page 2", "Page 3"};  
  
    public MonAdaptateur(FragmentManager fm) { super(fm); }  
  
    @Override  
    public Fragment getItem(int position) { return MonFragment.newInstance(pages[position]); }  
  
    @Override  
    public int getCount() { return pages.length; }  
}  
  
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ViewPager vp = (ViewPager) findViewById(R.id.viewpager);  
        vp.setAdapter(new MonAdaptateur(getSupportFragmentManager()));  
    }  
}
```

fabien.brissonneau@gmail.com

136

Disposer le fragment

- ▶ Dérivée de Fragment
 - Support-v4

```
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {  
    View vue = inflater.inflate(R.layout.view_pager,parent,false);  
    return vue;  
}  
  
@Override  
public void onViewCreated(View view, Bundle savedInstanceState) {  
    super.onViewCreated(view,savedInstanceState);  
    TextView msgTextView = (TextView)view.findViewById(R.id.texte);  
    msgTextView.setText(msg);  
}
```

```
public class MonFragment extends Fragment {  
    public String msg;  
  
    public static Fragment newInstance(final String msg) {  
        MonFragment mf = new MonFragment();  
        Bundle bdl = new Bundle(1);  
        bdl.putString("EXTRA_TEXT",msg);  
        mf.setArguments(bdl);  
        return mf;  
    }  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        if(getArguments() != null) {  
            msg = getArguments().getString("EXTRA_TEXT");  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

137

Exercice 5

fabien.brissonneau@gmail.com

138

Evénements généraux

- ▶ Gestion des touches : onKeyDown, onKeyUp

```
public boolean onKeyDown(int keyCode, KeyEvent evt) {
    super.onKeyDown(keyCode, evt);
    if(keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
        Log.v(TAG, "Volume up");
    }
    else if(keyCode == KeyEvent.KEYCODE_ZOOM_IN) {
        Log.v(TAG, "Zoom in");
    }
    return true;
}
```

fabien.brissonneau@gmail.com

139

La saisie

- ▶ Pour surveiller la saisie de texte, implémenter TextWatcher

```
public class MainActivity extends AppCompatActivity implements TextWatcher {

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }

    @Override
    public void afterTextChanged(Editable s) {
    }

    EditText edt = (EditText) findViewById(R.id.edition);
    edt.addTextChangedListener(this);
}
```

fabien.brissonneau@gmail.com

140

Surveiller la saisie

- ▶ Seule afterTextChanged permet de modifier le paramètre

```
@Override  
public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
    Log.v(TAG, "avant " + s.toString());  
}  
  
@Override  
public void onTextChanged(CharSequence s, int start, int before, int count) {  
    Log.v(TAG, "pendant " + s.toString());  
}  
  
@Override  
public void afterTextChanged(Editable s) {  
    TextView texte = (TextView) findViewById(R.id.texte);  
    texte.setText("longueur " + s.toString().length());  
    Log.v(TAG, "longueur " + s.toString());  
}
```

fabien.brissonneau@gmail.com

141

Toucher

- ▶ Implémenter OnTouchListener et abonner

```
public class MainActivity extends AppCompatActivity implements View.OnTouchListener{  
  
    @Override  
    public boolean onTouch(View v, MotionEvent event) {  
        if(event.getAction() == MotionEvent.ACTION_UP) {  
            Log.v(TAG, "Up ! ");  
            return true;  
        }  
        return false;  
    }  
  
    EditText edt = (EditText) findViewById(R.id.edition);  
    edt.setOnTouchListener(this);  
}
```

fabien.brissonneau@gmail.com

142

Réseaux et services Web

- ▶ API réseaux
- ▶ Services Web
- ▶ Traitements asynchrones

fabien.brissonneau@gmail.com

143

API réseaux

- ▶ Les permissions

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    ...
```
- ▶ Vérifier la connexion
 - Utilise CONNECTIVITY_SERVICE
 - Obtention d'un objet NetworkInfo
- ▶ Suivre les changements de connectivité
 - En abonnant un PhoneStateListener

fabien.brissonneau@gmail.com

144

Vérifier la connexion

- ▶ Utiliser getSystemService(CONNECTIVITY_SERVICE)

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);

NetworkInfo infos = cm.getActiveNetworkInfo();

if(infos != null && infos.isConnected()) {
    Toast.makeText(MainActivity.this, "La connexion est OK !", Toast.LENGTH_SHORT).show();
}
else
```

- ▶ Exploiter les types de réseau

- Wifi
- Bluetooth
- Mobile

```
Toast.makeText(this, "Type "+infos.getTypeName(), Toast.LENGTH_SHORT).show();
if(infos.getType() == ConnectivityManager.TYPE_WIFI) {
    Toast.makeText(this, "C'est du wifi !", Toast.LENGTH_SHORT).show();
}
```

fabien.brissonneau@gmail.com

145

Suivre la connectivité

- ▶ Abonner un Listener

```
PhoneStateListener psl = new PhoneStateListener() {
    final String TAG = "RESEAU";
    @Override
    public void onDataConnectionStateChanged(int state) {
        switch(state) {
            case TelephonyManager.DATA_CONNECTED :
                Log.v(TAG, "appareil connecté");
                break;
            case TelephonyManager.DATA_DISCONNECTED :
                Log.v(TAG, "appareil déconnecté");
                break;
            case TelephonyManager.DATA_CONNECTING :
                Log.v(TAG, "appareil en cours de connexion");
                break;
            case TelephonyManager.DATA_SUSPENDED :
                Log.v(TAG, "appareil connecté sans transfert de données");
                break;
        }
        super.onDataConnectionStateChanged(state);
    }
};
```

fabien.brissonneau@gmail.com

146

Services Web

- ▶ Connexion à une URL
 - Grâce à la classe HttpURLConnection
- ▶ Parser du JSON
- ▶ Parser du XML
 - Avec la classe XmlPullParser

fabien.brissonneau@gmail.com

147

Connexion grâce à une URL

- ▶ Ouvrir la connexion et récupérer les données via un flux
- ▶ Interdit dans le thread ihm

```
@Override  
protected String doInBackground(URL... url) {  
    HttpURLConnection cxt = null;  
    try {  
        cxt = (HttpURLConnection)url[0].openConnection();  
        BufferedInputStream bis = new BufferedInputStream(cxt.getInputStream());  
        int inChar;  
  
        while((inChar = bis.read()) != -1) {  
            buffer.append((char)inChar);  
        }  
        cxt.disconnect();  
    }  
    catch(Exception exp) {  
        Log.v(TAG, exp.toString());  
    }  
    return buffer.toString();  
}
```

fabien.brissonneau@gmail.com

148

Parser du JSON

► Le format JSON

```
try {
    JSONObject js = new JSONObject(str);

    JSONObject data = js.getJSONObject("data");

    JSONObject mesure = data.getJSONObject("measurements");

    String vent = mesure.getString("wind_speed_avg");

    Log.v(TAG, "vent " + vent);

} catch (Exception exp) {
    Log.v(TAG, " ERREUR avec JSON " + exp);
}
```

```
{
    "doc": "http://developers.pioupiou.fr/api/live/",
    "license": "http://developers.pioupiou.fr/data-licensing",
    "attribution": "(c) contributors of the Pioupiou wind network
<http://pioupiou.fr>",

    "data": [
        {
            "id": 212,
            "meta": {
                "name": "Pioupiou 212"
            },
            "location": {
                "latitude": 46.697897,
                "longitude": 0.655904,
                "date": "2016-03-11T16:02:38.000Z",
                "success": true
            },
            "measurements": {
                "date": "2016-03-15T13:34:32.000Z",
                "pressure": 1011.4,
                "wind_heading": 67.5,
                "wind_speed_avg": 8.25,
                "wind_speed_max": 14,
                "wind_speed_min": 2.75
            },
            "status": {
                "date": "2016-03-15T13:34:32.000Z",
                "snr": 36.85,
                "state": "on"
            }
        }
    ]
}
```

fabien.brissonneau@gmail.com

149

Parser du XML

► Pour un fichier ou un retour

```
try {
    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
    XmlPullParser parser = factory.newPullParser();

    InputStream in = getApplicationContext().getAssets().open("data.xml");
    parser.setInput(in, null);

    int evt = parser.getEventType();
    while(evt != XmlPullParser.END_DOCUMENT) {
        if(evt == XmlPullParser.START_DOCUMENT) {
            Log.v(TAG, " début du document");
        }
        else if(evt == XmlPullParser.START_TAG) {
            Log.v(TAG, " début de balise");
        }
        else if(evt==XmlPullParser.END_TAG) {
            Log.v(TAG, " fin de balise");
        }
        evt = parser.next();
    }
    in.close();
}
catch(Exception exp) {
    Log.v(TAG, " ERREUR avec XML "+exp);
}
```

fabien.brissonneau@gmail.com

150

Traitements asynchrones

- ▶ L'intérêt est de dégager les threads d'ihm des traitements bloquants
- ▶ La tâche est longue et empêche le retour utilisateur
- ▶ Les APIs peuvent aussi lever des exceptions
 - HttpURLConnection lève NetworkOnMainThreadException
- ▶ Les possibilités de traitements asynchrones
 - AsyncTask
 - IntentServices
 - Looper
 - BroadcastReceiver
 - AlarmManager

fabien.brissonneau@gmail.com

151

AsyncTask

- ▶ Dériver une classe de AsyncTask
 - Spécifier le type des paramètres au lancement de la tâche
 - Spécifier le type du retour du traitement
 - Spécifier le type de la donnée de progression
- ▶ 3 méthodes
 - doInBackground : le traitement à réaliser
 - onProgressUpdate : appelée pour présenter l'avancée du traitement
 - onPostExecute : appelée lorsque le traitement est fini
- ▶ Simple, mais ne survit pas à l'activité

fabien.brissonneau@gmail.com

152

Implémenter AsyncTask

- ▶ Le traitement

```
class MaTache extends AsyncTask<URL,Void,String> {  
    final String TAG = "RESEAU";  
    StringBuilder buffer = new StringBuilder();  
    @Override  
    protected String doInBackground(URL... url) {  
        HttpURLConnection cxt = null;  
        try {  
            cxt = (HttpURLConnection)url[0].openConnection();  
            BufferedInputStream bis = new BufferedInputStream(cxt.getInputStream());  
            int inChar;  
  
            while((inChar = bis.read()) != -1) {  
                buffer.append((char)inChar);  
            }  
            cxt.disconnect();  
        }  
        catch(Exception exp) {  
            Log.v(TAG, exp.toString());  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

153

Dériver AsyncTask

- ▶ La classe dérivée en classe interne
- ▶ Accès aux données privées de la classe englobante

```
        }  
        cxt.disconnect();  
    }  
    catch(Exception exp) {  
        Log.v(TAG, exp.toString());  
    }  
    return buffer.toString();  
}  
  
@Override  
protected void onPostExecute(String result) {  
    TextView resultat = (TextView) findViewById(R.id.resultat);  
    String donnees = buffer.toString();  
    resultat.setText(donnees);  
    parseJSON(donnees);  
}  
}
```

fabien.brissonneau@gmail.com

154

Les services

- ▶ Composants Android sans ihm
- ▶ Survivent plus longtemps que les activités
- ▶ Correspond à
 - Une classe dérivant de Service
 - Une déclaration dans le manifest

```
|     </intent-filter>
|   </activity>
<service android:name=".MonService" />
</application>
```

```
public class MonService extends Service {
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

fabien.brissonneau@gmail.com

155

Lancer un service

- ▶ Utiliser un Intent
- ▶ Le service s'exécute dans le thread de l'activité
- ▶ On peut lié le service à l'activité de façon à les faire interagir

```
Intent getIntent = new Intent(this, MonService.class);
startService(getIntent);
```

fabien.brissonneau@gmail.com

156

Une classe de service

- Hérite de Service
- Méthodes à redéfinir :
 - Méthode onCreate
 - Méthode onStartCommand (démarrage via startService)
 - Méthode onBind (démarrage via bindService)
 - Méthode onDestroy

Cycle de vie 1

- Un service peut être démarré par une activité
- Dans ce cas, il tourne même si l'activité n'est plus en avant plan
- Il est arrêté dans de rares cas
- Un autre composant peut l'arrêter avec stopService
- Lui-même peut s'arrêter avec stopSelf

Cycle de vie 2

- Un service peut être lié par une activité
- Dans ce cas, il est détruit lorsque plus aucune activité n'est liée
- La méthode onStartCommand n'est pas appelée, mais onBind
- L'activité appelante est quelconque

Thread

- Le service tourne dans le thread appelant
- Pour des tâches consommatrices, il est préférable de créer des threads dédiés

Implémenter un IntentService

- ▶ La classe IntentService réalise une tâche dans un « worker thread »
- ▶ Pas de blocage du thread principal

```
public class MonService extends IntentService {  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        final String TAG="INTENTSERVICE";  
        StringBuilder buffer = new StringBuilder();  
        HttpURLConnection cxt = null;  
        try {  
            String str = intent.getStringExtra("url");  
            Log.v(TAG, "URL en cours "+str);  
            URL url = new URL(str);  
            cxt = (HttpURLConnection)url.openConnection();  
            BufferedInputStream bis = new BufferedInputStream(cxt.getInputStream());  
            int inChar;|
```

fabien.brissonneau@gmail.com

161

Lancer un service en passant des arguments

- ▶ Associer des « extras » à l'Intent
- ▶ Les « extras » sont récupérés du côté du service

```
Intent getIntent = new Intent(this, MonService.class);  
getIntent.putExtra("url","http://api.pioupiou.fr/v1/live/212");  
startService(getIntent);
```

fabien.brissonneau@gmail.com

162

Utiliser un PendingIntent

- ▶ Un objet PendingIntent encapsule un Intent
- ▶ Lancement d'un service ...

```
final int GET_REQUEST_CODE=42;
PendingIntent pendingResult = createPendingResult(GET_REQUEST_CODE, new Intent(), 0);
Intent getIntent = new Intent(this, MonService.class);
getIntent.putExtra("url", "http://api.pioupiou.fr/v1/live/212");
getIntent.putExtra("retour", pendingResult);
startService(getIntent);
```

fabien.brissonneau@gmail.com

163

Traitement du PendingIntent

- ▶ Par le service ...

```
@Override
protected void onHandleIntent(Intent intent) {
    final int RESULT_CODE=42;
    final String TAG="INTENTSERVICE";
    StringBuilder buffer = new StringBuilder();
    HttpURLConnection cxt = null;
    try {
        String str = intent.getStringExtra("url");
        PendingIntent retour = intent.getParcelableExtra("retour");

        Log.v(TAG,"URL en cours de traitement"+str);
        URL url = new URL(str);
        cxt = (HttpURLConnection)url.openConnection();

        BufferedReader in = new BufferedReader(
            new InputStreamReader(cxt.getInputStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            buffer.append(inputLine);
        }
        cxt.disconnect();

        Intent result = new Intent();
        String chaineResultante = buffer.toString();
        result.putExtra("retour", chaineResultante);

        retour.send(this, RESULT_CODE, result);
    }
```

fabien.brissonneau@gmail.com

164

Traitement final du retour

- ▶ La méthode createPendingResult suppose la présence de la méthode onActivityResult

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == GET_REQUEST_CODE) {
            switch (resultCode) {

                case 42:
                    String donnees = data.getStringExtra("retour");
                    TextView resultat = (TextView) findViewById(R.id.resultat);
                    resultat.setText(donnees);
                    parseJSON(donnees);
                    break;
            }
        }
    }
```

fabien.brissonneau@gmail.com

165

Les BroadcastReceiver

- ▶ Sert à recevoir des événements

```
public class MonReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //blah
    }
}
```

- ▶ Abonné aux événements à recevoir

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

<receiver android:name=".MonReceiver">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

fabien.brissonneau@gmail.com

166

Implémentation BroadcastReceiver

```
public class MonReceiver extends BroadcastReceiver {  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
  
        if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {  
            Bundle bundle = intent.getExtras();  
            if(bundle != null) {  
                Object[] pdus = (Object[])bundle.get("pdus");  
                final SmsMessage[] messages = new SmsMessage[pdus.length];  
                int pos=0;  
                for(Object msgPdu : pdus) {  
                    messages[pos] = SmsMessage.createFromPdu((byte[])msgPdu, "3gpp");  
                    ++pos;  
                }  
  
                for (SmsMessage message : messages){  
                    final String msgBody = message.getMessageBody();  
                }  
            }  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

167

Enregistrement dynamique

- ▶ Dans les méthodes onResume / onPause

```
@Override  
protected void onResume() {  
    super.onResume();  
    IntentFilter filter= new IntentFilter();  
    filter.addAction("android.provider.Telephony.SMS_RECEIVED");  
    bd = new BroadcastReceiver() {  
        @Override  
        public void onReceive(Context context, Intent intent) {  
            //blah  
        }  
    };  
    registerReceiver(bd, filter);  
}  
  
@Override  
protected void onPause() {  
    super.onPause();  
    unregisterReceiver(bd);  
}
```

fabien.brissonneau@gmail.com

168

La gestion des alarmes

► Utiliser AlarmManager

```
AlarmManager mgr = (AlarmManager) getSystemService(Context.ALARM_SERVICE);  
  
Intent intent = new Intent(getApplicationContext(),MonAlarme.class);  
PendingIntent pi = PendingIntent.getBroadcast(this, 0, intent, 0);  
mgr.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,1000L, pi);
```

► Les modes

- ELAPSED_REALTIME ou RTC
 - Temps écoulé depuis le boot ou bien temps lié à l'horloge
- ELAPSED_REALTIME_WAKEUP ou RTC_WAKEUP
 - L'alarme réveillera l'appareil si besoin

fabien.brissonneau@gmail.com

169

AlarmManager

- Méthodes set, setWindow, setExact
- Méthodes setRepeating,...
- Méthode cancel()
- Mise en oeuvre

```
public class MonAlarme extends BroadcastReceiver {  
  
    </activity>  
    <receiver android:name=".MonAlarme">  
        </application>  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Log.v("RECEIVE", "Bien reçu ");  
    }  
}
```

fabien.brissonneau@gmail.com

170

Exercice 6

fabien.brissonneau@gmail.com

171

Persistance de données

- ▶ Stockage SharedPreferences
 - Propre à l'activity, nommé ou non
 - Propre à l'application, en rapport avec les « PreferencesActivity »
- ▶ Système de fichier
 - Fichiers propres à l'application ou non
- ▶ SQLite
 - Stockage structuré

fabien.brissonneau@gmail.com

172

SharedPreferences

- ▶ Stockage en clés-valeurs
- ▶ Pour des valeurs primitives
- ▶ Persistantes jusqu'à la désinstallation de l'application
- ▶ Possibilités de gérer plusieurs fichiers de préférences
 - getPreferences(mode)
 - getSharedPreferences(nom, mode)
 - MODE_PRIVATE seul utilisable actuellement (API 23)
- ▶ Puis accès aux couples clé-valeur via un Editor
- ▶ Lié aux PreferenceActivity, global à l'application
 - ▶ PreferenceManager.getDefaultSharedPreferences()

fabien.brissonneau@gmail.com

173

SharedPreferences

- ▶ Utiliser apply ou commit pour positionner une valeur
- ▶ Utiliser getxx pour récupérer une valeur

```
SharedPreferences pref = getPreferences(MODE_PRIVATE);
//positionner une valeur
SharedPreferences.Editor editor = pref.edit();
editor.putInt("valeur",maValeur);
boolean ok = editor.commit();
//récupérer une valeur
maValeur = pref.getInt("valeur", 42);
```

fabien.brissonneau@gmail.com

174

Ecran d'accès aux préférences

- ▶ Un fichier XML à décrire dans /xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
<PreferenceCategory android:title="Notifications">
    <SwitchPreference
        android:key="notifications"
        android:title="Autorise les notifications"/>
</PreferenceCategory>
<PreferenceCategory android:title="Données">
    <CheckBoxPreference
        android:summary="Sauvegarde"
        android:title="Autorise les sauvegardes"/>
</PreferenceCategory>
</PreferenceScreen>
```

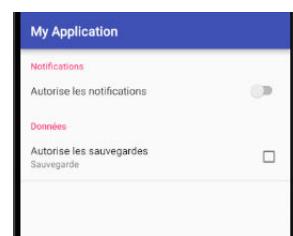
fabien.brissonneau@gmail.com

175

Ecran d'accès aux préférences

- ▶ Dériver une classe de PreferenceFragment

```
public class MesPreferences extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```



- ▶ Et créer une activité qui charge ce fragment

fabien.brissonneau@gmail.com

176

Système de fichier

- ▶ Stockage interne
 - Sauvegarde propre à une application
 - Suppression des fichiers lors de la désinstallation
 - Possibilité d'utiliser un espace temporaire (cache)
- ▶ Stockage externe
 - Externe ou non à l'appareil
 - Disponibilité testable
 - Propre à l'application ou partagé

fabien.brissonneau@gmail.com

177

Stockage interne en écriture

- ▶ Utiliser la méthode openFileOutput

```
public void ecrireLesDonnees(View v) {
    String nomDeFichier = "fichier_ecriture.txt";

    try (FileOutputStream fos = openFileOutput(nomDeFichier, Context.MODE_PRIVATE)) {
        fos.write(donnees.getBytes());
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

fabien.brissonneau@gmail.com

178

Stockage interne en lecture

- ▶ Utiliser la méthode openFileInput

```
String nomDeFichier = "fichier_ecriture.txt";
byte[] buffer = new byte[512];
try (FileInputStream fos = openFileInput(nomDeFichier) ) {
    fos.read(buffer);
    donnees = new StringBuilder().append(new String(buffer)).toString();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

fabien.brissonneau@gmail.com

179

Fichiers de « cache »

- ▶ getCacheDir() donne le répertoire du cache
- ▶ getFilesDir() donne le répertoire des fichiers de l'application
 - /data/data/mon-petit-package/files

```
String nomDeFichier = getCacheDir() + "/fichier_ecriture.txt";
byte[] buffer = new byte[512];
try (FileInputStream fos = openFileInput(nomDeFichier) ) {
    fos.read(buffer);
    donnees = new StringBuilder().append(new String(buffer)).toString();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

fabien.brissonneau@gmail.com

180

Tester le stockage externe

- Le stockage externe peut être présent ou pas

```
public void testerLesDonneesExternes(View v) {  
    String etat = Environment.getExternalStorageState();  
    if(Environment.MEDIA_MOUNTED.equals(etat)) {  
        //blah  
    }  
    else if(Environment.MEDIA_MOUNTED_READ_ONLY.equals(etat)) {  
        //blah  
    }  
    else if(Environment.MEDIA_REMOVED.equals(etat)) {  
        //blah  
    }  
}
```

fabien.brissonneau@gmail.com

181

Utiliser les fichiers de l'application

- Répertoires privés à l'application, disparaissent avec elle
- Utiliser getExternalFilesDir(null)
 - Répertoire racine
- Ou avec paramètres :
 - Environment.DIRECTORY_DCIM, .DIRECTORY_MUSIC,...
- Ne pas oublier les permissions

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

```
public void ecrireLesDonneesExternes(View v){  
    String nomDeFichier = getExternalFilesDir(null)+"/fichier_ecriture.txt";  
  
    try (FileOutputStream fos = new FileOutputStream(nomDeFichier) ) {  
        fos.write(donnees.getBytes());  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

fabien.brissonneau@gmail.com

182

Utiliser des fichiers partagés

- ▶ Les répertoires sont partagés avec les autres applications
 - Le contenu ne disparaît pas lors de la désinstallation
- ▶ Le paramétrage est le même que précédemment

```
File chemin =
    Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

File fichier = new File(chemin, "fichier_ecriture.txt");

try (FileOutputStream fos = new FileOutputStream(fichier)) {
```

fabien.brissonneau@gmail.com

183

Répertoires

Application directories

Method	Result
getCacheDir()	/data/data/package/cache
getFilesDir()	/data/data/package/files
getFilesDir().getParent()	/data/data/package

Application External storage directories

Method	Result
getExternalCacheDir()	/storage/sdcard0/Android/data/package/cache
getExternalFilesDir(null)	/storage/sdcard0/Android/data/package/files
getExternalFilesDir(DIRECTORY_ALARMS)	/storage/sdcard0/Android/data/package/files/Alarms
getExternalFilesDir(DIRECTORY_DCIM)	/storage/sdcard0/Android/data/package/files/DCIM
getExternalFilesDir(DIRECTORY_DOWNLOADS)	/storage/sdcard0/Android/data/package/files/Download
getExternalFilesDir(DIRECTORY_MOVIES)	/storage/sdcard0/Android/data/package/files/Movies
getExternalFilesDir(DIRECTORY_MUSIC)	/storage/sdcard0/Android/data/package/files/Music

External storage directories

Method	Result
Environment.getExternalStorageDirectory()	/storage/sdcard0
Environment.getExternalStoragePublicDirectory(DIRECTORY_ALARMS)	/storage/sdcard0/Alarms
Environment.getExternalStoragePublicDirectory(DIRECTORY_DCIM)	/storage/sdcard0/DCIM
Environment.getExternalStoragePublicDirectory(DIRECTORY_DOWNLOADS)	/storage/sdcard0/Download
Environment.getExternalStoragePublicDirectory(DIRECTORY_MOVIES)	/storage/sdcard0/Movies
Environment.getExternalStoragePublicDirectory(DIRECTORY_MUSIC)	/storage/sdcard0/Music

Exercice 7

fabien.brissonneau@gmail.com

185

SQLite

- Un stockage local en base de données est possible
- Ce stockage est privé à l'application
- Utilise 3 composants
 - Cursor représente l'accès aux données
 - SQLiteDatabase est la base elle-même
 - SQLiteOpenHelper est une classe qui aide à écrire les accès

fabien.brissonneau@gmail.com

186

SQLiteOpenHelper

- ▶ Dériver et implémenter pour créer et mettre à jour

```
public class SQLPersonneHelper extends SQLiteOpenHelper {  
  
    public SQLPersonneHelper(Context context, String name, SQLiteDatabase  
        super(context, name, factory, version);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int n  
    }  
}
```

fabien.brissonneau@gmail.com

187

Exemple d'implémentation

- ▶ Préparer la description de la table

```
public class SQLPersonneHelper extends SQLiteOpenHelper {  
  
    public static final String TABLE = "Personnes";  
    public static final String COL_ID = "id";  
    public static final String COL_NAME ="nom";  
    public static final String COL_FNAME = "prenom";  
  
    private static final String CREATE_BDD = "CREATE TABLE "+  
        TABLE+ " (" +COL_ID+" INTEGER PRIMARY KEY AUTOINCREMENT, "+  
            COL_NAME +" TEXT NOT NULL, "+  
            COL_FNAME +" TEXT NOT NULL );";  
  
    public SQLPersonneHelper(Context context, String name, SQLiteDatabase.Curs  
        super(context, name, factory, version);  
    }
```

fabien.brissonneau@gmail.com

188

Exemple d'implémentation

- ▶ Gérer la création et la mise à jour

```
@Override  
public void onCreate(SQLiteDatabase db) {  
    db.execSQL(CREATE_BDD);  
}  
  
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    db.execSQL("DROP TABLE "+TABLE);  
    onCreate(db);  
}
```

fabien.brissonneau@gmail.com

189

Ecrire les données

```
public void ecrireLesDonnees(View v) {  
    final int VERSION = 1;  
    SQLPersonneHelper personnes = new SQLPersonneHelper(this, "MABASE", null, VERSION);  
    SQLiteDatabase sql = personnes.getWritableDatabase();  
  
    ContentValues valeurs = new ContentValues();  
    valeurs.put(SQLPersonneHelper.COL_NAME, "Brissonneau");  
    valeurs.put(SQLPersonneHelper.COL_FNAME, "Fabien");  
    sql.insert(SQLPersonneHelper.TABLE, null, valeurs);  
  
    sql.close();  
}
```

fabien.brissonneau@gmail.com

190

Lire les données

- ▶ Exécuter une requête via query
- ▶ Possible de lier ListView aux données via des adaptateurs

```
public void lireLesDonnees(View v) {  
    final int VERSION = 1;  
    SQLPersonneHelper personnes = new SQLPersonneHelper(this, "MABASE", null, VERSION);  
    SQLiteDatabase sql = personnes.getReadableDatabase();  
    Cursor c = sql.query(SQLPersonneHelper.TABLE,  
        new String[]{SQLPersonneHelper.COL_ID,SQLPersonneHelper.COL_NAME, SQLPersonneHelper.COL_FNAME},  
        null, null, null, null, null);  
    if(c.getCount() != 0) {  
        ListView liste = (ListView) findViewById(R.id.donnees);  
        liste.setAdapter(new MonAdaptateur(this, c, 0));  
    }  
    else  
        c.close();  
}
```

fabien.brissonneau@gmail.com

191

Création d'un adaptateur

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <TextView  
        android:id="@+id/nom"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
    <TextView  
        android:id="@+id/prenom"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
</LinearLayout>
```

▶ Colonne identité

- _id (ici COL_ID)

```
public class MonAdaptateur extends CursorAdapter {  
    public MonAdaptateur(Context context, Cursor cursor, int flags) {  
        super(context, cursor, 0);  
    }  
    @Override  
    public View newView(Context context, Cursor cursor, ViewGroup parent) {  
        return LayoutInflater.from(context).inflate(R.layout.item_personne, parent, false);  
    }  
    @Override  
    public void bindView(View view, Context context, Cursor cursor) {  
        TextView txtnom = (TextView) view.findViewById(R.id.nom);  
        TextView txtprenom = (TextView) view.findViewById(R.id.prenom);  
        String nom = cursor.getString(cursor.getColumnIndexOrThrow(SQLPersonneHelper.COL_NAME));  
        String prenom = cursor.getString(cursor.getColumnIndexOrThrow(SQLPersonneHelper.COL_FNAME));  
        txtnom.setText(nom);  
        txtprenom.setText(prenom);  
    }  
}
```

fabien.brissonneau@gmail.com

192

Implémenter un Content Provider

- ▶ Permet de partager les données
- ▶ Le système lui-même partage des informations via des ContentProvider
- ▶ Doit proposer un URI
 - Par exemple « content://autorité/table/id »
- ▶ Des méthodes CRUD (insert, query, update, delete)
- ▶ Il faut déclarer le fournisseur dans le manifest
 - Mettre en relation la classe et le nom logique de l'autorité

```
<provider android:name="android.ei6.biz.applibasededonnees.MonContentProvider"  
|         android:authorities="eixa6" />
```

fabien.brissonneau@gmail.com

193

Implémenter un Content Provider

- ▶ Dériver une classe de ContentProvider
- ▶ Implémenter les méthodes proposées

```
public class MonContentProvider extends ContentProvider {  
  
    @Override  
    public boolean onCreate() {  
        return false;  
    }  
  
    @Nullable  
    @Override  
    public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {  
        return null;  
    }  
}
```

fabien.brissonneau@gmail.com

194

Exemple d'implémentation

- ▶ Ici le nom de la table (dans l'uri) n'est pas utilisé

```
public Uri insert(Uri uri, ContentValues contentValues) {  
    SQLiteDatabase db=dbHelper.getWritableDatabase();  
    try {  
        long id = db.insertOrThrow(SQLPersonneHelper.TABLE,null,contentValues);  
        return ContentUris.withAppendedId(uri,id);  
    }  
    finally {  
        db.close();  
    }  
}
```

fabien.brissonneau@gmail.com

195

Exemple de requête

- ▶ Suivant l'Uri, la requête est exécutée différemment

```
public Cursor query(Uri uri, String[] proj, String sel, String[] selArgs, String ordre) {  
    long id = getId(uri);  
    SQLiteDatabase db=dbHelper.getReadableDatabase();  
    if(id<0) {  
        return db.query(SQLPersonneHelper.TABLE,proj,  
                        sel,selArgs,null,null,ordre);  
    }  
    else {  
        return db.query(SQLPersonneHelper.TABLE,proj,  
                        SQLPersonneHelper.COL_ID+"="+id,selArgs,null,null,null);  
    }  
}
```

fabien.brissonneau@gmail.com

196

Utiliser un Content Provider

- ▶ Exemple correspondant au code précédent

```
ContentValues personne = new ContentValues();
personne.put(SQLPersonneHelper.COL_NAME, "Bazin");
personne.put(SQLPersonneHelper.COL_FNAME, "Marie");
Uri uri = getContentResolver().insert(MonContentProvider.CONTENT_URI, personne);
```

- ▶ Ici l'uri est

```
public static final Uri CONTENT_URI = Uri.parse("content://eixa6/personnes");
```

Utiliser une requête sur ContentProvider

- ▶ La requête retourne un curseur

```
public void lireLesDonneesProvider(View v) {
    Cursor c = getContentResolver().query(MonContentProvider.CONTENT_URI, null, null, null, null);
    if(c.getCount() != 0) {
        ListView liste = (ListView) findViewById(R.id.donnees);
        liste.setAdapter(new MonAdaptateur(this, c, 0));
    }
    else
        c.close();
}
```

Accéder aux Contacts

- ▶ Utiliser le Content Provider des contacts

```
private Cursor getContacts() {  
  
    Uri uri = ContactsContract.Contacts.CONTENT_URI;  
    String[] projection = new String[] { ContactsContract.Contacts._ID,  
                                         ContactsContract.Contacts.DISPLAY_NAME };  
    String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '"  
        + ("1") + "'";  
    String[] selectionArgs = null;  
    String sortOrder = ContactsContract.Contacts.DISPLAY_NAME  
        + " COLLATE LOCALIZED ASC";  
  
    return getContentResolver().query(uri, projection, selection, selectionArgs,  
                                     sortOrder);  
}
```

fabien.brissonneau@gmail.com

199

Utiliser le curseur

- ▶ Parcourir les contacts et mettre leurs noms dans une liste

```
public void listeContacts(View v) {  
    ArrayList<String> contacts = new ArrayList<>();  
    Cursor cursor = getContacts();  
  
    while (cursor.moveToNext()) {  
  
        String displayName = cursor.getString(cursor  
            .getColumnIndex(ContactsContract.Data.DISPLAY_NAME));  
        contacts.add(displayName);  
    }  
    ListView liste = (ListView) findViewById(R.id.liste);  
    ArrayAdapter<String> adaptateur =  
        new ArrayAdapter<String>(getApplicationContext(),  
                               android.R.layout.simple_list_item_1, contacts);  
    liste.setAdapter(adaptateur);  
}
```

fabien.brissonneau@gmail.com

200

Exercice 8

fabien.brissonneau@gmail.com

201

API et fonctionnalités multimédia

- ▶ Affichage de document
- ▶ Prise de photo
- ▶ Sons et micro
- ▶ Envoi et réception de SMS
- ▶ GoogleMap et GPS
- ▶ Push notification

fabien.brissonneau@gmail.com

202

Prise de photos

- ▶ Utiliser l'application Camera
- ▶ Pour démarrer une activité et récupérer un résultat
 - Utiliser startActivityForResult
- ▶ Pour s'assurer de la présence d'un appareil photo

```
<uses-feature android:name="android.hardware.camera2"/>
```

- ▶ Pour les permissions sur l'appareil photo et les fichiers

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

Prévisualiser la caméra

- ▶ Lancer

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

if(intent.resolveActivity(getApplicationContext())!=null) {
    startActivityForResult(intent, CODE_PREVIEW_IMAGE);
}
```

- ▶ Récupérer

```
if( requestCode == CODE_PREVIEW_IMAGE && resultCode == RESULT_OK) {
    Bundle extras = data.getExtras();
    Bitmap imageBitmap = (Bitmap) extras.get("data");
    detailImage.setImageBitmap(imageBitmap);
}
```

Stocker sous forme de fichier

- ▶ Il faut donner un nom au fichier image
 - ▶ Puis lancer le stockage

```
if(intent.resolveActivity(getApplicationContext())!=null) {  
  
    fichierPhoto = null;  
    try {  
        fichierPhoto = donneUnNomDeFichier();  
    }  
    catch(IOException io) {  
        Toast.makeText(getApplicationContext(), "Pas de carte", Toast.LENGTH_LONG).show();  
    }  
    if(fichierPhoto != null) {  
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(fichierPhoto));  
        startActivityForResult(intent, CODE_CAPTURE_IMAGE);  
    }  
    else if( requestCode == CODE_CAPTURE_IMAGE && resultCode == RESULT_OK) {  
  
        Intent intent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);  
        Uri contenu = Uri.fromFile(fichierPhoto);  
        intent.setData(contenu);  
        sendBroadcast(intent);  
    }  
}
```

fabien.brissonneau@gmail.com

205

Sons et micro

- ▶ Enregistrer le son et stocker dans un fichier
 - ▶ Nécessite les permissions

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

- #### ▶ Utiliser la classe MediaRecorder

```
private MediaRecorder enregistreur;  
private String nomDeFichier;
```

- ## ▶ Initialiser

```
nomDeFichier = Environment.getExternalStorageDirectory().getAbsolutePath() + "/enr.3gp";

enregistreur = new MediaRecorder();
enregistreur.setAudioSource(MediaRecorder.AudioSource.MIC);
enregistreur.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
enregistreur.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB);
enregistreur.setOutputFile(nomDeFichier);
```

fabien.brissonneau@gmail.com

206

Enregistrement

- ▶ Lancement et arrêt
 - Utiliser prepare() et start()
 - Utiliser release() et stop()

```
try {  
    enregistreur.prepare();  
}  
catch(IOException io) {  
    Log.v("MIC", "Probleme..." + io);  
}  
enregistreur.start();  
Log.v("MIC", "En cours d'enregistrement");  
  
enregistreur.stop();  
enregistreur.release();  
Log.v("MIC", "Enregistrement arrêté");
```

fabien.brissonneau@gmail.com

207

Restitution

- ▶ Utiliser prepare() et play()
- ▶ Levée d'exceptions à prévoir

```
MediaPlayer mdp = new MediaPlayer();  
mdp.setDataSource(nomDeFichier);  
mdp.prepare();  
mdp.start();
```

fabien.brissonneau@gmail.com

208

Envoi et réception de SMS

- ▶ Récupérer des informations sur l'appareil
- ▶ Gérer les appels
- ▶ Gérer les messages

Récupérer des informations

- ▶ TelephonyManager est un service système
- ▶ Les accès sont soumis à autorisation

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

TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
int typeTelephone = tm.getPhoneType();
Log.v("TEL", "Type " + typeTelephone);
String codeIMEI = tm.getDeviceId();
Log.v("TEL", "IMEI " + typeTelephone);
String numeroTelephone = tm.getLine1Number();
Log.v("TEL", "Numéro " + typeTelephone);
int etatDonnees = tm.getDataState();
Log.v("TEL", "Etat données " + typeTelephone);
String operateur = tm.getSimOperator();
Log.v("TEL", "Opérateur " + typeTelephone);
```

Sur les autorisations (Android6)

- ▶ Il faut demander les autorisations lors de l'accès au service

```
public void recupererLesInfos(View v) {  
  
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE)  
        == PackageManager.PERMISSION_GRANTED) {  
        accedeAuTelephone();  
    }  
    else {  
        Log.v("TELEPHONE","Pas de permission");  
        ActivityCompat.requestPermissions(this,  
            new String[]{Manifest.permission.READ_PHONE_STATE}, 42);  
    }  
}  
  
@Override  
public void onRequestPermissionsResult(int requestCode,  
    @NotNull String[] permissions,  
    @NotNull int[] grantResults) {  
  
    if (requestCode == 42) {  
        if (grantResults.length == 1 &&  
            grantResults[0] == PackageManager.PERMISSION_GRANTED) {  
            Log.v("TELEPHONE","ok ..");  
            accedeAuTelephone();  
        } else {  
            Log.v("TELEPHONE","toujours pas ...");  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

211

Gérer les appels

- ▶ Tester la disponibilité de la fonctionnalité
 - Peut passer par la déclaration <uses-feature ...>

```
PackageManager packageManager = getPackageManager();  
boolean telephoneDispo = packageManager.hasSystemFeature(PackageManager.FEATURE_TELEPHONY);
```

- ▶ Passer un appel

```
Intent appel = new Intent(Intent.ACTION_CALL, Uri.parse("tel:0674641636"));  
startActivity(appel);
```

- ▶ En version 6, penser à tester les permissions

fabien.brissonneau@gmail.com

212

Gérer les appels

- ▶ Soit utiliser un BroadcastReceiver
- ▶ Soit créer un PhoneStateListener

```
PhoneStateListener psl = new PhoneStateListener() {  
    @Override  
    public void onCallStateChanged(int state, String numero) {  
        super.onCallStateChanged(state,numero);  
        switch(state) {  
            case TelephonyManager.CALL_STATE_IDLE:  
                Log.v("ENTREES","rien à faire...");  
                break;  
            case TelephonyManager.CALL_STATE_OFFHOOK:  
                Log.v("ENTREES","en cours de comm");  
                break;  
            case TelephonyManager.CALL_STATE_RINGING:  
                Log.v("ENTREES","appel entrant");  
                break;  
        }  
    };  
    TelephonyManager telephony =  
        (TelephonyManager) this.getSystemService(Context.TELEPHONY_SERVICE);  
    telephony.listen(psl, PhoneStateListener.LISTEN_CALL_STATE);  
};
```

fabien.brissonneau@gmail.com

213

Gérer les messages

- ▶ 2 possibilités
 - Passer par l'application de message
 - Envoyer un message en direct
- ▶ Envoyer un message avec l'application message

```
Intent envoi = new Intent(Intent.ACTION_SENDTO, Uri.parse("sms:0674641636"));  
envoi.putExtra("sms_body","Envoi d'un SMS");  
startActivity(envoi);
```

fabien.brissonneau@gmail.com

214

Envoyer directement un SMS

- ▶ Déclarer la permission SEND_SMS

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

SmsManager mgr = SmsManager.getDefault();
mgr.sendTextMessage("0674641636", null, "Envoi d'un message", null, null);
```

- ▶ Rappel : les permissions ne sont plus accordées de la même façon en Android 6

Envoyer un SMS : plusieurs parties

- ▶ Le message est composé à partir d'une liste
- ▶ Cette liste peut être automatiquement constituée

```
SmsManager mgr = SmsManager.getDefault();
ArrayList<String> messages = new ArrayList<>();
messages = mgr.divideMessage("Ceci est un long long message qui sera divisé ...");
mgr.sendMultipartTextMessage("0674641636", null, messages, null, null);
```

Réceptionner un SMS

▶ Déclarer la permission

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

▶ Déclarer un receiver

```
<!-- -->
<receiver android:name=".MonReceiver" android:exported="false">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

▶ Créer la classe du receiver

```
public class MonReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.provider.Telephony.
```

fabien.brissonneau@gmail.com

217

Implémenter la réception

```
public class MonReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
            Bundle bundle = intent.getExtras();
            if(bundle != null) {
                Object[] pdus = (Object[])bundle.get("pdus");
                final SmsMessage[] messages = new SmsMessage[pdus.length];
                int pos=0;
                for(Object msgPdu : pdus) {
                    messages[pos] = SmsMessage.createFromPdu((byte[])msgPdu);
                    ++pos;
                }

                for (SmsMessage message : messages){
                    final String msgBody = message.getMessageBody();
                    final String msgPhonenb = message.getDisplayOriginatingAddress();

                    Log.v("BROADCAST", msgBody + " " + msgPhonenb);
                }
            }
        }
    }
}
```

fabien.brissonneau@gmail.com

218

Exercice 9

fabien.brissonneau@gmail.com

219

GoogleMap et GPS

► Utiliser les Google Play Services

```
compile 'com.google.android.gms:play-services:8.4.0'
```

► Déclarer les activités

```
<category android:name="android.intent.category.LAUNCHER">
</category>
<activity
    android:name=".MapsActivity"
    android:label="@string/title_activity_maps">
</activity>
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

► Utiliser une clé

- Voir /res/values/google_maps_api.xml

fabien.brissonneau@gmail.com

220

Utiliser un fragment

- ▶ Implémenter l'interface OnMapReadyCallback

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {  
  
    private GoogleMap mMap;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_maps);  
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
            .findFragmentById(R.id.map);  
        mapFragment.getMapAsync(this);  
    }  
}
```

fabien.brissonneau@gmail.com

221

GoogleMap et marqueurs

- ▶ Lorsque la carte est obtenue
- ▶ Placer des marqueurs

```
@Override  
public void onMapReady(GoogleMap googleMap) {  
    mMap = googleMap;  
  
    LatLng rochefort = new LatLng(45.9497, -0.9879);  
  
    mMap.addMarker(new MarkerOptions().position(rochefort).title("Marque sur Rochefort"));  
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(rochefort, 15.0f));  
}
```

fabien.brissonneau@gmail.com

222

Services de localisation

- ▶ Soit utiliser directement les senseurs de l'appareil
 - GPS
- ▶ Soit passer par les Google Play Services
 - Maps
 - Fit
 - Analytics
 - Drive
 - Ads
 - Wear
 - ...

fabien.brissonneau@gmail.com

223

La localisation par Google Services

- ▶ Déclarer les permissions

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

- ▶ Créer un GoogleApiClient

```
if (mGoogleApiClient == null) {  
    mGoogleApiClient = new GoogleApiClient.Builder(this)  
        .addConnectionCallbacks(this)  
        .addOnConnectionFailedListener(this)  
        .addApi(LocationServices.API)  
        .build();  
}
```

fabien.brissonneau@gmail.com

224

Le client se connecte

- Gérer la connexion et la déconnexion

```
@Override  
protected void onStart() {  
    mGoogleApiClient.connect();  
    super.onStart();  
}  
  
@Override  
protected void onStop() {  
    mGoogleApiClient.disconnect();  
    super.onStop();  
}
```

fabien.brissonneau@gmail.com

225

Implémentations nécessaires

- La disponibilité du service de localisation
- Les erreurs de connexion

```
public class MainActivity extends AppCompatActivity  
    implements LocationListener ,  
    GoogleApiClient.ConnectionCallbacks,  
    GoogleApiClient.OnConnectionFailedListener  
{
```

fabien.brissonneau@gmail.com

226

Utiliser le GoogleApiClient

- ▶ Implémentation de onConnected
- ▶ Pour la dernière localisation connue

```
@Override  
public void onConnected(Bundle bundle) {  
  
    if (checkSelfPermission(  
        Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED)  
    {  
        mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);  
  
        if (mLastLocation != null) {  
            EditText latitudeText = (EditText) findViewById(R.id.editText);  
            EditText longitudeText = (EditText) findViewById(R.id.editText2);  
  
            latitudeText.setText(String.valueOf(mLastLocation.getLatitude()));  
            longitudeText.setText(String.valueOf(mLastLocation.getLongitude()));  
        }  
    }  
}
```

fabien.brissonneau@gmail.com

227

Suivi de la localisation

- ▶ Il faut enregistrer une requête

```
LocationServices.FusedLocationApi.  
    requestLocationUpdates(mGoogleApiClient, mLocationRequest, this);
```

- ▶ Suivi continu
 - Coûteux

```
LocationRequest mLocationRequest;  
  
protected void creerLaDemande() {  
    mLocationRequest = new LocationRequest();  
    mLocationRequest.setInterval(10000);  
    mLocationRequest.setFastestInterval(5000);  
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);  
}  
  
@Override  
public void onLocationChanged(Location location) {  
    EditText latitudeText = (EditText) findViewById(R.id.editText);  
    EditText longitudeText = (EditText) findViewById(R.id.editText2);  
  
    latitudeText.setText(String.valueOf(location.getLatitude()));  
    longitudeText.setText(String.valueOf(location.getLongitude()));  
}
```

fabien.brissonneau@gmail.com

228

Retrouver une adresse

```
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;

public void trouverUneAdresse() throws IOException {
    Geocoder geocoder = new Geocoder(this, Locale.getDefault());
    List<Address> addresses = geocoder.getFromLocation(
        mLastLocation.getLatitude(),
        mLastLocation.getLongitude(),
        1);
}
```

fabien.brissonneau@gmail.com

229

Utiliser le GPS de l'appareil

► LocationManager

```
Criteria criteria = new Criteria();
locationManager = (LocationManager) getActivity().getSystemService(Context.LOCATION_SERVICE);
provider = locationManager.getBestProvider(criteria, false);
```

```
Location courante = locationManager.getLastKnownLocation(provider);

if( courante == null) {
    courante =new Location("");
    courante.setLatitude(45.771974);
    courante.setLongitude(-1.148018);
}
```

► Location

► Et autres ...

```
Location eixa6 = new Location("");
eixa6.setLatitude(45.950010);
eixa6.setLongitude(-0.987354);

float distance = courante.distanceTo(eixa6);

distance = (int) (distance/1000);
final TextView tv = (TextView) getActivity().findViewById(R.id.textView);
tv.setText("Vous êtes à "+distance+" km de eixa6 !");
```

fabien.brissonneau@gmail.com

230

Notification

▶ Paramétrage par un Builder

```
Notification.Builder bld = new Notification.Builder(MainActivity.this)
    .setSmallIcon(R.drawable.eixa6)
    .setContentTitle("eixa6 vous signale")
    .setContentText("Une info est arrivée...");
```

▶ Affichage

```
NotificationManager mgr = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mgr.notify(1, bld.build());
```

fabien.brissonneau@gmail.com

231

Lier la notification à l'activité

▶ Permettre un accès à l'activité

- Back retourne sur le

```
Intent ri= new Intent(MainActivity.this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(
    MainActivity.this,
    0,
    ri,
    PendingIntent.FLAG_UPDATE_CURRENT
);
bld.setContentIntent(pi);
```

▶ Avec prise en charge de la pile des activités

```
Intent ri= new Intent(MainActivity.this, MainActivity.class);
TaskStackBuilder tsk = TaskStackBuilder.create(MainActivity.this);
tsk.addParentStack(MainActivity.class);
tsk.addNextIntent(ri);
PendingIntent pi = tsk.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
bld.setContentIntent(pi);
```

fabien.brissonneau@gmail.com

232

Déploiement

- ▶ Prérequis au déploiement
- ▶ GooglePlay
- ▶ Internationalisation

fabien.brissonneau@gmail.com

233

Le déploiement

- ▶ Problème des rétro-compatibilités
- ▶ Publication
- ▶ Génération

fabien.brissonneau@gmail.com

234

Problème des rétro-compatibilités

- ▶ Utilisation des bibliothèques de support v7
- ▶ Tester le niveau du SDK utilisé pour compiler

fabien.brissonneau@gmail.com

235

Publication

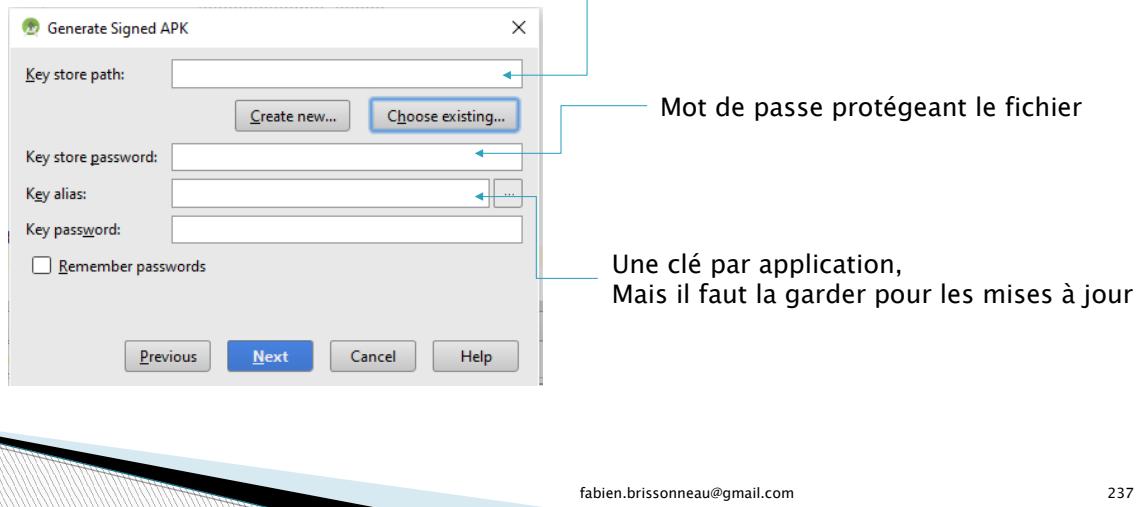
- ▶ Eviter tout le code destiné au mode debug
 - Log.d n'est pas utilisé, seuls restent Log.e Log.i, Log.w
 - Enlever Log.v
- ▶ L'application doit déclarer label et icon
- ▶ L'application doit déclarer toutes ses permissions

fabien.brissonneau@gmail.com

236

Génération

- Il faut générer un package signé



GooglePlay

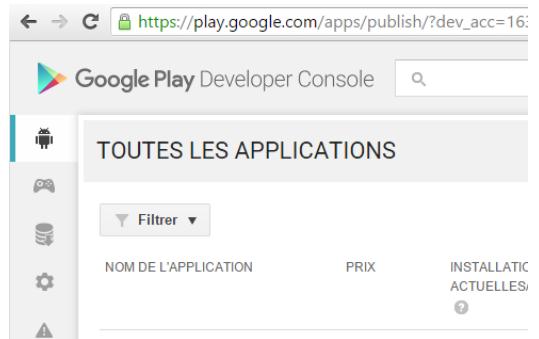
- Il faut créer un compte de développeur sur Google Play

Register for a Publisher Account

- Visit the [Google Play Developer Console](#).
- Enter basic information about your **developer identity** — name, email address, and so on. You can modify this information later.
- Read and accept the [Developer Distribution Agreement](#) for your country or region. Note that apps and store listings that you publish on Google Play must comply with the Developer Program Policies and US export law.
- Pay a \$25 USD [registration fee](#) using Google payments. If you don't have

Publier une application

- ▶ Charger le fichier apk
- ▶ Fournir des descriptions
 - Courte et longue
- ▶ Fournir 2 copies d'écran
 - Voir plus
- ▶ Une image haute définition
 - 512x512
- ▶ Une image de présentation



fabien.brissonneau@gmail.com

239

En entreprise

- ▶ Distribution par mail ou site web
 - Un lien vers le fichier apk
- ▶ Recopie de fichier
- ▶ Problème potentiel de recopies

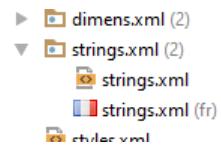
fabien.brissonneau@gmail.com

240

L'internationalisation

- ▶ Les ressources sont localisées en fonction
 - La valeur par défaut (values/strings.xml)
 - La valeur localisée (values/fr/strings.xml)
- ▶ Pour tester, changer la langue de l'émulateur

```
C:\Users\fabien\AppData\Local\Android\sdk\platform-tools>adb shell  
root@generic_x86:/ # setprop persist.sys.locale fr-CA;stop;sleep 5;start  
root@generic_x86:/ # setprop persist.sys.locale fr-FR; stop; sleep 5; start  
root@generic_x86:/ # setprop persist.sys.locale en-US; stop; sleep 5; start  
root@generic_x86:/ # [ ]
```



fabien.brissonneau@gmail.com

241

Fin



fabien.brissonneau@gmail.com

242

Annexes

- ▶ Senseurs et permissions
- ▶ Fragments et layouts
- ▶ Menus, drawer et webservices

fabien.brissonneau@gmail.com

243

Compléments Android

Senseurs, Permissions

Sommaire

- ▶ Les senseurs
- ▶ Le positionnement
- ▶ Les permissions Android 6

Les senseurs

- ▶ Lister les senseurs
- ▶ Utiliser un senseur

Lister les senseurs

- ▶ Récupérer un accès au service

```
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- ▶ Trouver la liste des senseurs disponibles

```
List<Sensor> laListe = sensorManager.getSensorList(Sensor.TYPE_ALL);  
  
for(Sensor sensor : laListe) {  
    Log.v(TAG,sensor.getName());  
}
```

Utiliser le senseur d'orientation

- ▶ Pour connaître l'orientation de l'appareil
- ▶ Enregistrer un listener

```
sensorManager.registerListener(new SensorEventListener() {  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
        if(event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {  
            textViewX.setText(Float.toString(event.values[0]));  
        }  
    }  
},
```

- ▶ Spécifier type et délai

```
    sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),  
    SensorManager.SENSOR_DELAY_NORMAL);  
},
```

Enregistrement / déenregistrement

- ▶ Enregistrer le listener dans onResume
 - Consomme des ressources
- ▶ Déenregistrer le listener dans onPause

Le positionnement

- ▶ Utiliser soit LocationManager, soit les services Google
- ▶ Avec LocationManager,
 - Récupération de la dernière position connue
 - Mise à jour des données
 - Abonnement à la position

```
Location loc = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

- ▶ Les permissions précision fine ou grossière

```
import static android.Manifest.permission.ACCESS_FINE_LOCATION;
import static android.content.pm.PackageManager.PERMISSION_GRANTED;

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

Les permissions au runtime

- ▶ Depuis Android 6
- ▶ Tester si on a la permission

```
if (ActivityCompat.checkSelfPermission(this, ACCESS_FINE_LOCATION) != PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this, ACCESS_COARSE_LOCATION) != PERMISSION_GRANTED) {  
    requestPermissions(new String[] {ACCESS_FINE_LOCATION}, REQUESTE_POSITION);  
}  
return;  
}
```

- ▶ Sinon, la demander

```
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {  
    if(requestCode == REQUESTE_POSITION) {  
        if(grantResults[0]==PERMISSION_GRANTED) {  
            recupererLaPosition();  
        }  
    }  
}
```

Mise à jour des coordonnées

- ▶ Mise à jour une fois

```
locationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, new LocationListener() {  
    @Override  
    public void onLocationChanged(Location loc) {  
        if (loc != null) {  
            latitudeTxt2.setText(Double.toString(loc.getLatitude()));  
            longitudeTxt2.setText(Double.toString(loc.getLongitude()));  
        }  
    }  
});
```

- ▶ Ou en continu
 - requestLocationChanges

Compléments Android

Fragments, Layout

Sommaire

- ▶ Usage des fragments
- ▶ Prise en charge automatique des layout

Usage des fragments

- ▶ Définition
- ▶ Layout principal
- ▶ Changement de fragment

Définition

- ▶ Un Fragment est une sous-partie d'activité
 - Une classe Java
 - Un layout

```
public class PremierFragment extends Fragment {  
  
    public PremierFragment() {  
        // Required empty public constructor  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                            Bundle savedInstanceState) {  
        View view = inflater.inflate(R.layout.fragment_premier, container, false);  
  
        return view;  
    }  
}
```

Layout principal

- ▶ Les fragments remplacent une partie du layout principal
 - Soit directement, avec la balise fragment
 - Utiliser android:name pour spécifier la classe de fragment
 - Soit dynamiquement
 - Réserver la place

```
<FrameLayout  
    android:id="@+id/partiel1"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
</FrameLayout>
```

Remplacer un fragment

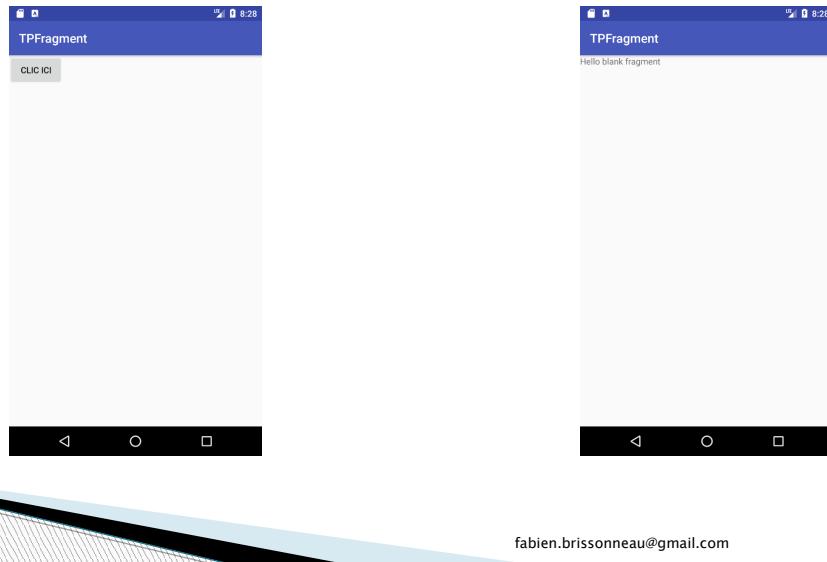
- ▶ Utiliser le FragmentManager

```
FragmentManager mgr = getSupportFragmentManager();  
mgr.beginTransaction().replace(R.id.partiel1, new PremierFragment()).commit();
```

- ▶ Sur clic bouton :

```
FragmentManager mgr = getSupportFragmentManager();  
mgr.beginTransaction().replace(R.id.partiel1, new SecondFragment()).commit();
```

Résultat



fabien.brissonneau@gmail.com

259

Prise en charge automatique du layout

- ▶ Créer un layout spécifique
- ▶ Déetecter un layout

Créer un layout spécifique

- ▶ Par exemple un layout pour mode paysage

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/parent">

    <fragment
        android:name="android.ei6.biz.tpfragment.PremierFragment"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"/>

    <fragment
        android:name="android.ei6.biz.tpfragment.SecondFragment"
        android:id="@+id/fragment2"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"/>

</LinearLayout>
```

DéTECTER le fragment

- ▶ Ne pas charger dynamiquement le fragment s'il est déjà là

```
Fragment secondFragment = (Fragment)mgr.findFragmentById(R.id.fragment2);

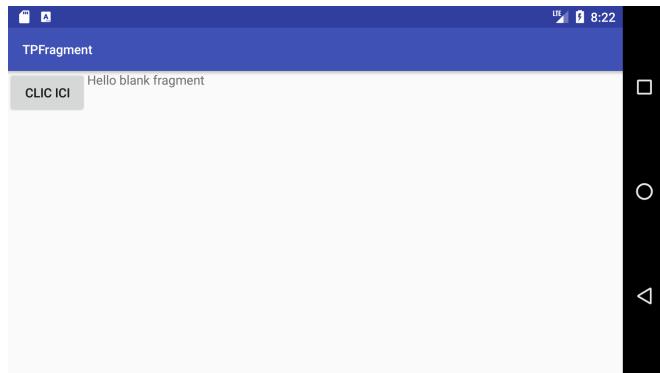
if(secondFragment == null) {
    mgr.beginTransaction().replace(R.id.partie1, new PremierFragment()).commit();
}
```

- ▶ Ou bien

```
View v = findViewById(R.id.partie1);

if(v != null) {
    mgr.beginTransaction().replace(R.id.partie1, new PremierFragment()).commit();
}
```

Résultat



Animations sur transactions

- ▶ Il est possible d'animer le changement entre les fragments

```
FragmentTransaction tr = mgr.beginTransaction();
tr.setCustomAnimations(android.R.anim.slide_out_right, android.R.anim.slide_in_left);
tr.replace(R.id.partiel, new SecondFragment());
tr.commit();
```

Compléments Android

Menus, Drawer et WebServices

Sommaire

- ▶ Barre de menus
- ▶ Navigation drawer
- ▶ Executer un webservice

Barre de menus

- ▶ Créer le fichier de description de menu
- ▶ Faire apparaître les menus
- ▶ Traiter l'utilisation des items

Créer le fichier de menu

- ▶ Créer le répertoire res/menu
- ▶ Créer un fichier xml contenant la description du menu
- ▶ Pour chaque item
 - Id : l'identifiant
 - Title : le nom de l'entrée
 - Icon : l'icon associée
 - onClick : la méthode correspondante
 - showAsAction : visibilité de l'item

Le fichier de menu

- ▶ Ce fichier s'appelle menu_principal.xml

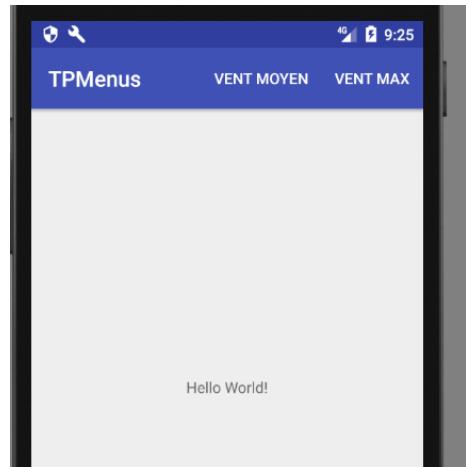
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/vent_moyen"
        android:title="@string/vent_moyen"
        app:showAsAction="always"/>
    <item
        android:id="@+id/vent_max"
        android:title="@string/vent_max"
        app:showAsAction="always"/>
</menu>
```

Faire apparaître les menus

- ▶ Redéfinir la méthode onCreateOptionsMenu(Menu menu)

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

Affichage



Traiter la sélection

- ▶ Redéfinir la méthode `onOptionsItemSelected(MenuItem item)`

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch( item.getItemId() ) {  
        case R.id.vent_moyen :  
            return true;  
        case R.id.vent_max :  
            return true;  
        default :  
            return super.onOptionsItemSelected(item);  
    }  
}
```

Navigation drawer

- ▶ Préparer le layout principal
- ▶ Charger le drawer
- ▶ Réagir sur les items sélectionnés

Préparer le layout principal

- ▶ Utiliser un android.support.v4.widget.DrawerLayout
- ▶ Attention à la compatibilité

- ▶ Créer la place pour le contenu principal

- ▶ Un widget pour le menu drawer

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:labelText="android.eie.biz.tpnavigation.MainActivity"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/emplacement_principal"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </FrameLayout>

    <ListView
        android:id="@+id/drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start">
    </ListView>
```

Préparer un layout pour l'item

- ▶ Dans un fichier de layout séparé

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

Charger le drawer

- ▶ Le plus simple est de mettre en place une ListView affichant des TextView avec des chaînes de caractères

```
<resources>
    <string name="app_name">TPnavigation</string>
    <string-array name="drawer">
        <item>Vent moyen</item>
        <item>Vent rafale</item>
    </string-array>
</resources>
```

- ▶ Lier le drawer avec les items (dans onCreate)

```
ListView drawer = (ListView) findViewById(R.id.drawer);
String[] lesItems = getResources().getStringArray(R.array.drawer);
drawer.setAdapter(new ArrayAdapter<String>(this,R.layout.drawer_item,lesItems));
```

Réagir sur la sélection

- ▶ Enregistrer un listener sur le drawer

```
drawer.setOnItemClickListener(new ListView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        //  
    }  
});
```

Fermer le drawer

- ▶ Le DrawerLayout a un id

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.v4.widget.DrawerLayout  
    android:id="@+id/principal"  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match parent"
```

- ▶ Récupérer cet id

- ▶ Utiliser

```
drawerLayout = (DrawerLayout) findViewById(R.id.principal);  
  
public void onItemClick(AdapterView<?> parent, View view, int pc  
    drawerLayout.closeDrawer(android.view.Gravity.START);  
}
```

Gérer un bouton Home

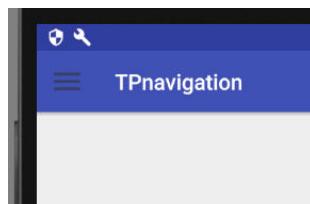
- ▶ Pour ouvrir le drawer ou permettre la navigation en retour
- ▶ Changer l'icône
- ▶ Réagir sur l'activation

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_action_name);
```

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    int id = item.getItemId();  
  
    if(id == android.R.id.home) {  
        if(drawerLayout.isDrawerOpen(Gravity.START))  
            drawerLayout.closeDrawer(Gravity.START);  
        else  
            drawerLayout.openDrawer(Gravity.START);  
  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

La bouton de navigation

- ▶ Ici un drawable menu, à remplacer au besoin



Executer un webservice

- ▶ Executer un appel web
- ▶ Parser le format JSON
- ▶ Traiter l'asynchrone
- ▶ Afficher les données

Executer un appel web

- ▶ Plusieurs possibilités
- ▶ Choix ici de java.net

```
URL url = new URL("http://api.pioupiou.fr/v1/live/512");
BufferedReader in = new BufferedReader(
    new InputStreamReader(
        url.openStream()));

String str = "";
String inputLine;
while ((inputLine = in.readLine()) != null)
    str += inputLine;
```

- ▶ Penser aux permissions

Parser JSON

- ## ▶ Utiliser org.json.JSONObject

```
JSONObject obj = new JSONObject(str);
in.close();
JSONObject data = obj.getJSONObject("data");
JSONObject arr = data.getJSONObject("measurements");
vent_moyen = arr.getString("wind speed avg");
```

Traiter asynchrone

- ▶ Plusieurs possibilités
 - ▶ Si AsyncTask, alors implémenter
 - doInBackground
 - onPostExecute

```
new AsyncTask<Void, Void, String>() {
    @Override
    protected String doInBackground(Void... params) {
        String vent_moyen="";
        try {
            URL url = new URL("http://api.pioupiou.fr/v1/live/512");
            BufferedReader in = new BufferedReader(

```

Afficher les données

- ▶ Accès aux éléments d'ihm

```
@Override  
protected void onPostExecute(String valeur) {  
    TextView texte = (TextView) findViewById(R.id.texte);  
    texte.setText(valeur);  
}
```