Licence 3 MAI Mathématiques Appliquées à l'info (MAI)

Développement mobile

Séquence 2: Initiation à la programmation Android

Mr. Boubou CAMARA

Plan

Séquence 2 : Initiation à la programmation Android

II.0 Introduction
II.1 Notions essentielles
2.1.1 activités (activity)
2.1.2 intentions (intent)
2.1.3 services (service)
2.1.4 fournisseur de contenu (content provider)

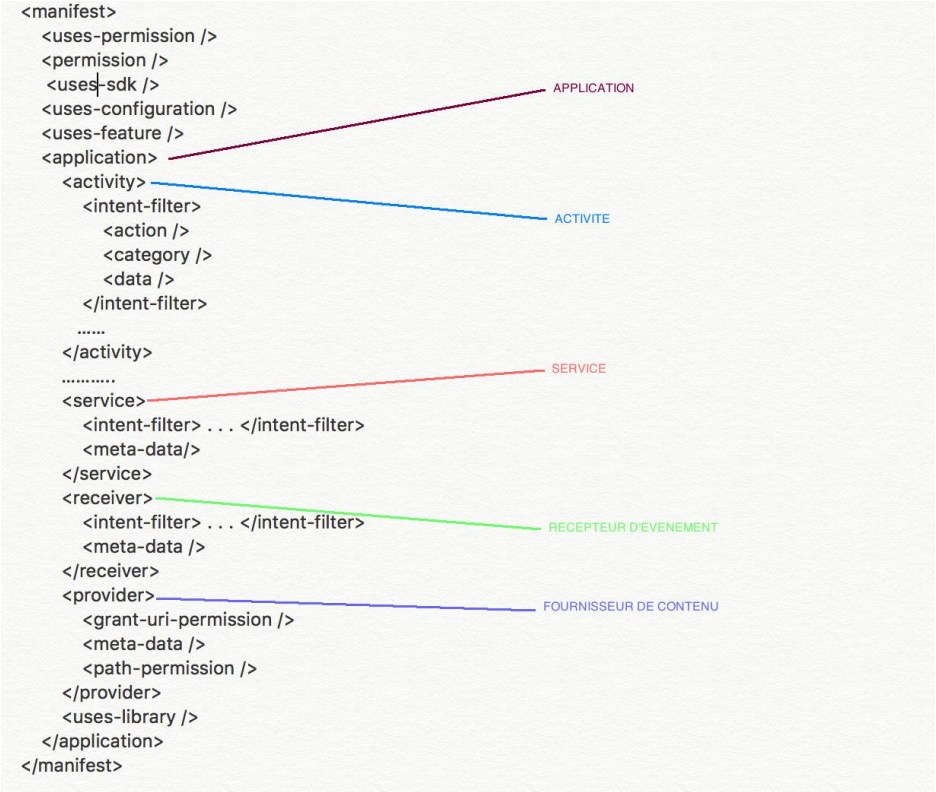
II.2 La structure d'un projet android
2.2.1 Cas de l'application XamSaGox
2.2.1 Description des différents éléments de XamSaGox

Introduction - Développement Android - Notions essentielles

Les blocs de base ou les composants fondamentaux de Android sont :

- AndroidManifest.xml
- Les permissions (élément <uses-permission android:name=« nomPermission"/>)
- les activités (**Activity**, package **android.app**)
- les intentions (**Intent**, package **android.content**)
- les services (**Service**, package **android.app**)
- ~~les fournisseurs de contenu (**ContentProvider**, package **android.app**)~~
- ~~les Récepteurs d'Evénements (**BroadcastReceiver**, package **android.app**)~~
- ~~les fragments (**Fragment**)~~

Développement Android - Notions essentielles - Le manifeste




```
<manifest>
  <uses-permission />
  <permission />
   <uses-sdk />
  <uses-configuration />
  <uses-feature />
  <application>
    <activity>
      <intent-filter>
        <action />
        <category />
        <data />
      </intent-filter>
      ……
    </activity>
    ……
    <service>
      <intent-filter> . . . </intent-filter>
      <meta-data/>
    </service>
    <receiver>
      <intent-filter> . . . </intent-filter>
      <meta-data />
    </receiver>
    <provider>
      <grant-uri-permission />
      <meta-data />
      <path-permission />
    </provider>
    <uses-library />
  </application>
</manifest>
```

AndroidManifest.xml

Le fichier AndroidManiftest.xml est le fichier manifeste du projet Android, qui contient la configuration principale de l'application. On y déclare les différents composants de l'application tels que: les activités, les services, les récepteurs d'événement, les fournisseurs de contenu, etc.

Le manifest contient également les déclarations optionnels des intentions implicites (**Implicit Intent**) associés aux composants (voir section sur les intentions pour les définitions) et d'autres types d'informations tels que les permissions, et les contraintes de permission… Voir la documentation de android à l'adresse https://developer.android.com/guide/topics/manifest/manifest-intro.html#filestruc pour avoir les détails de tous les éléments constitutifs du fichier de manifest.

L'élément élément <uses-permission android:name=« nomPermission »/> permet de demander, à l'exécution, à l'utilisateur une autorisation qui doit être accordée à l'application pour qu'elle fonctionne correctement.

Exemples :

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

Développement Android - notions essentielles - « Activity »

définition: Une activité est une instance de « **Activity** », une classe dans le SDK Android. Une activité est responsable de la gestion des interactions de l'utilisateur avec un écran de information.

Toute application, à l'exception des application de type services, a au moins une activité dont la première est celle qu'on voit au démarrage de l'application.

Déclaration et Implantation d'une activité

L'implantation d'une activité se fait en :

- créant une classe qui étend android.app.Activity
- déclarant l'activité dans le fichier de manifeste
- création du fichier de layout associé à l'activité

Fort heureusement, Android Studio crée le squelette de l'activité, une entrée de l'Activité dans le manifeste est le fichier de vue dont le nom est bâti à partir du non de l'activité - d'un seul coup.

Le développeur n'a alors qu'à compléter les items. Voir la page suivante pour un exemple d'un projet android studio.

Développement Android - notions essentielles - « Activity »

déclaration de l'activité dans le manifeste

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <!-- The meta-data tag is required if you support API level 15 and lower -->
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

Développement Android - notions essentielles - « Activity »

déclaration de la vue associée à l'activité

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout 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"
    tools:context="sn.uvs.mysecondapplication.MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/quel_est_ton_nom"
        />
    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="41dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text=""
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Développement Android - notions essentielles - « Activity »

déclaration de la vue associée à l'activité (suite 2/2)

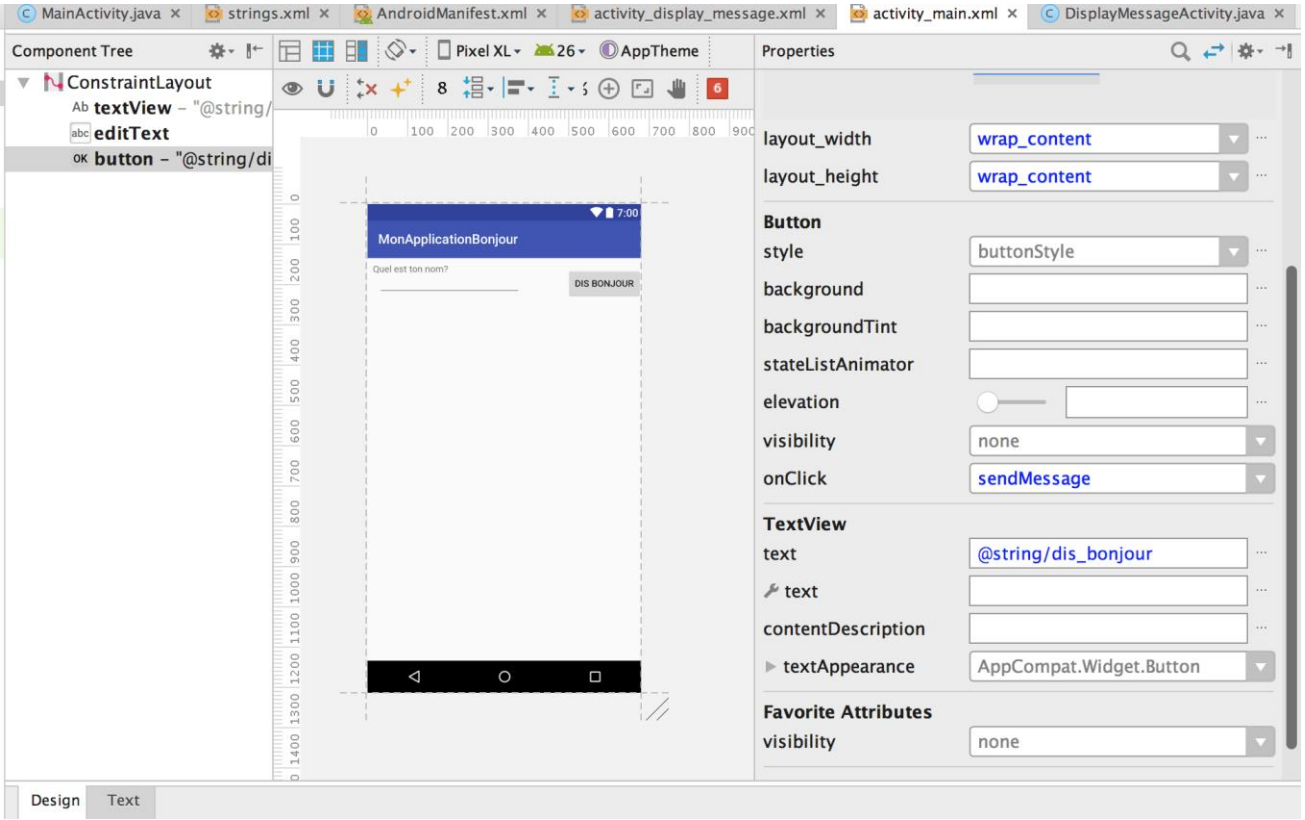
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="0dp"
     android:keyboardNavigationCluster="false"
     android:onClick="sendMessage"
     android:text="@string/dis_bonjour"
     app:layout_constraintBaseline_toBaselineOf="@+id/editText"
     app:layout_constraintRight_toRightOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Développement Android - notions essentielles - « Activity »

déclaration de la classe de l'activité

```
9   import java.sql.Timestamp;
10  import java.util.Calendar;
11  import java.util.Date;
12  import java.util.GregorianCalendar;
13
14  public class MainActivity extends AppCompatActivity {
15      public static final String EXTRA_MESSAGE = "sn.uvs.mysecondapp.MESSAGE";
16      @Override
17      protected void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_main);
20      }
21      /** Invoqué quand l'utilisateur clique sur le bouton « Dis bonjour! » */
22      public void sendMessage(View view) {
23          Intent intent = new Intent(this, DisplayMessageActivity.class);
24          EditText editText = (EditText) findViewById(R.id.editText);
25          String inputMessage = (editText.getText().toString().isEmpty())?"A vous":editText.getText().toString();
26          String message="";
27          Date dateAndTimeNow = new Date();
28          Date midi = new Date();
29          Calendar calendar = GregorianCalendar.getInstance();
30          int timeOfDay = calendar.get(GregorianCalendar.HOUR_OF_DAY);
31          if (timeOfDay >=12) {
32              message = "Bonsoir "+inputMessage +"!";
33          } else {
34              message = "Bonjour "+inputMessage+"!";
35          }
36          intent.putExtra(EXTRA_MESSAGE, message);
37          startActivity(intent);
38      }
39  }
```

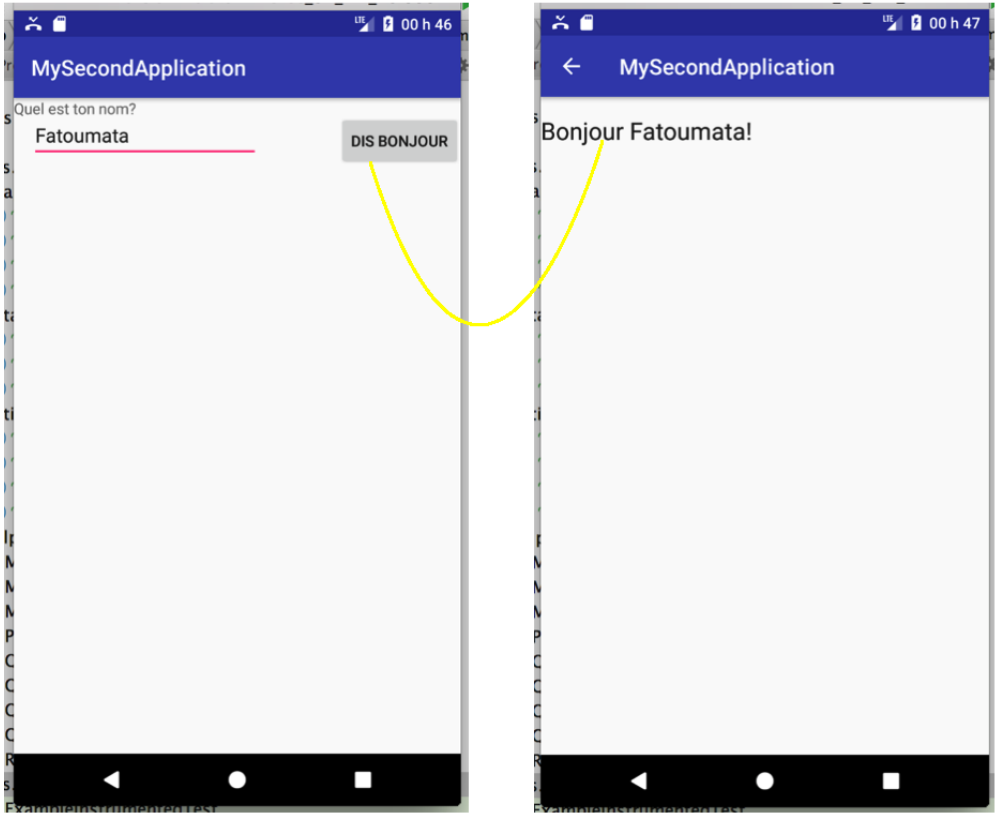
notions essentielles - « Activity » - Vue associée



déclaration de la classe de l'activité

En utilisant l'interface graphique de Android Studio, on peut associer l'événement onClick du bouton « Dis Bonjour » à la méthode sendMessage de la classe MainActivité.

Notions essentielles - Exemple de projet avec 2 activités



Execution du projet

A chacune des fenêtre est associé une activité:

1) la fenêtre principale correspond à l'activité principale MainActivity
2) la seconde correspond à l'activité DisplayMessageActivity
3) La transition entre les deux activités est assurée par l'intention créée à la ligne 10 du diapo précédent. (Le concept d'Intention sera abordé dans les pages suivantes).

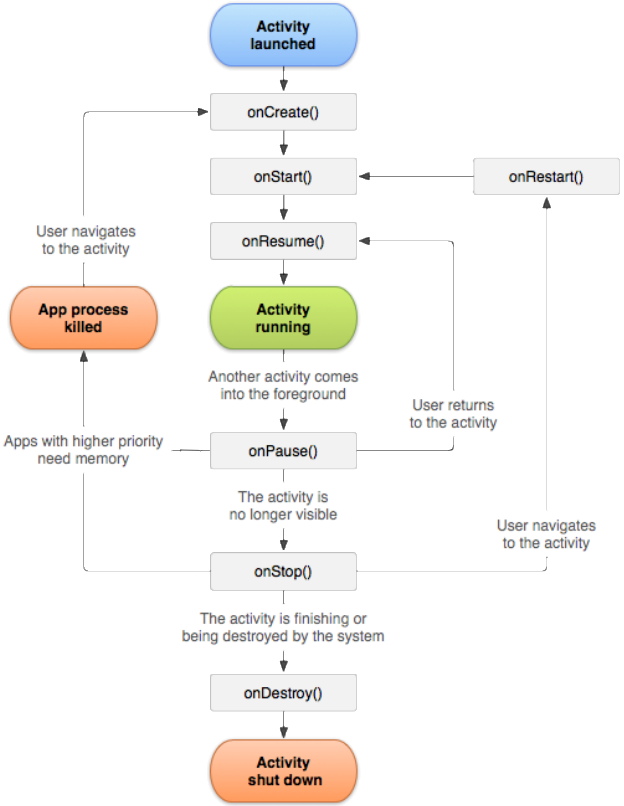
Développement Android - Activité - cycle de vie

Cycle de vie d'une activité

Le cycle de vie d'une activité voit son état transiter entre les quatre valeurs ci-dessous. A chaque transition, une méthode de sa classe est invoquée par le Système d'exploitation qui lui notifie ce changement.

Etat	signification
nonExistant	N'est pas dans la mémoire et n'est donc pas visible - et certainement pas actif dans le premier plan.
arrêté	Une activité arrêtée n'est pas visible sur l'écran: elle est en arrière-plan et est susceptible d'être supprimé par le système lorsque sa mémoire est nécessaire. Une activité est arrêtée quand une autre activité entre au premier plan et devient active. Par exemple, lorsque vous répondez à un appel, l'application du téléphone devient active et l'application précédemment utilisée est arrêtée.
en pause	une activité en pause visible sur l'écran mais n'a pas le focus; comme quand une boîte de dialogue d'alerte est affichée. L'usager ne peut pas interagir avec l'activité interrompue jusqu'à ce qu'elle devienne actif - par exemple, après qu'il ait fermé une boîte de dialogue d'alerte.
en reprise	Une activité active est visible à l'écran et "a le focus", c'est-à-dire est au premier plan. Vous pouvez interagir avec l'activité actuellement au premier plan

Développement Android - Activité - cycle de vie



Transitions entre les états d'une activité

A mesure qu'une activité transite entre entre ses différents états, le moteur d'exécution Android appelle diverses méthodes du cycle de vie de l'activité, qui sont toutes définies par la classe Activity dans le package android.app.

Méthode	signification
onCreate()	Invoqué quand Called when the activity is first created
onStart()	Appelé juste après sa création ou par la méthode de redémarrage après onStop (). Ici l'activité commence devenir visible pour l'utilisateur
onResume()	Appelé lorsque l'activité est visible pour l'utilisateur et que l'utilisateur peut interagir avec celui-ci
onPause()	Appelé lorsque le contenu de l'activité n'est pas visible car l'utilisateur reprend l'activité précédente
onStop()	Appelé lorsque l'activité n'est pas visible pour l'utilisateur car une autre activité l'a remplacé
onRestart()	Appelé lorsque l'utilisateur arrive à l'écran ou reprend l'activité qui a été arrêtée
onDestroy()	Appelé lorsque l'activité n'est pas en arrière-plan

Android - notions essentielles - Les intentions ou « Intent »

La classe Intent (Intention en français) est une classe qui permet de faire communiquer des composantes Android entre elles : à l'intérieur d'une même application ou entre des composantes d'application différentes.

Dans le projet MySecondApplication de la page 11,
- l'instruction **Intent intent = new Intent(this, DisplayMessageActivity.class);**
crée une intention dans l'activité MainActivity, à passer la main à l'activity **DisplayMessageActivity**
- l'instruction **intent.putExtra(EXTRA_MESSAGE, message);**
stocke de l'information dans l'intention, en vue d'être utilisé dans l'activité destination.
- l'instruction **startActivity(intent) :** fait s'executer l'activité cible.

Android - notions essentielles - Les types d'intentions

Il existe deux types d'intention :

- **intention explicite**, telle qu'on l'a vu dans l'exemple de l'application MySecondApplication; ici on précise, lors de la création de l'intention, le composant qu'on cible : **Intent intent = new Intent(this, DisplayMessageActivity.class)**
- **intention implicite :** pour une intention implicite par contre, le composant source ne précise pas le composant cible qu'il veut faire démarrer, mais plutôt l'action qu'il veut faire faire, ainsi que le type de donnée que le composant cible doit gérer; et le système Android se charge de trouver le composant qui remplit ces critères; quand il y' a plus de deux composants, le système Android donne le choix à l'utilisateur.

Exemples d'actions d' intention

Action	Description	Data URI Scheme
Intent.ACTION_VIEW	Visualiser quelque chose	
Intent.ACTION_CALL	Appeler un numéro de téléphone	

Android - notions essentielles - Intention Implicite - Exemple

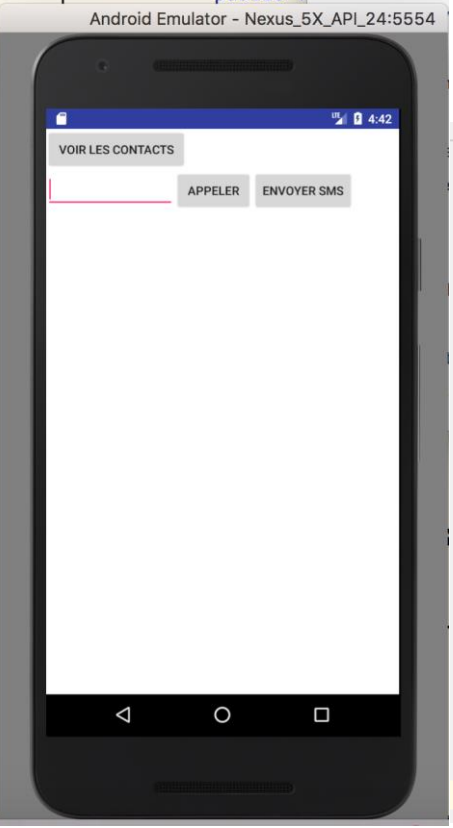
Exemple d'application utilisant trois intention implicites

dans ce qui va suivre on va concevoir et implanter une application android qui permet de réaliser 3 actions :

- Voir les contacts
- Envoyer un SMS vers un numéro de téléphone donné
- Appeler un numéro de téléphone donné

Action	Description	Data URI Scheme
Intent.ACTION_VIEW	Visualiser quelque chose	
Intent.ACTION_CALL	Appeler un numéro de téléphone	

Android - notions essentielles - Intention Implicite - Exemple



Exemple d'application utilisant trois intention implicites

1. Créer un nouveau projet Android Studio avec son activité MainActivity
2. mettre à jour le manifest avec le contenu ci-dessous

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="sn.uvs.devmobile.implicitintentapplication">
4      <uses-permission android:name="android.permission.READ_CONTACTS"/>
5      <uses-permission android:name="android.permission.CALL_PHONE"/>
6      <uses-permission android:name="android.permission.SEND_SMS"/>
7      <application
8          android:allowBackup="true"
9          android:icon="@mipmap/ic_launcher"
10         android:label="ImplicitIntentApplication"
11         android:roundIcon="@mipmap/ic_launcher_round"
12         android:supportsRtl="true"
13         android:theme="@style/AppTheme">
14         <activity android:name=".MainActivity">
15             <intent-filter>
16                 <action android:name="android.intent.action.MAIN" />
17
18                 <category android:name="android.intent.category.LAUNCHER" />
19             </intent-filter>
20         </activity>
21     </application>
22
23 </manifest>
```

Android - notions essentielles - Intention Implicite - Exemple

Les lignes :
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
demandent à l'usager du téléphone ou l'application est installée l'autorisation pour l'application, de lire ses contacts, d'utiliser son téléphone pour appeler ou pour envoyer des sms.

3. remplacer le contenu de la vue activity_main.xml de l'activité par le code suivant:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff"
    android:orientation="vertical" >
    <Button
        android:id="@+id/btnLireContacts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Voir les contacts " />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <EditText
            android:id="@+id/numeroTelephone"
            android:layout_width="150dip"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/btnAppeler"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=" Appeler " />

        <Button
            android:id="@+id/btnSms"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Envoyer SMS" />
</LinearLayout>
</LinearLayout>
```

Android - notions essentielles - Intention Implicite - Exemple

4. remplacer le contenu de l'activité principale par la suivante:

```
14  public class MainActivity extends Activity {
15      Button lireContacts,appeler,envoyerSms;
16      EditText phone;
17      @Override
18      public void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21          lireContacts=(Button)findViewById(R.id.btnLireContacts);
22          lireContacts.setOnClickListener((v) → {
25                  Intent intentionVue=new Intent();
26                  intentionVue.setAction(android.content.Intent.ACTION_VIEW);
27                  intentionVue.setData(ContactsContract.Contacts.CONTENT_URI);
28                  startActivity(intentionVue);
29          });
31          phone=(EditText)findViewById(R.id.numeroTelephone);
32          appeler=(Button)findViewById(R.id.btnAppeler);
33          appeler.setOnClickListener((v) → {
36                  try {
37                  String uri = "tel:" + phone.getText().toString();
38                  Intent callIntent = new Intent(Intent.ACTION_CALL, Uri.parse(uri));
39                  startActivity(callIntent);
40                  } catch (SecurityException se) {
41                      Toast.makeText(getApplicationContext(), "Votre appel a échoué...", Toast.LENGTH_LONG).show();
42                      se.printStackTrace();
43                  }
44          });
46          envoyerSms=(Button)findViewById(R.id.btnSms);
47          envoyerSms.setOnClickListener((v) → {
50                  Intent sms=new Intent();
51                  String uri = "smsto:" + phone.getText().toString();
52                  Intent viewIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
53                  startActivity(viewIntent);
54          });
56      }
57  }
```

Android - notions essentielles - Intention Implicite - Exemple

5. Essayer les 3 fonctions : LireContacts, Envoyer SMS et AppelerTelephone

```
14  public class MainActivity extends Activity {
15      Button lireContacts,appeler,envoyerSms;
16      EditText phone;
17      @Override
18      public void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21          lireContacts=(Button)findViewById(R.id.btnLireContacts);
22          lireContacts.setOnClickListener((v) → {
25                  Intent intentionVue=new Intent();
26                  intentionVue.setAction(android.content.Intent.ACTION_VIEW);
27                  intentionVue.setData(ContactsContract.Contacts.CONTENT_URI);
28                  startActivity(intentionVue);
29          });
31          phone=(EditText)findViewById(R.id.numeroTelephone);
32          appeler=(Button)findViewById(R.id.btnAppeler);
33          appeler.setOnClickListener((v) → {
36                  try {
37                  String uri = "tel:" + phone.getText().toString();
38                  Intent callIntent = new Intent(Intent.ACTION_CALL, Uri.parse(uri));
39                  startActivity(callIntent);
40                  } catch (SecurityException se) {
41                      Toast.makeText(getApplicationContext(), "Votre appel a échoué...", Toast.LENGTH_LONG).show();
42                      se.printStackTrace();
43                  }
44          });
46          envoyerSms=(Button)findViewById(R.id.btnSms);
47          envoyerSms.setOnClickListener((v) → {
50                  Intent sms=new Intent();
51                  String uri = "smsto:" + phone.getText().toString();
52                  Intent viewIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
53                  startActivity(viewIntent);
54          });
56      }
57  }
```

Android - notions essentielles - « Service »

Définition

Un service Android est un composant d'application sans interface utilisateur qui exécute des tâches en arrière-plan. Contrairement aux activités, les services n'ont pas d'interface utilisateur visuelle. Ils sont utilisés pour implémenter des opérations d'arrière-plan de longue durée ou une API de communication riche qui peut être appelée par d'autres applications.

Service VS Thread

Un service est simplement un composant qui peut s'exécuter en arrière-plan, même si l'utilisateur n'interagit pas avec votre application. Vous devez donc créer un service uniquement si c'est ce dont vous avez besoin.

Si une application doit effectuer un travail en dehors de son thread principal, mais seulement lorsque l'utilisateur interagit avec l' application, vous devez créer un nouveau thread.

Android - notions essentielles - « Service »

Déclaration de service dans le manifeste

SYNTAX:

```
<service android:description="string resource"
         android:directBootAware=["true" | "false"]
         android:enabled=["true" | "false"]
         android:exported=["true" | "false"]
         android:icon="drawable resource"
         android:isolatedProcess=["true" | "false"]
         android:label="string resource"
         android:name="string"
         android:permission="string"
         android:process="string" >
    . . .
</service>
```

CONTENU DANS :

<application>

PEUT CONTENIR:

<intent-filter>

<meta-data>

Déclare un service (une sous-classe de service) comme l'un des composants de l'application. Tous les services doivent être représentés par des éléments <service> dans le fichier manifeste. Tout ce qui n'est pas déclaré ne sera pas vu par le système et ne sera jamais exécuté.

Android - notions essentielles - « Service »

Caractéristiques de services

- Service Démarré (Started Service)

Un service démarré est un service qui est lancé par d'autres composants d'application, tels qu'une activité ou un récepteur de diffusion. Ils peuvent fonctionner indéfiniment en arrière-plan jusqu'à ce qu'ils soient arrêtés ou détruits par le système pour libérer des ressources.

- Service Lié (Bound Service)

Un service lié est similaire à un service démarré, sauf qu'un service démarré ne renvoie généralement pas de résultats ou n'autorise pas l'interaction avec le composant qui l'a lancé. D'un autre côté, un service lié permet au composant de lancement (client) d'interagir avec le service et de recevoir des résultats.

Ainsi, par exemple, un service lié peut lire un fichier audio et envoyer des données concernant le démarrage / la pause / l'arrêt audio et le temps écoulé jusqu'au composant Activité de lancement afin que l'interface utilisateur puisse être mise à jour en conséquence.

- Service Local (Local Service)

- Service Distant (Remote Service)

C’est un service lancé dans un autre processus, différent de celui qui a démarré notre application.

Pour communiquer avec un tel service, on peut utiliser l'AIDL (Android Interface Definition

Language). Nous n’aborderons pas ce type de service dans ce cours.

Android - notions essentielles - « Service »

Caractéristiques de services

- Service Démarré (Started Service)

Un service démarré est un service qui est lancé par d'autres composants d'application, tels qu'une activité ou un récepteur de diffusion. Ils peuvent fonctionner indéfiniment en arrière-plan jusqu'à ce qu'ils soient arrêtés ou détruits par le système pour libérer des ressources.

- Service Lié (Bound Service)

Un service lié est similaire à un service démarré, sauf qu'un service démarré ne renvoie généralement pas de résultats ou n'autorise pas l'interaction avec le composant qui l'a lancé. D'un autre côté, un service lié permet au composant de lancement (client) d'interagir avec le service et de recevoir des résultats.

Ainsi, par exemple, un service lié peut lire un fichier audio et envoyer des données concernant le démarrage / la pause / l'arrêt audio et le temps écoulé jusqu'au composant Activité de lancement afin que l'interface utilisateur puisse être mise à jour en conséquence.

- Service Local (Local Service)

- Service Distant (Remote Service)

C’est un service lancé dans un autre processus, différent de celui qui a démarré notre application.

Pour communiquer avec un tel service, on peut utiliser l'AIDL (Android Interface Definition

Language). Nous n’aborderons pas ce type de service dans ce cours.

Android - notions essentielles - « Service »

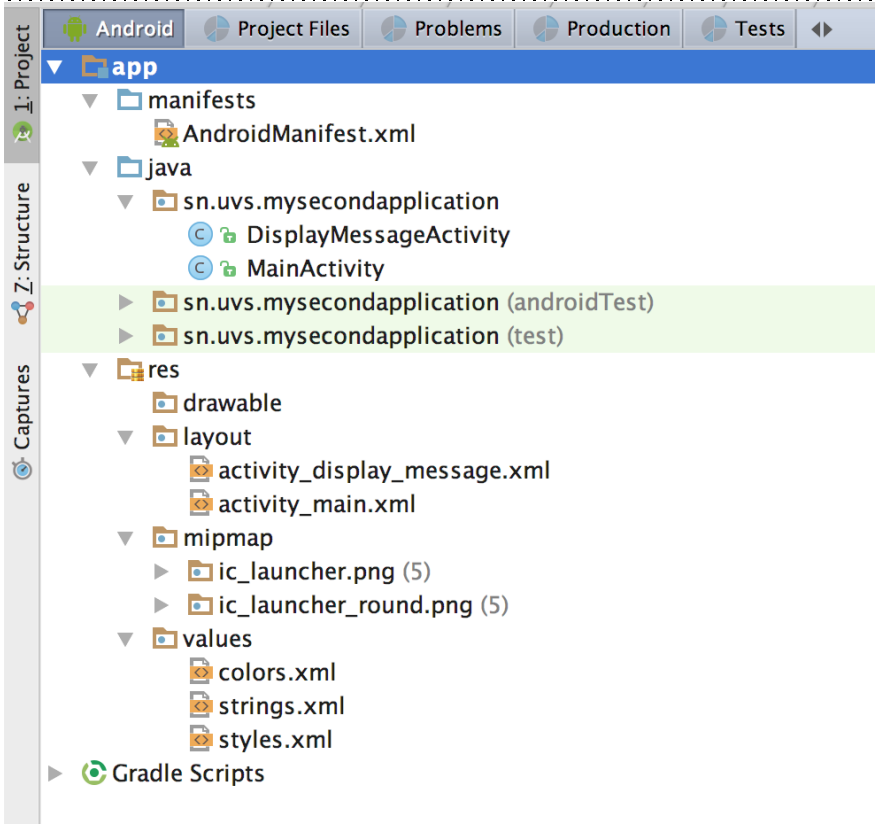
Implantation de service

Il existe deux façons d'implanter un Service : en étendant la classe Service, ou en étendant la classe IntentService (qui est elle-même une sous-classe de Service, et qui fournit un mécanisme commode pour la gestion de requêtes asynchrones)

```
public class MonServiceDemarre extends Service {
  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
      //TODO faire quelque chose d'utile
      return Service.START_NOT_STICKY;
  }

  @Override
  public IBinder onBind(Intent intent) {
    //TODO for communication return IBinder implementation
    return null;
  }
}
```

Android - Structure d'un projet Android



Vues d'un projet Android

Il existe plusieurs vues d'un projet Android. La vue par défaut est :

- la vue Android du projet (onglet Android)
- mais il y' a aussi les vues :
  - projet
  - tests

Vue Android du projet

- app
  - manifests : contient le/les manifests
  - java : contient les fichiers java, ainsi que les fichiers de test
  - res : resources
    - drawable: repertoire par défaut des images
    - layout : repertoire par défaut des dispositions
    - mipmap : repertoire par défaut des icônes
    - values : repertoire des valeurs
- Gradle Scripts : repertoire contenant les scripts de compilation et de construction des fichiers de déploiement de l'application

Développement Android - Application XamSaGoX

L'application Xamsagox a pour objet de tester/renforcer les connaissances historiques, géographiques et culturelles du joueur. Les questions porteront sur des

Architecture de l'application :

- MVC : Pattern Modele View Controller
- Modele : - Quiz(id, catid, question, options, reponse, image)
  - Categorie(id, nom, image, type_de_quiz)

Questions

Mr. Boubou CAMARA

boubou.camara@yahoo.ca