JAVA embarqué et développement Android

Android – *Applications et activités*

Composition d'une application

- Une application est composé de plusieurs activités, Chacune gère un écran d'interaction avec l'utilisateur et est définie par une classe Java,
- Une application complexe peut aussi contenir :
  - des services : ce sont des processus qui tournent en arrière-plan,
  - des fournisseurs de contenu : ils représentent une sorte de base de données,
  - des récepteurs d'annonces : pour gérer des événements globaux envoyés par le système à toutes les applications.

Déclaration d'une application

- Le fichier AndroidManifest.xml déclare les éléments d'une application, avec un '.' devant le nom des activités :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
   <application android:icon="@drawable/app_icon.png" ... >
          <activity android:name=".MainActivity"
                    android:label="@string/main" ... />
           <activity android:name=".EditActivity"
                    android:label="@string/edit" ... />
          ...
   </application>
</manifest>
```

- <application> est la seule branche sous la racine <manifest> et ses filles sont des <activity>.

Sécurité des applications

- Chaque application est associée à un UID (compte utilisateur Unix) unique dans le système. Ce compte les protège les unes des autres. Cet UID peut être défini dans le fichier AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...
        android:sharedUserId="ma.ensa.myapplication1">
        ...
</manifest>
```

- Définir l'attribut android:sharedUserId avec une chaîne identique à une autre application, et signer les deux applications avec le même certificat, permet à l'une d'accéder à l'autre.
- Voir : http://developer.android.com/guide/topics/manifest/manifest-element.html

Sécurité des applications

- Syntaxe <manifest> :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="string"
        android:sharedUserId="string"
        android:sharedUserLabel="string resource"
        android:versionCode="integer"
        android:versionName="string"
        android:installLocation=["auto" | "internalOnly" | "preferExternal"] >
    . . .
</manifest>
```

- Voir : http://developer.android.com/guide/topics/manifest/manifest-element.html

Autorisation d'une application

- Une application doit déclarer les autorisations dont elle a besoin : accès à internet, caméra, carnet d'adresse, GPS, etc.
- Cela se fait en rajoutant des éléments dans le manifeste :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ...
   <uses-permission
         android:name="android.permission.CAMERA"/>
   <uses-permission
         android:name="android.permission.INTERNET"/>
   <uses-permission
         android:name="android.permission.CALL_PHONE"/>
         ...
</manifest>
```

Démarrage d'une application

- L'une des activités est marquée comme démarrable de l'extérieur :

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

- Un <intent-filter> déclare les conditions de démarrage d'une activité, ici il dit que c'est l'activité principale.

Android – les *Intents*

Android – Principe des Intents

- Les activités sont démarrées à l'aide d'*Intents*.

- Les *Intents* permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications.
- Le but des *Intents* est de déléguer :
  - une autre activité de l'application courante,
  - une action à un autre composant,
  - ou une autre application.
    - Par exemple, composer un numéro de téléphone
    - ou envoyer ou recevoir des messages (SMS/MMS), etc.

Android – Principe des Intents

- Un objet Intent peut contenir les informations suivantes :
  - Le nom du composant ciblé (facultatif)
  - L'action à réaliser, sous forme de chaine de caractères
  - Les données : contenu MIME et URI
  - Des données supplémentaires sous forme de paires clef/valeur
  - Une catégorie pour cibler un type d'application
  - Des drapeaux (informations supplémentaires)
- On peut envoyer des Intents informatifs pour faire passer des messages. Mais on peut aussi envoyer des Intents suivant à lancer une nouvelle activité.

- URI : Uniform Resource Identifier
- URL : Uniform Resource Locator

Lancement d'une activité par programme

- Soit une application contenant deux activité : Activ1 et Activ2. La première lance la seconde par :

```
Intent intent = new Intent(Activ1.this, Activ2.class);
startActivity(intent);
```

- L'instruction startActivity démarre Active2. Celle-ci se met devant Activ1 qui se met alors en sommeil,
- Dans certain cas, il ne faut pas mettre ***this*** mais faire appel à ***getApplicationContext()*** si l'objet manipulant l'Intent n'hérite pas de Context.
- Ce bout de code est employé par exemple lorsqu'un bouton, un menu, etc. est cliqué. Seule contrainte : que ces deux activités soient déclarées dans AndroidManifest.xml.

Lancement d'une application Android

- S'il s'agit de passer la main à une autre application, on donne au constructeur de *l'Intent* les données et l'URI cible: l'OS est chargé de trouver une application pouvant répondre à l'Intent.
- Pour ouvrir le navigateur sur un URL spécifique :

```
String url = "http://www.ensaf.ac.ma/..";
intent = new Intent (Intent.ACTION_VIEW, Uri.parse(url));
startActivity(intent);
```

- L'action VIEW avec un URI (généralisation d'un *URL*) est interprétée par Android, cela fait ouvrir automatiquement le navigateur.

Lancement d'une application Android

- Pour émettre un appel depuis votre application, il ne suffira pas uniquement de créer un objet Intent avec l'action *ACTION_CALL*. Il vous faudra également ajouter la permission *android.permission.CALL_PHONE* :

```
Uri uri= Uri.parse("tel:066123456");
Intent intent = new Intent (Intent.ACTION_CALL, uri);
startActivity(intent);
```

- Spécifier les permissions de l'application

Version < sdk-23

```
<manifest ….
 <uses-permission android:name="android.permission.CALL_PHONE" />
</manifest>
```

Version >= sdk-23

```
<uses-permission-sdk-23 android:name="android.permission.CALL_PHONE"/>
```

Android – Applications

Fonctionnement d'une application

- Au début, le système Android lance l'activité qui est marquée *action=MAIN* et *catégorie=LAUNCHER* dans *AndroidManifest.xml*.
- Ensuite d'autres activités peuvent être démarrées. Chacune se met « devant » les autres comme sur une pile. Deux cas sont possibles:
  - La précédente activité se termine, on ne revient pas dedans. Par exemple, une activité où on tape son login et son mot de passe lance l'activité principale et se termine, ou par exemple terminer son appel téléphonique et de revenir à l'activité ayant initié l'appel.
  - La précédente activité attend la fin de la nouvelle car elle lui demande un résultat en retour. Par conséquent, une activité peut vouloir récupérer un code de retour de l'activité "enfant". On utilise pour cela la méthode **startActivityForResult** qui envoie un code de retour à l'activité enfant.

Lancement sans attente

- Rappel, pour lancer Activ2 à partir de Activ1 :

```
Intent intent = new Intent(Activ1.this, Activ2.class);
startActivity(intent);
```

- On peut demander la **terminaison** de this après lancement de Activ2 ainsi :

```
Intent intent = new Intent(Activ1.this, Activ2.class);
startActivity(intent);
finish();
```

- finish() fait terminer l'activité courante. L'utilisateur ne pourra pas faire back dessus, car elle disparaît de la pile.

Transport d'informations dans un Intent

- Les **Intent** servent aussi à transporter des informations d'une activité à l'autre : *les extras*.
- Voici comment placer des données dans un Intent :

```
envoyer.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        intent = new Intent( packageContext: MainActivity.this, Main2Activity.class);

        intent.putExtra( name: "x", x);
        intent.putExtra( name: "y", y);
        intent.putExtra( name: "somme", somme);
        startActivity(intent);
    }
});
```

```
Bundle bundle = new Bundle();
bundle.putInt("x", x);
bundle.putInt("y", y);
bundle.putInt("somme", somme);
intent.putExtras(bundle);
startActivity(intent);
```

```
Bundle bundle = new Bundle();
bundle.putInt("x", x);
bundle.putInt("y", y);
bundle.putInt("somme", somme);
startActivity(intent, bundle);
```

- putExtra(nom, valeur) rajoute un couple (nom, valeur) dans l'intent.

Extraction d'informations d'un Intent

- Ces instructions récupèrent les données d'un *Intent* :

```
Intent intent = getIntent();
String name = intent.getStringExtra("nom");
Integer age = intent.getIntExtra("age", -1);
boolean present= intent.getBooleanExtra("presence",
false);
```

- getIntent() retourne l'*Intenet* qui a démarré cette activité.
- **get*Type*Extra**(nom, valeur par défaut) retourne la valeur de ce nom si elle en fait partie, la valeur par défaut sinon.
- Dans le cas de Bundle :

```
Bundle b = getIntent().getExtras();
int x = b.getInt( key: "x");
int y = b.getInt( key: "y");
int somme = b.getInt( key: "somme");
text1.setText("x: " + x + "+ y: " + y + " = " + somme );
```

Lancement avec attente de résultat

- Le lancement d'une activité avec attente de résultat est plus complexe. Il faut définir un *code d'appel* RequestCode fourni au lancement.

```
private static final int APPEL_ACTIV2 = 1;
Intent intent = new Intent(Activ1.this, Activ2.class);
startActivityForResult(intent, APPEL_ACTIV2);
```

- Ce code identifie l'activité lancée, afin de savoir plus tard que c'est d'elle qu'on revient. Par exemple, on pourrait lancer au choix plusieurs activités : édition, copie, suppression d'informations. Il faut pouvoir les distinguer au retour.

Voir : http://developer.android.com/training/basics/intents/result.html

Lancement avec attente de résultat, suite

- Ensuite, il faut définir une méthode de *callback* qui est appelée lorsqu'on revient dans notre activité (le code de retour est dans la méthode **onActivityResult**) :

```
@Override
protected void onActivityResult(
            int requestCode, int resultCode, Intent data)
{
            if (resultCode == Activity.RESULT_CANCELED) return;
            // selon le code d'appel
            switch (requestCode) {
                        case APPEL_ACTIV2: // on revient de Activ2
                                    …
            }
}
```

```
    if (requestCode == APPEL_ACTIV2)
        Toast.makeText(this, "Code de requête récupéré",
            Toast.LENGTH_LONG). show();
}
```

Méthode onActivityResult

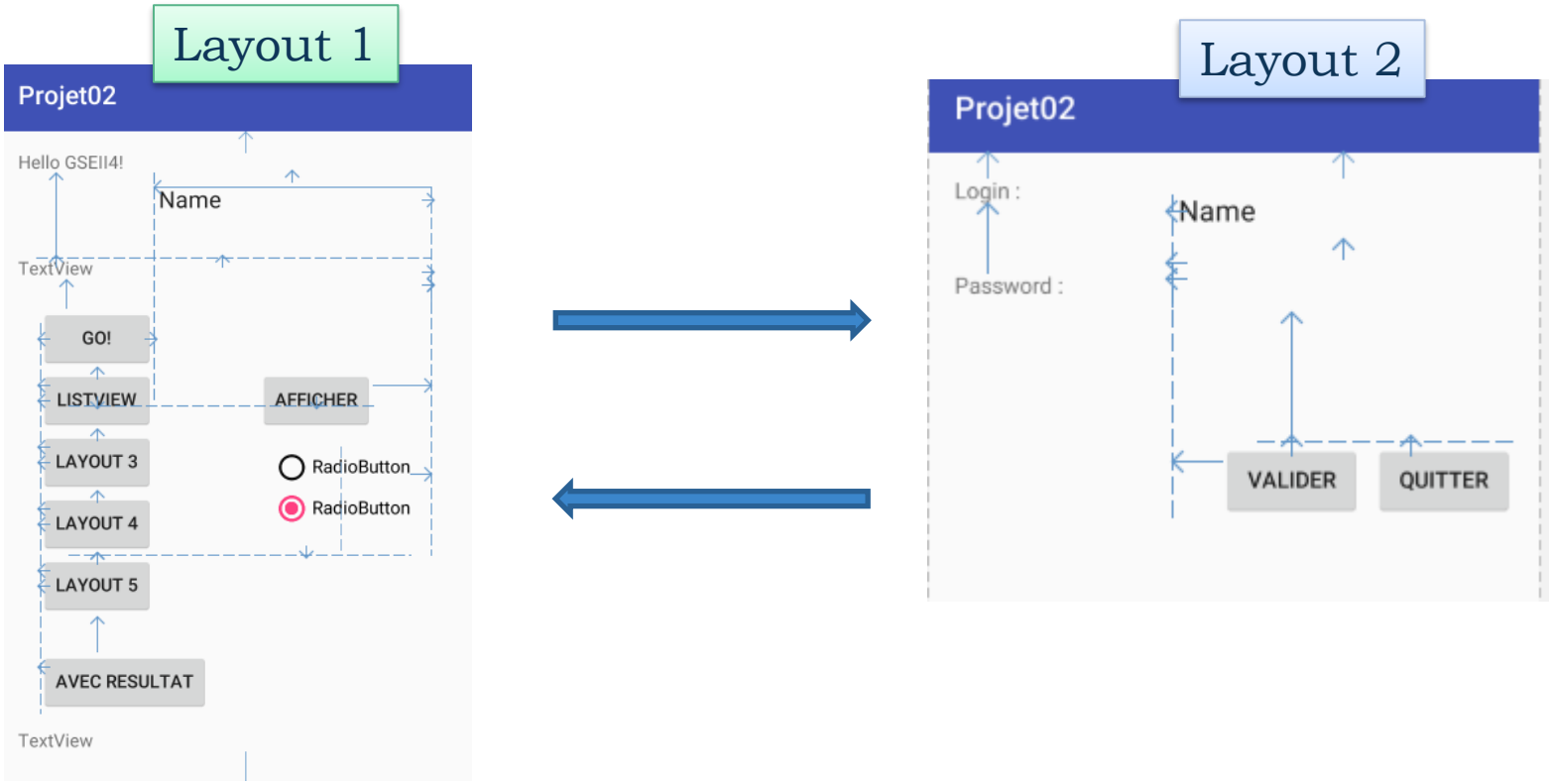
- onActivityResult(int requestCode, int resultCode, Intent data)
  - **requestCode** est le code d'appel de **startActivityForResult**
  - **resultCode** vaut soit **RESULT_CANCELED** soit **RESULT_OK**
  - **data** est fourni par l'autre activité.
- Ces deux dernières viennent d'un appel à **setResult(resultCode, data)** . Par exemple, l'autre activité se termine ainsi :

```
setResult(RESULT_OK, getIntent());
finish();
```

- Ou par :

```
setResult(RESULT_CANCELED);
finish();
```

Exemple (1/2)


Layout 1
Projet02
Hello GSEII4!
Name
TextView
GO!
LISTVIEW
AFFICHER
LAYOUT 3
RadioButton
RadioButton
LAYOUT 4
LAYOUT 5
AVEC RESULTAT
TextView
Layout 2
Projet02
Login :
Password :
Name
VALIDER
QUITTER

Exemple (2/2)

Layout 1

```
final Button b6 = (Button) findViewById(R.id.button6);
b6.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent1 = new
Intent(getApplicationContext(), Main2Activity.class);
        startActivityForResult(intent1, 2); //l'activité
est relancée avec le code de requete 2
        //finish();
    }
});
```

```
public void onActivityResult(int requestCode, int
resultCode, Intent data){
super.onActivityResult(requestCode, resultCode, data);
TextView tmessage=(TextView)findViewById(R.id.tmessage);

if(resultCode== Activity.RESULT_CANCELED) return;

if(requestCode==2) {
    String message1 = data.getStringExtra("Message");
    tmessage.setText(message1);

    Toast.makeText(MainActivity.this, message1,
Toast.LENGTH_LONG).show();
    }
}
```

Layout 2

```
TextView textView1=(TextView)findViewById(R.id.textView4);
final Button bValider = (Button)
findViewById(R.id.button6);
bValider.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        EditText tLogin =
(EditText)findViewById(R.id.editText3);
        EditText tPassorwd =
(EditText)findViewById(R.id.editText4);
        String message = tLogin.getText().toString()+" et
le mot de passe est:"+tPassorwd.getText().toString();
        Intent intent = new Intent();
        intent.putExtra("Message",message);
        setResult(2, intent);
        finish();
    }
});
```

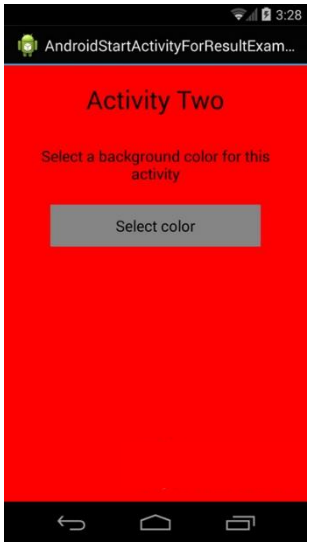
Pattern

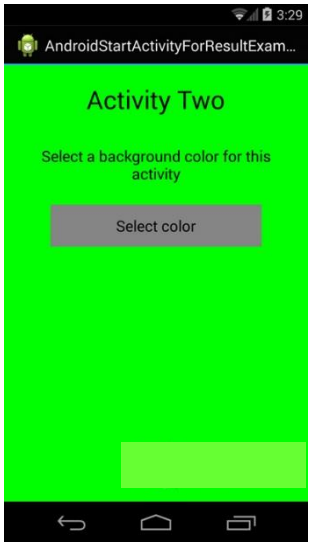
- Patterns are compiled regular expressions. In many cases, convenience methods such as String.matches, String.replaceAll and String.split will be preferable, but if you need to do a lot of work with the same regular expression, it may be more efficient to compile it once and reuse it. The Pattern class and its companion, Matcher, also offer more functionality than the small amount exposed by String.

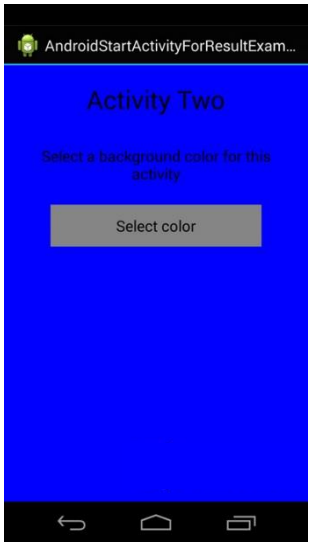
```
Pattern p = Pattern.compile(".+@.+\\.[a-z]+");
    //le 'matcher' va comparer le 'pattern' avec le 'string' passé en argument
    Matcher m = p.matcher(loginTxt);
    //Si l'adresse mail saisie ne correspond pas au format d'une
    //adresse mail on affiche un message d'erreur
    if (!m.matches()){
        Toast.makeText(getApplicationContext(),"Erreur : Email",Toast.LENGTH_LONG).show();
    //Pour eviter l'exécution des opérateurs qui suivent
    return;
}

//le cas des différents champs qui ne doivent pas être vides
if (loginTxt.equals("")||passTxt.equals("")){
    Toast.makeText(MainActivity.this,"Champs vide",Toast.LENGTH_LONG).show();
    return;
}
```

Exercice 1

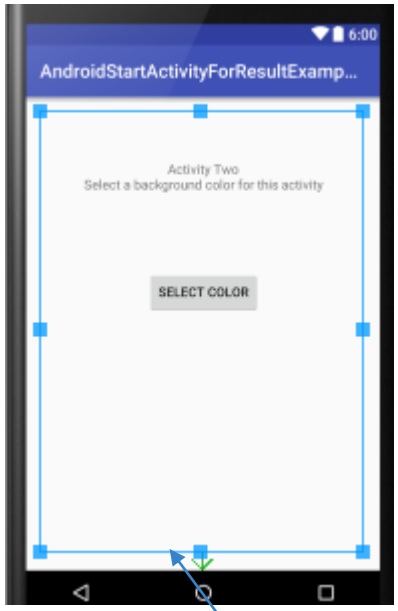

3:28
AndroidStartActivityForResultExam...
Activity Two
Select a background color for this activity
Select color

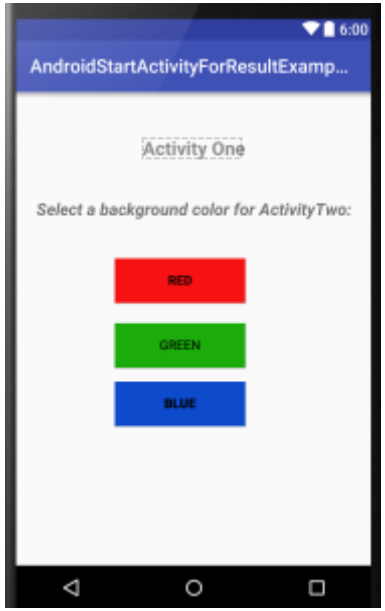

3:29
AndroidStartActivityForResultExam...
Activity Two
Select a background color for this activity
Select color


AndroidStartActivityForResultExam...
Activity Two
Select color


3:28
AndroidStartActivityForResultExam...
Activity One
Select a background color for ActivityTwo:
Red
Green
Blue

Exercice 1 : solution à compléter





LinearLayout : ContainerTwo

Exercice 1 : solution à compléter

```
public class MainActivity extends AppCompatActivity {
    LinearLayout containerTwo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        containerTwo = (LinearLayout) findViewById(R.id.containerTwo);
        Intent intent = getIntent();
        String color = intent.getStringExtra("color");
        changeBackground(color);       }
        public void openActivityOne(View v) {
        Intent intent = new Intent(MainActivity.this, Main2Activity.class);
        startActivity(intent);    }
    public void changeBackground(String color) {
       if (color.equals("red")) {  containerTwo.setBackgroundColor(Color.RED);
       } else if (color.equals("green")) {containerTwo.setBackgroundColor(Color.GREEN);
       } else if (color.equals("blue")) {containerTwo.setBackgroundColor(Color.BLUE);
       }    }  }
```

Exercice 1 : solution à compléter

```
public class Main2Activity extends AppCompatActivity {
        @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);}
```

```
public void redBackground(View v) {
      Intent intent = new Intent(this,
            Main2Activity.class);
      intent.putExtra("color", "red");
      startActivityForResult(intent, 1);
      finish();
   }
```

```
public void greenBackground(View v) {
      Intent intent = new Intent(
            this, Main2Activity.class);
      intent.putExtra("color", "green");
      startActivityForResult(intent, 1);
      finish();
   }
```

```
public void blueBackground(View v) {
      Intent intent = new Intent(this,
            Main2Activity.class);
      intent.putExtra("color", "blue");
      startActivityForResult(intent, 1);
      finish();
   }
}
```