Chapitre 2: Les composantes d'une application Android

Développement mobile sous Android

Composantes applicatifs

- **Activitiés (activity)**: c'est le bloc de base d'une application composé d'une interface graphique qui interagit avec l'utilisateur
- **Services**: est un composant qui fonctionne en tâche de fond, de manière invisible.
- **Fournisseurs de contenu (Content provider)**: permet de gérer et de partager les informations.
- **Gadgets (Widget):** est un composant graphique qui s'installe sur le bureau Android.

Les éléments d'interaction

- **L'objet Intent**:
  il permet de diffuser des messages en demandant la réalisation d'une action.
- **Le récepteur d'intents**:
  il permet à une application d'être à l'écoute des autres afin de répondre aux objets Intent qui lui sont destinés et qui sont envoyés par d'autres composants applicatifs.
- **Notification**:
  une notification signale une information à l'utilisateur sans interrompre ses actions en cours.

Permissions

Certaines opérations sont réalisables à condition d'en obtenir la permission. Ces actions sont de plusieurs formes :

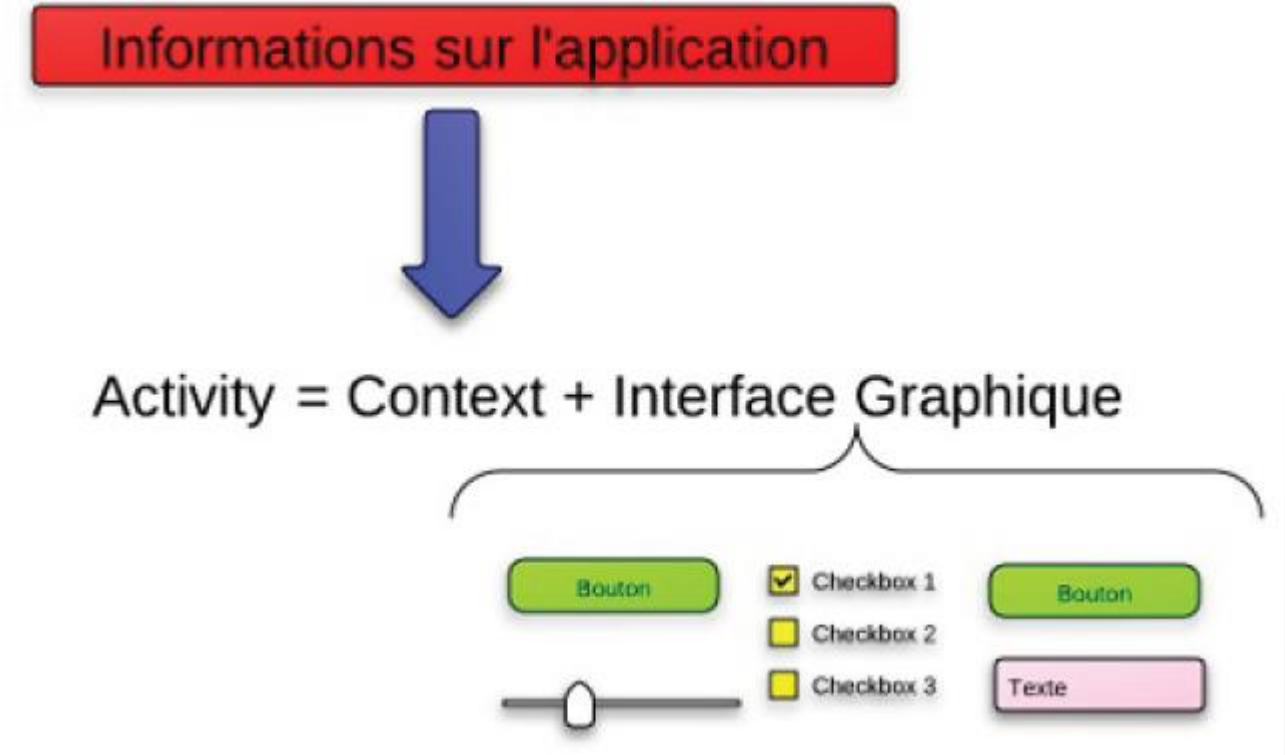
- opérations pouvant entraîner un surcoût (connexion, échange de données, envoi de SMS par exemple) ;
- utilisation de données personnelles (accès à vos contacts, à votre compte Google, exploitation de vos informations linguistiques entre autres) ;
- accès au matériel du téléphone (prise de clichés, écriture sur la carte mémoire...).

Cycle de vie d'une application Android

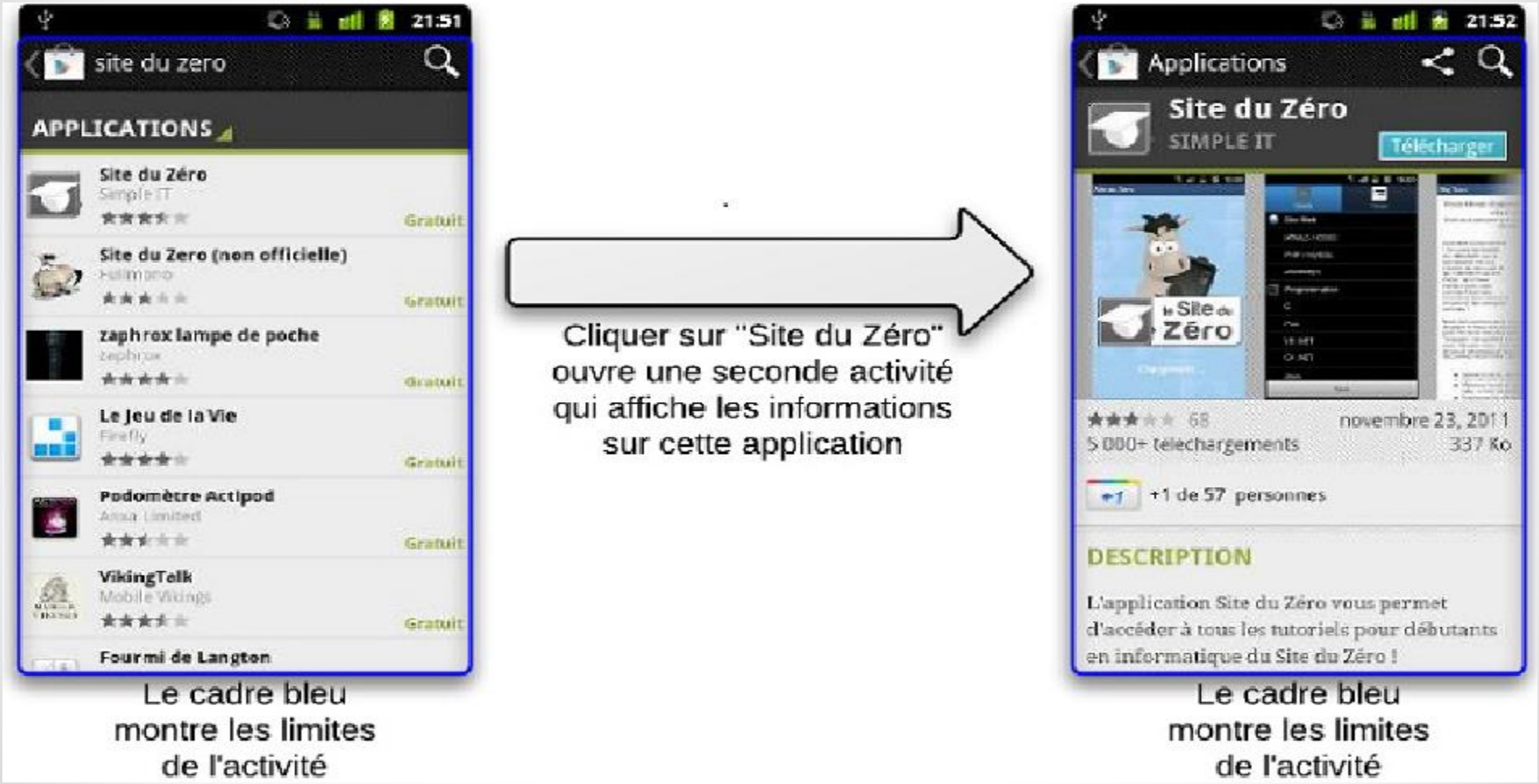
Processus	Priorité	Description
Processus actif	Maximale	Processus au premier plan qui héberge des applications dont des composants interagissent avec l'utilisateur. C'est un processus qu'Android protège en libérant des ressources.
Processus visible	Élevée	Processus visible mais inactif, il contient des activités au moins en partie visibles mais pas au premier plan.
Processus d'un service démarré	Élevée	Processus qui réalise des actions sans interface graphique visible. Il a donc une priorité moindre que les processus vus plus haut, mais il est considéré comme de premier plan à la différence des processus en tâche de fond. Il est conservé le plus longtemps possible.
Processus en tâche de fond	Faible	Processus hébergeant des activités non visibles ou des services non démarrés. Ces processus plus nombreux seront plus facilement éliminés en fonction de leur utilisation.
Processus en fin de vie	Minimale	Android dispose d'un cache d'applications qui ont terminé leur cycle de vie, à des fins d'optimisation lors de leur redémarrage.

L'activité

Une activité est constitué d'un contexte d'une application et d'une seule et unique interface graphique



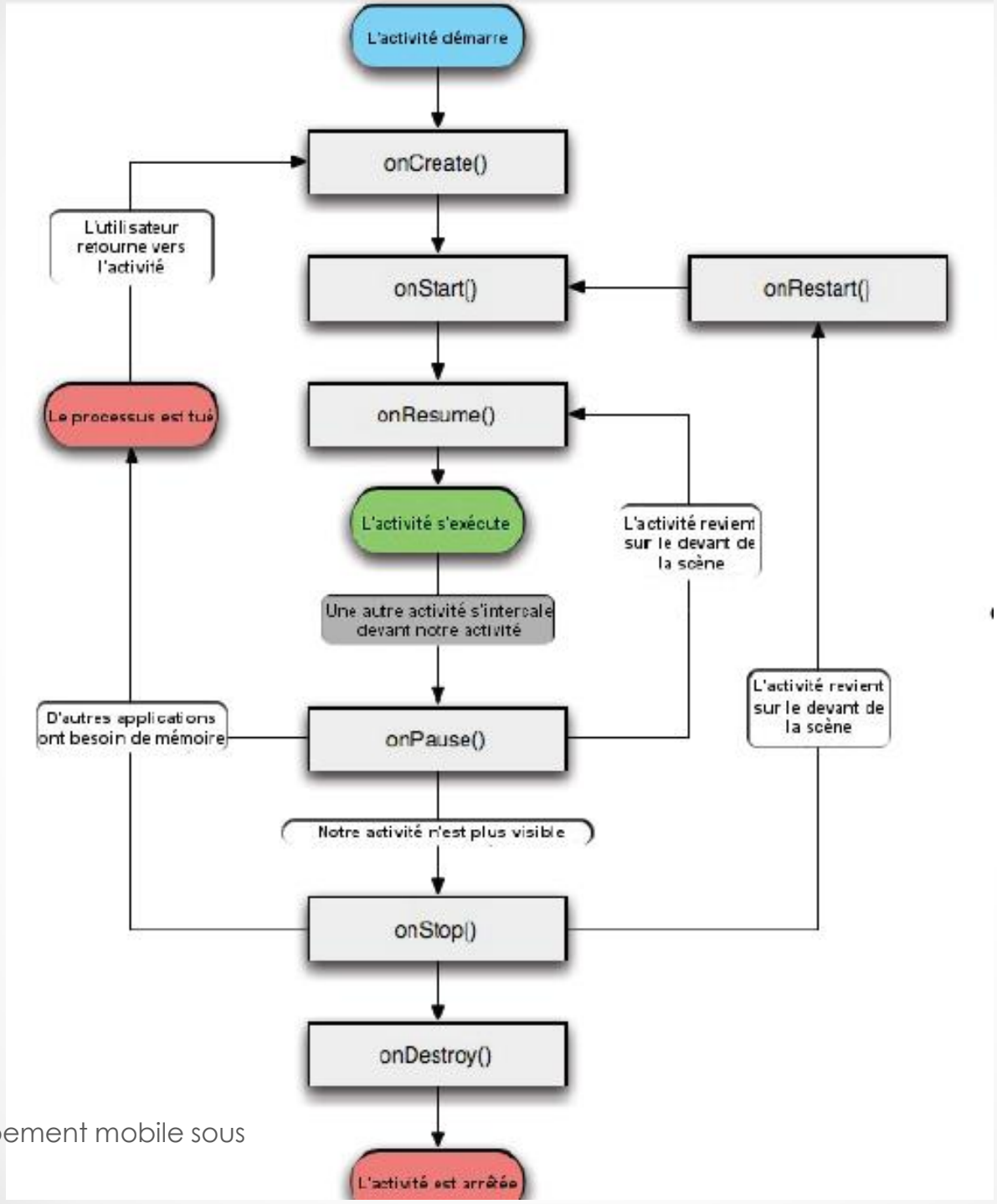
Une activité lance une autre



Cycle de vie d'une activité

Les états principaux d'une activité sont:

- **Active (*active*)** : activité visible qui détient le focus utilisateur et attend les entrées utilisateur. C'est l'appel à la méthode ***onResume***, à la création ou à la reprise après pause qui permet à l'activité d'être dans cet état. Elle est ensuite mise en pause quand une autre activité devient active grâce à la méthode ***onPause*** ;
- **Suspendue (*paused*)** : activité au moins en partie visible à l'écran mais qui ne détient pas le focus. La méthode ***onPause*** est invoquée pour entrer dans cet état et les méthodes ***onResume*** ou ***onStop*** permettent d'en sortir ;
- **Arrêtée (*stopped*) :** activité non visible. C'est la méthode ***onStop*** qui conduit à cet état.


L'activité démarre
onCreate()
L'utilisateur retourne vers l'activité
onStart()
onRestart()
Le processus est tué
onResume()
L'activité s'exécute
L'activité revient sur le devant de la scène
Une autre activité s'intercale devant notre activité
D'autres applications ont besoin de mémoire
onPause()
L'activité revient sur le devant de la scène
Notre activité n'est plus visible
onStop()
onDestroy()
L'activité est arrêtée

Cycle de vie d'une activité

```
import android.app.Activity;
import android.os.Bundle;
public final class TemplateActivity extends Activity {
/**
* Appelée lorsque l'activité est créée.
* Permet de restaurer l'état de l'interface
* utilisateur grâce au paramètre savedInstanceState.
*/
public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                // Placez votre code ici
}
/**
* Appelée lorsque que l'activité a fini son cycle de vie.
* C'est ici que nous placerons notre code de libération de
* mémoire, fermeture de fichiers et autres opérations
* de "nettoyage".
*/
@Override
public void onDestroy(){
                // Placez votre code ici
                super.onDestroy();
}
```

Cycle de vie d'une activité

```
/**
* Appelée lorsque l'activité démarre. Permet d'initialiser les contrôles.*/
@Override
public void onStart(){
        super.onStart();
        // Placezvotre code ici
}
/**
* Appelée lorsque l'activité passe en arrière plan.
* Libérez les écouteurs, arrêtez les threads, votre activité
* peut disparaître de la mémoire.
*/
@Override
public void onStop(){
        // Placez votre code ici
        super.onStop();
}
```

Cycle de vie d'une activité

```
/**
* Appelée lorsque l'activité sort de son état de veille.
*/
@Override
public void onRestart(){
        super.onRestart();
        //Placez votre code ici
}

/**
* Appelée lorsque que l'activité est suspendue.
* Stoppez les actions qui consomment des ressources.
* L'activité va passer en arrière-plan.
*/
@Override
public void onPause(){
        //Placez votre code ici
        super.onPause();
}
```

Cycle de vie d'une activité

```
/**
* Appelée après le démarrage ou une pause.
* Relancez les opérations arrêtées (threads).
* Mettez à jour votre application et vérifiez vos écouteurs.
*/
@Override
public void onResume(){
                super.onResume();
                // Placez votre code ici
}
/**
* Appelée lorsque l' activité termine son cycle visible.
* Sauvez les données importantes.
*/
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
                // Placez votre code ici
                // sans quoi l'activité aura perdu son état
                // lors de son réveil
                super.onSaveInstanceState(savedInstanceState);
}
/**
* Appelée après onCreate.
* Les données sont rechargées et l'interface utilisateur.
* est restaurée dans le bon état.
*/
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
                super.onRestoreInstanceState(savedInstanceState);
                //Placez votre code ici
}
}
```

Les vues (view)

- C'est les éléments de constructions d'une interface graphique d'une activité
- Exemple: textview, Editview, Button…
- Chaque écran Android contient un arbre d'élément view
- Les vues peuvent etre disposées dans une activités soit par une description XML ou par code Java.

Les Ressources

Type de ressource	Répertoire associé	Description
Valeurs simples	`res/values`	Fichiers XML convertis en différents types de ressources. Ce répertoire contient des fichiers dont le nom reflète le type de ressources contenues : 1. `arrays.xml` définit des tableaux ; 2. `string.xml` définit des chaînes de caractères ; 3. ...
Drawables	`res/drawable`	Fichiers `.png`, `.jpeg` qui sont convertis en bitmap ou `.9.png` qui sont convertis en "9-patches" c'est-à-dire en images ajustables. Note : à la construction, ces images peuvent être optimisées automatiquement. Si vous projetez de lire une image bit à bit pour réaliser des opérations dessus, placez-la plutôt dans les ressources brutes.
Layouts	`res/layout`	Fichiers XML convertis en mises en page d'écrans (ou de parties d'écrans), que l'on appelle aussi gabarits.
Animations	`res/anim`	Fichiers XML convertis en objets animation.
Ressources XML	`res/xml`	Fichiers XML qui peuvent être lus et convertis à l'exécution par la méthode `resources.getXML`.
Ressources brutes	`res/raw`	Fichiers à ajouter directement à l'application compressée créée. Ils ne seront pas convertis.

Fichier de configuration

- Chaque application Android nécessite un fichier de configuration: AndroidManifeste.xml

- Il décrit :
  - le contexte de l'application
  - les activités
  - les services
  - les récepteurs d'intents (Broadcast Receivers)
  - les fournisseur de contenu
  - Et les permissions

Fichier de configuration

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
        <uses-permission />
        <permission />
        <permission-tree />
        <permission-group />
        <instrumentation />
        <uses-sdk />
        <uses-configuration />
        <uses-feature />
        <supports-screens />
        <application>
                <activity>
                        <intent-filter>
                                <action />
                                <category />
                                <data />
                        </intent-filter>
                        <meta-data />
                </activity>
                <activity-alias>
                        <intent-filter> . . . </intent-filter>
                        <meta-data />
                </activity-alias>
```

Fichier de configuration

```
            <service>
                <intent-filter> . . . </intent-filter>
                <meta-data/>
            </service>
            <receiver>
                <intent-filter> . . . </intent-filter>
                <meta-data />
            </receiver>
            <provider>
                <grant-uri-permission />
                <path-permission />
                <meta-data />
            </provider>
            <uses-library />
        </application>
</manifest>
```

Fichier de configuration

- **<uses-permission />** définit les permissions nécessaires
- **<application>** un seul nœud est nécessaire pour définir le contenue de l'application
- **<activity>** déclare une activité présentée à l'utilisateur.
- **<service>** déclare un composant de l'application en tant que service (pas d'interface graphique, tout se déroulera en tâche de fond de votre application).
- **<receiver>** déclare un récepteur d'objets « Intent ». Cet élément permet à l'application de recevoir ces objets alors qu'ils sont diffusés par d'autres applications ou par le système
- **<provider>** déclare un fournisseur de contenu qui permettra d'accéder aux données gérées par l'application.