ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Matière :

Développement mobile

Chapitre 2:
Création d'applications et découverte des activités

Préparé par: Pr. Zakia EL UAHHABI
Edité par : Dr. M. BELATAR

Plan du cours

1. Composants d'une Application Android
   - Activité
   - Services
   - Récepteurs de diffusion
   - Fournisseurs de contenu
2. Ressources

Composants d'une Application Android

- Une application Android est composée des éléments suivants:
  - ☞ Activité;
  - ☞ Services;
  - ☞ Récepteurs de diffusion (Broadcast Receivers);
  - ☞ Fournisseurs de contenu (Content Providers).

Composants d'une Application Android

Activité: Définition

- ☞ Elle correspond à la partie présentation de l'application;
- ☞ Elle peut être assimilée à un **écran** structuré par un ensemble de vues et de contrôles composant son interface;
- ☞ Pour chaque écran de votre application, vous devrez donc créer une activité.
- ☞ Une activité = un programme + une interface
- ☞ Une activité peut interagir :
  -avec l'utilisateur en lui demandant des données;
  -avec d'autres activités ou services en émettant des intentions ou fournissant du contenu.

Composants d'une Application Android

Activité: Définition

Vues (Views)

- Ce sont les éléments de l’interface graphique que l’utilisateur voit et sur lesquels il pourra agir.

Les “contrôles” sont des “Views”

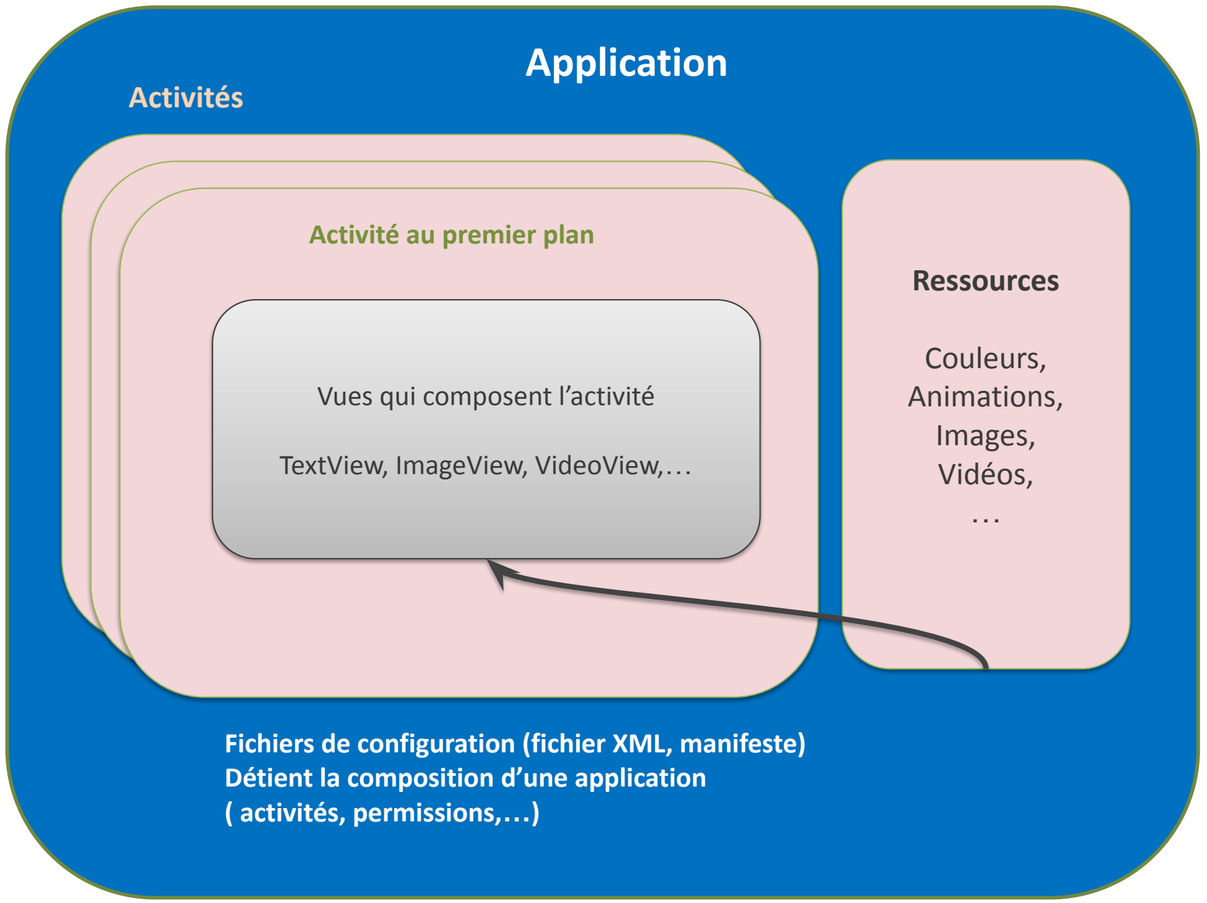
- Boutons,
- Champs de saisie,
- Case à cocher, etc.

Composants d'une Application Android

Activité: Définition

Composants d'une Application Android

Activité: Définition

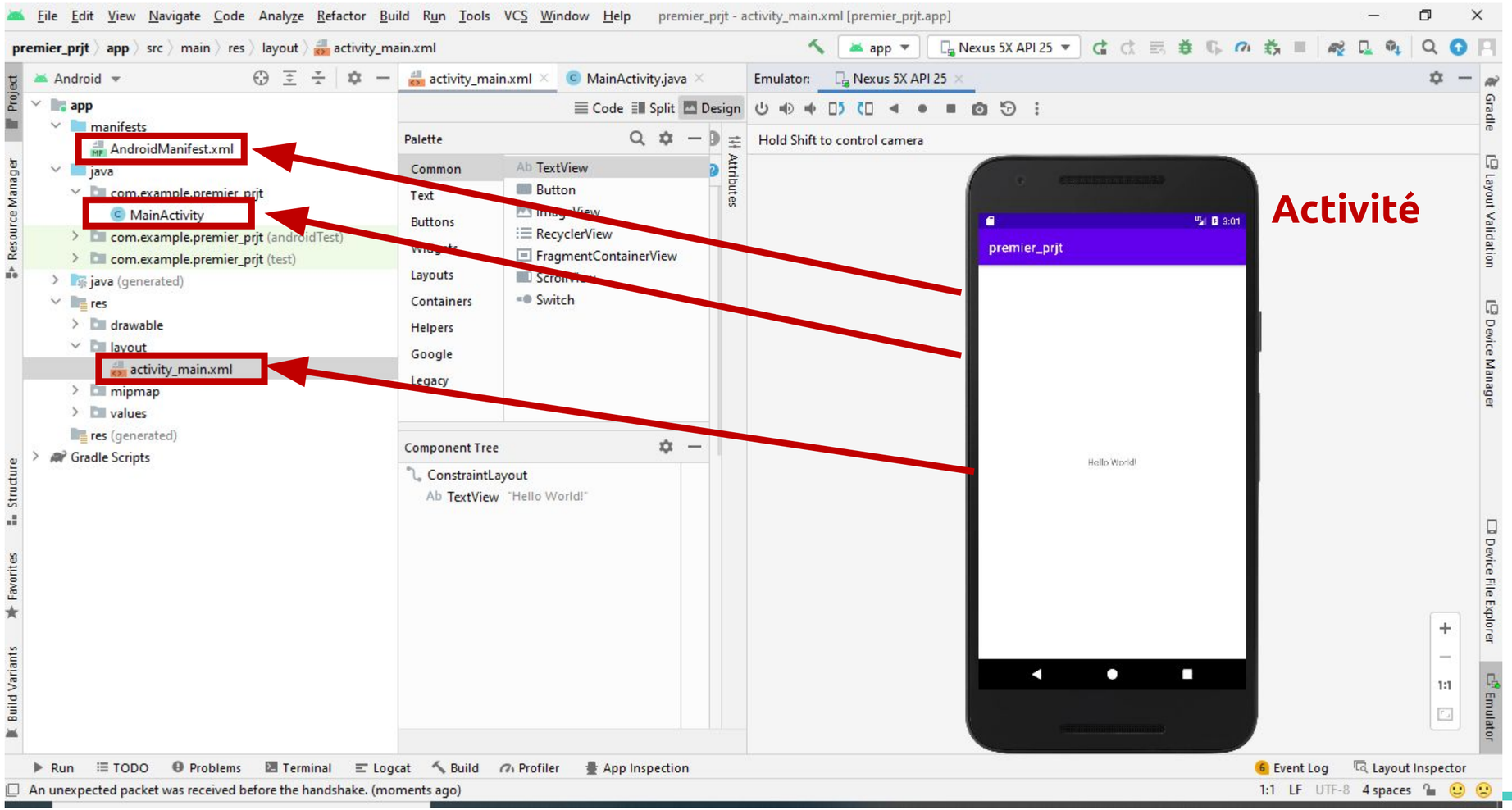


Composants d'une Application Android

Activité: Fichiers de base

Chaque activité est associée à trois principaux fichiers:

- Fichier **Androidmanifest.xml** où elle est déclarée
- Fichier **activity_main.xml** qui décrit son design
- Fichier **MainActivity** pour la contrôler



Composants d'une Application Android

Activité: Fichiers de base

☞ Fichier **manifest.xml**

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

☞ Un fichier **manifest.xml** est un fichier de configuration est composé d'une racine (le tag **manifest**) et d'une suite de nœuds enfants qui définissent l'application.

☞ Chaque activité doit être déclarée dans le **Manifest** pour être visible par le système. On utilise la balise **<activity>**

☞ La balise **<intent-filter>** indique la première activité qui se lance au démarrage de l'application.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.premier_prjt">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="premier_prjt"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Premier_prjt">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Composants d'une Application Android

Activité: Fichiers de base

- Fichier **manifest.xml**

- La balise **<intent-filter>** indique la première activité qui se lance au démarrage de l'application.

- Au début, le système Android lance l'activité qui est marquée dans AndroidManifest.xml:

  – action=MAIN
  – catégorie=LAUNCHER.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.premier_prjt">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="premier_prjt"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Premier_prjt">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Composants d'une Application Android

Activité: Cycle de vie

Chaque application Android nécessite une activité possède quatre états:

- **Active**: l'activité est lancée par l'utilisateur, elle s’exécute au premier plan;
- **En pause**: mettre en pause quand une autre activité devient active;
- **Stoppée:** activité est totalement cachée, pas de code exécuté, Ses informations sont conservées,
- **Morte** : l'activité n'est pas lancée.

Composants d'une Application Android

Activité: Cycle de vie

- Classe qui hérite de Activity ou d'une classe dérivée de Activity (par exemple de MapActivity pour utiliser Google maps, ListActivity ou TabActivity pour des interfaces particulières)

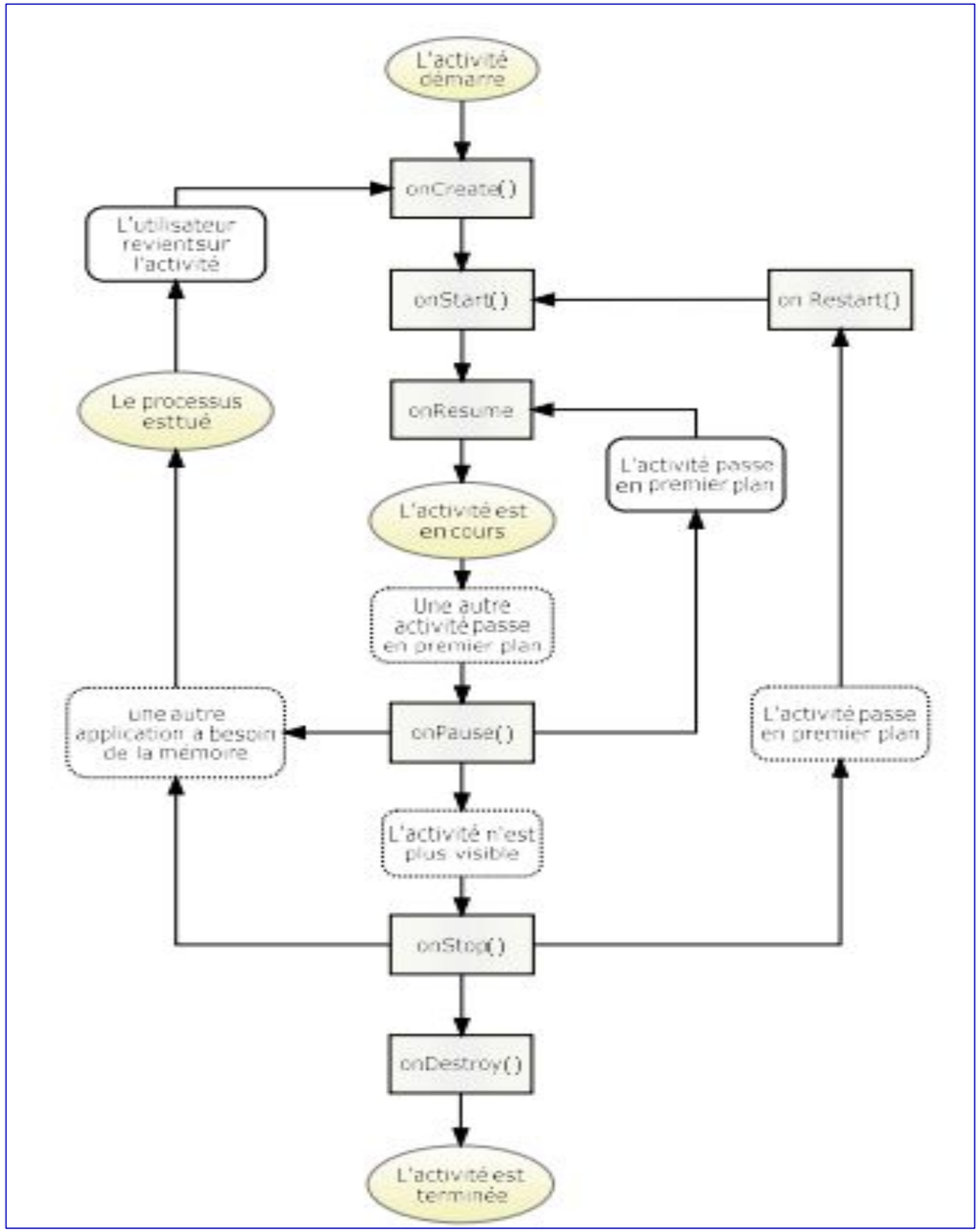
```
import android.app.Activity;
import android.os.Bundle;
import android.support.annotation.Nullable;

public class MyActivity extends Activity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    protected void onStart() {
        super.onStart();
    }
    @Override
    protected void onResume() {
        super.onResume();
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
    }
    @Override
    protected void onStop() {
        super.onStop();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
    }
    @Override
    protected void onSaveInstanceState(Bundle
outState) {
        super.onSaveInstanceState(outState);
    }
}
```

Composants d'une Application Android

Activité: Cycle de vie

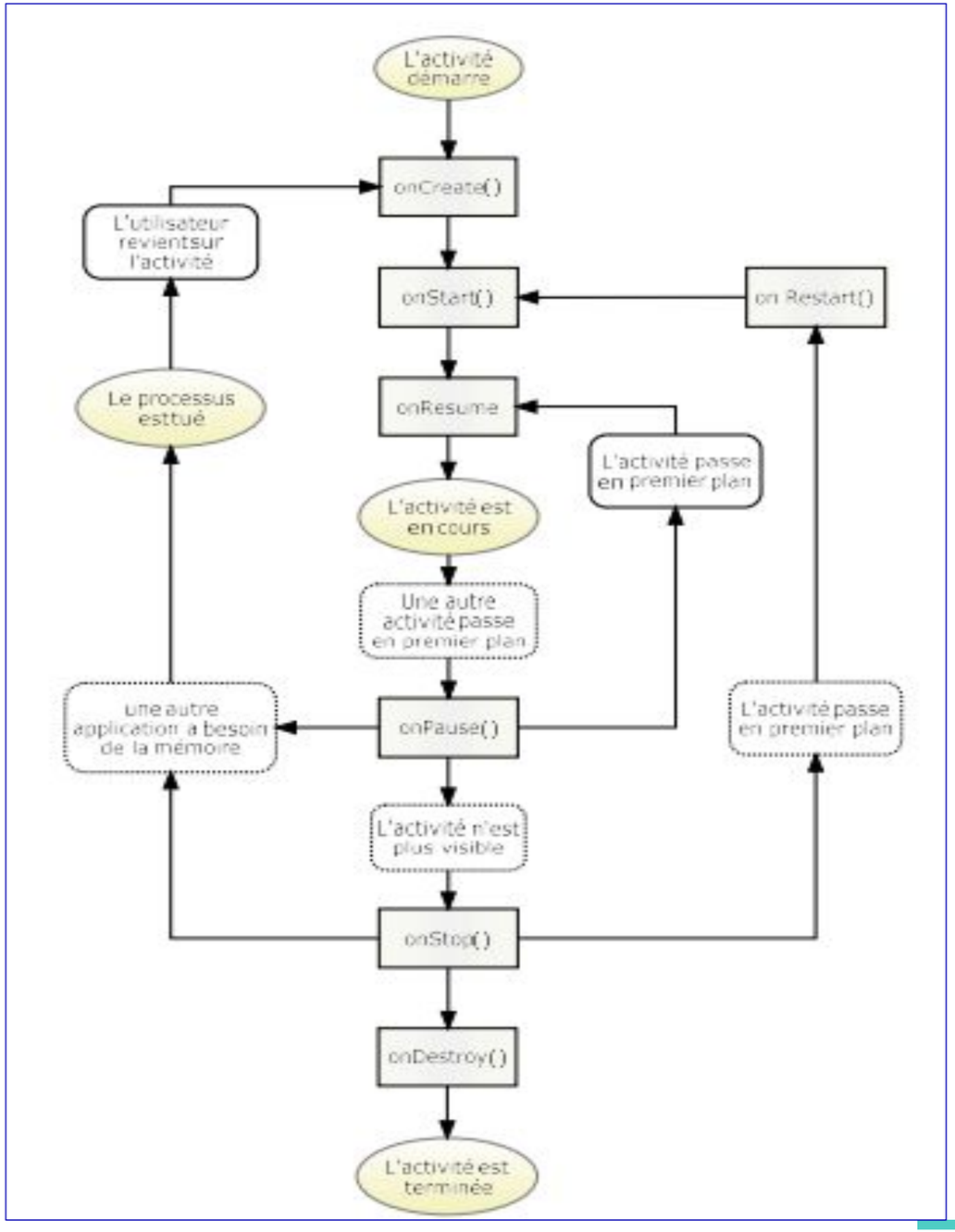
- Lorsque l'activité est lancée, le système Android appelle les méthodes ***onCreate***, ***onStart*** et ***onResume***.
- Lorsque l'activité s'arrête, le système Android appelle les méthodes ***onPause***, ***onStop***, et ***onDestroy***.
- Lorsque l'activité n'est plus au premier plan, mais qu'elle est tout de même affichée, ***onPause*** est appelée. lorsque l'activité retourne au premier plan, ***onResume*** est appelée.
- Lorsque l'activité n'est plus affichée ***onStop*** est appelée.
- Lorsque l'activité est de nouveau affiche, ***onRestart***, ***onStart*** et ***onResume*** sont appelées.
- Lorsque le système Android décide de tuer votre application, il appelle la méthode onSaveInstanceState qui vous permet de sauvegarder l'état de votre activité. Cet état sera passe en paramètre à la méthode ***onCreate***, afin que vous puissiez restaurer l'état de l'activité.



Composants d'une Application Android

Activité: Cycle de vie

- **onCreate** lors de la création.
- **onDestroy** lorsque l'activité se termine. la destruction opère quand quelqu'un appelle la méthode finsh() ou quand le système qui décide de tuer l'activité pour économiser l'espace.
  OnCreate() devra à nouveau être exécuté pour obtenir à nouveau l'activité.
- **onStart** lorsque l'activité démarre ou redémarre.
- **onPause**: Méthode exécutée à chaque fois que:
  -l'utilisateur passe à une autre activité,
  -le système a besoin de libérer de la mémoire
- **onResume** lorsque l'activité revient en premier plan.
- **onStop**:
  -lorsque l'activité n'est plus visible.
  -Libération des ressources
- **onRestart** lorsque l'activité redevient visible.



Composants d'une Application Android

- Une application Android est composée des éléments suivants:
  - ☞ Activité;
  - ☞ Services;
  - ☞ Récepteurs de diffusion (Broadcast Receivers);
  - ☞ Fournisseurs de contenu (Content Providers).

Composants d'une Application Android

Services

- Un service est une opération exécutée en tâche de fond et fonctionne indépendamment de toute activité.
- Un service = un programme sans interface
- Il permet à une application d'indiquer au système d'exploitation qu'il a besoin d'effectuer une opération longue en dehors du cycle normal de l'activité.

Composants d'une Application Android

Récepteurs de diffusion

- Les récepteurs de diffusion gèrent la communication entre le système d'exploitation Android et les applications.
- La communication entre les composants des applications Android se fait par envoie de message via des « **Intent** ».

Composants d'une Application Android

- Différents composants applicatifs Android et les classes associées

Nom	Classe ou paquetage concerné(e)
Activité	android.app.Activity
Service	android.app.Service
Fournisseurs de contenu (content provider)	android.content.ContentProvider
Récepteur de diffusion (Broadcast Receiver)	android.content.BroadcastReceiver

Composants d'une Application Android

- Déclaration des différents composants applicatifs Android dans AndroidManifest:

- **Activité**

```
<activity>
<intent-filter>
        …
</intent-filter>
</activity>
```

- **Service**

```
<service>
<intent-filter>
        …
</intent-filter>
</service>
```

- **Récepteurs de diffusion**

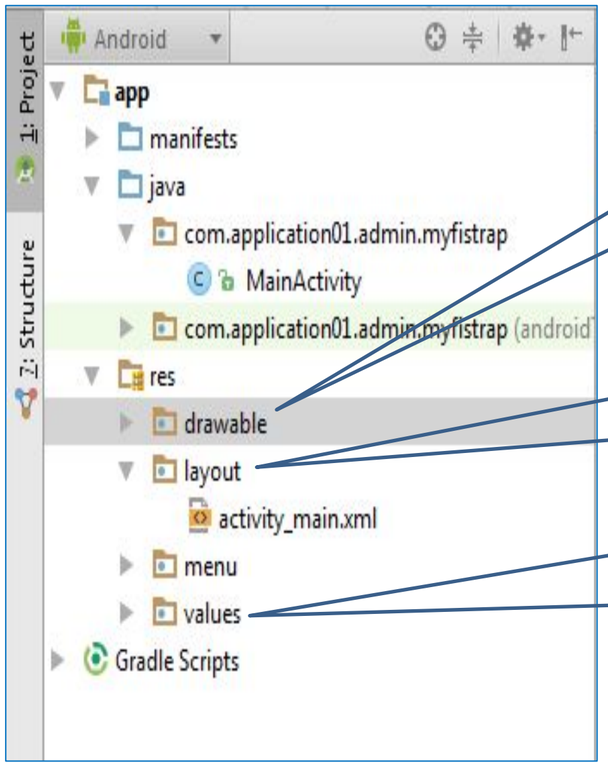
```
<receiver>
<intent-filter>
        …
</intent-filter>
</receiver>
```

- **Fournisseur de contenu**

```
<provider>
<grant-uri-permission …/>
<path-permission …/>
</provider>
```

Ressources

- Les ressources sont des fichiers externes – ne contenant pas d'instruction – qui sont utilisés par le code et liés à une application au moment de sa construction.
  Elles sont décrites très souvent dans des fichiers XML.



drawable: Fichiers qui contiennent des images (les images de type PNG, JPEG ou encore GIF), images ajustables.

layout: Fichiers XML qui représentent la mise en page d'écrans (ou de parties d'écrans), que l'on appelle aussi gabarits.

values: Fichiers XML representent les différents types de ressources comme des chaînes de caractères, des dimensions, des couleurs, etc

Ressources

Type	Description
Dessin et image (res/drawable)	On y trouve les images matricielles (les images de type PNG, JPEG ou encore GIF) ainsi que des fichiers XML qui permettent de décrire des dessins simples (par exemple des cercles ou des carrés).
Mise en page ou interface graphique (res/layout)	sert à définir les interfaces d'une application
Menu (res/menu)	Les fichiers XML pour pouvoir constituer des menus.
Donnée brute (res/raw)	Données diverses au format brut. Ces données ne sont pas des fichiers de ressources standards, on pourrait y mettre de la musique ou des fichiers HTML par exemple.
Différentes variables (res/values)	Il est plus difficile de cibler les ressources qui appartiennent à cette catégorie tant elles sont nombreuses. On y trouve entre autre des variables standards, comme des chaînes de caractères, des dimensions, des couleurs, etc
res/mipmap	les ressources de différentes densités pour les icones de lancement

Ressources

- Voici quelques exemples plus précis de l’utilisation de ressources dans le développement Android :

  - ☞ Elles peuvent être utilisées pour afficher une application en plusieurs langues.

  - ☞ Elles permettent également de changer l’orientation de l’écran (mode portrait ou mode paysage).

  - ☞ Les ressources permettent d’adapter l’application aux différentes tailles et types d’écrans (smartphone, tablette…).

Ressources

- Pour manipuler facilement dans le code java des activités les différentes ressources déclarées dans l'application, Android crée une classe nommée **R** qui sera utilisée pour se référer aux ressources dans le code:

- La classe **R** est générée automatiquement par Android,

- Elle permet à l'application d'accéder aux ressources (dans le code Java):

  Exemple: activity_main.xml => **R**.layout.activity_main

- Elle contient des classes internes dont les noms correspondent aux types de ressources (id, drawable, layout …)

- Elle est constituée à partir des fichiers placés dans les sous répertoires du répertoire **res.**

Ressources

- La classe **R** contient des classes internes dont les noms correspondent aux types de ressources (id, drawable, layout …)

```
public final class R
 {
public static final class string
 {
public static final int app_name=0x7f080000;
 public static final int message=0x7f080001;
 }
public static final class layout {
 public static final int main=0x7f030000;
 }
public static final class menu {
 public static final int main_menu=0x7f050000;
 public static final int context_menu=0x7f050001; }
 ...
```

Ressources

Accès aux ressources

- Référencement d'une ressource dans un fichier XML. La forme générale est :
  "@type/identificateur"

Par exemple :

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/login" />
```

@string/login fait référence à une chaine contenue dans un fichier XML placé dans le répertoire res/values et définie comme suit :

```
<resources>
    <string name="login">Entrer votre login</string>
</resources>
```

- Référencement d'une ressource dans le code. La forme générale est
  R.type.identificateur ou bien getResources().getString(R.string.hello_world)

Par exemple :

R.string.login fait référence à la même chaine

Ressources

Valeurs simples

- Il est possible de déclarer dans des fichier XML,
  - ☞ des chaînes de caractères,
  - ☞ des dimensions,
  - ☞ des couleurs,
  - ☞ des tableaux (de chaînes de caractères ou d’entiers) comme valeurs simples.

Ressources

Création des ressources

Couleurs

- Couleur définit une valeur RVB (rouge, vert et bleu) et une transparence.

  Il existe différents formats dont la syntaxe globale est la suivante :

  ```
  <color name=nom_couleur>valeur_de_la_couleur</color>
  ```

- Définition d'un fichier de ressources de couleurs:

  ```
  <resources>
  <color name="vert">#48E10BF</color>
  </resources>
  ```

- Utilisation des couleurs (exemple avec la déclaration de la couleur « vert »):
  - Java : R.color.vert

Ressources

Création des ressources

Chaînes de caractères et texte formaté

- Les chaînes de caractères avec un format optionnel peuvent être utilisées comme ressources.
  syntaxe : <string name=nom_chaine>valeur_de_la_chaine</string>

- utilisation de trois balises HTML standard (gras), <i> (italique) et <u> (souligné) :
  <string name="ex">Un texte mis en forme </string>

- si vous voulez utiliser des guillemets ou des apostrophes, vous devez les échapper en les faisant précéder du caractère slash ('\'):
  <string name="ex2">Voici l\'application</string>

- exemples de déclaration de chaînes de caractères dans un fichier de ressources de chaînes de caractères

```
<resources>
    <string name="app_name">Calculatrice</string>
    <string name="login">Entrer votre login</string>
</resources>
```

Ressources

Création des ressources

Chaînes de caractères et texte formaté

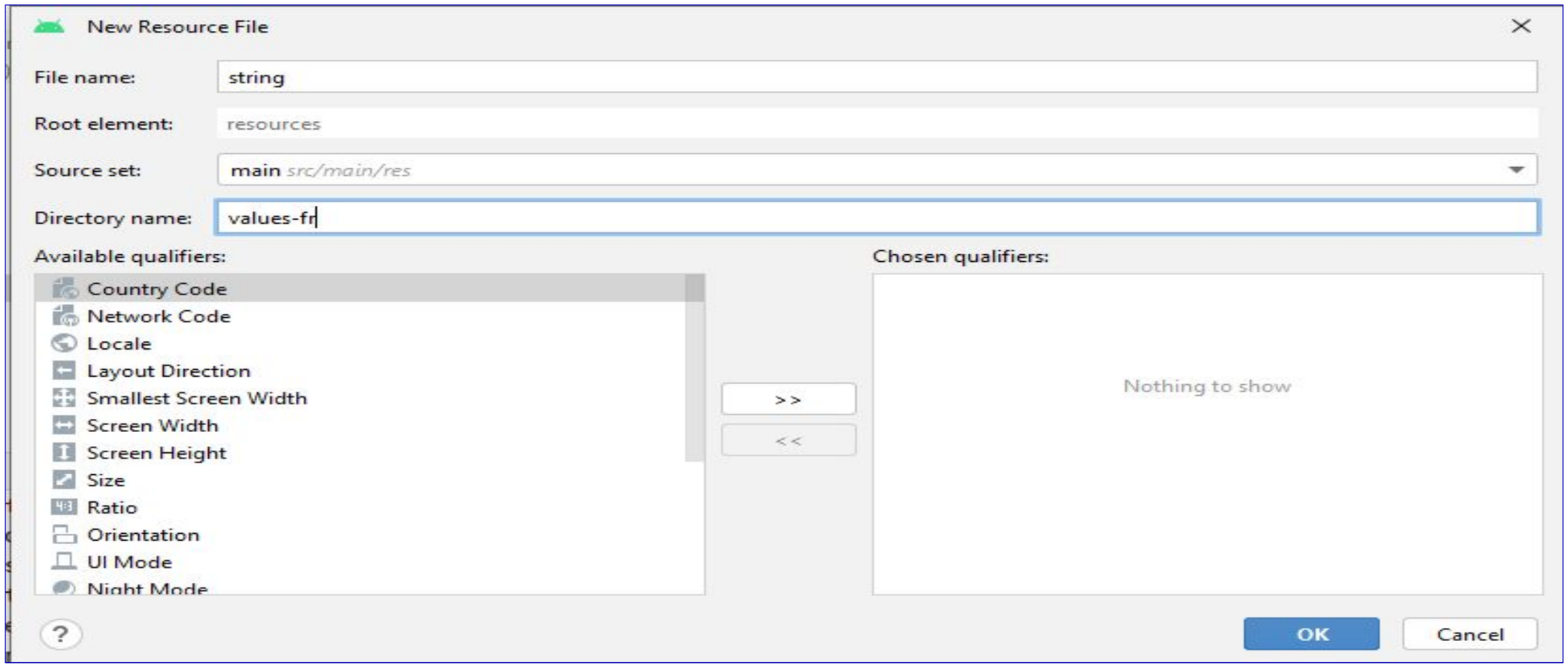
- Le système de ressources permet de gérer très facilement l'internationalisation d'une application.
- Il suffit de créer des répertoires values-XX ou XX est le code de la langue que l'on souhaite implanter.
- On place alors dans sous répertoire le fichier string.xml contenant les chaînes traduites associées aux même clefs que dans values/string.xml.
- **Exemple:**

values-fr/string.xml
values-en/string.xml

Ressources

Création des ressources

- **Chaînes de caractères et texte formaté**



Ressources

Création des ressources

Unités de mesure (« dimensions »)

- Les dimensions sont la plupart du temps référencées dans les styles et les mises en page.
- Elles peuvent servir de constantes entre différents éléments d'écrans.
- Unités prises en charge par Android :
  - px (pixels),
  - in (pouces),
  - mm (millimètres),
  - pt (points),
  - dp (density-independant pixel),
  - sp (scale-independant pixel).
- Définition d'un fichier de ressources de dimensions

```
<resources>
    <dimen name="taille_texte">5sp</dimen>
</resources>
```

Ressources

Création des ressources

Styles:

- Un style est un ensemble de propriétés s'appliquent à un widget.
- Définition d'un fichier de ressources de styles

```
<resources>
<style name="style_txt">
   <item
name="android:textColor">@color/vert</item>
   <item name="android:textSize">25px</item>
  </style>
</resources>
```

styles.xml

```
<TextView style="@style/style_txt"
      android:id="@+id/textView2"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
      android:text="@string/hello"
      />
```

activity.xml