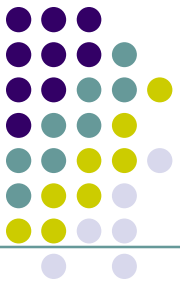




Année académique 2017-2018



Développement Mobile sous Android



Références

1. Android, Guide de développement d'applications Java pour Smartphones et Tablettes. Sylvain HEBUTERNE, Édition : ENI - 562 pages , 3^e édition, 1^{er} août 2016.
2. Livres francophones sur le développement mobile sous Android.
<https://android.developpez.com/livres/>
3. <https://developer.android.com>
4. <https://android.developpez.com/cours/>
5. Tout comprendre sur les Fragments
<http://mathias-seguy.developpez.com/tutoriels/android/comprendre-fragments>
6. Youtube.com



Sommaire

- Introduction
- Création de votre première application
- Prise en charge de différents appareils
- Sauvegarde des données
- Interaction avec d'autres applications
- Prise en compte des permissions systèmes
- Intégration des éléments multimédias
- Les meilleurs pratiques
- Test d'une application Android
- Distribution et monétisation sur Google Play

Introduction





Créer votre première application

Objectifs:

- *Dans cette partie, nous allons vous apprendre comment créer votre première application Android.*
- *Vous apprendrez à créer un projet Android avec Android Studio et à exécuter une version débuggable de l'application.*
- *Vous apprendrez également quelques principes fondamentaux du design de l'application Android, y compris la façon de créer une interface utilisateur simple et de gérer l'entrée des utilisateurs.*



Créer votre première application

Créer un projet Android

Dans cette partie, on verra comment créer un nouveau projet Android avec Android Studio et décrire certains des fichiers du projet.

1. Dans Android Studio, créez un nouveau projet:
 - Si vous n'avez pas de projet ouvert, dans la fenêtre **Welcome to Android Studio**, cliquez sur **Start a new Android Studio project**.
 - Si vous avez ouvert un projet, sélectionnez **File > New Project**.
2. Dans l'écran **New Project**, entrez les valeurs suivantes:
 - **Application Name** : "My First App"
 - **Company Domain** : "example.com"
 - On peut bien modifier l'emplacement du projet, mais laissons les autres options telles qu'elles sont.
3. Cliquez sur **Next**.



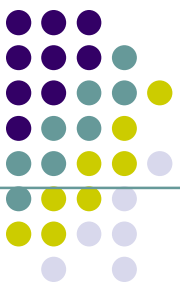
Créer votre première application

Créer un projet Android

4. Dans l'écran **Target Android Devices**, conservez les valeurs par défaut, puis cliquez sur **Suivant** .
5. Dans l'écran **Add an Activity to Mobile**, sélectionnez **Empty Activity** et cliquez sur **Suivant** .
6. Dans l'écran **Customize the Activity** , conservez les valeurs par défaut, puis cliquez sur **Terminer** .

Après un certain traitement, Android Studio ouvre l'IDE. Maintenant, prenez un moment pour passer en revue les fichiers les plus importants.

Tout d'abord, assurez-vous que la fenêtre **Project** est ouverte (sélectionnez **View > Tool Windows > Project**) et la vue **Android** est sélectionnée dans la liste déroulante en haut de cette fenêtre. Vous pouvez ensuite voir les fichiers suivants:



Créer votre première application

Créer un projet Android

- **app > java > com.example.myfirstapp > MainActivity.java**

C'est l'activité principale (le point d'entrée de votre application). Lorsqu'on crée et exécute l'application, le système lance une instance de cette activité et charge sa mise en page.

- **app > res > layout > activity_main.xml**

Ce fichier XML définit la mise en page de l'interface utilisateur de l'activité. Il contient un élément TextView avec le texte "Hello world!".

- **app > manifests > AndroidManifest.xml**

Le fichier manifeste décrit les caractéristiques fondamentales de l'application et définit chacun de ses composants.

- **Gradle Scripts > build.gradle**

On voit deux fichiers avec ce nom: un pour le projet et un pour le module "app". Chaque module possède son propre fichier **build.gradle**, mais ce projet possède actuellement un seul module. Vous travaillerez plus avec le fichier **build.gradle** du module pour configurer la façon dont les outils Gradle compilent et développent votre application. .



Créer votre première application

Exécutez votre application

Dans la partie précédente, un projet Android qui affiche "Hello World" a été créé. Cette partie montre comment exécuter l'application sur un périphérique réel ou un émulateur

Exécuter sur un véritable périphérique

- Configurez votre appareil comme suit:
 1. Connectez votre appareil à votre machine de développement avec un câble USB. Si vous développez sur Windows, vous devrez installer le pilote USB approprié pour votre appareil. Pour plus d'aide sur l'installation des pilotes, voir l'adresse <https://developer.android.com/studio/run/oem-usb.html>.
 2. Activez le **débogage USB** sur votre appareil en allant dans **Paramètres > Options de développeur**.



Remarque: sur Android 4.2 et plus, les **options de développeur** sont cachées par défaut. Pour le rendre disponible, allez dans **Paramètres > À propos du téléphone** et appuyez sur **Build number** sept fois. Revenez à l'écran **précédent** pour trouver les **options de développeur**.



Créer votre première application

Exécutez votre application

Exécuter sur un véritable périphérique

- Exécutez l'application depuis Android Studio  le suit:
 1. Dans Android Studio, cliquez sur le module **app** dans la fenêtre **Project** , puis sélectionnez **Run > Run** (ou cliquez sur  dans la barre d'outils).
 2. Dans la fenêtre **Select Deployment Target**, sélectionnez votre appareil, puis cliquez sur **OK** .

Android Studio installe l'application sur votre appareil connecté et le démarre.




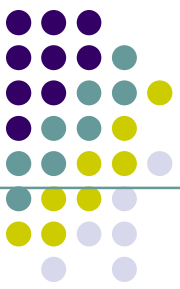
Créer votre première application

Exécutez votre application

Exécuter sur un émulateur

Avant d'exécuter votre application sur un émulateur, vous devez créer une définition de périphérique virtuel Android ou Android Virtual Device (AVD). Une AVD spécifie les caractéristiques d'un téléphone Android, d'une tablette, d'un Android Wear ou d'un appareil Android TV que vous souhaitez simuler dans l'émulateur Android.

- **Créez une définition AVD comme suit:**
 1. Lancez le Gestionnaire de périphériques virtuels Android en sélectionnant **Tools > Android > AVD Manager** ou en cliquant sur l'icône AVD Manager  de la barre d'outils.
 2. Dans l'écran **Your Virtual Devices**, cliquez sur **Create Virtual Device**.
 3. Dans l'écran **Select Hardware**, sélectionnez un périphérique de téléphone, tel que Pixel, puis cliquez sur **Next**.
 4. Dans l'écran **System Image**, cliquez sur **Download** sur l'une des images système recommandées. Acceptez les termes pour compléter le téléchargement.



Créer votre première application

Exécutez votre application

Exécuter sur un émulateur

- **Créez une définition AVD comme suit :**
 5. A la fin du téléchargement, sélectionnez l'image système dans la liste et cliquez sur **Next**.
 6. Sur l'écran suivant, acceptez les paramètres de configuration proposés et cliquez sur **Finish**.
 7. De retour dans l'écran **Your Virtual Devices**, sélectionnez le périphérique que vous venez de créer et cliquez sur **Launch this AVD in the emulator**.
- **Exécuter l'application :**

Pendant que l'émulateur démarre, fermez la fenêtre du Gestionnaire de périphériques virtuels Android et revenez à votre projet afin que vous puissiez exécuter l'application:

1. Une fois que l'émulateur est démarré, cliquez sur le module **app** dans la fenêtre **Project** , puis sélectionnez **Run > Run**
2. Dans la fenêtre **Select Deployment Target**, sélectionnez l'émulateur et cliquez sur **OK**

Android Studio installe l'application sur l'émulateur et la démarre.



Créer votre première application

Créer une interface utilisateur simple

Dans cette partie, vous utiliserez l'éditeur de mise en page Android Studio pour créer une mise en page qui comprend une zone de texte et un bouton. Dans la leçon suivante, vous allez faire fonctionner l'application en appuyant sur le bouton en envoyant le contenu de la zone de texte à une autre activité.

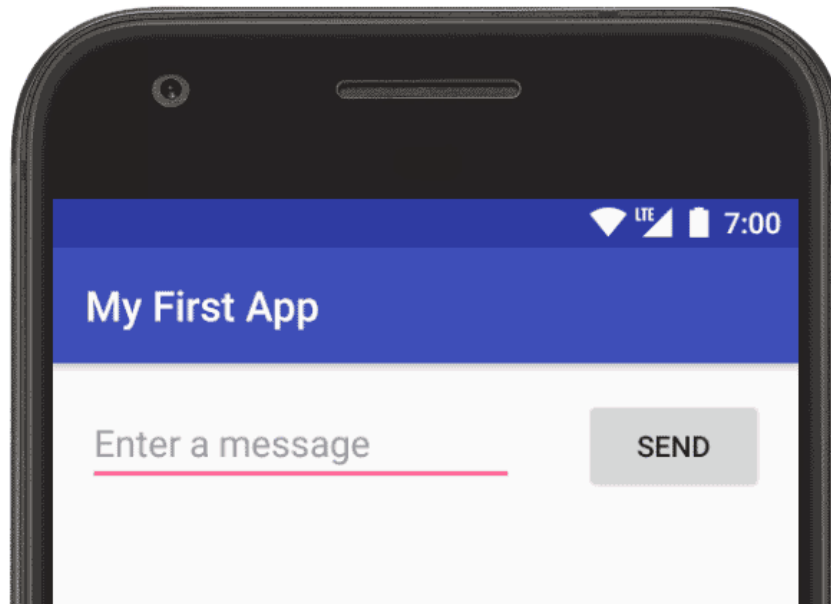


Figure 1. Capture d'écran de la mise en page finale



Créer votre première application

Créer une interface utilisateur simple

- **A propos de l'interface utilisateur**

L'interface utilisateur d'une application Android est construite à l'aide d'une hiérarchie de *mises en page* (*Layout*) (objets `ViewGroup`) et *widgets* (objets `View`).

Les mises en page sont des conteneurs invisibles qui contrôlent la visualisation de leurs éléments enfants sur l'écran.

Les widgets sont des composants UI tels que des boutons et des zones de texte.

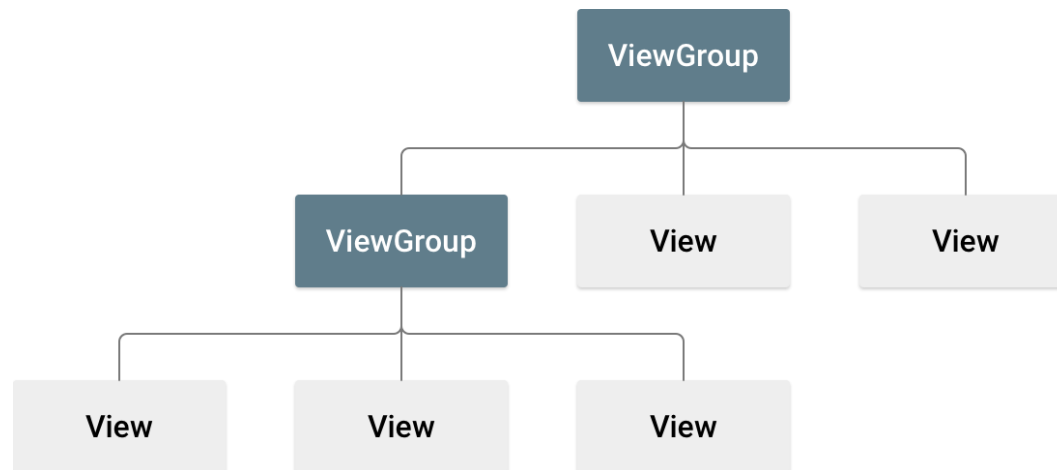


Figure 2. Illustration de la façon dont les objets `ViewGroup` forment des branches dans la mise en page et contiennent les objets `View`



Créer votre première application


Créer une interface utilisateur simple

- **Configuration de l'éditeur de mise en page**

Android fournit un vocabulaire XML pour les classes ViewGroup et View, de sorte que la plupart de vos UI est définie dans les fichiers XML.

Cependant, au lieu de vous apprendre à écrire un peu de XML, cette partie vous montre comment créer une mise en page à l'aide de l'éditeur de mise en page de Android Studio, ce qui facilite la création d'une mise en page en faisant glisser des vues

Pour commencer, configurez votre espace de travail comme suit:






1. Dans la fenêtre Projet de Android Studio, ouvrez **app > res > layout > activity_main.xml**.
2. Pour donner plus de place à l'éditeur de mise en page, cachez la fenêtre **Project** en sélectionnant **View > Tool Windows > Project** (ou cliquez sur **Project**  situé sur le côté gauche d'Android Studio
3. Si votre éditeur affiche la source XML, cliquez sur l'onglet **Design** en bas de la fenêtre.



Créer votre première application

Créer une interface utilisateur simple

- **Configuration de l'éditeur de mise en page**

1. Cliquez sur **Show Blueprint**  de sorte que seule la disposition du plan soit visible.
2. Assurez-vous que **Show Constraints** est activée. Ce qui correspond à l'icône .
3. Assurez-vous que la connexion automatique est désactivée. Ce qui correspond à l'icône .
4. Cliquez sur **Default Margins (Marges par défaut)**  **8** dans la barre d'outils et sélectionnez **16** (vous pouvez encore ajuster la marge pour chaque vue plus tard).
5. Cliquez sur **Device in Editor (Appareil dans l'éditeur)**  dans la barre d'outils et sélectionnez **Pixel XL**

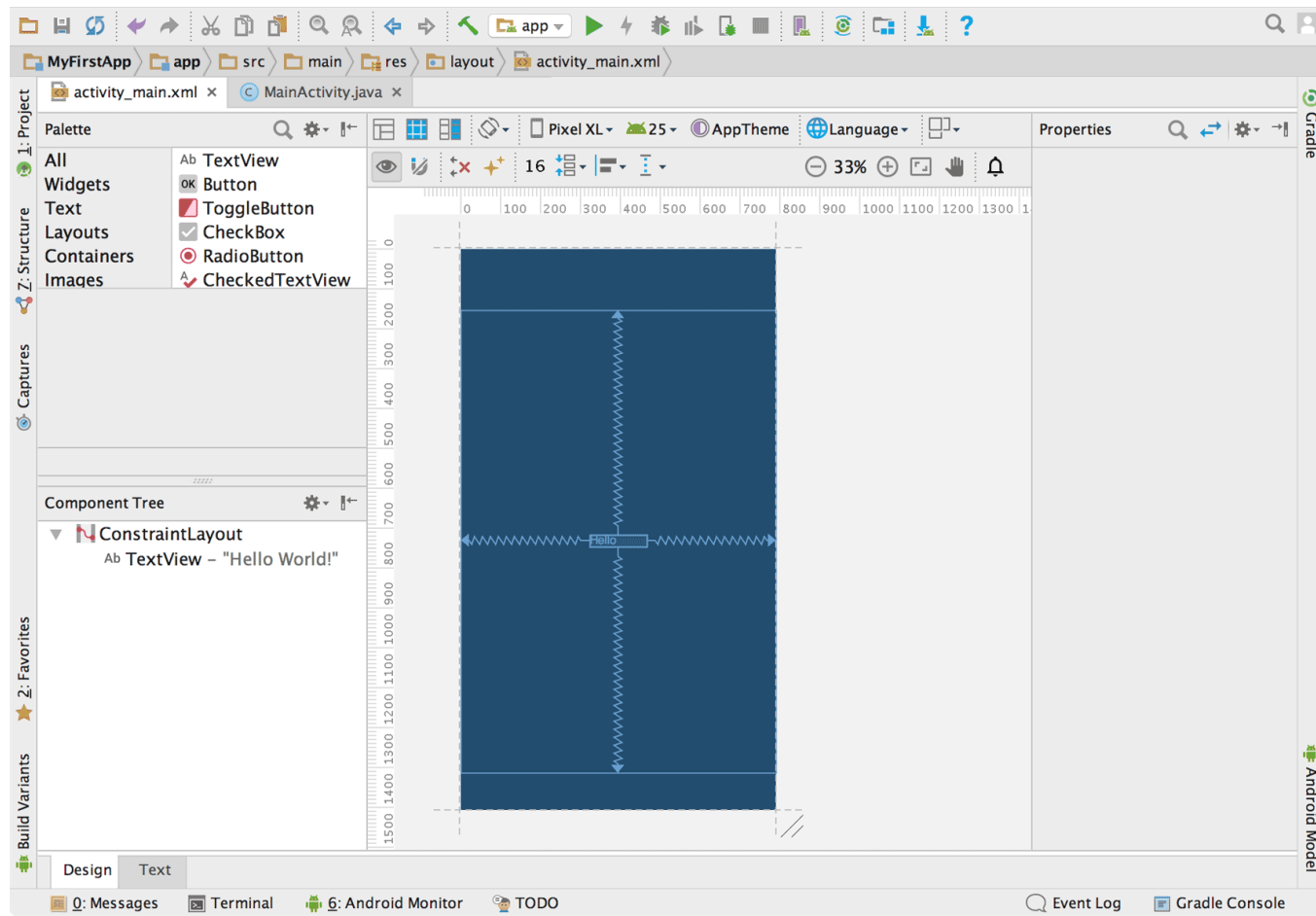


Créer votre première application

Créer une interface utilisateur simple

- Configuration de l'éditeur de mise en page

Votre éditeur devrait maintenant regarder comme le montre la figure 3.





Créer votre première application

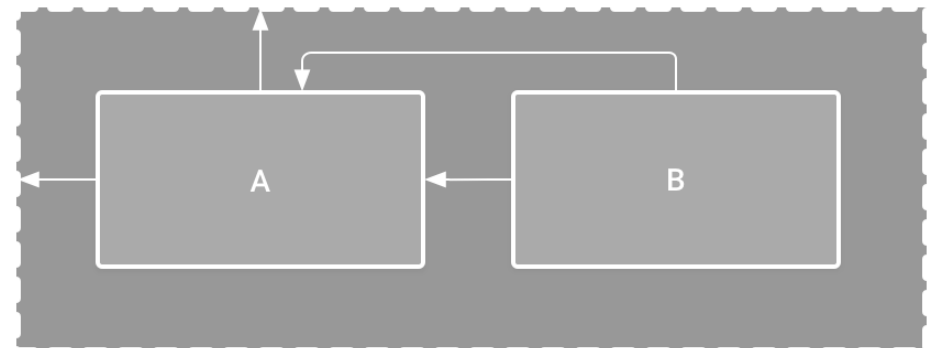
Créer une interface utilisateur simple

- **La fenêtre Component Tree**

1. La fenêtre **Component Tree** (**Arbre de composant**) en bas à gauche montre la hiérarchie des vues de la mise en page. Dans ce cas, la vue racine est `ConstraintLayout`, contenant seulement un objet `TextView`.
2. **ConstraintLayout** est une mise en page qui définit la position de chaque vue en fonction des contraintes des vues frères et à la mise en page parentale. De cette façon, vous pouvez créer des mises en page simples et complexes avec une hiérarchie à vue plate. Cela évite le besoin de dispositions imbriquées, ce qui peut augmenter le temps requis pour dessiner l'interface utilisateur.

Exemple : vous pouvez déclarer la disposition suivante (figure 4):

- La vue A apparaît à 16dp du haut et à 16dp à gauche de la disposition parentale.
- La vue B apparaît 16dp à droite de la vue A et est alignée horizontalement avec la vue A.



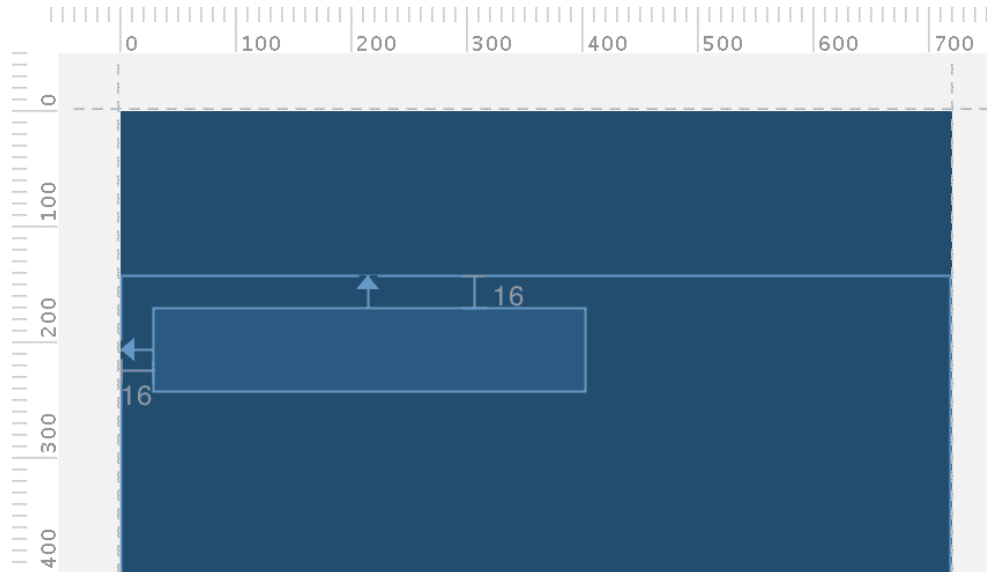


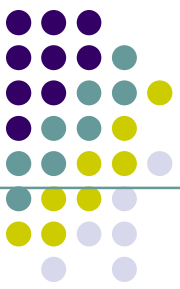
Créer votre première application

Créer une interface utilisateur simple

- **Ajout d'une zone de texte**

1. Tout d'abord, vous devez supprimer ce qui est déjà dans la mise en page. Faites un clique-droit sur `TextView` dans la fenêtre de l'arbre des composants, puis appuyez sur **Supprimer**.
2. Dans la fenêtre **Palette** à gauche, cliquez sur **Text** dans le volet gauche, puis faites glisser **Plain Text** (texte brut) dans l'éditeur de conception et déposez-le près du haut de la mise en page. Il s'agit d'un widget `EditText` qui accepte l'entrée de texte brut.





Créer votre première application

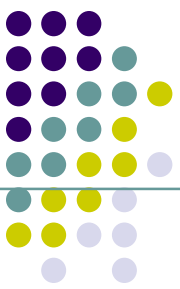
Créer une interface utilisateur simple

- **Ajout d'une zone de texte**

1. Cliquez sur la vue dans l'éditeur de conception (design editor). Vous pouvez maintenant voir les poignées de redimensionnement sur chaque coin (carrés) et les ancrages de contrainte de chaque côté (cercles).

Pour un meilleur contrôle, vous pouvez zoomer sur l'éditeur.


2. Cliquez et maintenez l'ancre sur le côté supérieur, puis faites-le glisser jusqu'à ce qu'il s'enclenche en haut de la disposition et relâche. C'est une contrainte - il spécifie que la vue devrait être 16dp du haut de la mise en page (car vous avez défini la marge par défaut à 16dp).
3. De même, créez une contrainte du côté gauche de la vue sur le côté gauche de la mise en page



Créer votre première application

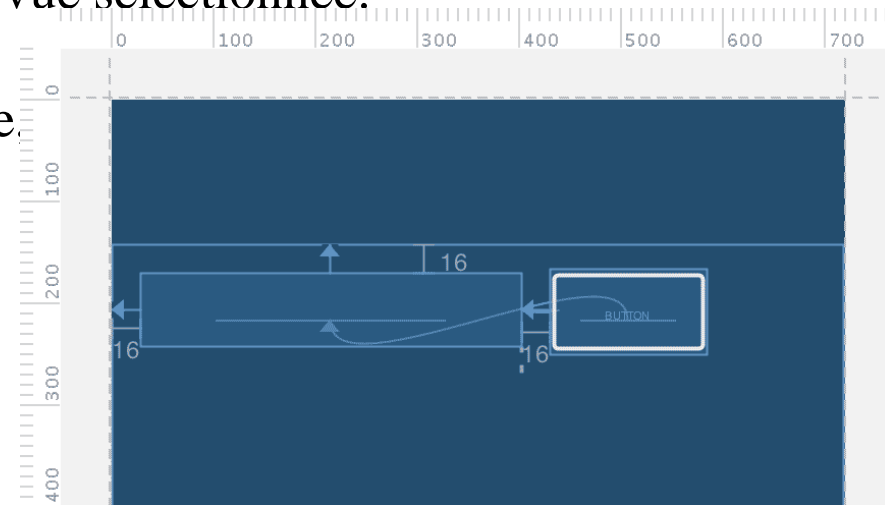
Créer une interface utilisateur simple

- **Ajout d'un bouton**

1. Dans la fenêtre **Palette**, cliquez sur **Widgets** dans le volet gauche, puis faites glisser le bouton dans l'éditeur de conception et déposez-le près du côté droit.
2. Créez une contrainte du côté gauche du bouton sur le côté droit de la zone de texte.
3. Pour limiter les vues dans un alignement horizontal, vous devez créer une contrainte entre les lignes de base du texte. Cliquez donc sur le bouton, puis cliquez sur **Baseline Constraint**  (Contrainte de ligne de base), qui apparaît dans l'éditeur de conception directement en dessous de la vue sélectionnée.

L'ancre de ligne de base apparaît dans le bouton. Cliquez et maintenez sur cette ancre puis faites-la glisser vers l'ancre de ligne de base qui apparaît dans la zone de texte.

Le résultat devrait ressembler à la capture d'écran de la figure 6.






Créer votre première application

Créer une interface utilisateur simple

- **Changer les chaînes d'interface utilisateur**

Cliquez sur **Afficher le design** (Show Design ) dans la barre d'outils pour prévisualiser l'interface utilisateur. Notez que l'entrée de texte est pré-remplie avec "Name" et que le bouton est intitulé "Button". Donc maintenant vous allez changer ces chaînes.

1. Ouvrez la fenêtre **Project** , puis **sélectionnez res>values> strings.xml**.

Il s'agit d'un fichier de ressources de chaîne où vous devez spécifier toutes vos chaînes d'interface utilisateur. Cela vous permet de gérer toutes les chaînes UI dans un seul emplacement, ce qui facilite la recherche, la mise à jour et la localisation (par rapport aux chaînes de codage dur de votre mise en page ou de votre code d'application).


1. Cliquez sur **Open editor** (Ouvrir l'éditeur) en haut de la fenêtre de l'éditeur. Cela ouvre l'éditeur de traductions, qui fournit une interface simple pour ajouter et modifier vos chaînes par défaut, et permet de garder toutes vos chaînes traduites organisées.

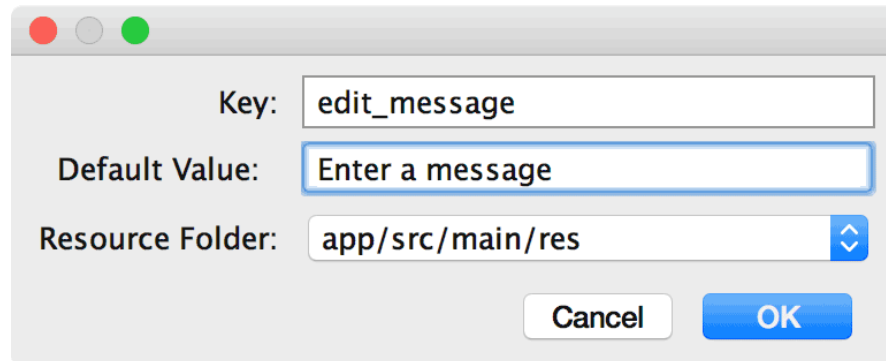


Créer votre première application

Créer une interface utilisateur simple

- **Changer les chaînes d'interface utilisateur**

1. Cliquez sur **Add Key**  (Ajouter une clé) pour créer une nouvelle chaîne comme "texte d'avis" pour la zone de texte.
 - a. Entrez "edit_message" pour le nom de la clé.
 - b. Entrez "Entrez un message" pour la valeur.
 - c. Cliquez sur OK .



2. Ajoutez une autre clé nommée "button_send" avec une valeur de "Envoyer".




Créer votre première application

Créer une interface utilisateur simple

- **Changer les chaînes d'interface utilisateur**

Maintenant, vous pouvez définir ces chaînes pour chaque vue. Revenez ensuite au fichier de mise en page en cliquant sur **activity_main.xml** dans la barre d'onglets et ajoutez les chaînes comme suit:

1. Cliquez sur la zone de texte dans la mise en page et, si la fenêtre Propriétés n'est pas déjà visible sur la droite, cliquez sur **Properties** sur la barre latérale droite.
2. Localisez la propriété **hint** (suggestion), puis cliquez sur **Pick a Resource**  (Choisir une ressource) à droite de la zone de texte. Dans la boîte de dialogue qui apparaît, double-cliquez sur **edit_message** de la liste.
3. Toujours visualiser les propriétés de la zone de texte, supprimez également la valeur de la propriété **text** (actuellement définie sur "Nom").
4. Cliquez maintenant sur le bouton dans la mise en page, localisez la propriété de **text**, cliquez sur **Pick a Resource**, puis sélectionnez **button_send**



Créer votre première application

Créer une interface utilisateur simple

- **Rendre la taille de la zone de texte flexible**

Pour créer une mise en page qui répond aux différentes tailles d'écran, vous allez maintenant faire rendre la zone de texte flexible pour compléter tous les espaces horizontaux restants (après avoir comptabilisé le bouton et les marges).

1. Créez une contrainte du côté droit du bouton sur le côté droit de la disposition parentale. Cela définit maintenant la largeur totale que les deux vues sont disponibles (que vous pouvez maintenant remplir avec la zone de texte).
2. Ajoutez une contrainte du côté droit de la zone de texte sur le côté gauche du bouton. Il pourrait sembler qu'il soit déjà là, mais en fait, vous ajoutez une contrainte bidirectionnelle entre les deux vues. Les deux vues sont donc contraintes les unes aux autres. C'est ce qu'on appelle une chaîne (comme indiqué par la chaîne entre les vues) et permet d'ajouter des options de mise en page supplémentaires.

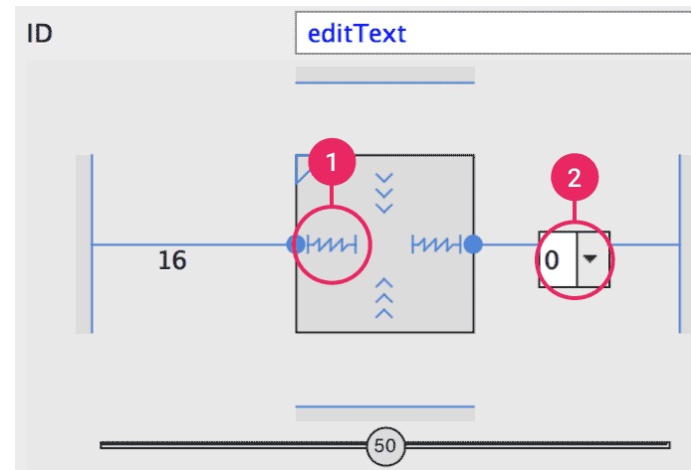


Créer votre première application

Créer une interface utilisateur simple

- **Rendre la taille de la zone de texte flexible**

1. Ouvrez la fenêtre Propriétés pour la zone de texte, puis cliquez sur l'indicateur de largeur jusqu'à ce qu'il soit configuré pour correspondre à **Match Constraints**, comme l'indique la légende 1 de la figure ci-dessous. "**Match Constraints**" signifie que la largeur est maintenant déterminée par les contraintes horizontales et les marges. Par conséquent, la zone de texte s'étend pour compléter l'espace horizontal.



1. Cependant, les deux vues sont séparées par 32dp au lieu de 16dp, car les deux vues ont des marges. Ainsi, tout en affichant les propriétés de la boîte de texte, modifiez la marge droite sur 0, comme indiqué par la légende 2 de la figure ci-dessus.

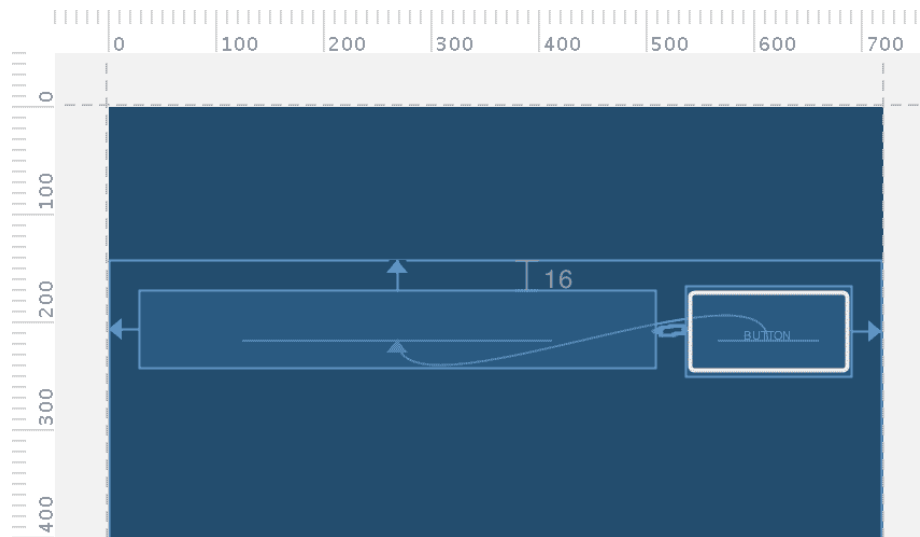


Créer votre première application

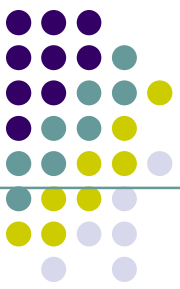
Créer une interface utilisateur simple

- **Rendre la taille de la zone de texte flexible**

Maintenant, la mise en page est terminée et devrait apparaître comme la figure 9.



La zone de texte s'étend pour compléter l'espace restant



Créer votre première application

Créer une interface utilisateur simple

- **Rendre la taille de la zone de texte flexible**

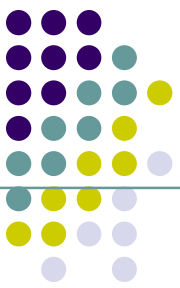
Le code XML de la mise en page finale devrait ressembler à ceci :

- **Exécuter l'application**

```
<?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="com.example.myfirstapp.MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="0dp"
        android:layout_marginTop="16dp"
        android:hint="@string/edit_message"
        android:inputType="textPersonName"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/button"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_send"
        app:layout_constraintBaseline_toBaselineOf="@+id/editText"
        app:layout_constraintLeft_toRightOf="@+id/editText"
        app:layout_constraintRight_toRightOf="parent" />
</android.support.constraint.ConstraintLayout>
```



Créer votre première application

Commencer une autre activité

Après avoir terminé l'activité précédente, vous avez obtenu une application qui affiche une activité (un seul écran) avec un champ de texte et un bouton. Dans cette nouvelle partie, vous allez ajouter dans MainActivity un code qui démarre une nouvelle activité pour afficher le message lorsque l'utilisateur tape sur Envoyer .

Répondre au bouton d'envoi

Ajoutez dans MainActivity.java une méthode appelée par le bouton comme suit:

1. Dans le fichier **app>java>com.example.myfirstapp>MainActivity.java**, ajoutez la déclaration de la méthode sendMessage() comme indiqué ci-dessous:

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /** Called when the user taps the Send button */  
    public void sendMessage(View view) {  
        // Do something in response to button  
    }  
}
```



Créer votre première application

Commencer une autre activité

Répondre au bouton d'envoi

Vous pouvez voir une erreur car Android Studio ne peut pas résoudre la classe View utilisée comme argument. Cliquez donc pour placer votre curseur sur la déclaration View, puis effectuez une correction rapide en appuyant sur Alt + Entrée (ou Option + Entrée sur Mac). (Si un menu apparaît, sélectionnez **Import class**).

2. Revenez maintenant au fichier **activity_main.xml** pour appeler cette méthode à partir du bouton:
 - a. Cliquez pour sélectionner le bouton dans l'Éditeur de mise en page.
 - b. Dans la fenêtre **Propriétés**, localisez la propriété **onClick** et sélectionnez **sendMessage [MainActivity]** dans la liste déroulante .

Remarques: Les détails suivants de cette méthode qui sont nécessaires pour que le système le reconnaisse compatible avec l'attribut android:onClick . Plus précisément, la méthode doit être déclarée avec :

- Un accès public
- Une valeur de retour nulle
- Une View comme seul paramètre (c'est l'objet View qui a été cliqué)



Créer votre première application

Commencer une autre activité

Construire une intention

- Une Intent est un objet qui fournit une liaison d'exécution entre des composants distincts, tels que deux activités. Une Intent représente "l'intention de l'application de faire quelque chose". Vous pouvez utiliser les intents pour une grande variété de tâches, mais dans cette leçon, votre intention commence une autre activité.
- Dans `MainActivity.java`, ajoutez la constante `EXTRA_MESSAGE` et le code de `sendMessage()` comme indiqué ici:

```
public class MainActivity extends AppCompatActivity {  
    public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    /** Called when the user taps the Send button */  
    public void sendMessage(View view) {  
        Intent intent = new Intent(this, DisplayMessageActivity.class);  
        EditText editText = (EditText) findViewById(R.id.editText);  
        String message = editText.getText().toString();  
        intent.putExtra(EXTRA_MESSAGE, message);  
        startActivity(intent);  
    }  
}
```



Créer votre première application

Commencer une autre activité

Construire une intention

- Android Studio va de nouveau rencontrer des problèmes pour résoudre les erreurs de symboles. Alors appuyez sur Alt + Entrée (ou Option + Retour sur Mac). Vos importations devraient se terminer comme suit:

```
Importer android.content.Intent;  
Importer android.support.v7.app.AppCompatActivity;  
Importer android.os.Bundle;  
Importer android.view.View;  
Importer android.widget.EditText;
```

- Une erreur demeure sur **DisplayMessageActivity**, mais il sera résolue dans la suite.

Voici ce qui se passe dans **sendMessage()** :

1. Le constructeur **Intent** prend deux paramètres:
 - Un **Context** comme premier paramètre (**this** est utilisé car la classe **Activity** est une sous-classe de **Context**)
 - La Class du composant de l'application à laquelle le système doit fournir l'intention (dans ce cas, l'activité qui doit être lancée).



Créer votre première application

Commencer une autre activité

Construire une intent

Voici ce qui se passe dans `sendMessage()` :

1. La méthode `putExtra()` ajoute la valeur `EditText` à l'intent . Une Intent peut transporter des types de données en tant que paires de valeur-clé appelées extras.
 - La clé est une constante publique `EXTRA_MESSAGE` car l'activité suivante utilise cette clé pour retrouver la valeur du texte.
 - C'est une bonne pratique de définir des clés pour les extras d'intent en utilisant le nom de package de votre application comme préfixe.
 - Cela garantit que les clés soient uniques, au cas où votre application interagirait avec d'autres applications.
2. La méthode `startActivity()` démarre une instance de `DisplayMessageActivity` spécifiée par l'intention.
3. Maintenant, vous devez créer cette classe



Créer votre première application

Commencer une autre activité

Créer la deuxième activité

1. Dans la fenêtre **Projet**, cliquez avec le bouton droit de la souris sur le dossier de l'application et sélectionnez **New > Activity > Empty Activity**
 2. Dans la fenêtre **Configure Activity**, entrez "DisplayMessageActivity" pour Activity Name et cliquez sur Terminer (laissez toutes les autres propriétés définies par défaut).
- ✓ Android Studio effectue automatiquement trois choses:
 - Crée le fichier **DisplayMessageActivity.java**.
 - Crée le fichier de mise en page correspondant **activity_display_message.xml**.
 - Ajoute l'élément **<activity>** requis dans **AndroidManifest.xml**.
 - ✓ Si vous exécutez l'application et appuyez sur le bouton sur la première activité, la deuxième activité démarre mais est vide. C'est parce que la deuxième activité utilise la disposition vide fournie par le modèle




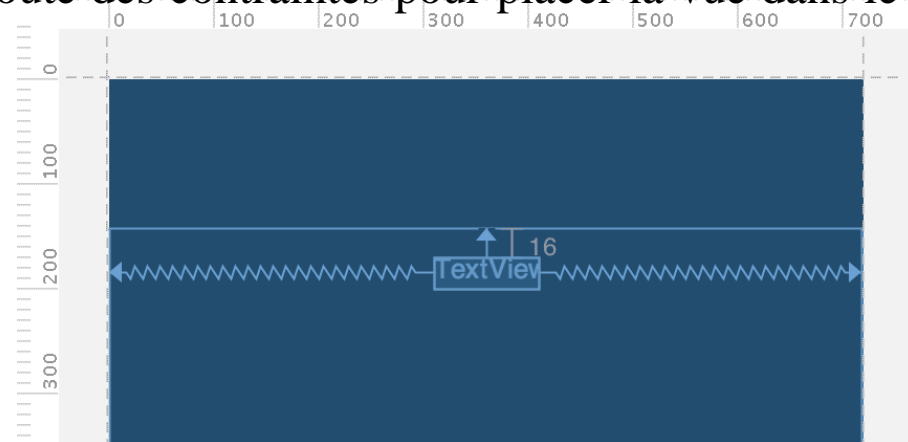
Créer votre première application

Commencer une autre activité

Ajouter un text view (une vue de texte)

La nouvelle activité comprend un fichier de mise en page vide, alors maintenant, vous ajouterez une vue de texte où le message apparaîtra.

1. Ouvrez le fichier **app > res > layout > activity_display_message.xml**
2. Cliquez sur **Turn On Autoconnect**  dans la barre d'outils.
3. Dans la fenêtre **Palette**, faites glisser une **TextView** dans la mise en page et placez-la près du sommet de la mise en page, près du centre, de sorte qu'elle s'arrête à la **ligne verticale** qui apparaît, puis laissez-la. **Autoconnect** ajoute des contraintes pour placer la vue dans le centre horizontal.
4. Créez une autre contrainte du haut de la vue du texte vers le haut de la mise en page, de sorte qu'elle apparaît comme indiqué sur la figure ci-contre.



Exercice : Effectuez quelques ajustements au style de texte en développant **textAppearance** dans la fenêtre **Propriétés** et modifiez les attributs tels que **textSize** et **textColor**



Créer votre première application

Commencer une autre activité

Afficher le message

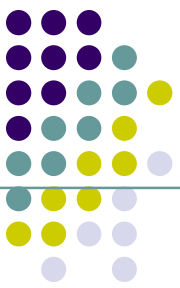
Maintenant, vous modifiez la deuxième activité pour afficher le message transmis par la première activité.

1. Dans `DisplayMessageActivity.java`, ajoutez le code suivant à la méthode `onCreate()`

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

    // Obtenez l'intention qui a commencé cette activité et extrayez la chaîne
    Intent intention = getIntent ();
    String message = intention.getStringExtra (MainActivity.EXTRA_MESSAGE);

    // Capture le TextView de la mise en page et définissez la chaîne comme son texte
    TextView textView = (TextView) findViewById (R.id.textview);
    textView.setText (message);
}
```



Créer votre première application

Commencer une autre activité

Afficher le message

1. Appuyez sur Alt + Entrée (ou Option + Retour sur Mac) pour importer les classes manquantes. Vos importations devraient se terminer comme suit:

```
Importer android.content.Intent;  
Importer android.support.v7.app.AppCompatActivity;  
Importer android.os.Bundle;  
Importer android.view.ViewGroup;  
Importer android.widget.TextView;
```



Créer votre première application

Commencer une autre activité

Ajouter un bouton de retour

- Chaque écran de votre application qui n'est pas l'entrée principale devrait avoir un bouton de retour pour permettre à l'utilisateur de retourner à l'écran parent logique dans la hiérarchie de l'application.
- Pour cela, vous devez déclarer quelle activité est le parent logique dans le fichier `AndroidManifest.xml`.
- Alors ouvrez le fichier **app>Manifests>AndroidManifest.xml**, localisez la balise `<activity>` correspondant à `DisplayMessageActivity` et la remplacer par ce qui suit:

```
<activity android:name=".DisplayMessageActivity"
          android:parentActivityName=".MainActivity" >
    <!-- The meta-data tag is required if you support API
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

- Le système ajoute automatiquement le bouton de retour dans la barre de l'application



Créer votre première application

Commencer une autre activité

Exécuter l'application

- Maintenant, exécutez l'application à nouveau en cliquant sur **Apply Changes** ⚡ (Appliquer les modifications) dans la barre d'outils.
- Lorsqu'il s'ouvre, saisissez un message dans le champ de texte et appuyez sur Envoyer pour voir le message apparaître dans la deuxième activité.





Créer votre première application

Exercice :

En se basant sur l'activité ci-dessous, réaliser une application Android permettant d'avoir une calculatrice simple.





Prise en charge de différents appareils

Motivations

- Les appareils Android sont de plusieurs formes et de tailles à travers le monde. Avec une large gamme de types d'appareils, vous avez la possibilité de d'accéder à un grand public avec votre application.
- Afin d'avoir le plus de succès possible sur Android, votre application doit s'adapter à différentes configurations de périphériques. Certaines des variantes importantes que vous devriez considérer incluent différentes langues, les tailles d'écran et les versions de la plate-forme Android.
- Dans cette partie, vous apprendrez à utiliser des fonctionnalités de plate-forme de base qui utilisent des ressources alternatives et d'autres fonctionnalités afin que votre application puisse fournir une expérience utilisateur optimisée sur une variété d'appareils compatibles avec Android, en utilisant un seul package d'application (APK).



Prise en charge de différents appareils

Prise en charge de différentes langues et cultures

- Les applications incluent des ressources spécifiques à une culture particulière.
- C'est une bonne pratique de séparer les ressources propres à la culture du reste de votre application. Android résout les ressources spécifiques à la langue et à la culture en fonction des paramètres régionaux du système en utilisant le répertoire des ressources (**res**).
- Vous pouvez spécifier des ressources adaptées à la culture des personnes utilisant votre application.
- **Exemple** : la capture d'écran suivante montre une chaîne et des ressources drawable (étirables) dans les paramètres régionaux par défaut de l'appareil (en_US) et les paramètres régionaux espagnols (es_ES).





Prise en charge de différents appareils

Prise en charge de différentes langues et cultures

Créer des répertoires locaux et des fichiers de ressources

- Pour prendre en charge des paramètres régionaux d'une langue, créez des répertoires supplémentaires à l'intérieur de `res/`. Le nom de chaque répertoire devrait respecter le format suivant: `<Type de ressource> -b + <code de langue> [+ <code de pays>]`
- **Exemple :** `values-b+es/` contient des ressources chaîne pour l'espagnole. De même, `mipmap-b+es+ES/` contient des icônes pour la langue espagnole de l'Espagne. Android charge les ressources appropriées en fonction des paramètres régionaux de l'appareil au moment de l'exécution.
- Notons que les fichiers suivants sont différents pour différentes langues:

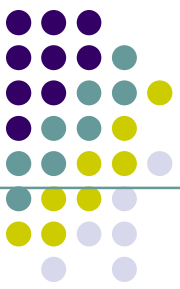
- Chaînes `/values/strings.xml` (Par défaut)

```
<resources>
  <string name="hello_world">Hello World!</string>
</resources>
```

- Chaînes `/values-es/strings.xml` (Espagnol)

```
<resources>
  <string name="hello_world">¡Hola Mundo!</string>
</resources>
```

```
MyProject/
  res/
    values/
      strings.xml
    values-b+es/
      strings.xml
    mipmap/
      country_flag.png
    mipmap-b+es+ES/
      country_flag.png
```



Prise en charge de différents appareils

Prise en charge de différentes langues et cultures

Utilisez les ressources dans votre application

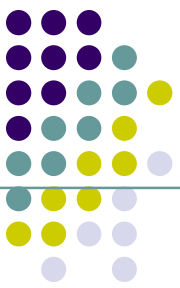
- Vous pouvez faire référence aux ressources depuis le code source et depuis les fichiers XML à l'aide de l'attribut **name** de la ressource.
- A partir du code source:
 - Syntaxe : **R.<type de la ressource >.<nom de la ressource >**

```
// Obtenez une Resources chaîne à partir des Resources de votre Resources
String hello = getResources().getString(R.string.hello_world);

// Ou fournir une ressource de chaîne à une méthode qui requiert une chaîne
TextView textView = new TextView(this);
TextView.setText(R.string.hello_world);
```

- A partir d'un fichier XML:
 - Syntaxe : **@<type de la ressource >/<nom de la ressource>**

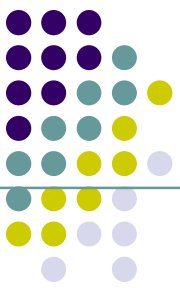
```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/country_flag" />
```



Prise en charge de différents appareils

Prise en charge de différents écrans

- Android classe les écrans des appareils à l'aide de deux propriétés générales: la taille (size) et la densité (density).
- Vous devriez vous attendre à ce que votre application soit installée sur des périphériques avec des écrans avec diverses tailles et densités.
- C'est pourquoi, vous devez inclure des ressources alternatives qui optimisent l'apparence de votre application suivant ces tailles et ces densités d'écran.
- Il existe quatre tailles d'écran: **small, normal, large, xlarge**
- Et quatre densités d'écran: **low (ldpi), medium (mdpi), high (hdpi), extra high (xhdpi)**
- Pour déclarer différentes mises en page et bitmaps à utiliser sur différents écrans, vous devez placer ces ressources alternatives dans des répertoires distincts, comme pour les chaînes de langue.
- Soyez également conscient que l'orientation des écrans (paysage ou portrait) est considérée comme une variation de la taille de l'écran, tant d'applications doivent réviser la mise en page afin d'optimiser l'expérience de l'utilisateur dans chaque orientation



Prise en charge de différents appareils

Prise en charge de différents écrans

Créer différentes mises en page

- Pour optimiser votre interface utilisateur sur différentes tailles d'écran, vous devez créer un fichier XML de mise en page unique pour chaque taille d'écran souhaitée.
- Chaque mise en page doit être sauvegardée dans le répertoire de ressources approprié, nommé avec un suffixe `-<screen_size>`.
- Par exemple, une mise en page unique pour les grands écrans devrait être sauvegardée sous `res/layout-large/`.

Exemple 1: (Une mise en page par défaut et une mise en page alternative pour les *grands* écrans)

```
MyProject/  
  res/  
    layout/  
      main.xml  
    layout-large/  
      main.xml
```

Remarque : Les noms de fichiers doivent être exactement les mêmes, mais leur contenu est différent pour fournir une interface utilisateur optimisée pour la taille d'écran correspondante.



Prise en charge de différents appareils

Prise en charge de différents écrans

Créer différentes mises en page

- Pour référencer le fichier de mise en page dans votre application on met :

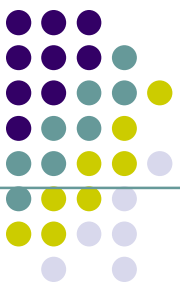
```
@Passer outre
Protected void onCreate (Bundle savedInstanceState) {
    Super.onCreate (savedInstanceState);
    SetContentView (R.layout.main);
}
```

Exemple 2: (Une mise en page alternative pour l'orientation paysage)

```
MyProject/
  res/
    layout/
      main.xml
    layout-land/
      main.xml
```

```
MyProject/
  res/
    layout/           # default (portrait)
      main.xml
    layout-land/      # landscape
      main.xml
    layout-large/     # large (portrait)
      main.xml
    layout-large-land/ # large landscape
      main.xml
```

Remarque : Par défaut, le fichier layout/main.xml est utilisé pour l'orientation portrait.



Prise en charge de différents appareils

Prise en charge de différents écrans

Créer différents bitmaps

- Vous devez toujours fournir des ressources bitmap qui sont correctement mises à l'échelle pour chacun des densités : faible, moyenne, haute et très haute densité. Cela vous permet d'obtenir une bonne qualité graphique et des performances sur toutes les densités d'écran.
- Pour générer ces images, vous devez commencer avec votre ressource brute en format vectoriel et générer les images pour chaque densité à l'aide de l'échelle de taille suivante:
 - Xhdpi: 2.0
 - Hdpi: 1.5
 - Mdpi: 1,0 (baseline)
 - Ldpi: 0.75
- Cela signifie que si vous générez une image 200x200 pour les périphériques xhdpi, vous devez générer la même ressource à 150x150 pour hdpi, 100x100 pour mdpi et 75x75 pour les périphériques ldpi.



Prise en charge de différents appareils

Prise en charge de différents écrans

Créer différents bitmaps

- Ensuite, placez les fichiers dans le répertoire de ressources drawable approprié:

```
MyProject/  
  res/  
    drawable-xhdpi/  
      awesomeimage.png  
    drawable-hdpi/  
      awesomeimage.png  
    drawable-mdpi/  
      awesomeimage.png  
    drawable-ldpi/  
      awesomeimage.png
```

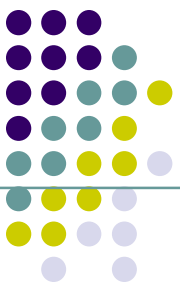
- Ainsi chaque fois que vous faites référence à `@drawable/awesomeimage`, le système sélectionne le bitmap approprié en fonction de la densité de l'écran.



Prise en charge de différents appareils

Prise en charge des versions de la plateforme

- Même si les dernières versions d'Android fournissent souvent de superbes API pour votre application, vous devriez continuer à prendre en charge les anciennes versions d'Android jusqu'à ce que plus de périphériques soient mis à jour.
- Cette partie vous montre comment tirer parti des dernières API tout en continuant à prendre en charge les anciennes versions.
- Le tableau de bord pour les versions de la plate-forme est mis à jour régulièrement pour afficher la distribution des périphériques actifs exécutant chaque version d'Android, en fonction du nombre d'appareils visitant Google Play Store. Généralement, c'est une bonne pratique de supporter environ 90% des dispositifs actifs, tout en axant votre application sur la dernière version.
- **Remarque :** Afin de fournir les meilleures fonctionnalités dans plusieurs versions Android, vous devez utiliser la bibliothèque de prise en charge des versions de Android dans votre application, ce qui vous permet d'utiliser plusieurs API de plate-forme récentes sur les anciennes versions



Prise en charge de différents appareils

Prise en charge des versions de la plateforme

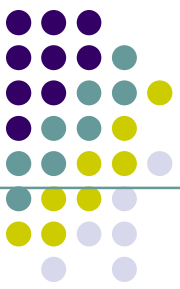
Spécifier les niveaux minimum et cible de l'API

- Le fichier **AndroidManifest.xml** décrit les détails de votre application et identifie les versions d'Android qu'il prend en charge.
- Plus précisément, les attributs **minSdkVersion** et **targetSdkVersion** de l'élément **<uses-sdk>** qui désignent respectivement le version minimale de l'API sur lequel l'application peut tourner et la version de l'API à partir de laquelle on pourra exploiter à fond l'application.

Exemple :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >  
    <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="15" />  
    ...  
</manifest>
```

- À mesure que de nouvelles versions d'Android sont diffusées, certains styles et comportements peuvent changer.
- Pour permettre à votre application de profiter de ces modifications et de s'assurer qu'elle correspond au style des plateformes des utilisateurs, vous devez définir la valeur de **targetSdkVersion** à la dernière version Android disponible.



Prise en charge de différents appareils

Prise en charge des versions de la plateforme

Vérifiez la version du système au moment de l'exécution

- Android fournit un code unique pour chaque version de plate-forme dans la classe des constantes **Build**. Utilisez ces codes dans votre application pour créer des conditions qui garantissent que le code, qui dépend des niveaux supérieurs de l'API, soit exécuté uniquement lorsque ces API sont disponibles sur le système.

```
private void setUpActionBar() {  
    // S'assurer qu'il fonctionne sur Honeycomb ou supérieur pour utiliser les API ActionBar  
  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {  
        ActionBar actionBar = getActionBar();  
        actionBar.setDisplayHomeAsUpEnabled(true);  
    }  
}
```

Remarque:

- Lors de l'analyse des ressources XML, Android ignore les attributs XML qui ne sont pas pris en charge par le périphérique actuel.
- Ainsi en toute sécurité, on peut utiliser des attributs XML pris en charge que par des versions plus récentes, sans se soucier si les anciennes versions bugent si elles rencontrent ce code.



Prise en charge de différents appareils

Prise en charge des versions de la plateforme

Vérifiez la version du système au moment de l'exécution

Remarque (suite):

- Exemple, si `targetSdkVersion="11"`, votre application inclut `ActionBar` par défaut sur Android 3.0 ou plus.
- Ensuite, pour ajouter des éléments de menu à la barre d'action, vous devez ajouter `android:showAsAction="ifRoom"` dans votre ressource XML de menu.
- Il est sécurisé de le faire dans un fichier XML de version croisée, car les anciennes versions d'Android ignorent simplement l'attribut `showAsAction` (c'est-à-dire que vous n'avez pas besoin d'une version distincte dans `res/menu-v11/`).



Prise en charge de différents appareils

Prise en charge des versions de la plateforme

Utiliser les styles et les thèmes de la plate-forme

- Android propose des thèmes d'interface utilisateur qui donnent aux applications l'apparence du système d'exploitation sous-jacent. Ces thèmes peuvent être appliqués à votre application dans le fichier manifeste. En utilisant ces styles et thèmes intégrés, votre application suivra naturellement la dernière apparence d'Android avec chaque nouvelle version.

- Pour que votre activité ressemble à une boîte de dialogue:

```
<activity android:theme="@android:style/Theme.Dialog">
```

- Pour que votre activité ait un fond transparent:

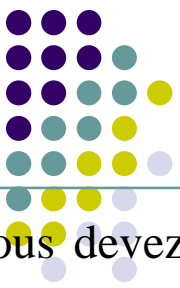
```
<activity android:theme="@android:style/Theme.Translucent">
```

- Pour appliquer à une activité votre propre thème défini dans [/res/values/styles.xml](#) :

```
<activity android:theme="@style/CustomTheme">
```

- Pour appliquer un thème à votre application entière (à toutes les activités), ajoutez l'attribut [android:theme](#) à l'élément [<application>](#) :

```
<application android:theme="@style/CustomTheme">
```

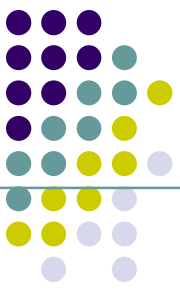


Création d'une IU dynamique fragmenté

- Pour créer une interface utilisateur (**IU**) dynamique et fragmenté sur Android, vous devez encapsuler les composants et les comportements d'activité dans des modules que vous pouvez échanger dans vos activités.
- Ces modules peuvent être créés avec la classe **Fragment** qui se comporte comme une activité imbriquée qui peut définir sa propre mise en page et gérer son propre cycle de vie.
- Lorsqu'un fragment spécifie sa propre mise en page, il peut être configuré en combinaisons différentes avec d'autres fragments dans une activité pour modifier votre configuration de mise en page pour différentes tailles d'écran (un petit écran peut afficher un fragment à la fois, mais un grand écran peut en afficher deux ou plus).
- Cette classe vous montre comment créer une interface utilisateur dynamique avec des fragments et optimiser l'expérience utilisateur de votre application pour les périphériques de différentes tailles d'écran tout en continuant à prendre en charge les périphériques qui exécutent des versions aussi anciennes que Android 1.6.

Nous allons voir dans la suite comment:

- **Créer un fragment**
- **Créer une interface utilisateur flexible**
- **Communiquer avec d'autres fragments**



Création d'une IU dynamique fragmenté

Création d'un fragment

- Vous pouvez penser à un fragment en tant que section modulaire d'une activité, qui a son propre cycle de vie, reçoit ses propres événements d'entrée et que vous pouvez ajouter ou supprimer pendant que l'activité est en cours d'exécution (une sorte de «sous-activité» que vous pouvez réutiliser dans différentes activités).
- Cette partie montre comment étendre la classe [Fragment](#) à l'aide de la bibliothèque de prise en charge ([Support Library](#)) afin que votre application reste compatible avec les périphériques exécutant des versions système aussi bas que Android 1.6.
- Avant de commencer cette partie, vous devez configurer votre projet Android pour utiliser la bibliothèque [Support Library](#) :
(voir <https://developer.android.com/topic/libraries/support-library/setup.html>)



Création d'une IU dynamique fragmenté

Créer un fragment

Créer une classe Fragment

- Pour créer un fragment, étendez la classe `Fragment`, puis neutralisez les méthodes de cycle de vie de la clé pour insérer la logique de votre application, comme dans une classe `Activity`.
- Une différence lors de la création d'un `Fragment` est que vous devez utiliser la fonction de rappel `onCreateView()` pour définir la mise en page. En fait, c'est le seul rappel dont vous avez besoin pour obtenir un fragment en cours d'exécution. Par exemple, voici un fragment simple qui spécifie sa propre mise en page:

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```



Création d'une IU dynamique fragmenté

Créer un fragment

Créer une classe Fragment

- Tout comme une activité, un fragment devrait implémenter d'autres rappels de cycle de vie qui permettent de gérer son état lorsqu'il est ajouté ou supprimé de l'activité et lorsque l'activité passe entre ses états de cycle de vie.
- Par exemple, lorsque la méthode `onPause()` de l'activité est appelée, tous les fragments de l'activité reçoivent un appel de `onPause()`



Création d'une IU dynamique fragmenté

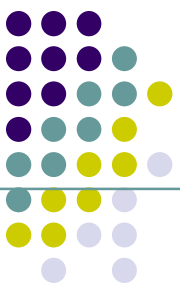
Créer un fragment

Ajouter un fragment à une activité en utilisant XML

- Alors que les fragments sont réutilisables dans les composants UI modulaires, chaque instance d'une classe `Fragment` doit être associée à une activité parent `FragmentActivity`.
- Vous pouvez réaliser cette association en définissant chaque fragment dans votre fichier XML de mise en page d'activité.

Remarque:

- **`FragmentActivity`** est une activité spéciale fournie dans la bibliothèque `Support Library` pour traiter des fragments sur des versions de système antérieures à l'API 11.
- Si la version système la plus basse que vous supportez est le niveau API 11 ou supérieur, vous pouvez utiliser une Activité régulière.



Création d'une IU dynamique fragmenté

Créer un fragment

Ajouter un fragment à une activité en utilisant XML

- Voici un exemple de fichier de mise en page qui ajoute deux fragments à une activité lorsque l'écran du périphérique est "large".

res/layout-large/news_articles.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
        android:id="@+id/headlines_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
        android:id="@+id/article_fragment"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
```



Création d'une IU dynamique fragmenté

Créer un fragment

Ajouter un fragment à une activité en utilisant XML

- Ensuite, appliquez la mise en page à votre activité:

- Si vous utilisez la bibliothèque **v7 appcompat**, votre activité devrait étendre **AppCompatActivity**, qui est une sous-classe de **FragmentActivity**.

```
Importer android.os.Bundle;
Importer android.support.v4.app.FragmentActivity;

Classe publique MainActivity extends FragmentActivity {
    @Passer outre
    Public void onCreate (Bundle savedInstanceState) {
        Super.onCreate (savedInstanceState);
        SetContentView (R.layout.news_articles);
    }
}
```

Remarque:

- Lorsque vous ajoutez un fragment à une mise en page d'activité en définissant le fragment dans le fichier XML de la mise en page, vous *ne pouvez pas* supprimer le fragment au moment de l'exécution.
- Si vous prévoyez d'échanger vos fragments pendant l'interaction avec l'utilisateur, vous devez ajouter le fragment à l'activité lorsque l'activité commence, comme indiqué dans la suite.



Création d'une IU dynamique fragmenté

Création d'IU flexible

- Lors de la conception de votre application, pour prendre en charge une large gamme de tailles d'écran, vous pouvez réutiliser vos fragments dans différentes configurations de mise en page afin d'optimiser l'interface utilisateur en fonction de l'espace d'écran disponible.
- Par exemple, sur un appareil combiné, il peut être approprié d'afficher un seul fragment à la fois pour une interface utilisateur à un seul élément. À l'inverse, vous pouvez définir des fragments côte à côte sur une tablette qui a une taille d'écran plus large pour afficher plus d'informations à l'utilisateur.



Légende : Deux fragments, affichés dans différentes configurations pour la même activité sur différentes tailles d'écran. Sur un grand écran, les deux fragments s'ajoutent côte à côte, mais sur un appareil combiné, un seul fragment correspond à la fois, de sorte que les fragments doivent se remplacer à fur et à mesure que l'utilisateur navigue.



Création d'une IU dynamique fragmenté

Création d'IU flexible

Ajouter un fragment à une activité lors de l'exécution

- Plutôt que de définir les fragments d'une activité dans le fichier de mise en page - comme indiqué dans la partie précédente avec l'élément `<fragment>`, vous pouvez ajouter un fragment à l'activité pendant son exécution. Ceci est nécessaire si vous prévoyez de modifier des fragments pendant la durée de vie de l'activité.
- Pour effectuer une transaction telle qu'ajouter ou supprimer un fragment, vous devez utiliser `FragmentManager` pour créer un `FragmentTransaction`, qui fournit des API pour ajouter, supprimer, remplacer et effectuer d'autres transactions de fragments.
- Si votre activité permet aux fragments d'être supprimés et de remplacés, vous devez ajouter le (s) fragment (s) initial (s) à l'activité pendant la méthode `onCreate()` de l'activité.
- Une règle importante lorsqu'on traite des fragments, en particulier lors de l'ajout de fragments au moment de l'exécution, est que votre mise en page d'activité doit inclure un conteneur `View` dans lequel vous pouvez insérer le fragment.



Création d'une IU dynamique fragmenté

Création d'IU flexible

Ajouter un fragment à une activité lors de l'exécution

- La mise en page suivante est une alternative à la mise en page présentée dans la partie précédente qui ne montre qu'un seul fragment à la fois. Pour remplacer un fragment par un autre, la mise en page de l'activité comprend un `FrameLayout` vide qui agit comme un conteneur de fragments.
- Notez que le nom du fichier est identique au fichier de mise en page de la partie précédente, mais le répertoire de mise en page n'a pas le qualificateur `large`, donc cette disposition est utilisée lorsque l'écran du périphérique est plus petit que `large` car l'écran ne peut contenir deux fragments à la fois.

`res/layout/news_articles.xml:`

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```




Création d'une IU dynamique fragmenté

Création d'IU flexible

Ajouter un fragment à une activité lors de l'exécution

- Dans votre activité, appelez `getSupportFragmentManager()` pour obtenir un `FragmentManager` à l'aide des API Support Library. Ensuite, appelez `beginTransaction()` pour créer un `FragmentTransaction` et appeler `add()` pour ajouter un fragment.
- Vous pouvez effectuer plusieurs transactions de fragment pour l'activité en utilisant le même `FragmentTransaction`. Lorsque vous êtes prêt à effectuer les modifications, vous devez appeler `commit()`.
- L'exemple suivant montre comment ajouter un fragment à la mise en page précédente:



Création d'une IU dynamique fragmenté

Création d'IU flexible Ajouter un fragment à une activité lors de l'exécution

Exemple :

Parce que le fragment a été ajouté au container `FrameLayout` au moment de l'exécution - au lieu de le définir dans la mise en page de l'activité avec un élément `<fragment>`.

L'activité peut supprimer le fragment et le remplacer par un autre.

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {

            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }

            // Create a new Fragment to be placed in the activity layout
            HeadlinesFragment firstFragment = new HeadlinesFragment();

            // In case this activity was started with special instructions from an
            // Intent, pass the Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            // Add the fragment to the 'fragment_container' FrameLayout
            getSupportFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
    }
}
```



Création d'une IU dynamique fragmenté

Création d'IU flexible

Remplacer un fragment avec un autre

- La procédure pour remplacer un fragment est semblable à celle pour ajouter un, mais elle nécessite la méthode `replace()` au lieu de `add()`.
- Gardez à l'esprit que lorsque vous effectuez des transactions de fragments, telles que remplacer ou supprimer un fragment, il est souvent approprié de permettre à l'utilisateur de naviguer vers l'arrière et d'annuler la modification.
- Pour permettre à l'utilisateur de naviguer vers l'arrière à travers les transactions de fragments, vous devez appeler `addToBackStack()` avant d'exécuter `FragmentTransaction`

Remarque:

- Lorsque vous supprimez ou remplacez un fragment et ajoutez la transaction à la pile arrière, le fragment supprimé est arrêté (non détruit). Si l'utilisateur retourne pour restaurer le fragment, il redémarre.
- Si vous n'ajoutez pas la transaction à la pile arrière, le fragment est détruit lorsqu'il est supprimé ou remplacé.



Création d'une IU dynamique fragmenté

Création d'IU flexible

Remplacer un fragment avec un autre

Exemple : (Remplacement d'un fragment par un autre)

```
// Create fragment and give it an argument specifying the article it should show
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentManager transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

- La méthode `addToBackStack()` prend un paramètre de chaîne optionnel qui spécifie un nom unique pour la transaction.
- Le nom n'est pas nécessaire, sauf si vous prévoyez d'effectuer des opérations de fragment avancées à l'aide des API `FragmentManager.BackStackEntry`.



Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

- Afin de réutiliser les composants Fragment UI, vous devez construire chacun comme un composant modulaire entièrement autonome qui définit sa propre mise en page et son comportement. Une fois que vous avez défini ces fragments réutilisables, vous pouvez les associer à une activité et les connecter avec la logique de l'application pour réaliser l'interface utilisateur intégrée globale.
- Souvent, vous voudrez qu'un Fragment communique avec un autre, par exemple pour modifier le contenu en fonction d'un événement utilisateur. Toute communication Fragment à Fragment se fait via l'activité associée. Deux fragments ne devraient jamais communiquer directement.



Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

Définir une interface

Pour permettre à un Fragment de communiquer avec son Activité, vous pouvez définir une interface dans la classe Fragment et l'implémenter dans l'Activité. Le Fragment capture l'implémentation de l'interface pendant sa méthode de cycle de vie `onAttach()` et peut ensuite appeler les méthodes d'interface pour communiquer avec l'Activité.

Exemple : Voici un exemple de communication de Fragment à Activity :

```
public class HeadlinesFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;

    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception
        try {
            mCallback = (OnHeadlineSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                + " must implement OnHeadlineSelectedListener");
        }
    }

    ...
}
```



Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

Définir une interface

- Maintenant le fragment peut envoyer des messages à l'activité en appelant la méthode `onArticleSelected()` (ou d'autres méthodes dans l'interface) à l'aide de l'instance `mCallback` de l'interface `OnHeadlineSelectedListener`.
- Par exemple, la méthode suivante dans le fragment est appelée lorsque l'utilisateur clique sur un élément de liste. Le fragment utilise l'interface de rappel pour délivrer l'événement à l'activité parentale.

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    // Send the event to the host activity
    mCallback.onArticleSelected(position);
}
```



Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

Implémenter l'interface

- Afin de recevoir des rappels d'événements à partir du fragment, l'activité qui l'héberge doit implémenter l'interface définie dans la classe de fragments.

Exemple : L'activité suivante implémente l'interface à partir de l'exemple ci-dessus.

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
    }
}
```




Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

Fournir un message à un fragment

- L'activité hôte peut envoyer des messages à un fragment en capturant l'instance `Fragment` avec `findFragmentById()`, puis appeler directement les méthodes publiques du `Fragment`.
- Par exemple, imaginez que l'activité ci-dessus peut contenir un autre fragment utilisé pour afficher l'élément spécifié par les données renvoyées dans la méthode de rappel ci-dessus. Dans ce cas, l'activité peut transmettre l'information reçue dans la méthode de rappel à l'autre fragment qui affichera l'élément:

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article

        ArticleFragment articleFrag = (ArticleFragment)
            getSupportFragmentManager().findFragmentById(R.id.article_fragment);

        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...

            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        } else {
```



Création d'une IU dynamique fragmenté

Communiquer avec d'autres fragments

Fournir un message à un fragment

- Exemple (suite):

```
// Otherwise, we're in the one-pane layout and must swap frags...

// Create fragment and give it an argument for the selected article
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentManager transaction = getSupportFragmentManager().beginTransaction();

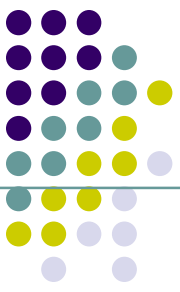
// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
    }
}
```



La sauvegarde des données

- La plupart des applications Android doivent sauvegarder des données, même si elles ne servent qu'à enregistrer des informations sur son état pendant la pause afin que la progression ne soit pas perdue.
- La plupart des applications non triviales doivent également sauvegarder les paramètres de l'utilisateur et certaines applications doivent gérer une grande quantité d'informations dans les fichiers et les bases de données. Cette partie vous présente les principales options de stockage de données dans Android, y compris:
 - L'enregistrement de paires de clés/valeurs de types de données simples dans un fichier de préférences partagées
 - L'enregistrement de fichiers arbitraires dans le système de fichiers Android
 - L'utilisation de bases de données gérées par SQLite



La sauvegarde des données

Sauvegarde de paires clé/valeur

- Si vous disposez d'une collection relativement petite de pair clé/valeurs que vous souhaitez sauvegarder, vous devez utiliser les API [SharedPreferences](#).
- Un objet [SharedPreferences](#) pointe sur un fichier contenant des paires de clé/valeur et fournit des méthodes simples pour les lire et les écrire.
- Chaque fichier [SharedPreferences](#) est géré par un framework et peut être privé ou partagé.
- Nous allons dans la suite montrer comment utiliser les API [SharedPreferences](#) pour stocker et récupérer des valeurs simples.
- **Remarque:** Les API [SharedPreferences](#) ne servent qu'à lire et à écrire des paires de clés/valeurs. A ne pas confondre avec les API [Preference](#), qui vous aident à créer une interface utilisateur avec les paramètres de votre application (même si elles utilisent [SharedPreferences](#) pour enregistrer les paramètres de l'application).



La sauvegarde des données

Sauvegarde de paires clé/valeur

Traitement des SharedPreferences

Vous pouvez créer un nouveau fichier de préférence partagé ou en accéder à un en appelant une des deux méthodes suivantes:

- `getSharedPreferences()` - à utiliser si vous avez besoin de plusieurs fichiers de préférences partagés identifiés par leur nom, que vous spécifiez avec le premier paramètre. Vous pouvez l'appeler ainsi à partir du `Context` dans votre application.
- `getPreferences()` - à utiliser à partir d'une Activity si vous devez utiliser un seul fichier de préférence partagé pour l'activité. Parce qu'il récupère un fichier de préférence partagé par défaut qui appartient à l'activité, vous n'avez pas besoin de fournir un nom.

Exemple : Le code suivant accède au fichier de préférences partagées qui est identifié par la chaîne `R.string.preference_file_key` et l'ouvre en mode privé afin que le fichier soit accessible uniquement par votre application.

```
//L'identifiant idPrefFile du SharedPreferences est une chaine de caracteres
SharedPreferences sharedPref = getSharedPreferences(idPrefFile,MODE_PRIVATE);
```



La sauvegarde des données

Sauvegarde de paires clé/valeur

Traitement des SharedPreferences

- Lors de la nomination de vos fichiers de préférence partagés, vous devez utiliser un nom qui est identifiable de manière unique à votre application, tel que `"com.example.myapp.PREFERENCE_FILE_KEY"`
- Sinon, si vous avez besoin d'un seul fichier de préférence partagé pour votre activité, vous pouvez utiliser la méthode `getPreferences()` :

```
SharedPreferences pref = getPreferences (MODE_PRIVATE) ;
```

- **Attention** : si vous créez un préférence partagé avec `MODE_WORLD_READABLE` ou `MODE_WORLD_WRITEABLE`, toutes les autres applications qui connaissent l'identifiant du fichier peuvent accéder à vos données



La sauvegarde des données

Sauvegarde de paires clé/valeur

Ecrire dans un fichier de préférences partagées

- Pour écrire dans un fichier de préférences partagées, créez un `SharedPreferences.Editor` en appelant `edit()` sur votre `SharedPreferences`.
- Passez les clés et les valeurs que vous voulez écrire en utilisant les méthodes `putInt()` et `putString()`. Ensuite, appelez `commit()` pour enregistrer les modifications.

Exemple :

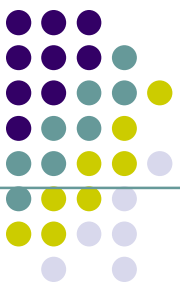
```
SharedPreferences sharedPref = getSharedPreferences(idPrefFile, MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putString("Expression", val);
editor.commit();
```

Lire des préférences partagées

Pour récupérer des valeurs à partir d'un fichier de préférences partagées, on appelle des méthodes telles que `getInt()` et `getString()`, fournissant la clé de la valeur souhaitée et éventuellement une valeur par défaut à renvoyer si la clé n'est pas présente.

Exemple :

```
SharedPreferences sharedPref = getSharedPreferences(idPrefFile, MODE_PRIVATE);
String val = sharedPref.getString("Expression", "");
```



La sauvegarde des données

Enregistrement de fichiers

- Android utilise un système de fichiers similaire aux systèmes de fichiers sur disque sur d'autres plates-formes. Cette partie décrit comment travailler avec le système de fichiers Android pour lire et écrire des fichiers avec les API File .
- Un objet File est adapté à la lecture ou à l'écriture de grandes quantités de données. Par exemple, il est bon pour les fichiers image ou tout ce qui a été échangé sur un réseau.
- Nous allons voir dans la suite montre comment effectuer des tâches basiques liées aux fichiers dans votre application. Cela suppose que vous connaissez les bases du système de fichiers Linux et les API d'entrée/sortie de fichier standard dans java.io



La sauvegarde des données

Enregistrement de fichiers

Choix du stockage (interne ou externe)

- Tous les appareils Android disposent de deux zones de stockage de fichiers: stockage "interne" et "externe".
- Ces noms datent au début d'Android, lorsque la plupart des appareils offraient une mémoire non volatile intégrée (stockage interne), plus un support de stockage amovible tel qu'une carte micro SD (stockage externe).
- Certains périphériques divisent l'espace de stockage permanent en partitions "internes" et "externes", donc même sans support de stockage amovible, il existe toujours deux espaces de stockage et le comportement de l'API est identique, même si le stockage externe est amovible ou non.
- Le tableau suivant résume les caractéristiques relatifs à chaque espace de stockage.



La sauvegarde des données

Enregistrement de fichiers

Choix du stockage (interne ou externe)

Stockage interne:	Stockage externe:
<ul style="list-style-type: none">• Il est toujours disponible.• Les fichiers enregistrés ici sont accessibles uniquement par votre application.• Lorsque l'utilisateur désinstalle votre application, le système supprime tous les fichiers de votre application depuis le stockage interne.	<ul style="list-style-type: none">• Il n'est pas toujours disponible, car l'utilisateur peut monter le stockage externe en tant que périphérique USB et, dans certains cas, le supprimer de l'appareil.• Il est lisible partout, donc les fichiers enregistrés ici peuvent être lus en dehors de votre appareil.• Lorsque l'utilisateur désinstalle l'application, le système supprime les fichiers de votre application ici uniquement si vous les enregistrez dans le répertoire depuis <code>getExternalFilesDir()</code>.
Le stockage interne est meilleur quand vous voulez être sûr que ni l'utilisateur ni d'autres applications ne peuvent accéder à vos fichiers.	Le stockage externe est le meilleur endroit pour les fichiers qui ne nécessitent pas de restrictions d'accès et pour les fichiers que vous souhaitez partager avec d'autres applications ou permettent à l'utilisateur d'accéder avec un ordinateur



La sauvegarde des données

Enregistrement de fichiers

Choix du stockage (interne ou externe):

Remarque :

- Bien que les applications soient installées par défaut sur le stockage interne, vous pouvez spécifier l'attribut `android:installLocation` dans votre manifeste afin que votre application puisse être installée sur un stockage externe.
- Les utilisateurs apprécient cette option lorsque la taille de l'APK est très grande et ont un espace de stockage externe supérieur à la mémoire interne.
- Pour plus d'informations, consultez la page relative à l'emplacement de l'installation <https://developer.android.com/guide/topics/data/install-location.html>



La sauvegarde des données

Enregistrement de fichiers

Autorisations pour le stockage externe

- Pour écrire sur le stockage externe, vous devez demander la permission avec **WRITE_EXTERNAL_STORAGE** dans votre fichier manifeste:

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

- **Attention :** Actuellement, toutes les applications ont la possibilité de lire le stockage externe sans autorisation spéciale. Cependant, cela changera dans une version ultérieure. Si votre application doit lire la mémoire externe (mais ne pas y écrire), vous devrez déclarer **READ_EXTERNAL_STORAGE**. Pour vous assurer que votre application continue de fonctionner comme prévu, vous devriez déclarer cette autorisation maintenant, avant que le changement ne prenne effet.

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



La sauvegarde des données

Enregistrement de fichiers

Autorisations pour le stockage externe

- si votre application utilise `WRITE_EXTERNAL_STORAGE`, elle a implicitement l'autorisation de lire le stockage externe.
- Vous n'avez besoin d'aucune autorisation pour enregistrer des fichiers sur le stockage interne. Votre application a toujours l'autorisation de lire et d'écrire des fichiers dans son répertoire de stockage interne



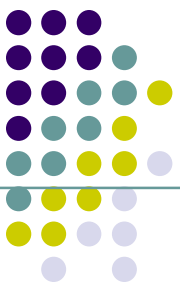
La sauvegarde des données

Enregistrement de fichiers

Enregistrer un fichier sur le stockage interne

- Lors de la sauvegarde d'un fichier dans un stockage interne, vous pouvez accéder au répertoire approprié en tant que File en appelant l'une des deux méthodes suivantes:
 - `getFilesDir()` qui renvoie un File représentant un répertoire interne pour votre application.
 - `getCacheDir()` Renvoie un File représentant un répertoire interne pour les fichiers de cache temporaire de votre application..
- Pour créer un nouveau fichier dans l'un de ces répertoires, vous pouvez utiliser le constructeur `File()`, en passant le File fourni par l'une des méthodes ci-dessus qui spécifie votre répertoire de stockage interne.

```
File file = new File(context.getFilesDir(), filename);
```
- Alternativement, vous pouvez appeler **`openFileOutput()`** pour obtenir un **`FileOutputStream`** qui écrit dans un fichier dans votre répertoire interne. Par exemple, voici comment écrire du texte dans un fichier.



La sauvegarde des données

Enregistrement de fichiers

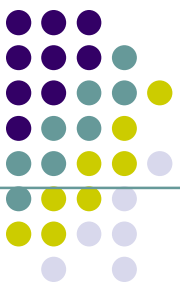
Enregistrer un fichier sur le stockage interne

```
try {  
    File file = new File(fileName);  
    FileOutputStream outputStream = openFileOutput(fileName, Context.MODE_APPEND);  
    outputStream.write(texte.getBytes());  
    outputStream.close();  
  
    textView.setText(texte);  
} catch (Exception e) {  
    textView.setText(e.toString());  
}
```

Supprimer un fichier

- Vous devez toujours supprimer les fichiers dont vous n'avez plus besoin. Le moyen le plus simple de supprimer un fichier consiste à supprimer la référence du fichier ouvert en appelant `delete()`.

```
myFile.delete();
```



La sauvegarde des données

Enregistrement de fichiers

Affichage d'un fichier sur le stockage interne

```
try{
    File file = new File(fileName);
    FileInputStream inputStream = openFileInput(fileName);

    InputStreamReader lecture=new InputStreamReader(inputStream);
    BufferedReader buff=new BufferedReader(lecture);
    String ch = "",ligne;
    while ((ligne=buff.readLine()) !=null) {
        ch += ligne+"\n";
    }

    buff.close();
}
catch (Exception e){
    textView.setText(e.toString());
}
```




La sauvegarde des données

Enregistrement dans une BD SQL

- L'enregistrement de données dans une base de données est idéal pour les données répétées ou structurées, telles que les informations de contact.
- Cette partie suppose que vous connaissez les bases de données SQL en général et vous aide à démarrer avec les bases de données SQLite sur Android.
- Les API nécessaires pour utiliser une base de données sur Android sont disponibles dans le package **android.database.sqlite** .



La sauvegarde des données

Enregistrement dans une BD SQL

Définir un schéma et un contract

- L'un des principaux fondamentaux des bases de données SQL est le schéma: une déclaration formelle de la manière dont la base de données est organisée.
- Le schéma se reflète dans les instructions SQL que vous utilisez pour créer votre base de données. Vous pourriez trouver utile de créer une classe complémentaire, connue sous le nom de classe de **contract**, qui spécifie explicitement la mise en page de votre schéma de façon systématique et auto-documentaire.
- Une classe contract est un conteneur pour les constantes qui définissent les noms des URI, des tables et des colonnes. Elle vous permet d'utiliser les mêmes constantes dans toutes les autres classes du même package. Cela vous permet de modifier un nom de colonne en un seul endroit et de le propager dans votre code.
- Une bonne façon d'organiser une classe contract consiste à mettre des définitions globales dans votre base de données dans le niveau racine de la classe. Ensuite, créez une classe interne pour chaque table qui énumère ses colonnes.



La sauvegarde des données

Enregistrement dans une BD SQL

Définir un schéma et un contract

- **Remarque :** En implémentant l'interface `BaseColumns`, votre classe interne peut hériter un champ de clé primaire appelé `_ID`. Ce n'est pas nécessaire, mais cela peut aider votre base de données à travailler harmonieusement avec le framework Android.

Exemple:

```
public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // make the constructor private.
    private FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
    }
}
```



La sauvegarde des données

Enregistrement dans une BD SQL

Créer la base de données avec SQL Helper

- Une fois que vous avez défini le schéma de la base de données, vous devez implémenter des méthodes qui créent et maintiennent la base de données et les tables. Voici quelques déclarations typiques qui créent et suppriment une table:

```
Private static final String SQL_CREATE_ENTRIES =  
    "CREATE TABLE" + FeedEntry.TABLE_NAME + "(" +  
    FeedEntry._ID + "INTEGER PRIMARY KEY," +  
    FeedEntry.COLUMN_NAME_TITLE + "TEXT", +  
    FeedEntry.COLUMN_NAME_SUBTITLE + "TEXT)";  
  
Private static final String SQL_DELETE_ENTRIES =  
    "DROP TABLE SI EXISTE" + FeedEntry.TABLE_NAME;
```

- Tout comme les fichiers que vous enregistrez sur le stockage interne du périphérique, Android stocke votre base de données dans un espace disque privé associé à l'application. Vos données sont sécurisées, car, par défaut, cette zone n'est pas accessible à d'autres applications.

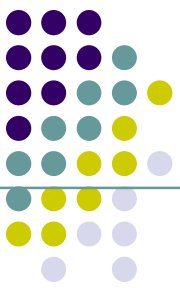


La sauvegarde des données

Enregistrement dans une BD SQL

Créer la base de données avec SQL Helper

- Un ensemble d'API utile est disponible dans la classe [SQLiteOpenHelper](#) . Lorsque vous utilisez cette classe pour obtenir des références à votre base de données, le système exécute les opérations potentiellement courantes de création et de mise à jour de la base de données uniquement lorsque cela est nécessaire et non pendant le démarrage de l'application. Tout ce que vous devez faire est d'appeler [getWritableDatabase\(\)](#) ou [getReadableDatabase\(\)](#).
- **Remarque:** Etant donné qu'elle peuvent prendre du temps à s'exécuter, assurez-vous d'appeler [getWritableDatabase\(\)](#) ou [getReadableDatabase\(\)](#) dans un thread en arrière-plan, par exemple avec [AsyncTask](#) ou [IntentService](#) .
- Pour utiliser [SQLiteOpenHelper](#) , créez une sous-classe qui remplace les méthodes de rappel [onCreate\(\)](#), [onUpgrade\(\)](#) et [onOpen\(\)](#). Vous pouvez également implémenter [onDowngrade\(\)](#), mais ce n'est pas obligatoire.



La sauvegarde des données

Enregistrement dans une BD SQL

Créer la base de données avec SQL Helper

Exemple : Le schéma suivant représente une implémentation de [SQLiteOpenHelper](#) qui utilise certaines commandes mentionnées ci-dessus.

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```



La sauvegarde des données

Enregistrement dans une BD SQL

Créer la base de données avec SQL Helper

- Pour accéder à votre base de données, instanciez votre sous-classe de `SQLiteOpenHelper` :

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getApplicationContext());
```



La sauvegarde des données

Enregistrement dans une BD SQL

Insertion dans une base de données

- On insère des données dans la base de données en passant un objet **ContentValues** à la méthode insert() :

```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_SUBTITLE, subtitle);

// Insert the new row, returning the primary key value of the new row
long newRowId = db.insert(FeedEntry.TABLE_NAME, null, values);
```

- Le premier argument de insert() est simplement le nom de la table.
- Le deuxième argument indique au framework ce qu'il faut faire dans le cas où ContentValues est vide (c.-à-d., Vous n'insérez pas de valeurs).
- Si vous spécifiez que le nom d'une colonne dans ContentValues, la structure insère une ligne et définit la valeur de cette colonne comme nulle. Si vous spécifiez null dans insert , comme dans cet exemple, on n'insère pas de ligne lorsqu'il n'y a pas de valeurs



La sauvegarde des données

Enregistrement dans une BD SQL

Lire les informations d'une base de données

- Pour lire à partir d'une base de données, on utilise la méthode `query()` en lui fournissant une requête.
- La méthode combine les éléments des méthode `insert()` et de `update()`, sauf que la liste des colonnes définit les données que vous souhaitez récupérer plutôt que les données à insérer.
- Les résultats de la requête vous sont retournés dans un objet `Cursor`.



La sauvegarde des données

Enregistrement dans une BD SQL

Lire les informations d'une base de données

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_SUBTITLE
};

// Filter results WHERE "title" = 'My Title'
String selection = FeedEntry.COLUMN_NAME_TITLE + " = ?";
String[] selectionArgs = { "My Title" };

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_SUBTITLE + " DESC";

Cursor cursor = db.query(
    FeedEntry.TABLE_NAME,           // The table to query
    projection,                     // The columns to return
    selection,                       // The columns for the WHERE clause
    selectionArgs,                  // The values for the WHERE clause
    null,                           // don't group the rows
    null,                           // don't filter by row groups
    sortOrder                       // The sort order
);
```



La sauvegarde des données

Enregistrement dans une BD SQL

Lire les informations d'une base de données

- Pour avoir les lignes du curseur, utilisez l'une des méthodes de Cursor, que vous devez toujours appeler avant de commencer à lire les valeurs.
- Puisque le curseur commence à la position -1, appeler `moveToNext()` place la "position de lecture" sur la première entrée dans les résultats et retourne si le curseur est déjà passé la dernière entrée dans l'ensemble résultats.
- Pour chaque ligne, vous pouvez lire la valeur d'une colonne en appelant l'une des méthodes get de Cursor, telles que `getString()` ou `getLong()`.
- Pour chacune des méthodes get, vous devez passer l'index de la colonne que vous désirez obtenir en appelant `getColumnIndex()` ou `getColumnIndexOrThrow()`.
- A la fin des itérations à travers les résultats, on appelle `close()` sur le curseur pour libérer ses ressources.

- **Exemple:**

```
List itemIds = new ArrayList<>();
while(cursor.moveToNext()) {
    long itemId = cursor.getLong(
        cursor.getColumnIndexOrThrow(FeedEntry._ID));
    itemIds.add(itemId);
}
cursor.close();
```



La sauvegarde des données

Enregistrement dans une BD SQL

Supprimer des informations d'une base de données

- Pour supprimer des lignes d'une table, vous devez fournir des critères de sélection qui identifient les lignes.
- L'API de base de données fournit un mécanisme pour créer des critères de sélection qui protègent contre l'injection SQL.
- Le mécanisme divise la spécification de sélection en une clause de sélection et des arguments de sélection.
- La clause définit les colonnes à examiner et vous permet également de combiner les tests de colonne.
- Les arguments sont des valeurs à tester qui sont liées à la clause. Étant donné que le résultat n'est pas traité comme une instruction SQL régulière, il est immunisé contre l'injection SQL.

```
// Define 'where' part of query.  
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";  
// Specify arguments in placeholder order.  
String[] selectionArgs = { "MyTitle" };  
// Issue SQL statement.  
db.delete(FeedEntry.TABLE_NAME, selection, selectionArgs);
```



La sauvegarde des données

Enregistrement dans une BD SQL

Mettre à jour une base de données

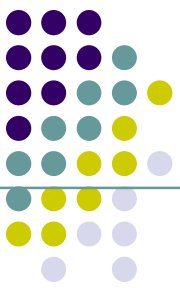
- Lorsque vous devez modifier un sous-ensemble de valeurs de base de données, utilisez la méthode `update()`.
- La mise à jour de la table combine la syntaxe des valeurs de contenu de `insert()` avec la syntaxe de `delete()` .

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the title
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
String[] selectionArgs = { "MyTitle" };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```



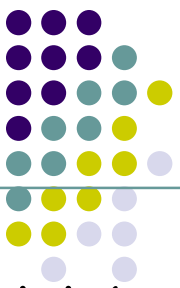
La sauvegarde des données

Enregistrement dans une BD SQL

Connexion persistante à une base de données

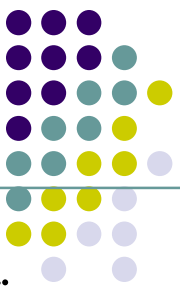
- Puisque `getWritableDatabase()` et `getReadableDatabase()` sont coûteux à appeler lorsque la base de données est fermée, vous devez laisser votre connexion à la base de données ouverte aussi longtemps que vous auriez probablement besoin d'y accéder.
- Généralement, il est optimal de fermer la base de données dans le `onDestroy()` dans l'appel de `Activity`

```
@Override
protected void onDestroy() {
    mDbHelper.close();
    super.onDestroy();
}
```



Interaction avec d'autres applications

- Une application Android comporte généralement plusieurs activités. Chaque activité affiche une interface utilisateur qui permet à l'utilisateur d'effectuer une tâche spécifique (par exemple, voir une carte ou prendre une photo).
- Pour mener l'utilisateur d'une activité à une autre, votre application doit utiliser la classe Intent pour définir "l'Intent" de votre application à faire quelque chose.
- Lorsque vous passez une Intent au système avec une méthode telle que `startActivity()`, le système utilise l'Intent pour identifier et démarrer le composant approprié de l'application. L'utilisation d'intents permet même à votre application de démarrer une activité contenue dans une application distincte.



Interaction avec d'autres applications

- Une Intent peut être explicite afin de démarrer un composant spécifique (une instance spécifique de Activity) ou implicite afin de démarrer tout composant capable de gérer l'action envisagée (par exemple, "capturer une photo").
- Cette partie montre comment utiliser une Intent pour effectuer certaines interactions de base avec d'autres applications, comme :
 1. démarrer une autre application,
 2. recevoir un résultat de cette application et
 3. rendre votre application capable de répondre aux intentions d'autres applications.



Interaction avec d'autres applications

Démarrer une autre application

- L'une des fonctionnalités les plus importantes d'Android est la capacité de l'application à envoyer l'utilisateur à une autre application en fonction d'une "action" qu'elle aimerait effectuer.
- Par exemple, si votre application possède l'adresse d'une entreprise que vous souhaitez afficher sur une carte, vous ne devez pas créer une activité dans votre application qui affiche une carte. Au lieu de cela, vous pouvez créer une demande pour afficher l'adresse en utilisant une Intent. Le système Android lance alors une application capable de montrer l'adresse sur une carte.
- Cette section montre comment créer une intent implicite pour une action particulière et comment l'utiliser pour démarrer une activité qui effectue l'action dans une autre application.



Interaction avec d'autres applications

Démarrer une autre application

Construire une intent implicite

- Les intentions implicites ne déclarent pas le nom de la classe du composant à démarrer, mais déclarent une action à exécuter. L'action spécifie ce que vous voulez faire, comme afficher, modifier, envoyer ou obtenir quelque chose. Les intents incluent aussi souvent les données associées à l'action, telles que l'adresse que vous souhaitez afficher ou le message que vous souhaitez envoyer. Selon l'intent que vous souhaitez créer, les données peuvent être une Uri , l'un des autres types de données, ou l'intent pourrait ne pas nécessiter de données du tout.
- Si vos données Uri , il existe un constructeur Intent() simple à utiliser pour définir l'action et les données.

Exemple 1 (Initier un appel téléphonique): voici comment créer une intent qui initie un appel téléphonique en utilisant les données Uri pour spécifier le numéro de téléphone

```
Uri number = Uri.parse("tel:771332357");  
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
```

Lorsque votre application appelle cette intent en appelant startActivity() , l'application Phone lance un appel au numéro de téléphone donné.



Interaction avec d'autres applications

Démarrer une autre application

Construire une intent implicite

Exemple 2 (Visualiser une carte):

// Point de la carte basé sur l'adresse

Uri location =

Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");

// Ou un point de la carte basé sur la latitude / longitude

// Uri location = Uri.parse("geo:37.422219,-122.08364?z=14");

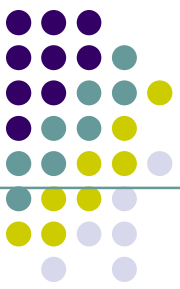
// le paramètre z est le niveau de zoom

Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

Exemple 3 (Visualiser une page web):

Uri webpage = Uri.parse("http://www.android.com");

Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);

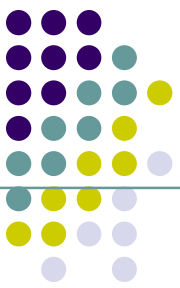


Interaction avec d'autres applications

Démarrer une autre application

Construire une intent implicite

- D'autres types d'intentions implicites nécessitent des données "supplémentaires" qui fournissent différents types de données, comme une chaîne. Vous pouvez ajouter une ou plusieurs données supplémentaires en utilisant les différentes méthodes `putExtra()`.
- Par défaut, le système détermine le type MIME approprié requis par une intent basée sur les données d'Uri incluses. Si vous n'incluez pas une Uri dans l'intent, vous devez généralement utiliser `setType()` pour spécifier le type de données associées à l'intent. Si vous définissez le type MIME, spécifiez les types d'activités qui devraient recevoir l'intention.
- Voici quelques autres intents qui ajoutent des données supplémentaires pour spécifier l'action souhaitée:



Interaction avec d'autres applications

Démarrer une autre application

Construire une intent implicite

Exemple 2 (Envoyer un email avec une pièce jointe:):

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);  
// L'intention n'a pas d'URI, alors déclarez le type MIME "texte / simple"  
emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);  
// destinataires  
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {"kenn@example.com"});  
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Objet de l'email");  
emailIntent.putExtra(Intent.EXTRA_TEXT, "Message texte de l'email");  
emailIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse("content://path/to/email/attachment"));  
// Vous pouvez également attacher plusieurs éléments en passant une ArrayList d'Uris
```



Interaction avec d'autres applications

Démarrer une autre application

Construire une intent implicite

Exemple 2 (Créer un événement de calendrier)

```
Intent calendarIntent = new Intent(Intent.ACTION_INSERT, Events.CONTENT_URI);
Calendar beginTime = Calendar.getInstance().set(2012, 0, 19, 7, 30);
Calendar endTime = Calendar.getInstance().set(2012, 0, 19, 10, 30);
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
beginTime.getTimeInMillis());
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
endTime.getTimeInMillis());
calendarIntent.putExtra(Events.TITLE, "Cours de Android");
calendarIntent.putExtra(Events.EVENT_LOCATION, "Salle 09");
```

Remarque:

Cette intent pour un événement de calendrier est prise en charge uniquement à partir de l'API 14.

Il est important que vous définissiez votre Intent comme étant aussi précis que possible. Par exemple, si vous souhaitez afficher une image à l'aide de l'intent ACTION_VIEW, vous devez spécifier un type MIME de `image/*`. Cela empêche les applications qui peuvent "afficher" d'autres types de données (comme une application de carte) d'être déclenchées par l'intent



Interaction avec d'autres applications

Démarrer une autre application

Vérifiez qu'une application peut recevoir une intent

- Bien que la plate-forme Android garantit que certaines intents puissent être utilisées dans l'une des applications intégrées (telles que l'application Téléphone, E-mail ou Calendrier), vous devez toujours inclure une étape de vérification avant d'invoquer une intent.
- **Attention:** si vous invoquez une intent sans qu'une application pouvant le gérer ne soit disponible, votre application va se planter.
- Pour vérifier qu'il existe une activité qui peut répondre une intent, appelez `queryIntentActivities()` pour obtenir une liste d'activités capables de gérer votre Intent . Si la List renvoyée n'est pas vide, vous pouvez utiliser l'intent en toute sécurité.

Exemple:

```
PackageManager packageManager = getPackageManager();  
List activities = packageManager.queryIntentActivities(intent,  
    PackageManager.MATCH_DEFAULT_ONLY);  
boolean isIntentSafe = activities.size() > 0;
```



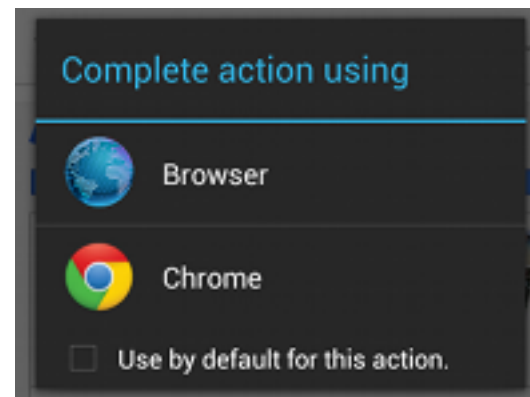
Interaction avec d'autres applications

Démarrer une autre application

Lancer une activité avec une intent

- Une fois que vous avez créé votre Intent et défini les informations supplémentaires, appelez `startActivity()` pour l'envoyer au système. Si le système identifie plus d'une activité pouvant gérer l'intent, il affiche une boîte de dialogue pour que l'utilisateur puisse sélectionner l'application à utiliser, comme le montre la figure ci-dessous. S'il n'y a qu'une seule activité qui gère l'intention, le système démarre immédiatement .

`StartActivity` (intention);





Interaction avec d'autres applications

Démarrer une autre application

Lancer une activité avec une intent

Exemple : Cet exemple montre comment créer une intention pour visualiser une carte, vérifier qu'une application existe pour gérer l'intention, puis démarrer:

```
// Construire l'intent
Uri location =
Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Vérifiez qu'il peut être résolu
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Démarrer une activité si elle est sûre
if (isIntentSafe) {
    startActivity(mapIntent);
}
```



Interaction avec d'autres applications

Démarrer une autre application

Afficher un sélecteur d'application

- Notez que lorsque vous démarrez une activité en passant votre Intent à `startActivity()` et qu'il y ait plus d'une application pouvant répondre à l'intent, l'utilisateur peut sélectionner l'application à utiliser par défaut (en sélectionnant une case au bas de la boîte de dialogue; Voir figure précédente).
- Ceci est agréable lors de l'exécution d'une action pour laquelle l'utilisateur veut généralement utiliser la même application à chaque fois, comme lors de l'ouverture d'une page Web (les utilisateurs utilisent probablement un seul navigateur Web) ou en prenant une photo (les utilisateurs préfèrent une caméra).
- Toutefois, si l'action à exécuter peut être gérée par plusieurs applications et que l'utilisateur puisse préférer une application différente à chaque fois, par exemple une action "partager", pour laquelle les utilisateurs peuvent avoir plusieurs applications par lesquelles ils peuvent partager un élément, vous devriez afficher explicitement une boîte de dialogue de choix comme illustré à la figure suivante.



Interaction avec d'autres applications

Démarrer une autre application

Afficher un sélecteur d'application

- La boîte de sélecteur oblige l'utilisateur à sélectionner l'application à utiliser pour l'action à chaque fois (l'utilisateur ne peut pas sélectionner une application par défaut pour l'action).
- Pour afficher le sélecteur, créez une Intent utilisant `createChooser()` et passez-le à `startActivity()`.

Exemple:

```
Intent intent = new Intent(Intent.ACTION_SEND);
```

```
...
```

```
// Toujours utiliser les ressources chaîne pour le texte de l'IU.
```

```
// Cette ressource peut contenir le texte "Partager cette photo avec"
```

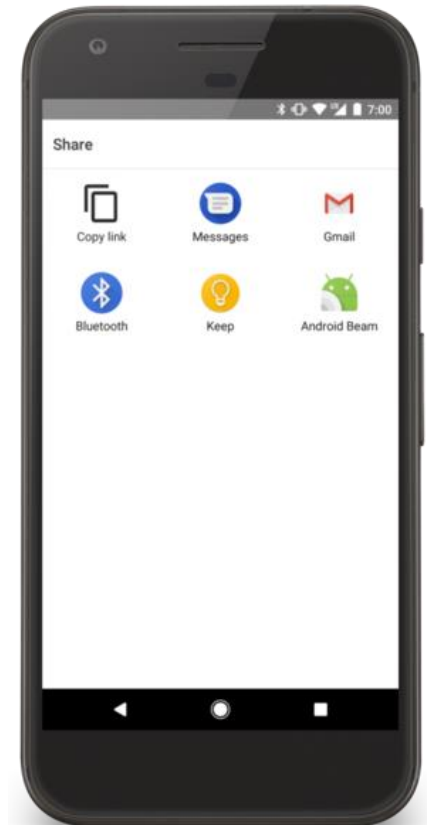
```
String title = getResources().getString(R.string.chooser_title);
```

```
// Créer l'intent qui montre le selecteur
```

```
Intent chooser = Intent.createChooser(intent, title);
```

```
// Vérifiez que l'intent résoudra au moins une activité
```

```
if (intent.resolveActivity(getPackageManager()) != null) {  
    startActivity(chooser);  
}
```





Interaction avec d'autres applications

Récupérer le résultat d'une activité

- Le fait de commencer une autre activité ne doit pas être à sens unique. Vous pouvez également lancer une autre activité et recevoir un résultat. Pour recevoir un résultat, on appelle **startActivityForResult()** (au lieu de **startActivity()**).
- Par exemple, votre application peut démarrer une application caméra et recevoir la photo capturée en conséquence. Ou, vous pouvez démarrer l'application Contacts pour que l'utilisateur puisse sélectionner un contact et vous recevrez les coordonnées en conséquence.
- Bien sûr, l'activité répondante doit être conçue pour renvoyer un résultat. Lorsque cela se produit, il envoie le résultat comme un autre objet Intent . Votre activité le reçoit dans le rappel de **onActivityResult()** .
- **Remarque:** On peut utiliser des intents explicites ou implicites lorsqu'on appelle **startActivityForResult()**. Lorsqu'on démarre une des activités pour recevoir un résultat, on doit utiliser une intent explicite pour s'assurer de recevoir le résultat attendu



Interaction avec d'autres applications

Récupérer le résultat d'une activité

Démarrer l'activité

- Il n'y a rien de spécial dans l'objet Intent qu'on utilise lors du démarrage d'une activité pour avoir un résultat, mais on doit passer un argument entier supplémentaire à la méthode `startActivityForResult()`.
- L'argument entier est un "code de requête" identifiant la requête. Lorsqu'on reçoit l'intent résultat, la fonction de rappel fournit le même code de requête afin que l'application puisse identifier correctement le résultat et pour bien le gérer.
- Exemple: Démarrer une activité qui permet à l'utilisateur de choisir un contact

```
Static final int PICK_CONTACT_REQUEST = 1; // Le code de la requête
```

```
...
```

```
Private void pickContact () {
```

```
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse ("contenu: // contacts"));
```

```
    PickContactIntent.setType(Phone.CONTENT_TYPE); /*Afficher les contacts utilisateur  
                                                    uniquement avec les numéros de téléphone */
```

```
    StartActivityResult (pickContactIntent, PICK_CONTACT_REQUEST);
```

```
}
```



Interaction avec d'autres applications

Récupérer le résultat d'une activité

Recevoir le résultat

- Lorsque l'utilisateur effectue l'activité et les retours suivants, le système appelle la méthode **onActivityResult()** de votre activité. Cette méthode comprend trois arguments:
 - Le code de la requête passé à `startActivityForResult()`.
 - Un code de résultat spécifié par la deuxième activité. Il s'agit de `RESULT_OK` si l'opération a réussi ou `RESULT_CANCELED` si l'utilisateur a été sauvegardé ou l'opération a échoué pour une raison quelconque.
 - Une intent qui porte les données de résultat.



Interaction avec d'autres applications

Récupérer le résultat d'une activité

Recevoir le résultat

- Par exemple, voici comment on peut gérer le résultat pour l'intent "choisir un contact":

@Passer outre

```
Protected void onActivityResult (int requestCode, int resultCode, Intent data) {  
    // On vérifie la requête à laquelle nous répondons  
    if (requestCode == PICK_CONTACT_REQUEST) {  
        if (resultCode == RESULT_OK) { // On s'assure que la requête a réussi  
            // L'utilisateur a choisi un contact.  
            // Les données d'Intent Uri identifient quel contact a été sélectionné.  
            // Faites quelque chose avec le contact ici (plus gros exemple ci-dessous)  
        }  
    }  
}
```

Dans cet exemple, l'Intent résultat retourné par l'application Contacts d'Android fournit un contenu Uri qui identifie le contact choisi par l'utilisateur.



Interaction avec d'autres applications

Récupérer le résultat d'une activité

Recevoir le résultat

- Afin de gérer efficacement le résultat, vous devez comprendre quel sera le format du résultat Intent. Le faire est facile lorsque l'activité qui renvoie un résultat est l'une de vos activités. Les applications incluses avec la plate-forme Android offrent leurs propres API auxquelles vous pouvez compter pour obtenir des données de résultats spécifiques. Par exemple, l'application Contact renvoie toujours un résultat avec l'URI de contenu qui identifie le contact sélectionné et l'application Camera renvoie un Bitmap dans les "data" supplémentaires (voir la classe sur la capture de photos).
- Le code ci-dessus montrant comment obtenir un résultat de l'application Contact sans entrer dans les détails sur la façon de lire les données du résultat. Cela nécessite une discussion plus avancée sur les fournisseurs de contenu.
- Le code suivant montre comment interroger les données résultat pour obtenir le numéro de téléphone du contact sélectionné.



Interaction avec d'autres applications

Récupérer le résultat d'une activité (Exemple)

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    // Vérifiez la demande à laquelle nous répondons  
    if (requestCode == PICK_CONTACT_REQUEST) {  
        if (resultCode == RESULT_OK) { // Assurez-vous que la demande a été réussie  
            Uri contactUri = data.getData(); // Obtenez l'URI qui pointe vers le contact sélectionné  
            // Nous n'avons besoin que de la colonne NUMBER, car il n'y aura qu'une seule ligne dans le résultat  
            String[] projection = {Phone.NUMBER};  
  
            // Exécute la requête sur le contact pour obtenir la colonne NUMBER  
            // Nous n'avons pas besoin d'un ordre de sélection ou de tri (il n'y a qu'un seul résultat pour l'URI donné)  
            // ATTENTION: la méthode query () doit être appelée à partir d'un thread distinct pour éviter de bloquer  
            // le thread de l'IU de votre application. (Pour simplifier l'exemple, ce code ne le fait pas).  
            // Considérez l'utilisation de CursorLoader pour effectuer la requête.  
            Cursor cursor = getContentResolver().query(contactUri, projection, null, null, null);  
            cursor.moveToFirst();  
  
            // Récupérer le numéro de téléphone depuis la colonne NUMBER  
            int column = cursor.getColumnIndex(Phone.NUMBER);  
            String number = cursor.getString(column);  
            // Faites quelque chose avec le numéro de téléphone ...  
        }  
    }  
}
```

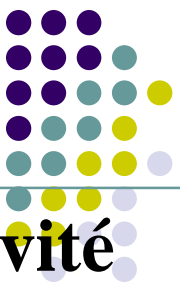


Interaction avec d'autres applications

Récupérer le résultat d'une activité

Recevoir le résultat

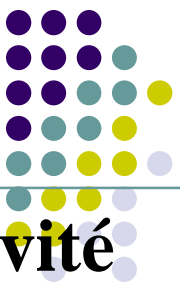
- **Remarque:**
 - Avant Android 2.3 (niveau API 9), l'exécution d'une requête sur le Contacts Provider (comme celui ci-dessus) exige que votre application déclare l'autorisation `READ_CONTACTS` (voir Sécurité et autorisations).
 - A partir de Android 2.3, l'application Contacts accorde à votre application une permission temporaire de lire à partir du fournisseur de contacts lorsqu'il vous renvoie un résultat. L'autorisation temporaire s'applique uniquement au contact spécifique demandé, de sorte que vous ne pouvez pas interroger un contact autre que celui spécifié par Uri l'intention, sauf si vous déclarez l'autorisation `READ_CONTACTS` .



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

- On a vu comment démarrer une activité d'une autre application à partir de votre application. Mais si votre application peut effectuer une action utile à une autre application, votre application devrait être prête à répondre aux demandes d'actions d'autres applications.
- Par exemple, si vous créez une application permettant de partager des messages ou des photos avec d'autres utilisateurs, il est dans votre intérêt de supporter l'Intent `ACTION_SEND` afin que ces utilisateurs puissent lancer une action de "partage" depuis une autre application et lancer votre application pour effectuer l'action.
- Pour cela, vous devez ajouter un élément `<intent-filter>` dans votre fichier de manifeste dans l'élément `<activity>` correspondant.
- Lorsque votre application est installée sur un périphérique, le système identifie vos filtres d'Intent et ajoute les informations à un catalogue interne d'Intents pris en charge par toutes les applications installées. Lorsqu'une application appelle `startActivity()` ou `startActivityForResult()`, avec une Intent implicite, le système trouve l'activité (ou les activités) pouvant répondre à l'Intent.



Interaction avec d'autres applications

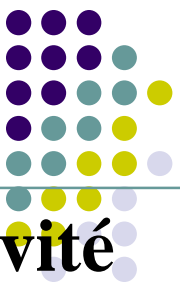
Autoriser d'autres applications à démarrer votre activité

Ajouter un filtre d'intention

- Afin de définir correctement ce que votre activité peut gérer, chaque filtre d'Intent que vous ajoutez devrait être aussi précis que possible en termes de type d'action et de données que l'activité accepte.
- Le système peut envoyer une Intent donnée à une activité si cette activité comporte un filtre d'Intent répond aux critères suivants de l'objet Intent:

Action

- Une chaîne nommant l'action à exécuter. Habituellement, l'une des valeurs définies par la plate-forme telles que ACTION_SEND ou ACTION_VIEW .
- A spécifier dans votre filtre d'Intent dans l'élément <action>. La valeur que vous spécifiez dans cet élément doit être le nom de chaîne complète de l'action, au lieu de la constante de l'API (voir les exemples ci-dessous).



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Ajouter un filtre d'intention

Data

- Une description des données associées à l'Intent.
- A spécifier dans votre filtre d'Intent avec l'élément <data>. En utilisant un ou plusieurs attributs dans cet élément, vous pouvez spécifier uniquement le type MIME, un préfixe URI, un schéma URI, ou une combinaison de ceux-ci et d'autres qui indiquent le type de données accepté.

Remarque: Si vous n'avez pas besoin de déclarer des informations spécifiques sur les données Uri (par exemple, lorsque votre activité gère un autre type de données «supplémentaires», au lieu d'un URI), vous devez spécifier uniquement l'attribut `android:mimeType` pour déclarer le type des données que votre activité gère, comme `text/plain` ou `image/jpeg`.



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

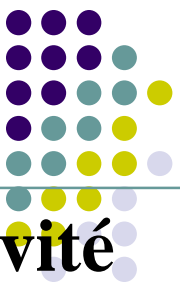
Ajouter un filtre d'intention

Category :

- Fournit un moyen supplémentaire de caractériser l'activité qui gère l'Intent, généralement liée au geste de l'utilisateur ou à l'emplacement à partir duquel elle a débuté. Il existe plusieurs catégories différentes prises en charge par le système, mais la plupart sont rarement utilisées. Cependant, toutes les intents implicites sont définies par `CATEGORY_DEFAULT` par défaut.
- A spécifier dans votre filtre d'intent avec l'élément `<category>`.

Exemple : Activité avec un filtre d'intent qui gère l'intent `ACTION_SEND` lorsque le type de données est soit un texte, soit une image:

```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    <data android:mimeType="image/*"/>
  </intent-filter>
</activity>
```



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Ajouter un filtre d'intention

- Chaque intent entrante spécifie une seule action et un type de données, mais il est possible de déclarer plusieurs instances des éléments `<action>`, `<category>` et `<data>` dans chaque `<intent-filter>` .
- Si deux paires d'actions et de données s'excluent mutuellement dans leurs comportements, vous devez créer des filtres d'intent distincts pour spécifier les actions acceptables lorsqu'ils sont associés aux types de données.
- Par exemple, supposons que votre activité gère à la fois du texte et des images pour les `ACTION_SEND` et `ACTION_SENDTO`. Dans ce cas, vous devez définir deux filtres d'intent distincts pour les deux actions, car une intention `ACTION_SENDTO` doit utiliser les données Uri pour spécifier l'adresse du destinataire en utilisant le schéma URI `send` ou `sendto`



Interaction avec d'autres applications

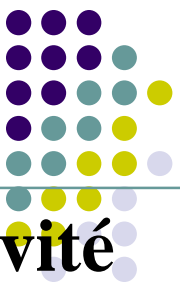
Autoriser d'autres applications à démarrer votre activité

Ajouter un filtre d'intention

Exemple:

```
<activity android:name="ShareActivity">
  <!--Filtre pour l'envoi de texte; Accepte l'action SENDTO avec les schéma URI sms -->
  <intent-filter>
    <action android:name="android.intent.action.SENDTO"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="sms" />
    <data android:scheme="smsto" />
  </intent-filter>

  <!--Filtre d'envoi de texte ou d'images; Accepte l'action SEND et les données texte ou d'image-->
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="image/*"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
</activity>
```

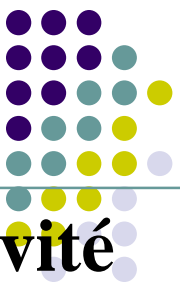



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Ajouter un filtre d'intention

- **Remarque:** Pour recevoir des intents implicites, vous devez inclure la catégorie `CATEGORY_DEFAULT` dans le filtre d'intent. Les méthodes `startActivity()` et `startActivityForResult()` traitent toutes les intents comme si elles avaient déclaré la catégorie `CATEGORY_DEFAULT`. Si vous ne le déclarez pas dans votre filtre d'intent, aucune intent implicite ne résoudra votre activité.



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Gérer l'intent dans votre activité

- Afin de décider des actions à prendre dans votre activité, vous pouvez lire l'Intent utilisée pour le lancer.
- Au fur et à mesure que votre activité démarre, appelez `getIntent()` pour récupérer l'Intent qui a commencé l'activité. Vous pouvez le faire à tout moment pendant le cycle de vie de l'activité, mais vous devez généralement le faire pendant les rappels anticipés tels que `onCreate()` ou `onStart()`.

Exemple:

@Override

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);
```

// Obtenir l'intent qui a commencé cette activité

```
Intent intent = getIntent();
```

```
Uri data = intent.getData();
```

// Déterminer ce qu'il faut faire en fonction du type d'intent

```
if (intent.getType().indexOf("image/") != -1) {
```

```
    // Traite les intents avec des données images ...
```

```
} else if (intent.getType().equals("text/plain")) {
```

```
    // Traite les intents avec du texte ...
```

```
}
```

```
}
```



Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Renvoyer un résultat

- Si vous souhaitez renvoyer un résultat à l'activité qui a invoqué le vôtre, appelez simplement setResult() pour spécifier le code de résultat et l'Intent résultat. Lorsque votre opération est terminée et que l'utilisateur doit revenir à l'activité d'origine, appelez finish() pour fermer (et détruire) votre activité.

- **Exemple :**

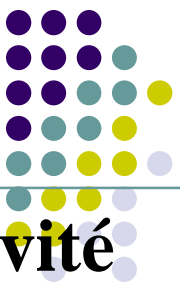
// Créer l'intent pour fournir des données résultat

```
Intent result = new Intent("com.example.RESULT_ACTION",  
                           Uri.parse("content://result_uri"));
```

```
setResult(Activity.RESULT_OK, result);
```

```
finish();
```

- Vous devez toujours spécifier un code résultat avec le résultat. Généralement, c'est RESULT_OK ou RESULT_CANCELED. Vous pouvez ensuite fournir des données supplémentaires avec une Intent , si nécessaire.



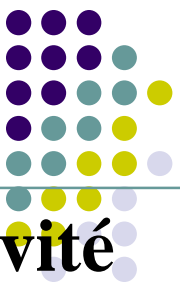
Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Renvoyer un résultat

- **Remarque:** le résultat est défini par `RESULT_CANCELED` par défaut. Donc, si l'utilisateur appuie sur le bouton Retour avant de terminer l'action et avant de définir le résultat, l'activité d'origine reçoit le résultat "canceled".
- Si vous devez simplement renvoyer un nombre entier qui indique l'une des nombreuses options de résultat, vous pouvez définir le code résultat à une valeur supérieure à 0. Si vous utilisez le code résultat pour livrer un nombre entier et que vous n'avez pas besoin d'inclure l' Intent , vous pouvez appeler `setResult()` et ne transmettre qu'un code résultat.
- **Exemple:**

```
setResult(RESULT_COLOR_RED);  
finish();
```

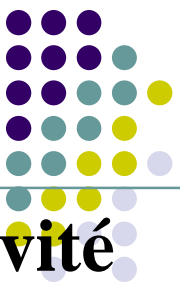


Interaction avec d'autres applications

Autoriser d'autres applications à démarrer votre activité

Renvoyer un résultat

- Dans ce cas, il ne peut y avoir qu'une poignée de résultats possibles, de sorte que le code résultat soit un nombre entier défini localement (supérieur à 0). Cela fonctionne bien lorsque vous renvoyez un résultat à une activité dans votre propre application, car l'activité qui reçoit le résultat peut faire référence à la constante publique pour déterminer la valeur du code résultat.
- **Remarque:** Il n'est pas nécessaire de vérifier si votre activité a été démarrée avec `startActivity()` ou `startActivityForResult()`. Appelez simplement `setResult()` si l'intention qui a commencé votre activité pourrait s'attendre à un résultat. Si l'activité d'origine avait appelé `startActivityForResult()`, alors le système délivre le résultat que vous fournissez à `setResult()` ; Sinon, le résultat est ignoré.

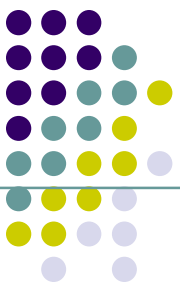


Prise en compte des permissions systèmes

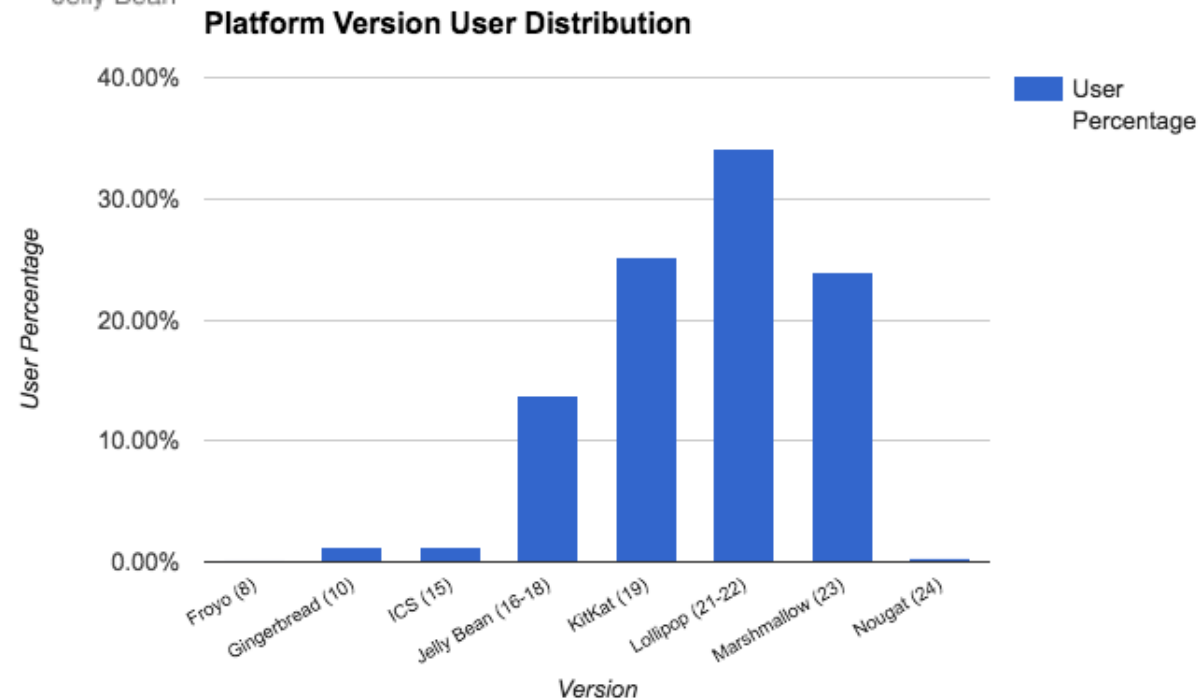
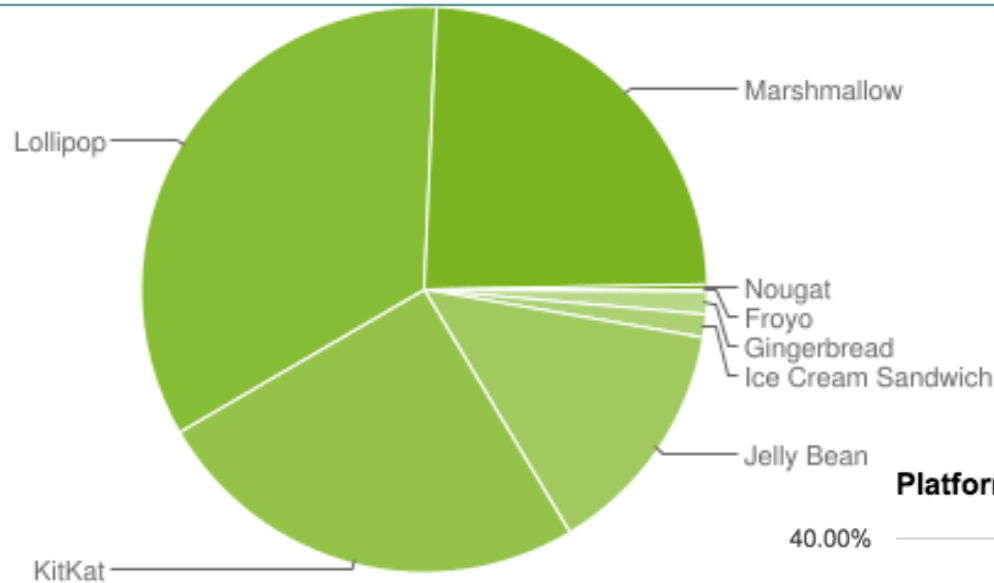
Autoriser d'autres applications à démarrer votre activité

Renvoyer un résultat

- Dans ce cas, il ne peut y avoir qu'une poignée de résultats possibles, de sorte que le code résultat soit un nombre entier défini localement (supérieur à 0). Cela fonctionne bien lorsque vous renvoyez un résultat à une activité dans votre propre application, car l'activité qui reçoit le résultat peut faire référence à la constante publique pour déterminer la valeur du code résultat.
- **Remarque:** Il n'est pas nécessaire de vérifier si votre activité a été démarrée avec `startActivity()` ou `startActivityForResult()`. Appelez simplement `setResult()` si l'intention qui a commencé votre activité pourrait s'attendre à un résultat. Si l'activité d'origine avait appelé `startActivityForResult()`, alors le système délivre le résultat que vous fournissez à `setResult()` ; Sinon, le résultat est ignoré.



Statistique sur l'utilisation des versions de Android





Utilisation de Google Play pour distribuer et monétiser

Liste des étapes à suivre:

1. Ouvrir un compte développeur Google Play:
 - **Adresse :** <https://play.google.com/apps/publish/signup/>
 - **Tutoriel:** http://blog.goodbarber.com/fr/Comment-ouvrir-un-compte-developpeur-Google-Play_a295.html
2. Soumettre son application dans Google Play :
 - **Adresse :** <https://play.google.com/apps/publish>
 - **Tutoriel :** http://blog.goodbarber.com/fr/Comment-soumettre-mon-application-dans-Google-Play_a302.html



Mooc sur Android

1. Udacity

<https://www.udacity.com/course/developing-android-apps--ud853>