# Fournisseurs de contenu

# Plan

- Explorer différent fournisseurs de contenus
- `Fournisseurs de contenu: MediaStore`, `CallLog`, `CalendarContract`, `UserDictionary`, `VoicemailContract`, `Settings`, and `ContactsContract`
- Accéder aux fournisseurs de contenus qui demandent de configure des permissions
- Ajouter, Modifier, Supprimer des Enregistrements

# Fournisseurs de contenu

- Une application peut accéder aux données d'autres applications en utilisant des interfaces de fournisseur de contenu
- Une application peut exposer des données internes à d'autres applications en devenant un fournisseur de contenu.
- Les fournisseurs de contenu sont utilisés pour accéder aux données utilisateur, telles que les contacts, les images, l'audio et la vidéo sur l'appareil, et bien plus encore.

# Fournisseurs de contenu

- Plusieurs applications fournis par le système Android exposent leurs données à travers les fournisseurs de contenu.
- Votre application peut accéder à des fournisseurs de contenu offrant des données provenant de différentes sources (fichiers, bases de données, etc.)
- Vous trouverez les fournisseurs de contenu fournis par Android dans le package `android.provider`.

# Fournisseurs de contenu

| Fournisseurs | Objectifs |
|---|---|
| `AlarmClock` | **Régler des alarmes (API Level 9)** |
| `CalendarContract` | **Calendrier (API Level 14)** |
| `CallLog` | **Appels** |
| `ContactsContract` | **Base de données des contacts** |
| `DocumentsProvider` | **Accès en lecture/écriture aux fichiers (API Level 19)** |
| `MediaStore` | **Données Audio/visuel interne/externe** |
| `SearchRecentSuggestions` | **Create appropriate search suggestions** |
| `Settings` | **System-wide device settings and preferences** |
| `Telephony` | **SMS et MMS (API Level 19)** |
| `UserDictionary` | **Dictionnaire des mots utilisés par l'utilisateur pour faire des prédictions de saisies au clavier** |
| `VoicemailContract` | **Voicemail (API Level 14)** |

# Fournisseur de contenu
## `MediaStore`

- Vous pouvez utiliser le fournisseur de contenu `MediaStore` pour accéder aux médias (audio, vidéo, images) situés sur vos supports de stockage interne et externe.
- Vous pouvez accéder à ces différents médias à travers leurs classes dans le package `android.provider.MediaStore`.

# Fournisseur de contenu
## `MediaStore`

- La plupart des classes de `MediaStore` permettent une interaction complète avec les données.
- Vous pouvez rechercher, ajouter, supprimer des fichiers de média.

# Fournisseur de contenu
## MediaStore

| Class | Purpose |
|---|---|
| Audio.Albums | **Gestion des fichiers audio organisés en albums.** |
| Audio.Artists | **Gestion des fichiers audio répartis par artiste.** |
| Audio.Genres | **Gestion des fichiers audio suivant leur genre** |
| Audio.Media | **Gestion des fichiers audio** |
| Audio.Playlists | **Gestion des fichiers audio en utilisant des playlists** |
| Audio.Radio | **Gestion des fichiers audio radio (API Level 21)** |
| Files | **Affichage des fichiers audio (API Level 11)** |
| Images.Media | **Gestion des fichiers image** |
| Images.Thumbnails | **Recherche des thumbnails des fichiers images** |
| Video.Media | **Gestion des fichiers video** |
| Video.Thumbnails | **Recherche des thumbnails des fichiers videos** |

# Permission pour accéder aux CallLog

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

# Ajouter la Permission pour accéder aux CallLog

```
if (ActivityCompat.checkSelfPermission(this,
            Manifest.permission.READ_CALL_LOG)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(UserActivity.this,
        PERMISSIONS_CALL_LOG, REQUEST_CALL_LOG);
} else {
    // permission already accepted, continue as usual
}
```

# Fournisseur de contenu
## `ContactsContract`

- Android par défaut une application permettant de gérer vos contacts. Les données sur vos contacts sont exposés à d'autres applications en utilisant un fournisseur de contenu
- Le fournisseur de contenu pour accéder aux contacts utilisateur s'appelait au départ `Contacts`.
- À partir de la version Android 2.0 (API Level 5), il a été introduit une autre classe de gestionnaire de contenu appelée `ContactContract`, incluant une sous-classe `ContactsContract.Contacts`.
- Le profil personnel de l'utilisateur est accessible à travers la classe `ContactsContract.Profile`.

# Fournisseur de contenu
## `ContactsContract`

- **Accès en lecture aux contacts**

  **<uses-permission android:name="android.permission.READ_CONTACTS" />**

- **Accès en écriture aux contacts**

  **<uses-permission android:name="android.permission.WRITE_CONTACTS" />**

- **Accès en lecture au profil de l'utilisateur**

  **<uses-permission android:name="android.permission.READ_PROFILE" />**

# Fournisseur de contenu

## ContactsContract

```
if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS)
    != PackageManager.PERMISSION_GRANTED
) {
    Log.i(TAG, "Contact permissions not granted. Requesting permissions.")
    requestPermissions(
        arrayOf(Manifest.permission.READ_CONTACTS), 123
    )
} else {
    val projection = arrayOf(
        ContactsContract.Contacts.DISPLAY_NAME,
        ContactsContract.CommonDataKinds.Phone.NUMBER
    )
    val loader = CursorLoader(
        this,
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        projection, null, null, "display_name desc limit 1"
    )
    val contactCursor: Cursor = loader.loadInBackground()!!
    Log.d(TAG, "Contacts Count: ${contactCursor.count}")
    if (contactCursor.count > 0) {
        val nameIndex: Int = contactCursor
            .getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)
        val phoneIndex: Int = contactCursor
            .getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER)
        contactCursor.moveToFirst()
        Log.d(TAG, "Name: " + contactCursor.getString(nameIndex))
        Log.d(TAG, "Phone: " + contactCursor.getString(phoneIndex))
    }
}
```

# Fournisseur de contenu
## ContactsContract

| Classe | Objectif |
|---|---|
| ContactsContract.CommonDataKinds | **Définit un nombre de colonnes fréquemment utilisées pour les contacts telles que email, nickname, phone, photo** |
| ContactsContract.Contacts | **Définit des données consolidées associées à un contact, peuvent contenir des aggrégations** |
| ContactsContract.Data | **Définit les données brutes associées à un seul contact** |
| ContactsContract.PhoneLookup | **Définit les colonnes de numéro de téléphone et peut être utilisé pour rechercher rapidement un numéro à des fins d'identification de l'appelant** |
| ContactsContract.StatusUpdates | **Définit des colonnes de reseaux sociaus et peuvent être utilisées pour chercher le statut d'un compte utilisateur** |
| ContactsContract.PinnedPositions | **Définit si un contact a été épinglé par l'utilisateur afin que les applications puissent présenter à l'utilisateur un ordre de ces contacts épinglés (API niveau 21)** |

# Mise à jour d'un contact

- Pemission:
  - `WRITE_CONTACTS`

# Insertion

```
val ops = ArrayList<ContentProviderOperation>()

var op =   ContentProviderOperation.newInsert(
        ContactsContract.RawContacts.CONTENT_URI)
    .withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, null)
    .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, null)

ops.add(op.build())
```

# Insertion

```
op = ContentProviderOperation.newInsert
        (ContactsContract.Data.CONTENT_URI)
    .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
    .withValue(ContactsContract.Data.MIMETYPE,
        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)

.withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
        prenom?.text.toString() + " " + nom?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.StructuredName.FAMILY_NAME,
        nom?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.StructuredName.GIVEN_NAME,
        prenom?.text.toString())

ops.add(op.build())
```

# Insertion

```
op = ContentProviderOperation.newInsert(
        ContactsContract.Data.CONTENT_URI)
    .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
    .withValue(ContactsContract.Data.MIMETYPE,
        ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE)
    .withValue(ContactsContract.CommonDataKinds.Email.ADDRESS,
        emailAddress?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.Email.TYPE,
        courriel_type.text.toString())

ops.add(op.build())
```

# Insertion

```
op = ContentProviderOperation.newInsert(
        ContactsContract.Data.CONTENT_URI)
    .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
    .withValue(ContactsContract.Data.MIMETYPE,
       ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
    .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER,
        phoneNumber?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
        ContactsContract.CommonDataKinds.Phone.TYPE_MAIN)

ops.add(op.build())
```

```
contentResolver.applyBatch(ContactsContract.AUTHORITY, ops)
```

# Mise à jour

```
val ops = ArrayList<ContentProviderOperation>()
val id = idTb?.text.toString()

val selectionClause = "${ContactsContract.Data.RAW_CONTACT_ID} = ?
        AND ${ContactsContract.Data.MIMETYPE} = ?"
val selectionArgs = arrayOf(id,
        ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)

var op = ContentProviderOperation.newUpdate(
        ContactsContract.Data.CONTENT_URI)
    .withSelection(selectionClause, selectionArgs)
   .withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
        prenom?.text.toString() + " " + nom?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.StructuredName.FAMILY_NAME,
        nom?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.StructuredName.GIVEN_NAME,
        prenom?.text.toString())

ops.add(op.build())
```

# Mise à jour

```
op = ContentProviderOperation.newUpdate(ContactsContract.Data.CONTENT_URI)
    .withSelection(selectionClause,
arrayOf(id, ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE))
    .withValue(ContactsContract.CommonDataKinds.Email.ADDRESS,
        emailAddress?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.Email.TYPE,
        courriel_type.text.toString())

ops.add(op.build())
```

# Mise à jour

```
op = ContentProviderOperation.newUpdate(ContactsContract.Data.CONTENT_URI)
    .withSelection(selectionClause, arrayOf(id,
        ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE))
    .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER,
        phoneNumber?.text.toString())
    .withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
        ContactsContract.CommonDataKinds.Phone.TYPE_MAIN)

ops.add(op.build())
```

# Supprimer tout

```
val hasWriteContactsPermission =
        checkSelfPermission(Manifest.permission.WRITE_CONTACTS)

if (hasWriteContactsPermission != PackageManager.PERMISSION_GRANTED) {
    requestPermissions(
        arrayOf(Manifest.permission.WRITE_CONTACTS),123)
} else {
    val ops = ArrayList<ContentProviderOperation>()
    var op =  ContentProviderOperation.newDelete(
        ContactsContract.RawContacts.CONTENT_URI)

    ops.add(op.build());

    contentResolver.applyBatch(ContactsContract.AUTHORITY, ops)
}
```

# Supprimer un enregistrement

```
val selectionArgs = arrayOf(idTb?.text.toString())
val selectionClause = "${ContactsContract.Data.RAW_CONTACT_ID} = ?"

val ops = ArrayList<ContentProviderOperation>()
var op =   ContentProviderOperation.newDelete(
        ContactsContract.RawContacts.CONTENT_URI)
                .withSelection(selectionClause, selectionArgs)

ops.add(op.build());

contentResolver.applyBatch(ContactsContract.AUTHORITY, ops)
```

# Références

- **Android API Guides: “Fournisseurs de contenus”:**
  - *http://d.android.com/guide/topics/providers/content-providers.html*
- **Android SDK Reference par rapport au package `android.provider` :**
  - *http://d.android.com/reference/android/provider/package-summary.html*
- **Android SDK Reference par rapport au fournisseur de contenu `CallLog`:**
  - *http://d.android.com/reference/android/provider/CallLog.html*
- **Android SDK Reference par rapport au fournisseur de contenu `Contacts`:**
  - *http://d.android.com/reference/android/provider/Contacts.html*
- **Android SDK Reference par rapport à `ContactsContract` :**
  - *http://d.android.com/reference/android/provider/ContactsContract.html*
- **Android SDK Reference par rapport au `MediaStore`:**
  - *http://d.android.com/reference/android/provider/MediaStore.html*