

JAVA ET ANDROID



Les données non SQL

Sommaire

Chapitre 1 - GESTION DES DONNEES.....	7
1.1 - Introduction.....	8
1.2 - Stratégies pour le stockage des données.....	9
Chapitre 2 - LES RESSOURCES : gestion des données privées et en lecture seule.....	14
2.1 - Raw Resource de type CSV.....	15
2.1.1 - Objectif.....	15
2.1.2 - Syntaxes IO.....	16
2.1.3 - Le problème des accents.....	17
2.1.4 - Affichage d'une Raw resource.....	18
2.1.4.1 - Le layout : fichier_ressources_lire.xml.....	19
2.1.4.2 - La classe FichierRessourcesLire.java.....	20
2.1.5 - Exercice : afficher un numéro de téléphone d'une ressource CSV suite à une sélection dans une liste triée de noms.....	21
2.2 - XML Resource.....	22
2.2.1 - Objectif.....	22
2.2.2 - Préambule XML.....	23
2.2.2.1 - Types de contenus dans les fichiers XML.....	23
2.2.2.2 - Principaux « éléments » XML.....	23
2.2.2.3 - Taxinomie (*) des parseurs XML.....	24
2.2.3 - Syntaxes XPP (XML Pull Parser).....	25
2.2.4 - Affichage d'une ressource XML de type Document dans un spinner.....	30
2.2.4.1 - Objectif.....	30
2.2.4.2 - Contexte.....	30
2.2.4.3 - Démarche.....	30
2.2.4.4 - Le layout xml_document_ressources_lire.xml.....	31
2.2.4.5 - XmlDocumentRessourcesLire.java.....	32
2.2.5 - Affichage d'une ressource XML de type Data dans un spinner.....	35
2.2.5.1 - Objectif.....	35
2.2.5.2 - Le fichier xml_data_ressources_lire.xml.....	36
2.2.5.3 - XmlDataRessourcesLire.java.....	37
2.2.6 - Exercice : XMLRessourcesLireExo (lire un fil RSS dans une ListView).....	40
2.3 - Ressource JSON.....	41
2.3.1 - Objectif.....	41
2.3.2 - Démarche.....	41

2.3.3 - Le fichier JSON.....	41
2.3.4 - Le layout json_ressources_lire.xml.....	42
2.3.5 - La classe JsonUtilitaires.java.....	43
2.3.6 - L'activité JsonRessourcesLire.java.....	44
Chapitre 3 - LES DATA : gestion de données en lecture/écriture et éventuellement publiques (CSV, XML, JSON).....	46
3.1 - Objectif de ce chapitre.....	47
3.2 - Le FileExplorer.....	48
3.3 - Quelques notions sur les fichiers avec Android.....	50
3.4 - Syntaxes pour travailler dans le dossier files.....	52
3.5 - Rappel de techniques de lecture et d'écriture classiques.....	53
3.6 - Lister le contenu d'un dossier.....	55
3.6.1 - Objectif.....	55
3.6.2 - Layout : data_fichiers_tests.xml.....	56
3.6.3 - Activité : DataFichiersTests.java.....	57
3.7 - Création d'un fichier ASCII stocké dans /data/data/n.d.p/files/	59
3.7.1 - Objectif.....	59
3.7.2 - Écran.....	59
3.7.3 - Layout.....	59
3.7.4 - Activité.....	60
3.8 - Lecture du contenu d'un fichier ASCII stocké dans data/data/n.d.p/files/.....	62
3.8.1 - Objectif.....	62
3.8.2 - Écran.....	62
3.8.3 - Layout.....	62
3.8.4 - Activité.....	63
3.9 - Gestion d'un fichier CSV stocké dans le téléphone.....	65
3.9.1 - Objectif.....	65
3.9.2 - Conception.....	65
3.9.3 - Le layout repertoire_csv.xml.....	68
3.9.4 - La diagramme de la classe Fichier.java (pour la gestion des données, le modèle).....	71
3.9.5 - Le diagramme de la classe RepertoireCSV.....	72
3.9.6 - Le code de la classe Fichier.java (classe utilitaire).....	73
3.9.7 - Le code de la classe RepertoireCSV.java (l'activité).....	76
3.9.8 - Exercice : supprimer un enregistrement d'un fichier CSV....	78
3.10 - Sous-dossier de files.....	79
3.10.1 - Objectif.....	79
3.10.2 - Démarche.....	79
3.10.3 - Layout.....	79
3.10.4 - Activité.....	80

Chapitre 4 - FICHIER et SD.....	82
4.1 - Lecture d'un fichier XML stocké dans du storage.....	83
4.2 - L'application RSS (XML) : créer et ajouter un fil.....	85
4.2.1 - Objectif.....	85
4.2.2 - Démarche.....	86
4.2.3 - Le fichier XML : rss.xml.....	86
4.2.4 - L'ajout des bibliothèques jdom-1.1.2.jar et jaxen.jar.....	87
4.2.5 - Syntaxes IO.....	88
4.2.6 - Syntaxes JDOM.....	89
4.2.7 - Le layout : creation flux rss.xml.....	90
4.2.8 - L'activité : CreationFluxRSS.java.....	92
4.2.9 - Exercice : visualiser les <title> du fichier rss.xml dans un Spinner.....	95
4.3 - Lecture d'un fichier JSON stocké dans du storage.....	96
4.3.1 - Objectif.....	96
4.3.2 - json_data_lire.xml.....	97
4.3.3 - JsonDataLire.java.....	98
4.4 - CRUD sur un fichier JSON stocké dans du storage.....	100
4.4.1 - Objectif.....	100
4.4.2 - Le layout json crud.xml.....	101
4.4.3 - La classe JSONUtilitaires.java.....	102
4.4.4 - L'activité JsonCRUD.java.....	103
Chapitre 5 - LES PREFERENCES.....	105
5.1 - Introduction.....	106
5.1.1 - Généralités.....	106
5.1.2 - L'écran d'accueil.....	106
5.1.3 - Diagrammes.....	107
5.1.3.1 - Diagramme de navigation.....	107
5.1.3.2 - Diagramme de composants.....	108
5.1.4 - Les fichiers XML ... généraux.....	109
5.1.4.1 - arrays.xml stocké dans /res/values/.....	109
5.1.4.2 - colors.xml stocké dans /res/values/.....	109
5.1.4.3 - menu_principal.xml.....	110
5.2 - La gestion personnalisée des préférences.....	111
5.2.1 - Présentation.....	111
5.2.2 - Objectif et écran.....	112
5.2.3 - Syntaxes.....	113
5.2.4 - Codes.....	116
5.2.4.1 - preferences_crud.xml.....	116
5.2.4.2 - PreferencesCRUD.java.....	118
5.2.4.3 - L'activité MenuPrincipal.java.....	122

5.2.5	- Fichiers générés.....	124
5.2.6	- Exercice : terminer le CRUD.....	125
5.2.6.1	- First.....	125
5.2.6.2	- Second.....	125
5.3	- Le framework Android pour gérer les préférences.....	126
5.3.1	- Objectif.....	126
5.3.2	- Démarche.....	129
5.3.3	- Syntaxes.....	130
5.3.4	- Codes.....	132
5.3.4.1	- preferences framework.xml stocké dans /res/layout/..	132
5.3.4.2	- PreferencesFramework.java.....	132
5.3.4.3	- MenuPrincipal.java.....	133
5.3.4.4	- La modification du fichier Manifest.....	136
5.3.5	- Fichier généré.....	137
5.3.6	- Exercice.....	138
5.4	- Les préférences système.....	139
5.4.1	- Objectifs.....	139
5.4.2	- Récupérer une préférence système.....	140
5.4.3	- Créer une préférence système.....	141
Chapitre 6	- FILEPROVIDER.....	142
Chapitre 7	- ANNEXES.....	144
7.1	- Histoire de contexte.....	145
7.2	- Lire un fichier image stocké sur la SD et l'afficher dans une ImageView.....	146
7.3	- Lire un fichier ASCII stocké dans les assets.....	146
7.4	- Lire un fichier stocké dans un sous-dossier de /data/data/.../files/.....	147
7.5	- Transférer un fichier de /res/ vers /data/data/files.....	148
7.5.1	- Transférer un fichier de /res/raw vers /data/data/files.....	148
7.5.2	- Transférer un fichier de /res/xml vers /data/data/files.....	149
7.6	- Les fichiers de données utilisés dans ce module.....	150
7.6.1	- repertoire.txt de /res/raw.....	150
7.6.2	- repertoire.csv du File System.....	150
7.6.3	- villes_data.xml : document de type DATA.....	151
7.6.4	- villes_doc.xml : document de type DOCUMENT.....	151
7.6.5	- communes_mixte.xml : document de type mixte.....	152
7.6.6	- rss.xml.....	153
7.6.7	- villes.json.....	154
7.6.8	- pays.json.....	155
7.6.9	- Autres fichiers .txt.....	156
7.6.9.1	- tintin.txt.....	156

7.6.9.2 - daltons.txt.....	156
7.6.9.3 - pieds_nickeles.txt.....	156
7.6.9.4 - asterix.txt.....	156
7.7 - Quelques classes utilitaires.....	157
7.7.1 - La classe Tableau.....	157
7.7.2 - La classe Fichier.....	159
7.7.3 - La classe XMLXPP.....	165
7.7.4 - La classe XML.....	166
7.8 - TP : l'application Lexique.....	167
7.9 - InputStream/FileInputStream (comparatif).....	169
7.10 - FileOutputStream (comparatif).....	170
7.11 - adb shell.....	171
7.12 - L'application Terminal.....	172
7.13 - Synthèse.....	173
7.13.1.1 - Le monde RES.....	173
7.13.1.2 - Le monde /data/data/n.d.p/files/.....	173
7.13.1.3 - Fichier dans /data/data/n.d.p/files et lecture.....	174

CHAPITRE 1 - GESTION DES DONNEES

1.1 - INTRODUCTION

Les applications Android utilisent des données SQL et « Non SQL »¹ (TXT, CSV, XML, JSON, ...).

Les données de type « Non SQL » peuvent être stockées dans les ressources de l'application, dans les data de l'application, dans la SD, dans des préférences.

C'est l'objet de ce document que de présenter cette gestion.

L'accès à des données « Non SQL » distantes en lecture et/ou écriture (principalement accessibles via le WEB en mode direct ou via des services web) est abordé dans un autre document.

L'accès à des données SQL locales ou distantes en lecture et/ou écriture est abordé dans 2 autres documents.

1 : « Non SQL » n'a aucun rapport avec les données « NoSQL ».

1.2 - STRATÉGIES POUR LE STOCKAGE DES DONNÉES

Il existe plusieurs possibilités pour le stockage des données.

Selon le format de stockage :

- ✓ Préférences,
- ✓ Données CSV,
- ✓ Données XML,
- ✓ Données JSON,
- ✓ Données SQL.

Selon le « lieu » de stockage :

1. en local dans les ressources,
2. en local sur les mémoires de masse du terminal,
3. sur un serveur distant (données distantes).

Cela dépend principalement de plusieurs facteurs :

1. taille des données,
2. type des données,
3. données en RO ou en RW,
4. taux de mise à jour des données,
5. données privées versus données publiques,
6. l'exigence de la rapidité d'accès aux données,
7. nécessité d'avoir des données actualisées en temps réel,
8. nécessité d'avoir accès aux données en mode déconnecté,
9. optimisation de l'utilisation de la batterie.

Notes :

une application qui utilise intensivement le réseau est une application coûteuse à double titre ; elle utilise intensément de la bande passante, et donc le forfait de l'utilisateur, et elle utilise intensément la batterie.

Quelle est la taille des données ?

- ✓ De petites tailles,
- ✓ De moyennes ou grandes tailles.

Quel est le type de données que l'on veut stocker ?

- ✓ Des données de type clé=valeur
- ✓ Des données au format text,
- ✓ Des données au format CSV,
- ✓ Des données XML,
- ✓ Des données JSON,
- ✓ Des données SQL.

La rapidité d'accès et de coût de la manipulation des données

Les données distantes sont coûteuses en temps, en consommation réseau, en utilisation d'énergie donc de la batterie.

Pour quelle stratégie ?

- ✓ Comme ressource de l'application ?
- ✓ Comme ressource du terminal ?
- ✓ Comme ressource étendue du terminal (Carte SD) ?
- ✓ Comme ressource externe au terminal (Réseau) ?

- ✓ Données accessibles pour la seule application ?
- ✓ Données accessibles pour plusieurs applications ?
- ✓ Données en lecture/écriture ?
- ✓ Données en lecture seule ?

Stockage/Type	Lecture seule	Lecture Écriture	Text	CSV	JSON	XML	SQL	Accès
/res/xxx/	x		x	x	x	x		Privé
/data/data/		x	x	x	x	x	x	Privé/Public
SD		x	x	x	x	x		Privé/Public
Externe (Site web, serveur de BD, ...)		x	x	x	x	x	x	Privé/Public

Note : Privé et Public se réfèrent à l'accès aux données au regard des applications Android. Des données privées sont des données qui ne sont accessibles que par l'application qui les stocke (que ce soit dans les ressources ou dans les « data » ou sur la SD. Les données publiques (partagées serait plus adéquat) sont des données accessibles potentiellement pas d'autres applications Android.

Où les stocker?

Dans les préférences

- ✓ des paires clés-valeurs.

Dans le dossier /res/

- ✓ Les fichiers textes, csv, json, etc dans le sous-dossier /raw/,
- ✓ Les fichiers XML dans le sous-dossier /xml/,
- ✓ Etc.

Dans le dossier /data/data

- ✓ Les fichiers dans le sous-dossier /files/,
- ✓ Les BD SQLite dans le sous-dossier /databases/.

Ailleurs ... sur le réseau ...

En résumé :

Si les données sont propres à l'application, sont en lecture seule et sont connues à la création de l'application alors elles peuvent être stockées dans des fichiers de type ressource donc dans le dossier /res/raw/ ou /res/xml/ de l'application.

Si les données sont, à l'inverse, communes à plusieurs applications Android et/ou en lecture/écriture elles doivent être stockées dans /data/data/...
Pour que les données soient partagées il faudra créer un ContentProvider et les utiliser avec un ContentResolver (cf la gestion des données SQL).

Enfin si les données sont communes à plusieurs applications (Android, Java, PHP, ...) elles sont stockées sur le réseau (Serveur Web, serveur de BD, ...). On s'approche des Web Services pas à pas !

Généralités sur la gestion des données

Ressource CSV : visualisation.

Ressource XML : visualisation.

Data CSV : CRUD.

Data XML : CRUD.

SQLite : CRUD.

Content Provider : si les données doivent être rendues publiques.

Content Resolver : selon le Content Provider.

	Paires clé/valeur	CSV	XML	SQL
Préférence	x	x	X	X
Ressource		Affichage de tous les enregistrements dans un TextView. Saisie d'un nom dans un EditText et affichage d'un enregistrement dans un TextView.		X
Data/files				
Data/databases				

CHAPITRE 2 - LES RESSOURCES : GESTION DES DONNÉES PRIVÉES ET EN LECTURE SEULE

2.1 - RAW RESOURCE DE TYPE CSV

2.1.1 - Objectif

Exploiter des données stockées dans un fichier de type CSV stockée dans /res/raw/.

Le fichier étant une ressource de l'application il n'est accessible que par cette application.

Et il est en lecture seule.

Lors de la création de l'apk il sera stocké dedans. Si vous allez voir les caractéristiques de l'application vous ne verrez que x ko pour l'application et 0 ko pour les données.

2.1.2 - Syntaxes IO

Ouverture en lecture d'une ressource du dossier /res/raw/ dans un **data stream**.
Cela peut être aussi une image ou un son ou une video.

```
InputStream is = contexte.getResources().openRawResource(R.raw.ressource);
```

Construction d'un InputStreamReader à partir d'un InputStream afin de pouvoir bufferiser ensuite.
Sans buffer la lecture se ferait en mode byte (*).

```
InputStreamReader isr = new InputStreamReader(is);
```

Bufferisation

```
BufferedReader br = new BufferedReader(isr);
```

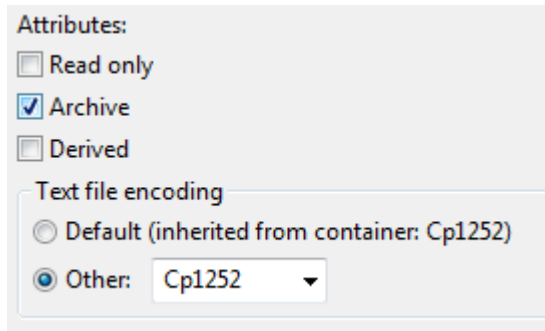
Lecture d'un enregistrement dans le buffer (une ligne jusqu'au retour chariot)

```
br.readLine();
```


2.1.3 - Le problème des accents

Si le fichier a été créé sous Windows ou dans Eclipse sous Windows il se peut qu'il soit encodé en Cp1252.

Dans Eclipse sélectionnez le fichier, cliquez droit puis propriétés et changez son jeu de caractères (UTF-8). Ouvrez le fichier et vérifiez les accents.



Autrement ...

(*) Sans InputStreamReader et sans buffer : **c'est plus lent mais les accents seront restitués correctement.**

```
// --- Sans buffer
InputStream is =
contexte.getResources().openRawResource(R.raw.repertoire);
StringBuilder lsbContenu = new StringBuilder("");
int liValeur = 0;
while((liValeur = is.read()) != -1) {
    if(liValeur != 13) { // Pour eliminer le RC (13) et ne garder que le
LF (10)
        lsbContenu.append((char)liValeur);
    }
}
lsContenu = lsbContenu.toString();
is.close();
```

Note :

La touche "Entrée" : c'est un retour chariot et un saut de ligne, qui s'appellent respectivement CR (carriage return : code ASCII 13 ou \r) et LF ou New Line (line feed : code ASCII 10 ou \n). Pour Unix c'est seulement le caractère LF.

2.1.4 - Affichage d'une Raw resource

Ceci est à comparer avec le paragraphe "Lecture d'un fichier CSV stocké dans le système de fichiers du téléphone" (cf plus loin).

Objectif

Afficher le contenu d'un fichier de type CSV nommé **repertoire.txt** stocké dans **/res/raw/** dans un TextView.



Démarche

Les 4 premières étapes sont réalisées avec l'assistant de création d'une application Android (New Android Application Project).

Création d'un nouveau projet nommé **Ressources**.

Création d'un package nommé **fr.pb.ressources**.

Création d'un layout (**fichier_ressources_lire.xml**) avec un TextView (**textViewFichier**).

Création d'une classe Java (**FichierRessourcesLire.java**) de type Activity et codage.

Création d'un dossier nommé **raw** dans le dossier **/res/** (clic droit / New Folder avec Eclipse. et Clic droit / New / Directory avec Android Studio).

Création du fichier **repertoire.txt** (avec Android Studio Clic droit / New / File).

Note : le contenu du fichier est sous vos yeux ou dans les annexes.

2.1.4.1 - Le layout : fichier_ressources_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:id="@+id/textViewFichier"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

2.1.4.2 - La classe FichierRessourcesLire.java

```
package fr.pb.ressources;

import java.io.*;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

/*
 * Lire les donnees du fichier CSV /res/raw/repertoire.txt
 * Les afficher dans un TextView
 * Au préalable le fichier est créé ou copié/collé dans /res/raw
 */
public class FichierRessourcesLire extends AppCompatActivity {

    private TextView textViewFichier;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fichier_ressources_lire);

        textViewFichier = findViewById(R.id.textViewFichier);

        StringBuilder lsbContenu = new StringBuilder("");
        String lsLigne = "";
        InputStream is;
        InputStreamReader isr;
        BufferedReader br;

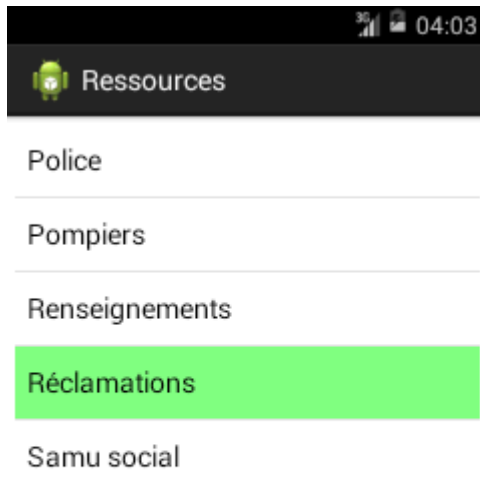
        try {
            // --- Avec un buffer
            is =
getBaseContext().getResources().openRawResource(R.raw.repertoire);
            isr = new InputStreamReader(is);
            br = new BufferedReader(isr);
            while ((lsLigne = br.readLine()) != null) {
                lsbContenu.append(lsLigne);
                lsbContenu.append("\n");
            }
            br.close();
            isr.close();
            is.close();
        }
        catch (Exception e) {
            lsbContenu.append(e.getMessage());
        }
        textViewFichier.setText(lsbContenu.toString());
    } /// on Create
} /// class
```

Par rapport à la lecture d'un fichier stocké dans le File System (cf plus loin) la seule ligne qui change est la ligne en gras d'ouverture du flux.

2.1.5 - Exercice : afficher un numéro de téléphone d'une ressource CSV suite à une sélection dans une liste triée de noms

La liste des noms **triée** est issue du fichier /res/raw/repertoire.txt.

L'utilisateur parcourt la liste, sélectionne un item et le numéro de téléphone correspondant est affiché.



13

cf corrigé dans android_java_exercices_corriges_2.doc

2.2 - XML RESOURCE

2.2.1 - Objectif

Récupérer le contenu – ou une partie - d'un document XML stocké dans les ressources donc dans /res/xml/ .

Utiliser l'API XML Pull Parser - une façade SAX XML - idoine pour cet objectif simple.

2.2.2 - Préambule XML

2.2.2.1 - Types de contenus dans les fichiers XML

- ✓ Données (plus pour les échanges que pour le stockage, cf les API de géolocalisation de Google).
- ✓ Configuration (cf le Manifest.xml).
- ✓ Interfaces (Layouts Android, Form Swing).

2.2.2.2 - Principaux « éléments » XML

Élément	Exemple XML	Exemple HTML
Prologue	<?xml version="1.0" ?>	
Élément vide	<pays id_pays='33' nom_pays='France' />	<input type='text' ... />
Élément « plein »	<pays></pays>	<form></form>
Balise ouvrante	<pays>	<form>
Balise fermante	</pays>	</form>
Balise ouvrante auto-fermée	<capitale nom='Paris' />	
Attribut	<capitale nom='Paris' />	<br class="nettoyeur" />
Élément enfant	<pays> France <capitale nom='Paris' /> </pays>	<form> <input type='text' ... /> </form>
Noeud texte	<pays>France</pays>	<label>Message</label>

2.2.2.3 - Taxinomie (*) des parseurs XML

Parser (analyseur) SAX (Simple API for XML : API Java standard),
Parser (analyseur) DOM (Document Object Model : API Java standard),
Parser (analyseur) xmlpull (API SAX de type façade). cf <http://www.xmlpull.org/> .

Et bien sûr il est possible dans une application d'ajouter d'autres bibliothèques, par exemple JDOM qui est une façade DOM.

Nous allons voir dans ce paragraphe la lecture d'une ressource XML via l'API XML de type **SAX** intégrée à Android (org.xmlpull.v1.XmlPullParser).
Les autres APIs sont aussi intégrées dans le bundle Android.

<http://www.xmlpull.org/v1/doc/api/org/xmlpull/v1/XmlPullParser.html>

Notes :

	SAX	DOM
Accès	Séquentiel	Séquentiel, direct, indexé
Événementiel	Oui	Non
Empreinte mémoire	Faible Chargement ligne à ligne	Forte Chargement de tout l'arbre DOM en mémoire
Sens de la lecture	Forward Only	Scrollable
RO/RW	RO	RW
CRUD	R	CRUD

Typologie des documents XML :

- ✓ Type DATA,
- ✓ Type DOC,
- ✓ Type mixte.

Pour plus de détails cf XML.odt et xml_java.odt.

(*) Taxinomie = Classification

2.2.3 - Syntaxes XPP (XML Pull Parser)

<http://xmllpull.org/>

<https://developer.android.com/reference/org/xmllpull/v1/XmllPullParser.html>

Android contient les packages suivants concernant XML :

android.sax,
android.xml,
android.xml.parsers,
android.w3c.dom,
android.xml.sax,
android.xml.sax.ext,
android.xml.helpers,
android.xmlpull.v1,
android.xmlpull.v1.sax2.

Importation de la classe XmlPullParser pour pouvoir gérer du XML

```
import org.xmlpull.v1.XmlPullParser;
```

Création d'une fabrique de parseurs

```
XmlPullParserFactory fabrique = XmlPullParserFactory.newInstance();
```

Paramétrage de la fabrique (présence d'espace de noms)

```
fabrique.setNamespaceAware(true | false);
```

Création d'un parseur à partir de la fabrique

```
XmlPullParser xpp = fabrique.newPullParser();
```

Affecte un contenu XML au parseur.

```
void xpp.setInput(InputStreamReader);  
ou  
void xpp.setInput(InputStream, Encodage);
```

Pour une ressource située dans /res/xml/ il est inutile de passer par une fabrique.
La méthode getXML() renvoie un XmlResourceParser.
Donc il est préférable de procéder ainsi.

```
XmlPullParser xpp = getResources().getXml(R.xml.ressource);
```

Note : si le fichier .xml est dans /res/raw/ l'exception suivante est levée : File res/raw/fichier.xml from xml type xml resource ID #0x7f050001. Il faudra procéder autrement.

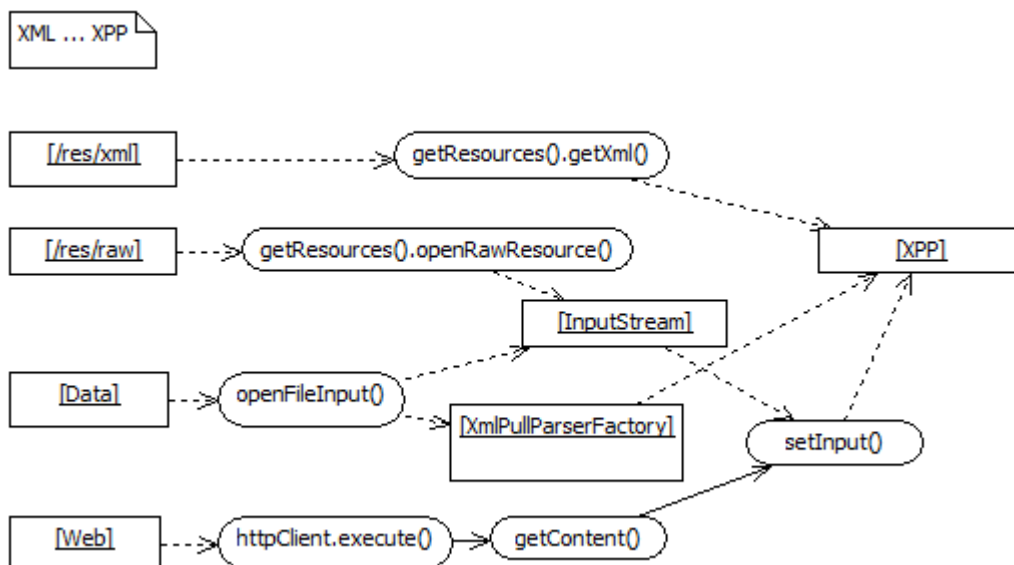
Si le fichier est dans /res/raw/ voilà ce qu'il faut faire ...

```
// --- via ... un InputStream
InputStream is = getResources().openRawResource(R.raw.rss);
XmlPullParserFactory fabrique = XmlPullParserFactory.newInstance();
//fabrique.setNamespaceAware(false); // facultatif
XmlPullParser xpp = fabrique.newPullParser();
InputStreamReader isr = new InputStreamReader(is);
xpp.setInput(isr);
```

ou plus court ...

```
// --- via ... un InputStream
InputStream is = getResources().openRawResource(R.raw.rss);
XmlPullParserFactory fabrique = XmlPullParserFactory.newInstance();
XmlPullParser xpp = fabrique.newPullParser();
xpp.setInput(is, null);
```

Résumé provisoire !



Récupération d'un événement SAX

```
int evenement = xpp.getEventType();
```

Quelques types d'événement :

XmlPullParser.START_DOCUMENT, XmlPullParser.END_DOCUMENT, XmlPullParser.START_TAG, XmlPullParser.END_TAG, XmlPullParser.TEXT.

Boucle jusqu'à la fin du document XML

```
while(xpp.getEventType() != XmlPullParser.END_DOCUMENT)
```

Si une balise ouvrante est trouvée

```
if(xpp.getEventType() == XmlPullParser.START_TAG)
```

Si le nom de la balise ouvrante est ...

```
if(xpp.getName().equals("nom d'élément"))
```

Alors on passe au "fragment" suivant – l'élément qui est de type NodeText - de l'élément et on récupère le texte.

```
xpp.next(); // --- Pour aller sur la feuille #text  
String lsTexte = xpp.getText();
```

cf aussi xpp.nextText() qui renvoie la valeur du nœud texte et passe au suivant.

Pour récupérer la valeur d'un attribut ce sera (n'oubliez pas de tester au préalable l'existence de l'attribut dans la balise) :

```
String lsTexte = xpp.getAttributeValue("NameSpace" | null, "nom de l'attribut");
```

ou

```
String lsTexte = xpp.getAttributeValue(index); // Index commence à 0
```

Test :

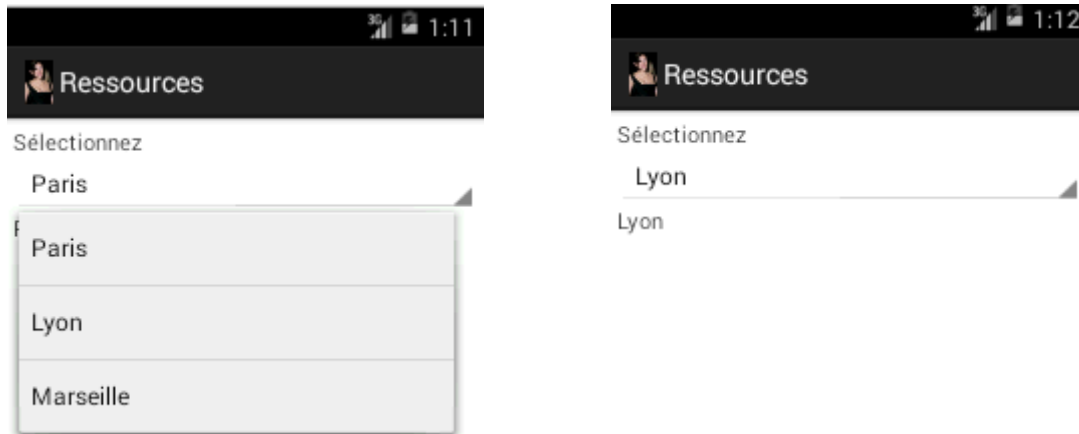
```
String lsValeurAttribut = xpp.getAttributeValue(null, "nom_attribut");  
if (lsValeurAttribut != null) {  
    // Code qui ajoute la valeur quelque part  
}
```

2.2.4 - Affichage d'une ressource XML de type Document dans un spinner

2.2.4.1 - Objectif

Afficher des feuilles #text d'un élément – l'élément <title> en l'occurrence - d'un fichier XML dans un Spinner.

C'est une ressource, elle sera donc en lecture seule.



2.2.4.2 - Contexte

Nous utiliserons le fichier **villes_doc.xml** (cf les annexes pour le contenu).
Puisque c'est une ressource le document XML sera stocké dans le dossier **/res/xml/** de l'application.

2.2.4.3 - Démarche

Le layout et l'activité peuvent être créés en une seule fois avec l'assistant.

Création d'un layout nommé **xml_document_ressources_lire.xml**. Avec un TextView, un spinner et un autre TextView pour l'affichage de la sélection.

Création d'une classe Java de type Activity nommée **XmlDocumentRessourcesLire.java**.

Création d'un dossier **xml** dans **/res/** (Clic droit dans **/res / New / Directory** avec Android Studio).
Création d'un fichier xml nommé **villes_doc.xml**.

Avec Android Studio : Clic droit / New / File

2.2.4.4 - Le layout xml_document_ressources_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélectionnez dans le document ..." />

    <Spinner
        android:id="@+id/spinnerresultats"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélection ..." />

</LinearLayout>
```

2.2.4.5 - XmlDocumentRessourcesLire.java

```
package fr.pb.ressources;

import java.util.ArrayList;
import java.util.List;

import org.xmlpull.v1.XmlPullParser;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;

/*
 * Lit des donnees XML stockees dans /res/xml/villes_doc.xml
 */
public class XmlDocumentRessourcesLire extends AppCompatActivity
implements OnItemClickListener {

    private Spinner spinnerresultats;
    private TextView textViewSelection;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.xml_document_ressources_lire);

        initInterface();

        List<String> listeVilles =
lireRessourceXMLDoc(R.xml.villes_doc, "nom_ville");

        // --- Le spinner avec les résultats
        ArrayAdapter<String> aareultats = new
ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
listeVilles);

aareultats.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
wn_item);

        // --- Affectation de l'ArrayAdapter à la liste du spinner
spinnerresultats.setAdapter(aareultats);

    } // / onCreate()

    // -----
    @Override
    public void onItemClick(AdapterView<?> parent, View v, int
position, long id) {
```



```

textViewSelection.setText(parent.getItemAtPosition(position).toString());
    }

    // -----
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        textViewSelection.setText("");
    }

    /**
     *
     * @param aiRessource
     * @param asBalise
     * @return
     */
    private List<String> lireRessourceXMLDoc(int aiRessource, String
asBalise) {
        // --- On crée une liste
        List<String> liste = new ArrayList();

        try {
            // --- On ouvre le document XML
            XmlPullParser xpp = getResources().getXml(aiRessource);

            // --- Tant que le document n'a pas été analyse jusqu'à
la fin
            while (xpp.getEventType() != XmlPullParser.END_DOCUMENT)
            {
                // --- Si c'est une balise ouvrante
                if (xpp.getEventType() == XmlPullParser.START_TAG)
                {
                    // --- Si la balise s'appelle [XXX]
                    if (xpp.getName().equals(asBalise)) {
                        // --- Pour aller sur le noeud #text
                        liste.add(xpp.nextText());
                    } // / IF balise "XXX" trouvee

                } // / IF start_tag

                // --- On passe a la balise suivante ou element
suivant
xpp.next();

            } // / WHILE parse

        } // / try

        catch (Exception e) {
            liste.add("Erreur" + e.getMessage());
        } // / catch

        return liste;

    } // / lireRessourceXML

    /**
     *
     */
    private void initInterface() {

```

```
// --- On pointe vers le label
textViewSelection = findViewById(R.id.textViewSelection);

// --- On pointe vers la liste deroulante
spinnerresultats = findViewById(R.id.spinnerresultats);

// --- Ajout d'un écouteur a la liste déroulante
spinnerresultats.setOnItemClickListener(this);

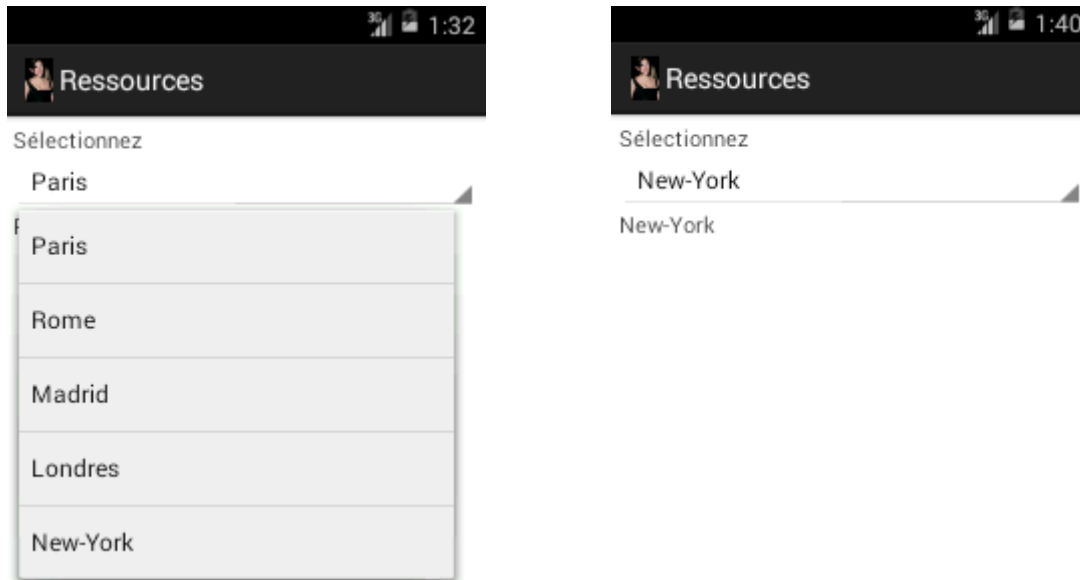
} // / initInterface

} // / Classe XmlDocumentRessourcesLire
```

2.2.5 - Affichage d'une ressource XML de type Data dans un spinner

2.2.5.1 - Objectif

Lire les valeurs d'un attribut.



Note : le contenu du layout est le même que dans l'exemple précédent.

Rappel :

Pour récupérer la valeur d'un attribut ce sera (n'oubliez pas de tester au préalable l'existence de l'attribut dans la balise) :

```
String lsTexte = xpp.getAttributeValue("NameSpace" | null, "nom de l'attribut");
```

ou

```
String lsTexte = xpp.getAttributeValue(index); // Index commence à 0
```

2.2.5.2 - Le fichier xml_data_ressources_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélectionnez dans les data ..." />

    <Spinner
        android:id="@+id/spinnerresultats"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélection ..." />

</LinearLayout>
```

2.2.5.3 - XmlDataRessourcesLire.java

```

package fr.pb.ressources;

import java.util.ArrayList;
import java.util.List;
import org.xmlpull.v1.XmlPullParser;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;

/*
 * Lit des donnees XML stockees dans /res/xml/villes_data.xml
 */
public class XmlDataRessourcesLire extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    private Spinner spinnerresultats;
    private TextView textViewSelection;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.xml_data_ressources_lire);

        initInterface();

        // Méthodes perso pour parser du XML type DATA
        List<String> listeVilles =
lireRessourceXMLData(R.xml.villes_data, "ville", "nom_ville");

        // --- Le spinner avec les résultats
        ArrayAdapter<String> aareultats = new
ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
listeVilles);

aareultats.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
wn_item);

        // --- Affectation de l'ArrayAdapter à la liste du spinner
        spinnerresultats.setAdapter(aareultats);

    } // / onCreate()

    // -----
    public void onItemClick(AdapterView<?> parent, View v, int
position, long id) {

textViewSelection.setText(parent.getItemAtPosition(position).toString());
    }

    // -----

```

```

public void onNothingSelected(AdapterView<?> arg0) {
    textViewSelection.setText("");
}

/**
 *
 * Renvoie une List contenant les valeurs d'un attribut XML type
DATA
 *
 * @param ressource
 * @param asBalise
 * @param asAttribut
 * @return
 */
private List<String> lireRessourceXMLData(int ressource, String
asBalise, String asAttribut) {
    // --- On cree une liste
    List<String> liste = new ArrayList<String>();

    try {
        // --- On ouvre le document XML
        XmlPullParser xpp = getResources().getXml(ressource);

        // --- Tant que le document n'apas ete analyse jusqu'a
la fin
        while (xpp.getEventType() != XmlPullParser.END_DOCUMENT)
        {

            // --- Si c'est une balise ouvrante
            if (xpp.getEventType() == XmlPullParser.START_TAG)
            {

                // --- Si la balise s'appelle ...
                if (xpp.getName().equals(asBalise)
                    && xpp.getAttributeCount() > 0) {
                    // --- Recuperation de la valeur d'un
attribut
                    String lsValeurAttribut =
xpp.getAttributeValue(null, asAttribut);
                    if (lsValeurAttribut != null) {
                        // Code qui ajoute la valeur
                        liste.add(lsValeurAttribut);
                    }

                    } // / IF balise trouvee

                } // / IF start_tag

                // --- On passe a la balise suivante ou element
suivant
                xpp.next();

            } // / WHILE parse

        } // / try

        catch (Exception e) {
            liste.add("Erreur" + e.getMessage());
        } // / catch

        return liste;
    }

```

```
    } // / lireRessourceXML

    // -----
    private void initInterface() {

        // --- On pointe vers le label
        textViewSelection = findViewById(R.id.textViewSelection);

        // --- On pointe vers la liste deroulante
        spinnerresultats = (Spinner)
        findViewById(R.id.spinnerresultats);

        // --- Ajout d'un ecouteur a la liste deroulante
        spinnerresultats.setOnItemClickListener(this);

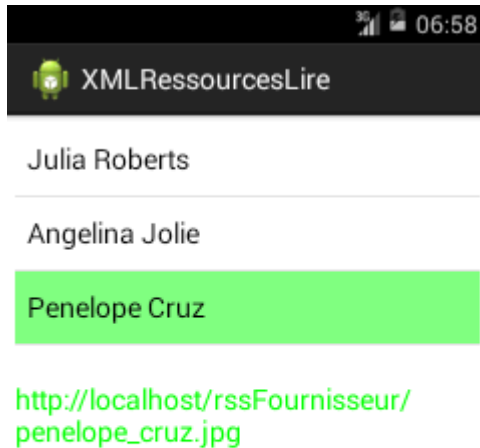
    } // / initInterface
} // / Classe XmlDataRessourcesLire
```

2.2.6 - Exercice : XMLRessourcesLireExo (lire un fil RSS dans une ListView)**Objectif**

Affichez la liste des noms des acteurs dans une ListView.

Suite à une sélection affichez l'URL complète de la photo de la correspondante.

Le fichier XML nommé [rss.xml] est stocké dans les ressources (/res/xml/).



cf corrigé dans android_java_exercices_corriges_2.doc

2.3 - RESSOURCE JSON

2.3.1 - Objectif

Gérer un fichier JSON et plus précisément afficher une liste de villes et récupérer un identifiant.



2.3.2 - Démarche

Utilisation de bibliothèque org.json intégrée à Android.

<https://developer.android.com/reference/org/json/package-summary.html>

Création d'une nouvelle activité nommée JsonRessourcesLire.java et un layout nommé json_ressources_lire.xml.

Création d'une classe nommée JsonUtilitaires avec une méthode nommée jsonIS2JsonArray().
Codage de l'activité.

2.3.3 - Le fichier JSON

Il est stocké dans /res/raw/.

villes.json

```
[
  {
    "cp": "75000",
    "nom": "Paris"
  },
  {
    "cp": "69000",
    "nom": "Lyon"
  },
  {
    "cp": "13000",
    "nom": "Marseille"
  }
]
```

2.3.4 - Le layout json_ressources_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélectionnez dans le JSON ..." />
    <Spinner
        android:id="@+id/spinnerresultats"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélection ..." />
</LinearLayout>
```

2.3.5 - La classe JsonUtilitaires.java

```
package fr.pb.utilitaires;

import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;
import java.io.*;

// -----
public class JSONUtilitaires {

    /**
     * @param is
     * @return
     */
    public static JSONArray jsonIS2JsonArray(InputStream is) throws
JSONException {

        /*
         Renvoie un tableau d'objets JSON
         Le tableau est une "copie" du fichier
         */
        JSONArray tableauJSON = null;

        StringBuilder lsbContenu = new StringBuilder();

        try {
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String lsLigne = "";
            while ((lsLigne = br.readLine()) != null) {
                lsbContenu.append(lsLigne);
                lsbContenu.append("\\n");
            }
            br.close();
            isr.close();
            is.close();

            // Object 2 JSONArray
            tableauJSON = new JSONArray(lsbContenu.toString());

        } catch (JSONException e) {
            JSONObject objetErreur = new JSONObject();
            objetErreur.put("Erreur JSON", objetErreur);
            tableauJSON.put(objetErreur);
        } catch (IOException e) {
            JSONObject objetErreur = new JSONObject();
            objetErreur.put("Erreur IO", objetErreur);
            tableauJSON.put(objetErreur);
        }
        return tableauJSON;
    } /// jsonIS2JsonArray
} /// class
```

2.3.6 - L'activité JsonRessourcesLire.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import java.io.InputStream;
import java.util.*;

import org.json.JSONArray;
import org.json.JSONObject;

import fr.pb.utilitaires.JSONUtilitaires;

/*
 * Lit des données JSON stockées dans /res/raw/villes.json
 * avec la bibliothèque org.json intégrée à Android
 */
public class JsonRessourcesLire extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    private Spinner spinnerresultats;
    private TextView textViewSelection;
    private JSONArray tableauJSON;

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

        initInterface();

        try {
            // Récupération d'un flux
            InputStream is = getResources().openRawResource(R.raw.villes);
            // Méthodes perso pour récupérer du JSON
            tableauJSON = JSONUtilitaires.jsonIS2JsonArray(is);
            List<String> listeNoms = new ArrayList();
            for (int i = 0; i < tableauJSON.length(); i++) {
                JSONObject jsonObject = (JSONObject) tableauJSON.get(i);
                listeNoms.add(jsonObject.get("nom").toString());
            }

            // --- Le spinner avec les résultats
            ArrayAdapter<String> aareultats = new
            ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
            listeNoms);

            aareultats.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
            wn_item);

            // --- Affectation de l'ArrayAdapter à la liste du spinner
            spinnerresultats.setAdapter(aareultats);
        } catch (Exception e) {
            textViewSelection.setText(e.getMessage());
        }
    }
}
```

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
    try {
        JSONObject objetJSON = (JSONObject) tableauJSON.get(position);
        textViewSelection.setText(objetJSON.get("cp").toString());
    } catch (Exception e) {
        textViewSelection.setText(e.getMessage());
    }
} /// onItemSelected

@Override
public void onNothingSelected(AdapterView<?> parent) {
    textViewSelection.setText("");
} /// onNothingSelected

// -----
private void initInterface() {
    // --- On pointe vers le label
    textViewSelection = findViewById(R.id.textViewSelection);
    // --- On pointe vers la liste deroulante
    spinnerresultats = (Spinner) findViewById(R.id.spinnerresultats);
    // --- Ajout d'un ecouteur a la liste deroulante
    spinnerresultats.setOnItemSelectedListener(this);
} // / initInterface
} /// classe
```

CHAPITRE 3 - LES DATA : GESTION DE DONNÉES EN LECTURE/ÉCRITURE ET ÉVENTUELLEMENT PUBLIQUES (CSV, XML, JSON)

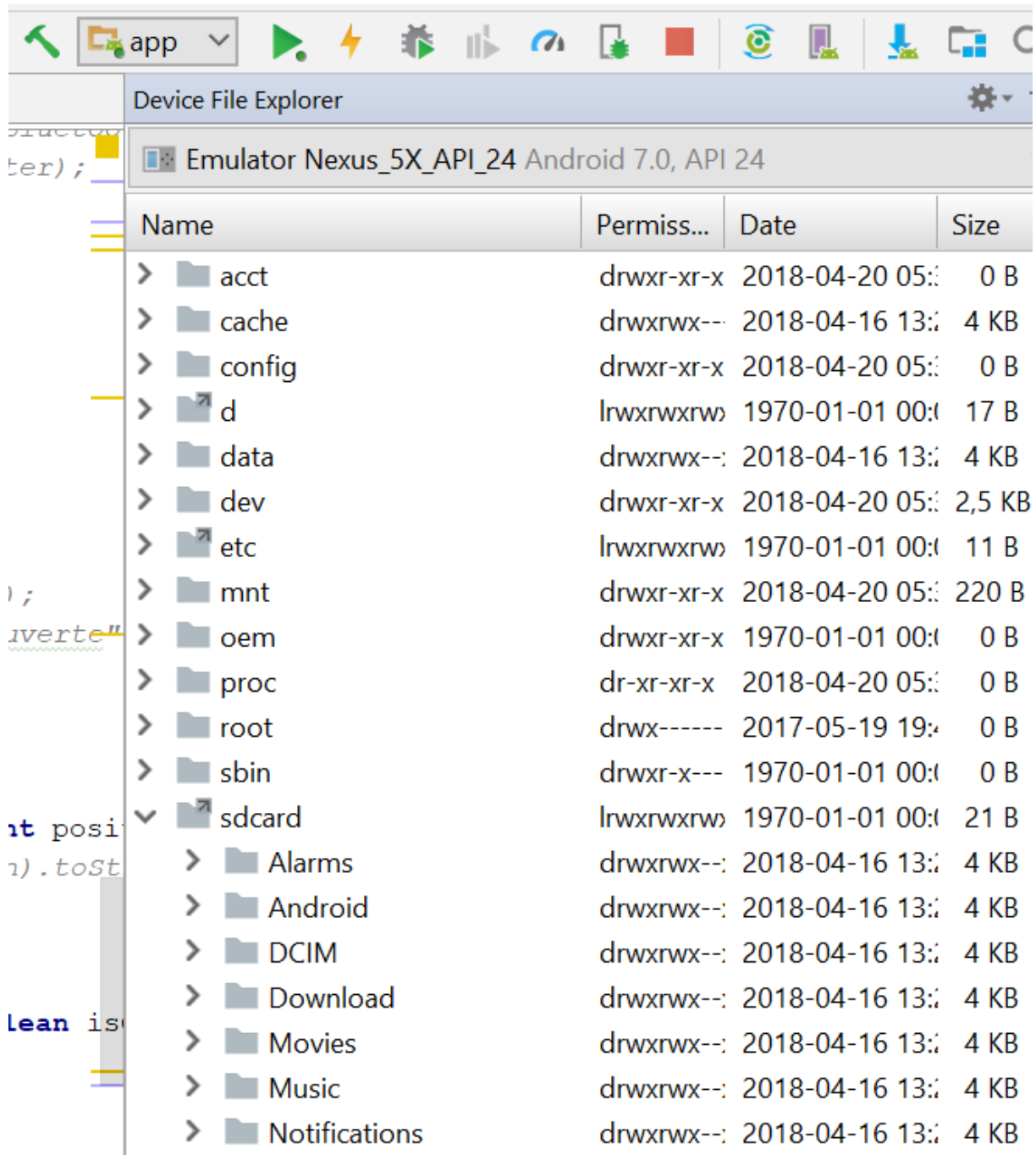
3.1 - OBJECTIF DE CE CHAPITRE

Gérer des données ASCII (de type CSV), XML et JSON en lecture/écriture.

3.2 - LE FILEEXPLORER

AndroidStudio V3

View / Tools Windows / Device File Explorer



Device File Explorer

Emulator Nexus_5X_API_24 Android 7.0, API 24

Name	Permiss...	Date	Size
> acct	drwxr-xr-x	2018-04-20 05:...	0 B
> cache	drwxrwx---	2018-04-16 13:...	4 KB
> config	drwxr-xr-x	2018-04-20 05:...	0 B
> d	lrwxrwxrwx	1970-01-01 00:...	17 B
> data	drwxrwx--:	2018-04-16 13:...	4 KB
> dev	drwxr-xr-x	2018-04-20 05:...	2,5 KB
> etc	lrwxrwxrwx	1970-01-01 00:...	11 B
> mnt	drwxr-xr-x	2018-04-20 05:...	220 B
> oem	drwxr-xr-x	1970-01-01 00:...	0 B
> proc	dr-xr-xr-x	2018-04-20 05:...	0 B
> root	drwx-----	2017-05-19 19:...	0 B
> sbin	drwxr-x---	1970-01-01 00:...	0 B
▼ > sdcard	lrwxrwxrwx	1970-01-01 00:...	21 B
> Alarms	drwxrwx--:	2018-04-16 13:...	4 KB
> Android	drwxrwx--:	2018-04-16 13:...	4 KB
> DCIM	drwxrwx--:	2018-04-16 13:...	4 KB
> Download	drwxrwx--:	2018-04-16 13:...	4 KB
> Movies	drwxrwx--:	2018-04-16 13:...	4 KB
> Music	drwxrwx--:	2018-04-16 13:...	4 KB
> Notifications	drwxrwx--:	2018-04-16 13:...	4 KB

Les données sont stockées dans /data/data/nom.du.package/.
Pour les fichiers, par défaut, le stockage est dans le dossier /data/data/nom.du.package/files/.
Le dossier /files sera créé automatiquement dès que le premier code sera exécuté.

fr.pb.datagestion	2013-12-07	06:14	drwxr-x--x	
cache	2013-12-07	06:14	drwxrwx--x	
files	2013-12-07	06:14	drwxrwx--x	
lib	2013-12-07	06:14	lrwxrwxrwx	-> /data/a...

Android utilise Linux comme OS.

La signification des lettres comme attributs des dossiers ou des fichiers :

La première :

- ✓ rien : file
- ✓ d : directory
- ✓ l : link

Les autres :

ce sont 3 groupes de 3 lettres

- ✓ r : read
- ✓ w : write
- ✓ x : execute qui signifie exécuter pour un exécutable et lister pour un dossier.

Le premier groupe correspond aux droits du propriétaire, le second au groupe auquel appartient le propriétaire et le troisième au reste du monde.

3.3 - QUELQUES NOTIONS SUR LES FICHIERS AVEC ANDROID

La plupart des notions sur les flux en Java sont applicables avec le SDK. Cependant quelques différences existent.

Pour la notion de contexte avec Android cf la page web suivante :
<http://developer.android.com/reference/android/content/Context.html>

Quelques méthodes propres au SDK et rappel de quelques méthodes Java :

Méthode	Description
File dir = Context. getDir (String name, int mode)	Retrouve, ou crée si nécessaire, un nouveau dossier dans lequel votre application peut stocker des fichiers de données. Le dossier est créé au niveau suivant : /data/data/nom.du.package/ Le nom du dossier sera préfixé de app_ !!!
File dir = Context. getFilesDir ()	Renvoie le dossier par défaut de stockage des fichiers. Est positionné dans /data/data/nom.du.package/files
File dir = Environment.getDataDirectory()	Pointe vers /data/
File dir = Environment.getExternalStorageDirectory() (**)	Pour la SD
String = dir.getAbsolutePath()	Renvoie le chemin absolu du type /data/data/nom.du.package/files (sans / à la fin) C'est du java.io, pas du SDK.
String = dir.getCanonicalPath() (*)	Renvoie le chemin absolu du type /data/data/nom.du.package/files (sans / à la fin) C'est du java.io, pas du SDK.
String[] t = dir.list()	Renvoie un tableau de String avec les noms des entrées du dossier
File[] t = dir.listFiles()	Renvoie un tableau de File (les entrées du dossier)
boolean = File.exists()	Teste l'existence du fichier ou du dossier
boolean = File.delete()	Supprime un fichier ou un dossier
boolean = File.renameTo(File)	Renomme un fichier ou un dossier
boolean = File.mkdir()	Crée un dossier

(*) à la différence de JavaSE getAbsolutePath() et getCanonicalPath() d'Android renvoie le même résultat. Avec JavaSE getAbsolutePath() ajoute le File.separator.

	Unix	Windows
File.separator	/	\\
File.pathSeparator	:	;

(**) cf le paragraphe 5.1 pour les permissions.

Les racines !

La racine de /data/data/nom.de.package/files

```
Context.getFilesDir().getAbsolutePath()
```

renvoie ceci par exemple (sans / final) : /data/data/fr.pb.ressources/files

Exemple :

```
Context contexte = getBaseContext();  
String lsCheminCompleet = contexte.getFilesDir().getAbsolutePath() +  
"/1.jpg";
```

La racine de la SD.

```
Environment.getExternalStorageDirectory().getAbsolutePath()
```

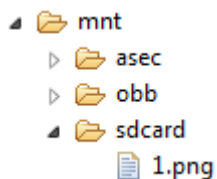
renvoie ceci par exemple (sans / final) : /mnt/sdcard

La SD dans l'explorateur.

Ceci n'est qu'un lien (le l minuscule) :

 sdcard	2014-02-06 04:02	lrwxrwxrwx	-> /mnt/sdcard
--	------------------	------------	----------------

Voilà la SD à explorer :



```
mnt  
├── asec  
├── obb  
└── sdcard  
    └── 1.png
```

3.4 - SYNTAXES POUR TRAVAILLER DANS LE DOSSIER FILES

Écriture

FileOutputStream : OutputStreamWriter : BufferedWriter : write.

```
// --- Ouverture du fichier en écriture : seulement si le fichier est stocké dans le dossier
/data/data/nom.du.package/files/
FileOutputStream fos = getBaseContext().openFileOutput("nomDuFichier", mode);
```

Modes : **MODE_APPEND** (ajout, si le fichier n'existe pas il est créé), **MODE_PRIVATE** (création et privé, si le fichier existe il est supprimé puis recréé), **MODE_WORLD_READABLE** (public en lecture, obsolète à partir de l'API 17), **MODE_WORLD_WRITEABLE** (public en écriture, obsolète à partir de l'API 17).

```
// --- Ecriture en mode String
OutputStreamWriter osw = new OutputStreamWriter(fos);
BufferedWriter bw = new BufferedWriter(osw);
bw.write("chaîne" + "\n");

// --- Ecriture en mode byte
BufferedOutputStream bos = new BufferedOutputStream(fos);
bos.write(byte); // Ou bos.write(byte[]);
```

Lecture

FileInputStream : InputStreamReader : BufferedReader : readLine.

```
// --- Ouverture du fichier en lecture : seulement si le fichier est stocké dans le dossier /data/data/
nom.du.package/files/
FileInputStream fis = getBaseContext().openFileInput("nomDuFichier");
```

```
// --- Lecture en mode String
InputStreamReader isr = new InputStreamReader(fis);
BufferedReader br = new BufferedReader(isr);
```

```
while((IsLigne = br.readLine()) != null) {
    sb.append(IsLigne);
    sb.append("\n");
}
```

```
// --- Lecture en mode byte
BufferedInputStream bis = new BufferedInputStream(fis);
while((octet = br.read()) != -1) {
```

Note : ouverture d'un fichier stocké ailleurs que dans le dossier par défaut (/data/data/nom.du.package/files/). Les méthodes **openFileOutput()** et **openFileInput()** **n'acceptent pas en argument un chemin contenant un séparateur** (en l'occurrence le /). Il faut donc ouvrir le fichier "à la mode Java". Ou voir aussi file:/// et URL et éventuellement Uri.
 URI : un URI (Uniform Resource Identifier, littéralement « identifiant uniforme de ressource ») identifie une ressource sur un réseau.
 URL : une URL (Uniform Resource Locator, littéralement « localisateur uniforme de ressource »), permet d'adresser les ressources du World Wide Web : document HTML, image, son, forum Usenet, boîte aux lettres électronique, etc.

[http://developer.android.com/reference/android/content/Context.html#openFileInput\(java.lang.String\)](http://developer.android.com/reference/android/content/Context.html#openFileInput(java.lang.String))

name The name of the file to open; can not contain path separators.

3.5 - RAPPEL DE TECHNIQUES DE LECTURE ET D'ÉCRITURE CLASSIQUES

Ce sont des codes Java.

Le tout avec les imports nécessaires (java.io.*).

Et les try ... catch ...

Tests d'existence et création de dossier

```
String lsDossier = getFilesDir().getAbsolutePath() + "/" + DOSSIER;
String lsFichier = lsDossier + "/" + FICHIER_TXT;

File dossier = new File(lsDossier);
// Si le dossier n'existe pas
if (!dossier.exists()) {
    dossier.mkdir();
    Toast.makeText(this, "Le dossier a été créé !",
        Toast.LENGTH_SHORT).show();
}

File fichier = new File(lsFichier);
// Si le fichier existe
if (fichier.exists()) {
```

Suppression d'un fichier

Supprime un fichier (il doit exister, donc il faut tester son existence !).

```
boolean File.delete()
```

Supprime un fichier s'il existe.

```
void File.deleteOnExit()
```

Suppression d'un dossier

Le dossier doit être vide.

La syntaxe est identique à celle de la suppression d'un fichier.

Écriture d'un fichier ASCII

FileWriter : BufferedWriter : write.

```
String lsFichier = getFilesDir().getAbsolutePath() + "/" + DOSSIER + "/" +  
FICHIER_TXT;  
FileWriter fw = new FileWriter(lsFichier, true); // true = mode append  
BufferedWriter bw = new BufferedWriter(fw);  
bw.write("Texte\n");  
bw.close();  
fw.close();
```

Le dossier sera en drwx et le fichier en rw.

Lecture d'un fichier ASCII

FileReader : BufferedReader : readLine.

```
String lsFichier = getFilesDir().getAbsolutePath() + "/" + DOSSIER + "/" +  
FICHIER_TXT;  
FileReader fr = new FileReader(lsFichier);  
BufferedReader br = new BufferedReader(fr);  
String lsLigne = br.readLine();  
br.close();  
fr.close();
```

Note : remplacez avantageusement "/" par File.separator !

3.6 - LISTER LE CONTENU D'UN DOSSIER

3.6.1 - Objectif

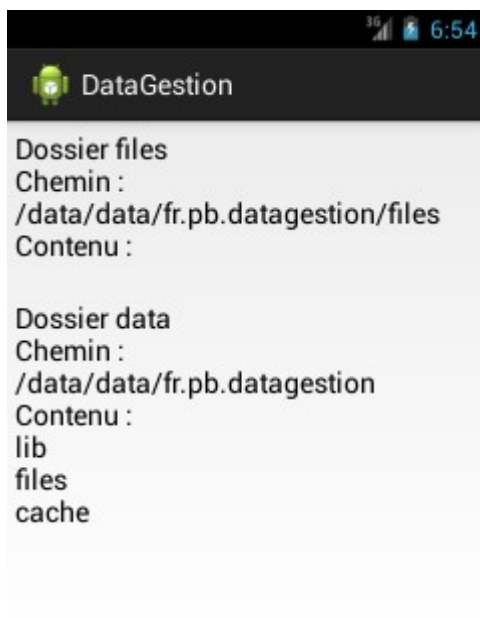
Pour démarrer avec la gestion des fichiers nous allons afficher le contenu de deux dossiers. Comme l'application est lancée pour la première fois les dossiers sont presque vides.

Premier affichage :

chemin et contenu du dossier par défaut de stockage de fichiers (/data/data/nom.du.package/files/).

Deuxième affichage :

chemin et contenu du dossier racine du dossier par défaut de stockage (stockage des fichiers, stockage de BDs (/data/data/nom.du.package/)).



3.6.2 - Layout : data_fichiers_tests.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:id="@+id/textViewContenuFiles"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dossier files"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textViewContenuData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dossier data"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```


3.6.3 - Activité : DataFichiersTests.java

```
package fr.pb.datagestion;

import java.io.*;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

/**
 *
 * @author pascal
 *
 */
public class DataFichierTests extends Activity {
    private TextView textViewContenuFiles;
    private TextView textViewContenuData;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.data_fichiers_tests);

        textViewContenuFiles =
findViewById(R.id.textViewContenuFiles);
        textViewContenuData = findViewById(R.id.textViewContenuData);

        testsDivers();

    } // / onCreate

    // -----
    private void testsDivers() {
        String lsChemin = "";
        String lsCheminAbsoluParent = "";
        Context contexte = getBaseContext();
        StringBuilder lsresultatsFiles = new StringBuilder();
        StringBuilder lsresultatsData = new StringBuilder();
        File dir;
        String[] tFichiers;
        File parent;
        File[] tDossiers;

        try {
            /*
             * Le dossier files donc /data/data/nom.du.package/files/
             */
            // Renvoie /data/data/fr.pb.data.fichier/files
            dir = contexte.getFilesDir();
            lsChemin = dir.getAbsolutePath();
            lsresultatsFiles.append("Dossier files\n");
            lsresultatsFiles.append("Chemin : \n");
            lsresultatsFiles.append(lsChemin);
            lsresultatsFiles.append("\n");
        }
```

```
// Liste les fichiers
tFichiers = dir.list();
lsbrésultatsFiles.append("Contenu : \n");
for (int i = 0; i < tFichiers.length; i++) {
    lsbrésultatsFiles.append(tFichiers[i]);
    lsbrésultatsFiles.append("\n");
}

/*
 * Le dossier parent donc /data/data/nom.du.package
 */
parent = dir.getParentFile();

// Renvoie /data/data/fr.pb.data.fichier
lsCheminAbsoluParent = parent.getAbsolutePath();
lsbrésultatsData.append("Dossier data\n");
lsbrésultatsData.append("Chemin : \n");
lsbrésultatsData.append(lsCheminAbsoluParent);
lsbrésultatsData.append("\n");

lsbrésultatsData.append("Contenu : \n");
tDossiers = parent.listFiles();
for (File file : tDossiers) {
    lsbrésultatsData.append(file.getName());
    lsbrésultatsData.append("\n");
}

} catch (Exception e) {
    lsbrésultatsFiles.append(e.getMessage());
    lsbrésultatsData.append(e.getMessage());
} finally {

} // / finally

// Affichages finals
textViewContenuFiles.setText(lsbrésultatsFiles.toString());
textViewContenuData.setText(lsbrésultatsData.toString());

} // / testsDivers

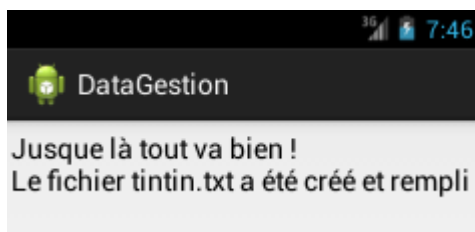
} // / class DataFichierTests
```

3.7 - CRÉATION D'UN FICHIER ASCII STOCKÉ DANS /DATA/DATA/N.D.P/FILES/

3.7.1 - Objectif

Créer un fichier ASCII nommé tintin.txt avec 3 enregistrements à l'intérieur.
Le fichier est stocké dans le dossier /data/data/nom.du.package/files/.
Si le fichier existe, il est supprimé puis recréé.
Un message est affiché.

3.7.2 - Écran



3.7.3 - Layout

data_fichier_ecrire.xml

Juste un TextView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:id="@+id/textViewMessageEcrire"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ecrire"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

3.7.4 - Activité

DataFichierEcrire.java

Puisque le fichier est stocké dans /data/data/nom.du.package/files/ on utilise les classes IO et les méthodes IO suivantes :

FileOutputStream
 OutputStreamWriter
 BufferedWriter

getBaseContext().openFileOutput(chemin, mode)
 write(String)
 close()

```
import java.io.*;
import android.content.Context;

import android.os.Bundle;
import android.widget.TextView;
import android.app.Activity;

// -----
public class DataFichierEcrire extends Activity {

    private TextView textViewMessageEcrire;
    private final String FICHIER_TXT = "tintin.txt";

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

        String lsContenu = "";
        lsContenu += "Tintin\n";
        lsContenu += "Milou\n";
        lsContenu += "Haddock\n";

        textViewMessageEcrire =
        findViewById(R.id.textViewMessageEcrire);

        textViewMessageEcrire.setText(ecrire(this, FICHIER_TXT,
        lsContenu));

    } // / onCreate

    /**
     *
     * @param contexte
     * @param psFichier
     * @param psContenu
     * @return
     */
    private String ecrire(Context contexte, String psFichier, String
    psContenu) {

        String lsMessage = "";
        FileOutputStream fos;
        OutputStreamWriter osw;
```

```
        BufferedWriter bw;

        try {
            fos = contexte.openFileOutput(psFichier,
Context.MODE_PRIVATE);
            osw = new OutputStreamWriter(fos);
            bw = new BufferedWriter(osw);
            bw.write(psContenu);

            bw.close();
            osw.close();
            fos.close();
            lsMessage = "Jusque là tout va bien !\nLe fichier " +
FICHIER_TXT + " a été créé et rempli";
        } catch (Exception e) {
            lsMessage = e.getMessage();
        }

        return lsMessage;

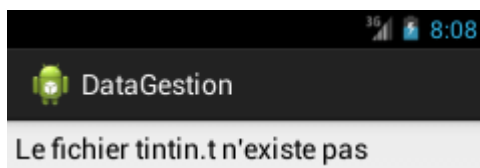
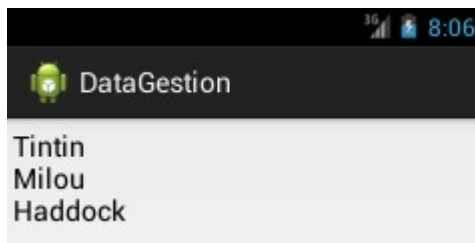
    } // / écrire
} // / DataFichierEcrire
```

3.8 - LECTURE DU CONTENU D'UN FICHIER ASCII STOCKÉ DANS DATA/DATA/N.D.P/FILES/

3.8.1 - Objectif

Afficher le contenu d'un fichier nommé tintin.txt stocké dans /data/data/nom.du.package/files/ dans un TextView.
Le test d'existence du fichier est réalisé.

3.8.2 - Écran



3.8.3 - Layout

data_fichier_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:id="@+id/textViewLecture"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ecrire"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

3.8.4 - Activité

DataFichierLire.java

Puisque le fichier est stocké dans /data/data/nom.du.package/files/ on utilise les classes IO et les méthodes IO suivantes :

FileInputStream
InputStreamReader
BufferedReader

getBaseContext().openFileInput(chemin)
readLine()
close()

```
import java.io.*;

import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;
import android.support.v7.app.AppCompatActivity;

// -----
public class DataFichierLire extends AppCompatActivity {

    private TextView textViewLecture;
    private final String FICHIER_TXT = "tintin.t";

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

        textViewLecture = findViewById(R.id.textViewLecture);

        textViewLecture.setText(lire(this, FICHIER_TXT));

    } // / onCreate

    /**
     *
     * @param contexte
     * @param psFichier
     * @return
     */
    private String lire(Context contexte, String psFichier) {

        File f;
        String lsChemin = contexte.getFilesDir().getAbsolutePath() +
        "/" + psFichier;
        FileInputStream fis;
        InputStreamReader isr;
        BufferedReader br;
        StringBuilder lsb = new StringBuilder();
        String lsLigne = "";

        try {
            f = new File(lsChemin);
            if (f.exists()) {
```

```
        fis = contexte.openFileInput(psFichier);
        isr = new InputStreamReader(fis);
        br = new BufferedReader(isr);

        while ((lsLigne = br.readLine()) != null) {
            if(lsLigne.trim().length() > 0) {
                lsb.append(lsLigne);
                lsb.append("\n");
            }
        }

        br.close();
        isr.close();
        fis.close();
    } else {
        lsb.append("Le fichier ");
        lsb.append(psFichier);
        lsb.append(" n'existe pas");
    }
} catch (FileNotFoundException e) {
    lsb.append(e.getMessage());
} catch (IOException e) {
    lsb.append(e.getMessage());
}
return lsb.toString();
} // / lire
} // / DataFichierLire
```


3.9 - GESTION D'UN FICHIER CSV STOCKÉ DANS LE TÉLÉPHONE

3.9.1 - Objectif

Gérer le fichier suivant : **repertoire.csv**

```
Nom;tel  
Tintin;+331010101  
Milou;+33606060606
```

Alors que précédemment le fichier CSV était stocké dans les ressources de l'application et utilisable uniquement par cette application en lecture seule, ici les données sont en lecture/écriture et potentiellement accessibles à d'autres applications (si les droits sont pour le groupe, mais surtout pour tout le mode -rw ou -r-).

Note : pour une lecture publique il serait préférable de créer un ContentProvider et autant de ContentResolver que nécessaire.

3.9.2 - Conception

1) Stocker des données dans le terminal dans le **dossier /data/data/...**

Les données seront potentiellement accessibles en lecture/écriture par toutes les applications.

Stocker des noms et des numéros de téléphone dans un fichier CSV nommé repertoire.csv.

A la différence des ressources (cf sections précédentes pour les strings, les colors, ..., les données raw ou les données xml) les données seront en lecture/écriture et potentiellement utilisables par plusieurs applications.




Si le fichier n'existe pas il est automatiquement créé. Non seulement le fichier est automatiquement créé mais si le dossier /data/data/nom.du.package/files n'existe pas ce dossier est créé.




2) Visualiser le contenu du fichier.

3) Ajouter un enregistrement dans le fichier.

Dans un deuxième temps vous gérerez un fichier ou plusieurs stockés dans le dossier **/data/data/nom.du.package/files/bandes_dessinees**

3) Interface de saisie et d'affichage

RepertoireCSV	
Dossier :	<input type="text"/>
Fichier:	<input type="text" value="repertoire.csv"/>
Nom :	<input type="text" value="Milou"/>
Téléphone :	<input type="text" value="+33602030405"/>
<div>  </div>	
Tintin;+33102030405	
Milou;+33602030405	

RepertoireCSV	
Dossier :	<input type="text" value="bandes_dessinees"/>
Fichier:	<input type="text" value="repertoire.txt"/>
Nom :	<input type="text" value="Milou"/>
Téléphone :	<input type="text" value="+33602030405"/>
<div>  </div>	
Tintin;+33102030405	
Milou;+331234567890	



Pour afficher les enregistrements dans la ListView.



Pour ajouter un enregistrement.



Pour supprimer un enregistrement (ce sera le sujet de l'exercice).

Démarche

Création d'un **layout** nommé **repertoire_csv.xml**.
Création d'une classe **ListActivity** nommée **RepertoireCSV.java**.
Création d'une classe **utilitaire** nommée **Fichier.java**.

Eventuellement :

Le fichier **repertoire.csv** est créé avec un éditeur de texte, sur c:/ par exemple, puis importé via le File Explorer dans le dossier /data/data/n.d.p/files/.
Vous devez lancer au moins une fois l'application pour que le dossier soit créé et accessible.

Par rapport à la lecture d'un fichier stocké dans /res/raw/ (cf plus haut) la seule ligne qui change est la ligne en gras.

Pour les ressources c'était :

```
InputStream is =  
getContext().getResources().openRawResource(R.raw.repertoire);  
InputStreamReader isr = new InputStreamReader(is);  
...
```

Pour les "data", s'il est situé dans /data/data/nom.du.package/files, ce sera :

```
InputStream is = getContext().openFileInput("repertoire.csv");  
InputStreamReader isr = new InputStreamReader(is);  
...
```

3.9.3 - Le layout repertoire_csv.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="Dossier : " />

        <EditText
            android:id="@+id/editTextDossier"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="bandes_dessinees" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="Fichier: " />

        <EditText
            android:id="@+id/editTextFichier"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="repertoire.csv" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="Nom : " />
```

```
<EditText
    android:id="@+id/editTextNom"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:inputType="textPersonName"
    android:text="Tintin" >

    <requestFocus />
</EditText>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:text="Téléphone : " />

    <EditText
        android:id="@+id/editTextTelephone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:inputType="phone"
        android:text="+33102030405" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageButton
        android:id="@+id/imageButtonGet"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:contentDescription="Get"
        android:src="@android:drawable/ic_input_get" />

    <ImageButton
        android:id="@+id/imageButtonAdd"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:contentDescription="Add"
        android:src="@android:drawable/ic_input_add" />

    <ImageButton
        android:id="@+id/imageButtonDelete"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:contentDescription="Delete"
        android:src="@android:drawable/ic_input_delete" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textViewMessage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Message" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <ListView
            android:id="@android:id/list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >
        </ListView>
    </LinearLayout>

</LinearLayout>
```

3.9.4 - La diagramme de la classe Fichier.java (pour la gestion des données, le modèle)

Fichier
contexte
setContexte(Context contexte) : void
ouvrirDataInput(String nomFichier) : String
ouvrirDataOutput(String nomFichier) : String
fermerInput() : String
fermerOutput() : String
ouvrirRawInput(int ressource) : String
ajouter(String asEnr) : String
lire() : String

La gestion du fichier est en mode octet pour pouvoir être utilisée de façon plus générique; la lecture peut être en mode ligne; par exemple pour lire un fichier stocké comme ressource. La méthode ouvrirRessourceInput() sera utilisée dans ce cas-là.

L'ajout d'un enregistrement est réalisé via un OutputStream.

A chaque ajout le fichier est ouvert, l'enregistrement est écrit puis le fichier est fermé.

Le fichier est ouvert en mode Append ie si le fichier existe, on ajoute l'enregistrement, si le fichier n'existe pas il est créé et on ajoute l'enregistrement.

Le fichier est lu via un InputStream.

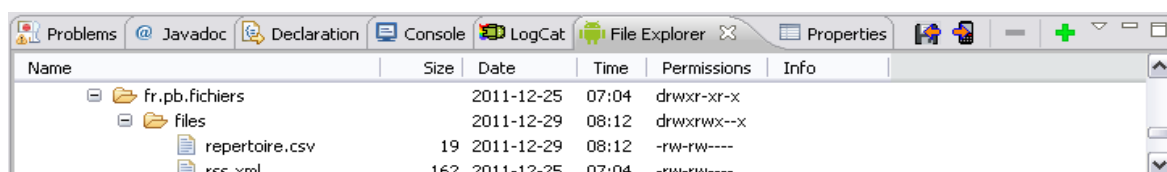
Où est caché le fichier ?

Dans le dossier /data/data/fr.pb.fichiers/files

Cf Travailler avec adb (Android Debug Bridge) dans les annexes.

<http://developer.android.com/guide/topics/data/data-storage.html#filesInternal>


et aussi le File Explorer d'Eclipse (data/data/ ...).



Le bouton  permet de créer un dossier.

Le bouton  permet de supprimer un fichier (Pas un dossier).

Le bouton  permet d'exporter un fichier.

Le bouton  permet d'importer un fichier.

Pour le stocker en dehors de l'application :

<http://developer.android.com/guide/topics/data/data-storage.html#filesExternal>

3.9.5 - Le diagramme de la classe RepertoireCSV

La classe **RepertoireCSV.java** (l'activité pour la gestion de la vue)

RepertoireCSV
isCheminFiles isCheminDossier isCheminFichier isEnrSelectionne editTextDossier editTextFichier editTextNom editTextTelephone imageButtonAdd imageButtonGet imageButtonDelete textViewMessage
onCreate(Bundle savedInstanceState) init() ajouterEnregistrement() supprimerEnregistrement() lireEnregistrements() onClick() onListItemClick()

La classe implémente l'interface `OnClickListener` (pour les clics des boutons).

Un objet `Fichier` est instancié dans chaque méthode de manipulation du fichier.

3.9.6 - Le code de la classe Fichier.java (classe utilitaire)

La gestion des fichiers avec Android diffère peu de la gestion classique en Java. L'ouverture du fichier diffère avec la méthode `openFileInput()` de la classe `Context` pour la lecture et la méthode `openFileOutput()` de la classe `Context` pour l'écriture.

```
package fr.pb.data.fichier;

import java.io.InputStream;
import java.io.OutputStream;

import android.content.Context;

// -----
public class Fichier {
// -----

    // --- Attributs
    private Context contexte;

    private OutputStream os;
    private InputStream is;

    // --- Méthodess
    // -----
    public void setContexte(Context contexte) {
        this.contexte = contexte;
    } /// setContexte

    // -----
    public String ouvrirDataInput(String nomFichier) {
        String lsMessage = "";

        try {
            // --- C'est la specificite
            this.is = this.contexte.openFileInput(nomFichier);
            lsMessage = "OK";
        }
        catch(Exception e) {
            lsMessage = e.getMessage();
        }
        return lsMessage;
    } /// ouvrirDataInput

    // -----
    public String fermerInput() {
        String lsMessage = "";

        try {
            this.is.close();
            lsMessage = "OK";
        }
        catch(Exception e) {
            lsMessage = e.getMessage();
        }
    }
}
```

```
    }
    return lsMessage;
} /// fermerInput

// -----
public String ouvrirDataOutput(String nomFichier) {
    String lsMessage = "";

    try {
        this.os = this.contexte.openFileOutput(nomFichier,
Context.MODE_APPEND); // MODE_PRIVATE par default
        lsMessage = "OK";
    }
    catch(Exception e) {
        lsMessage = e.getMessage();
    }
    return lsMessage;
} /// ouvrirDataOutput

// -----
public String fermerOutput() {
    String lsMessage = "";

    try {
        this.os.close();
        lsMessage = "OK";
    }
    catch(Exception e) {
        lsMessage = e.getMessage();
    }
    return lsMessage;
} /// fermerOutput

// ----- Pour les fichiers situés dans /res/raw/
public String ouvrirRawInput(int ressource) {
    String lsMessage = "";

    try {
        // --- C'est la specificite
        this.is =
this.contexte.getResources().openRawResource(ressource);
        lsMessage = "OK";
    }
    catch(Exception e) {
        lsMessage = e.getMessage();
    }
    return lsMessage;
}

// --- Ajoute un enregistrement octet par octet : renvoie un message
public String ajouter(String asEnr) {
    String lsMessage = "";
```

```
        try {
            for(int i = 0; i < asEnr.length(); i++) {
                char c = asEnr.charAt(i);
                this.os.write((byte)c);
            }
            this.os.write(10); // --- On est sur Unix ... pas de 13
donc
            this.os.flush();
            lsMessage = "Enregistrement ajouté";
        }
        catch (Exception e) {
            lsMessage = "Erreur : " + e.getMessage();
        }

        return lsMessage;
    } /// ajouter()

    // --- Lit le contenu du fichier octet par octet : renvoie une
String
    public String lire() {
        StringBuilder sbContenu = new StringBuilder("");

        try {
            int liValeur = 0;
            while((liValeur = this.is.read()) != -1) {
                sbContenu.append((char)liValeur);
            }
        }
        catch (Exception e) {
            sbContenu.append("Erreur : ");
            sbContenu.append(e.getMessage());
        }

        return sbContenu.toString();
    } /// lire()
} /// de la classe
```

3.9.7 - Le code de la classe RepertoireCSV.java (l'activité)

```

package fr.pb.data.fichier;

import android.app.ListActivity;
import android.os.Bundle;

import android.content.Context;
import android.view.View;

import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

/*
 * Gere le fichier (Ajout, visualisation) repertoire.csv stockee dans
 /data/data/fr.pb.data.fichier/files/
 * ou un autre d'ailleurs si on saisit un autre nom de fichier
 */

// -----
public class RepertoireCSV extends ListActivity {
// -----

    private EditText editTextFichier;
    private EditText editTextNom;
    private EditText editTextTel;

    private Button boutonAjouter;
    private Button boutonVoir;

    private TextView textViewMessage;

    private Context contexte;
    private Fichier fichier;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        // -----

        super.onCreate(savedInstanceState);
        setContentView(R.layout.repertoire_csv);

        // -----
        // --- Initialisation de l'interface statique
        // -----
        // --- Liaison "variables" et composants du layout
        editTextFichier = findViewById(R.id.editTextFichier);
        editTextNom      = findViewById(R.id.editTextNom);
        editTextTel      = findViewById(R.id.editTextTel);

        boutonAjouter = findViewById(R.id.boutonAjouter);
        boutonVoir     = findViewById(R.id.boutonVoir);

        textViewMessage = findViewById(R.id.textViewMessage);

        // -----

```

```
// --- Initialisation des variables de programme
// -----
contexte = getBaseContext();
fichier = new Fichier();
fichier.setContexte(contexte);

/*
 * Les procedures evenementielles
 */
// ----- AJOUTER
buttonAjouter.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        // --- Recuperation des nom et tel saisis
        // --- Fichier dans /data/data/

        fichier.ouvrirDataOutput(editTextFichier.getText().toString());

textViewMessage.setText(fichier.ajouter(editTextNom.getText().toString() +
";" + editTextTel.getText().toString()));
        fichier.fermerOutput();
    } /// de onClick

}); /// setOnClick de buttonAjouter

// ----- VOIR
buttonVoir.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        editTextNom.setText("");
        editTextTel.setText("");
        textViewMessage.setText("");

        // --- Fichier stocke dans /data/data/ en RW

        fichier.ouvrirDataInput(editTextFichier.getText().toString());
        String lsContenu = fichier.lire();
        // Remplir la ListView


        fichier.fermerInput();
    } /// de onClick

}); /// setOnClick de buttonVoir

} /// onCreate
} /// classe RepertoireCSV
```

3.9.8 - Exercice : supprimer un enregistrement d'un fichier CSV

L'utilisateur sélectionne un nom et appuie sur le bouton pour supprimer l'enregistrement.

L'interface est la même. Il faut coder le bouton Supprimer .

[cf corrigé dans android_java_exercices_corrige_2.doc](#)

3.10 - SOUS-DOSSIER DE FILES

3.10.1 - Objectif

Créer un sous dossier dans /data/data/n.d.p/files.

Créer un fichier ou plusieurs dans ce sous dossier.

Lister le contenu d'un dossier autre que /data/data/n.d.p/files/ !!!
Et surtout le contenu d'un fichier d'un de ses dossiers.

Afficher le contenu d'un fichier du sous-dossier.

Note : ouverture d'un fichier stocké ailleurs que dans le dossier par défaut (/data/data/nom.du.package/files/). Les méthodes **openFileOutput()** et **openFileInput()** **n'acceptent pas en argument un chemin contenant un séparateur** (en l'occurrence le /). Il faut donc ouvrir le fichier "à la mode Java". Ou voir aussi [file:///](#) et URL et éventuellement Uri.
Cf paragraphe 5.3.

3.10.2 - Démarche

Création d'un nouveau dossier nommé [bandes_dessinees] dans /data/data/nom.du.package/files/ avec DDMS/File Explorer.
Importation des fichiers tintin.txt, pieds_nickeles.txt, daltons.txt (cf les contenus des fichiers dans les annexes).
Importation des images associées !

3.10.3 - Layout

Le même que précédemment.

3.10.4 - Activité

Tests d'existence du sous dossier nommé « txt » et du fichier nommé « partage.txt ».

S'ils n'existent pas on les crée.

S'ils existent on affiche le contenu du fichier.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.*;

import java.io.*;
import java.net.URI;

public class DataFileSubFolder extends AppCompatActivity {

    private TextView textViewMessage;
    private final String SUB_FOLDER = "txt";
    private final String TXT_FILE = "partage.txt";

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

        textViewMessage = findViewById(R.id.textViewMessage);

        StringBuilder lsb = new StringBuilder("Read in Sub Folder\n");
        try {
            File fileDataFiles = getFilesDir();

            File subFolder = new File(fileDataFiles.getAbsolutePath() +
            "/" + SUB_FOLDER);
            if (!subFolder.exists()) {
                boolean ok = subFolder.mkdir();
                if (ok) {
                    File fileInSubFolder = new
                    File(fileDataFiles.getAbsolutePath() + "/" + SUB_FOLDER + "/" + TXT_FILE);
                    if (!fileInSubFolder.exists()) {
                        FileWriter fw = new FileWriter(fileInSubFolder);
                        fw.write("Partage1\nPartage2\nPartage3");
                        fw.close();
                        lsb.append("Jusque là tout va bien !\nLe fichier "
+ TXT_FILE + " a été créé et rempli\n");
                    }
                }
            }

            URI uriFichier = new URI("file://" +
            fileDataFiles.getAbsolutePath() + "/" + SUB_FOLDER + "/" + TXT_FILE);
            File file = new File(uriFichier);
            if (file.exists()) {
                lsb.append("Le fichier existe\n");
                FileReader fr = new FileReader(file);
                BufferedReader br = new BufferedReader(fr);
                String lsLigne;
                while ((lsLigne = br.readLine()) != null) {
                    lsb.append(lsLigne);
                }
            }
        } catch (IOException e) {
            Log.e("DataFileSubFolder", e.getMessage());
        }

        textViewMessage.setText(lsb.toString());
    }
}
```



```
        lsb.append("\n");
    }
    br.close();
    fr.close();
} else {
    lsb.append("Le fichier n'existe pas\n");
}
} catch (Exception e) {
    lsb.append(e.getMessage());
}

textViewMessage.setText(lsb.toString());
} /// onCreate
} /// class
```

CHAPITRE 4 - FICHER ET SD

4.1 - LECTURE D'UN FICHIER XML STOCKÉ DANS DU STORAGE

Storage : soit /data/data/n.d.p/files/ soit la SD.

Pour les syntaxes cf 2.2.2.

Rien de bien particulier sauf l'obtention de l'InputStream et la création du parseur.

Ouverture du fichier rss.xml stocké dans /data/data/nom.du.package/files/ et pas ailleurs (`openFileInput()` n'accepte pas les `File.separator`).

Après avoir instanciée une Factory et un parseur vous récupérez le contenu dans un flux.

```
InputStream is = getBaseContext().openFileInput("nom_du_document.xml");
```

Création d'un parseur à partir d'un InputStream.

```
XmlPullParser.setInput(InputStream, InputEncoding);
```

```
package fr.pb.data.xml;

import java.io.InputStream;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

/**
 *
 * Visualise les <title> de rss.xml stocke dans /data/data/...
 *
 * avec XmlPullParser
 *
 * @author pascal
 */
public class VisualiserFluxRssXpp extends Activity {

    private static final String FICHER = "rss.xml";
    private static final String BALISE = "title";

    private TextView textViewSelection;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
```

```

        setContentView(R.layout.visualiser_flux_rss_xpp);

        textViewSelection = findViewById(R.id.textViewSelection);

        StringBuilder lsbContenu = new StringBuilder("");

        // --- Mode XmlPullParser
        try {
            // --- Fabrique de XmlPullParser
            XmlPullParserFactory xppf =
XmlPullParserFactory.newInstance();
            // --- Creation d'un XmlPullParser
            XmlPullParser xpp = xppf.newPullParser();
            // --- Ouverture du fichier rss.xml
            // --- stocke dans /data/data/n.d.p/files/ et pas
ailleurs
            InputStream is = getBaseContext().openFileInput(FICHIER);
            // --- xpp.setInput(InputStream, InputEncoding);
            xpp.setInput(is, null);

            // --- Tant que le document n'a pas ete analyse jusqu'a
fin
            while (xpp.getEventType() != XmlPullParser.END_DOCUMENT)
            {
                // --- Si c'est une balise ouvrante
                if (xpp.getEventType() == XmlPullParser.START_TAG)
                {
                    // --- Recuperation d'un #text de balise
                    if (xpp.getName().equals(BALISE)) {
                        xpp.next(); // --- Pour aller sur noeud
#text
                        lsbContenu.append(xpp.getText());
                        lsbContenu.append("\n");
                    } // / IF balise trouvee
                } // / IF START_TAG

                // --- On passe au noeud
                xpp.next();
            } // / WHILE parse

            textViewSelection.setText(lsbContenu.toString());
        } catch (Exception e) {
            textViewSelection.setText("Erreur : " + e.getMessage());
        }

    } // / onCreate()
} // / class VisualiserFluxRssXpp

```

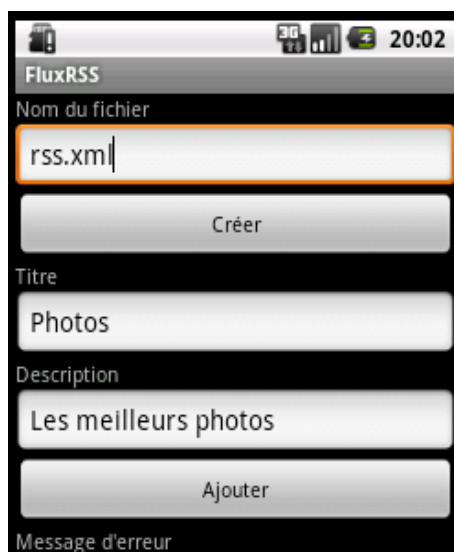
4.2 - L'APPLICATION RSS (XML) : CRÉER ET AJOUTER UN FIL

4.2.1 - Objectif

Stocker, dans /data/data/fr.pb.data.xml/, des fils RSS dans un fichier XML nommé **rss.xml** (cf un exemple de fichier à la page suivante).

Première étape : création du fichier rss.xml avec le prologue et les éléments <rss> et <channel> en mode fichier.

Deuxième étape : ajout d'un élément <title> et d'un élément <description> en mode DOM (avec **JDOM** qui est une façade pour faciliter l'utilisation de l'**API DOM** de Java – org.w3c.dom -).



Note :

il va falloir télécharger les bibliothèques **jaxen.jar** et **jdom-1.1.2.jar**.

4.2.2 - Démarche

Création d'un layout nommé **creation_flux_rss.xml** avec :

- Un TextView,
- Un EditText (etFichierXML),
- Un button (btCreerXML),
- Un TextView,
- Un EditText (etTitre),
- Un TextView,
- Un EditText (etDescription),
- Un button (btAjouterElements),
- Un TextView (tvMessageXML).

Création d'une classe Activity nommé **CreationFluxRSS.java**.

Ajout des bibliothèques **jaxen-x.x.x.jar** et **jdom-x.x.x.jar**.

Qui ont été téléchargées à : <http://www.jdom.org/downloads/>

Dézipper le fichier jdom-2.0.6.zip.

Copiez les fichiers jdom-2.0.6.jar et jaxen-1.1.6.jar dans le dossier /libs de votre application.

4.2.3 - Le fichier XML : rss.xml

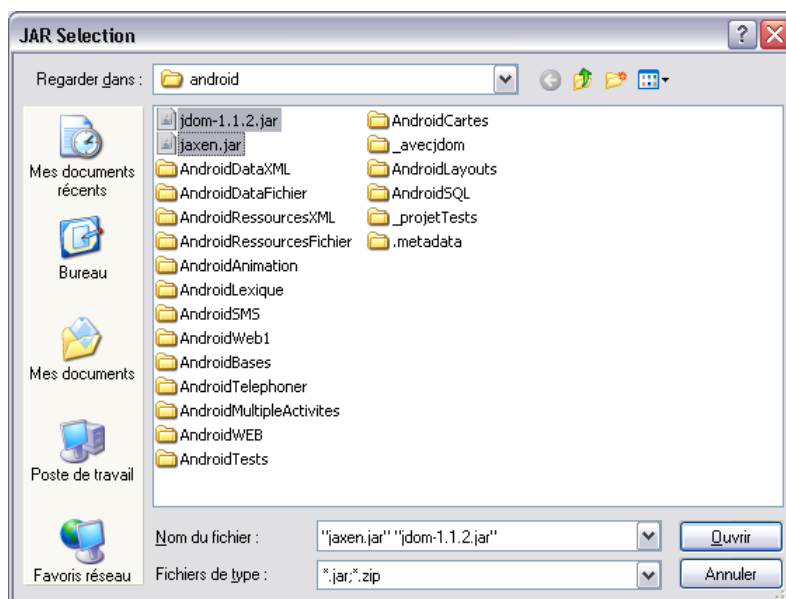
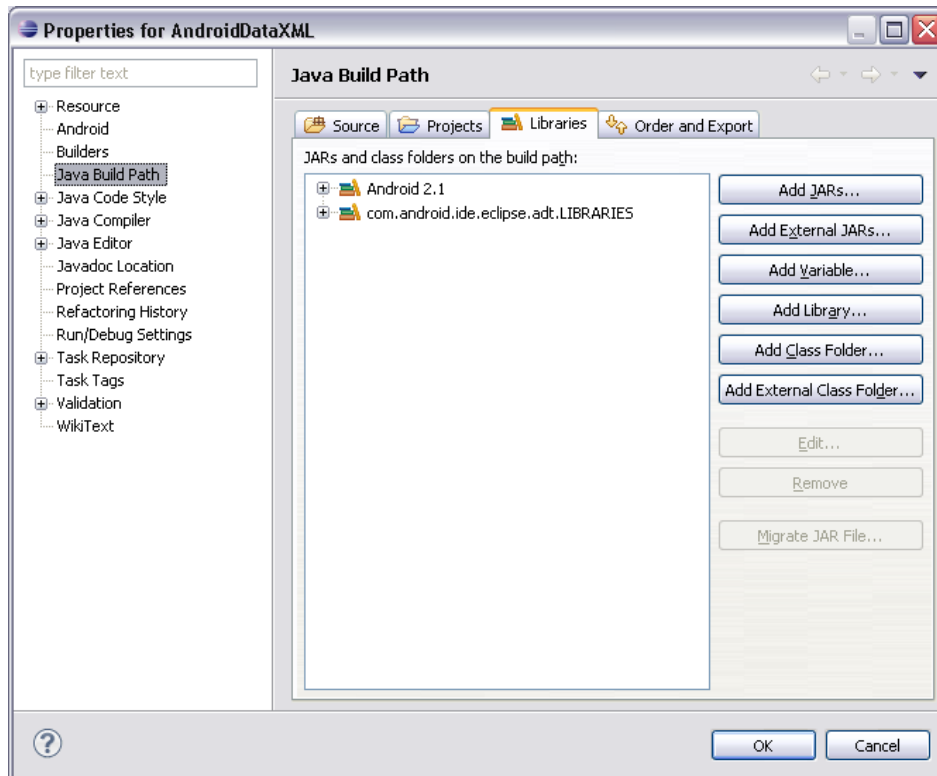
cf les annexes.

4.2.4 - L'ajout des bibliothèques jdom-1.1.2.jar et jaxen.jar

Copiez les fichiers jdom-x.x.x.jar et jaxen-x.x.x.jar dans le dossier /libs.

@obsolete !!!

Projet/Clic droit/Propriétés/Java Build Path/Add External JARs/



4.2.5 - Syntaxes IO

Fonctionnalité	Syntaxe
Ouvre un flux en écriture vers un fichier stocké dans /data/data/nom.du.package/files. Si le fichier n'existe pas il est créé.	FileOutputStream fos = getBaseContext(). openFileOutput ("nom du fichier", Context.MODE); Les modes Context.MODE_PRIVATE, Context.MODE_APPEND, Context.MODE_WORLD_READABLE, Context.MODE_WORLD_WRITEABLE.
Ouvre un OutputStreamWriter à partir d'un FileOutputStream pour écrire des caractères (pas des bytes).	OutputStreamWriter osw = new OutputStreamWriter(fos);
Ecrit les caractères à partir d'une chaîne.	osw.write(IsContenu);
Ouvre un flux en lecture vers un fichier stocké dans /data/data/nom.du.package/files.	FileInputStream fis = getBaseContext(). openFileInput ("nom de fichier");
Supprime un fichier stocké dans /data/data/nom.du.package/files.	getBaseContext().deleteFile("nom de fichier");
Ferme un flux, un reader, un writer, un buffer.	objet.close();

4.2.6 - Syntaxes JDOM

Les imports :

pour JDOM 1.x.x

```
import org.jdom.Document;  
import org.jdom.Element;  
import org.jdom.input.SAXBuilder;  
import org.jdom.output.Format;  
import org.jdom.output.XMLOutputter;
```

pour JDOM 2.x.x

```
import org.jdom2.Document;  
import org.jdom2.Element;  
import org.jdom2.input.SAXBuilder;  
import org.jdom2.output.Format;  
import org.jdom2.output.XMLOutputter;
```

Fonctionnalité	Syntaxe
Instancie un parseur SAX.	<code>SAXBuilder sxb = new SAXBuilder();</code>
Charge le flux en lecture dans un arbre DOM.	<code>Document arbreDom = sxb.build(fis);</code>
Récupère la racine du DOM.	<code>Element racine = arbreDom.getRootElement();</code>
Crée un élément DOM.	<code>Element élément = new Element("nom de l'élément DOM");</code>
Affecte un texte à l'élément.	<code>element.setText("texte");</code>
Ajoute un élément à un autre.	<code>racine.addContent(element);</code>
Crée la sortie.	<code>XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());</code>
Ecrit l'arbre DOM dans le flux de sortie.	<code>sortie.output(arbreDom, FileOutputStream);</code>

4.2.7 - Le layout : creation_flux_rss.xml

creation_flux_rss.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nom du fichier" />

    <EditText
        android:id="@+id/etFichierXML"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="rss.xml" />

    <Button
        android:id="@+id/btCreerXML"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="CréerXML" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Titre" />

    <EditText
        android:id="@+id/etTitre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Photos" >

        <requestFocus />
    </EditText>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Description" />

    <EditText
        android:id="@+id/etDescription"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Les meilleurs photos" />

    <Button
        android:id="@+id/btAjouterElements"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ajouter éléments" />
```

```
<TextView
    android:id="@+id/tvMessageXML"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Message d'erreur" />

</LinearLayout>
```

4.2.8 - L'activité : CreationFluxRSS.java

CreationFluxRSS.java

```
package fr.pb.data.xml;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

/*
 * Creation d'un fichier rss.xml stocke dans /data/data/...
 * Ajout d'elements avec JDOM
 */

//-----
public class CreationFluxRSS extends Activity {
//-----

    private EditText etFichierXML;
    private Button btCreerXML;

    private EditText etTitre;
    private EditText etDescription;
    private Button btAjouterElements;

    private TextView tvMessageXML;

    private Context contexte;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        // -----
        super.onCreate(savedInstanceState);
        setContentView(R.layout.creation_flux_rss);

        // -----
        // --- Initialisation de l'interface statique
        // -----
        // --- Liaison "variables" et composants du layout
        etFichierXML = findViewById(R.id.etFichierXML);
        btCreerXML = findViewById(R.id.btCreerXML);
    }
}
```

```
etTitre = findViewById(R.id.etTitre);
etDescription = findViewById(R.id.etDescription);
btAjouterElements = findViewById(R.id.btAjouterElements);

tvMessageXML = findViewById(R.id.tvMessageXML);

contexte = getBaseContext();

/*
 * Les procedures evenementielles
 */

// --- CREER
btCreerXML.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        String lsContenu = "";
        lsContenu += "<?xml version='1.0' encoding='utf-8' ?>\n";
        lsContenu += "<rss version='2.0'>\n";
        lsContenu += "\t<channel>\n";
        lsContenu += "\t</channel>\n";
        lsContenu += "</rss>";

        FileOutputStream fos = null;
        OutputStreamWriter osw = null;

        try {
            fos =
contexte.openFileOutput(etFichierXML.getText().toString(),
MODE_WORLD_WRITEABLE);
            osw = new OutputStreamWriter(fos);
            osw.write(lsContenu);
            osw.flush();
            osw.close();
            fos.close();
            tvMessageXML.setText("Fichier créé");
        } /// TRY

        catch (Exception e) {
            tvMessageXML.setText("Erreur : " + e.getMessage());
        }

        } /// de onClick

}); /// de setOnClickListener de btCreer

// --- AJOUTER
btAjouterElements.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

        // --- Les flux
        FileInputStream fis = null;
        FileOutputStream fos = null;

        try {
            // --- Creation des elements :

            SAXBuilder sxb = new SAXBuilder();
```

```
Document arbreDom = null;

// --- Ouvre le fichier XML et le place dans un
FileInputStream
fis =
contexte.openFileInput(etFichierXML.getText().toString());
// --- Charge le contenu du fichier dans le DOM
arbreDom = sxb.build(fis);
// --- Ferme le flux
fis.close();

// --- Recupere la racine du DOM
Element racine = arbreDom.getRootElement();
Element channel = racine.getChild("channel");

// --- Cree un element
Element titre = new Element("title");
titre.setText(etTitre.getText().toString());
Element description = new Element("description");

description.setText(etDescription.getText().toString());

// --- Ajoute des elements a <channel>
channel.addContent(titre);
channel.addContent(description);

// --- Supprime le fichier autrement il ajoute au lieu
de recreer

contexte.deleteFile(etFichierXML.getText().toString());

// --- Fermeture ou Output (dom2fichier)
// --- Ecriture sur le DD dans le document XML
// --- du contenu de l'arbre DOM qui est en RAM
// --- Outputs a JDOM document as a stream of bytes.
XMLOutputter sortie = new
XMLOutputter(Format.getPrettyFormat());
// --- Ouvre le fichier en mode creation/append
fos =
contexte.openFileOutput(etFichierXML.getText().toString(),
Context.MODE_APPEND);
// --- Ecriture sur le disque
sortie.output(arbreDom, fos);
fos.close();
tvMessageXML.setText("Élément ajouté");
} /// TRY

catch (Exception e) {
    tvMessageXML.setText("Erreur : " + e.getMessage());
}

} /// de onClick

}); /// de setOnClick de boutonAjouter

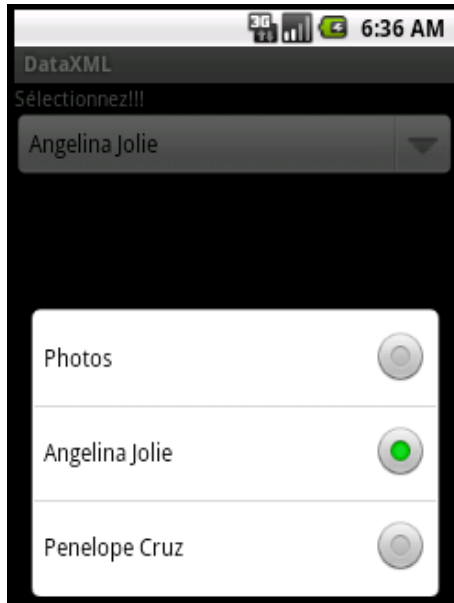
} /// onCreate()

} /// classe
```

4.2.9 - Exercice : visualiser les <title> du fichier rss.xml dans un Spinner

Objectif

Visualiser les <title> du fichier rss.xml dans une liste déroulante (spinner) en vue d'une suppression d'un élément.



[cf corrigé dans android_java_exercices_corriges_2.doc](#)

4.3 - LECTURE D'UN FICHIER JSON STOCKÉ DANS DU STORAGE

4.3.1 - Objectif

Afficher dans un Spinner les noms des pays et récupérer sur une sélection le code ISO2 du pays.



Storage : soit /data/data/n.d.p/files/ soit la SD.

Pour les syntaxes cf 2.3.x.

Rien de bien particulier sauf l'obtention de l'InputStream et la création du parseur.

Ouverture du fichier `pays.json` stocké dans `/data/data/nom.du.package/files/` et pas ailleurs (`openFileInput()` n'accepte pas les `File.separator`).

Codage de l'activité.

Réutilisation de la méthode `jsonIS2JsonArray()` de la classe `JsonUtilitaires`.

4.3.2 - json_data_lire.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélectionnez dans le JSON ..." />
    <Spinner
        android:id="@+id/spinnerresultats"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sélection ..." />
</LinearLayout>
```

4.3.3 - JsonDataLire.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.*;

import java.io.InputStream;
import java.util.*;

import org.json.JSONArray;
import org.json.JSONObject;

import fr.pb.utilitaires.JSONUtilitaires;

public class JsonDataLire extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    private Spinner spinnerresultats;
    private TextView textViewSelection;
    private JSONArray tableauJSON;

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

        initInterface();

        try {
            // Récupération d'un flux
            InputStream is = null;
            is = getBaseContext().openFileInput("pays.json");

            // Méthodes perso pour récupérer du JSON
            tableauJSON = JSONUtilitaires.jsonIS2JsonArray(is);
            List<String> listeNoms = new ArrayList<>();
            for (int i = 0; i < tableauJSON.length(); i++) {
                JSONObject jsonObject = (JSONObject) tableauJSON.get(i);
                listeNoms.add(jsonObject.get("nom").toString());
            }

            // --- Le spinner avec les résultats
            ArrayAdapter<String> aareultats = new
            ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
            listeNoms);

            aareultats.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
            wn_item);

            // --- Affectation de l'ArrayAdapter à la liste du spinner
            spinnerresultats.setAdapter(aareultats);
        } catch (Exception e) {
            textViewSelection.setText(e.getMessage());
        }
    } // onCreate
```

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
    try {
        JSONObject objetJSON = (JSONObject) tableauJSON.get(position);
        textViewSelection.setText(objetJSON.get("iso2").toString());
    } catch (Exception e) {
        textViewSelection.setText(e.getMessage());
    }
} /// onItemSelected

@Override
public void onNothingSelected(AdapterView<?> parent) {
    textViewSelection.setText("");
} /// onNothingSelected

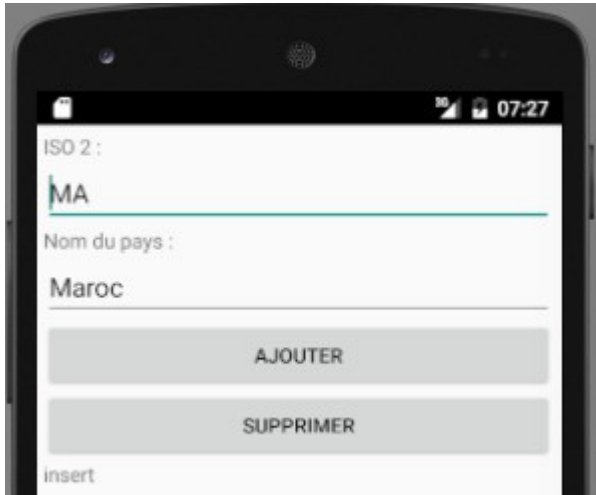
// -----
private void initInterface() {
    // --- On pointe vers le label
    textViewSelection = findViewById(R.id.textViewSelection);
    // --- On pointe vers la liste deroulante
    spinnerresultats = (Spinner) findViewById(R.id.spinnerresultats);
    // --- Ajout d'un ecouteur a la liste deroulante
    spinnerresultats.setOnItemSelectedListener(this);
} // / initInterface

} /// class
```

4.4 - CRUD SUR UN FICHIER JSON STOCKÉ DANS DU STORAGE

4.4.1 - Objectif

Ajouter un objet JSON dans le fichier pays.json.



Storage : soit /data/data/n.d.p/files/ soit la SD.

4.4.2 - Le layout json_crud.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ISO 2 :" />

    <EditText
        android:id="@+id/editTextISO2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nom du pays :" />

    <EditText
        android:id="@+id/editTextNomPays"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/buttonAjouter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ajouter" />

    <Button
        android:id="@+id/buttonSupprimer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Supprimer" />

    <TextView
        android:id="@+id/textViewMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message ..." />

</LinearLayout>
```

4.4.3 - La classe JSONUtilitaires.java

Enfin les méthodes insert() et delete() !

A vous de compléter !

4.4.4 - L'activité JsonCRUD.java

A vous de compléter !

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import org.json.JSONException;
import org.json.JSONObject;

import fr.pb.utilitaires.JSONUtilitaires;

public class JsonCRUD extends Activity implements View.OnClickListener {

    private EditText editTextISO2;
    private EditText editTextNomPays;
    private Button boutonAjouter;
    private Button boutonSupprimer;
    private TextView textViewMessage;

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

        initInterface();

        /*
        En test
        */
        editTextISO2.setText("MA");
        editTextNomPays.setText("Maroc");

    } /// onCreate

    /**
     *
     */
    private void initInterface() {
        editTextISO2 = findViewById(R.id.editTextISO2);
        editTextNomPays = findViewById(R.id.editTextNomPays);

        boutonAjouter = findViewById(R.id.boutonAjouter);
        boutonAjouter.setOnClickListener(this);

        boutonSupprimer = findViewById(R.id.boutonSupprimer);
        boutonSupprimer.setOnClickListener(this);

        textViewMessage = findViewById(R.id.textViewMessage);
        textViewMessage.setText("");
    } /// initInterface
```

```
@Override
public void onClick(View v) {

    // INSERT
    if (v == buttonAjouter) {
        String iso2 = editTextISO2.getText().toString();
        String nomPays = editTextNomPays.getText().toString();
        // A COMPLETER
    } /// buttonAjouter

    // DELETE
    if (v == buttonSupprimer) {
        String iso2 = editTextISO2.getText().toString();
        String nomPays = editTextISO2.getText().toString();
        // A COMPLETER
    } /// buttonSupprimer
    } /// onClick
} /// class
```


CHAPITRE 5 - LES PREFERENCES

5.1 - INTRODUCTION

5.1.1 - Généralités

Les préférences correspondent à certains choix de l'utilisateur faits soit au niveau d'une application soit au niveau du système. Elles sont sous forme de paires clé/valeur. Elles sont stockées physiquement sur le disque sous forme de fichiers XML. Cela rappelle les préférences des utilisateurs dans le monde web qui sont stockées dans des cookies.

Les valeurs qui sont stockées dans les préférences pourraient l'être dans des fichiers ou des BD gérés par vous-même ; l'avantage du système de Android des préférences est la gestion du CRUD XML est transparente pour le développeur, ce qui évite de long code en SAX et/ou DOM. Sans parler des problèmes de gérer le CRUD si vous choisissiez des fichiers CSV. De plus la gestion des préférences au niveau système ce révèle très lourd sans passer par la bibliothèque Android.

La gestion des préférences peut être réalisée de façon personnalisée ou en utilisant le framework Android.

Il est possible de gérer des préférences de niveau activité ou de niveau application comme de niveau système.

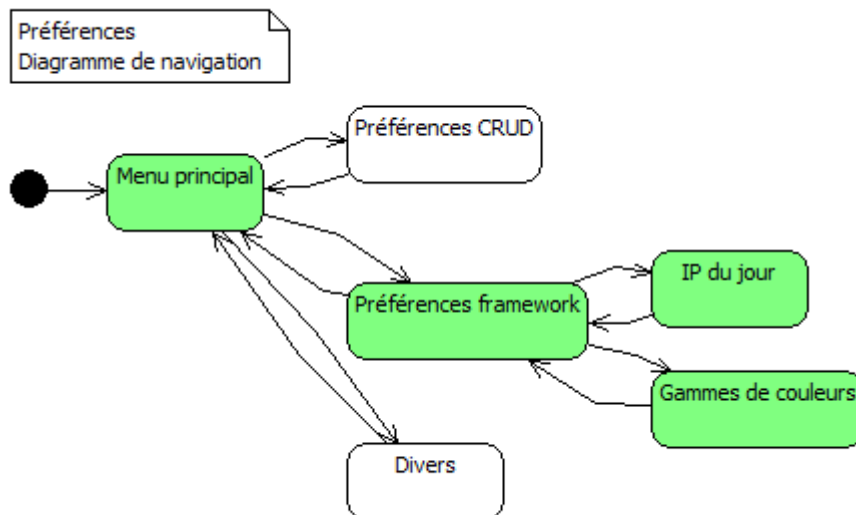
C'est l'objet de ce chapitre que de présenter ces modes de gestion des préférences.

5.1.2 - L'écran d'accueil

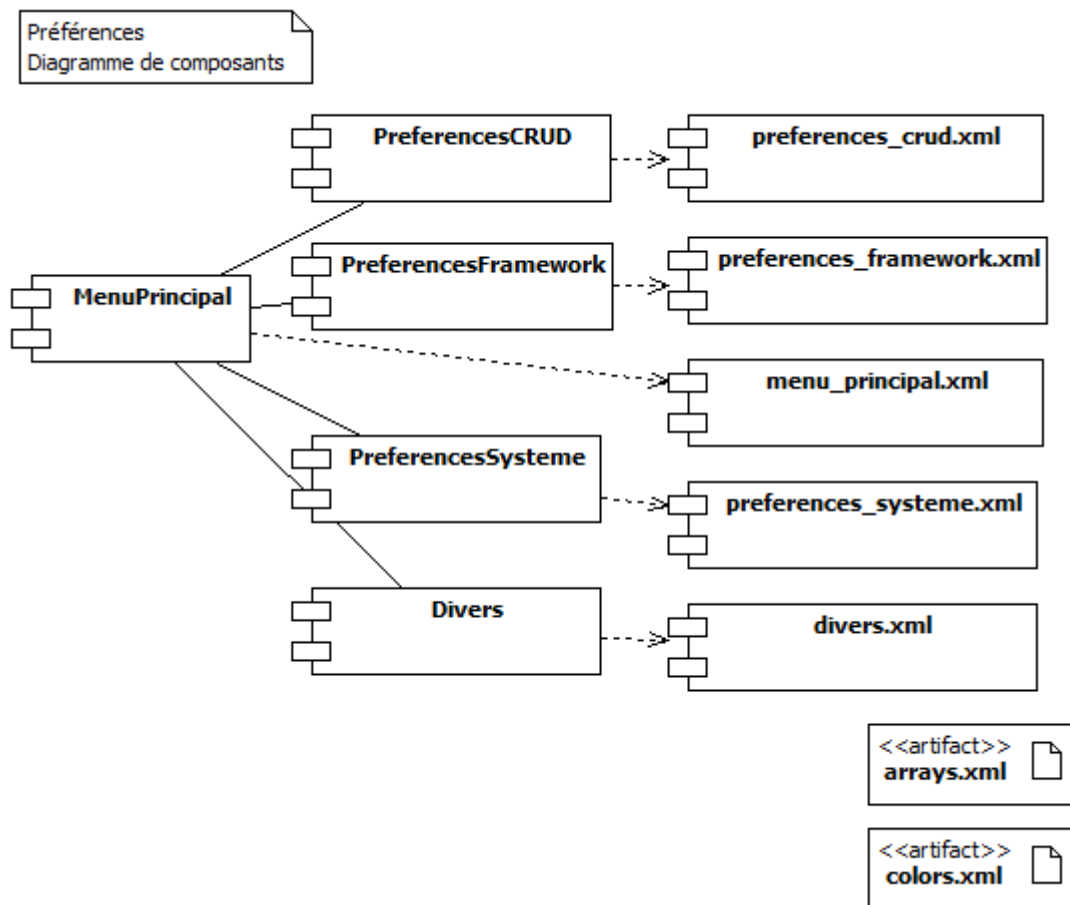


5.1.3 - Diagrammes

5.1.3.1 - Diagramme de navigation



5.1.3.2 - Diagramme de composants



5.1.4 - Les fichiers XML ... généraux

5.1.4.1 - arrays.xml stocké dans /res/values/

Les différents tableaux pour le menu principal et pour la ListView des préférences.

Le premier tableau est utilisé pour remplir la ListView du menu principal.

Les deux autres tableaux sont utilisés pour remplir la liste des gammes de couleurs de la ListView du framework. La première pour l'affichage et la deuxième pour les valeurs à stocker dans les préférences.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <array name="items_menu_principal">
        <item>Préférences (Sans Framework)</item>
        <item>Préférences réseau et couleurs (Framework)</item>
        <item>Divers</item>
    </array>

    <array name="liste_couleurs_fr">
        <item>Bleu</item>
        <item>Rouge</item>
        <item>Vert</item>
    </array>

    <array name="liste_couleurs">
        <item>blue</item>
        <item>red</item>
        <item>green</item>
    </array>

</resources>
```

5.1.4.2 - colors.xml stocké dans /res/values/

Ces données serviront pour coloriser les écrans suite aux modifications des préférences utilisateur.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <color name="red">#FF0000</color>
    <color name="pink">#FF99FF</color>
    <color name="green">#00FF00</color>
    <color name="light_green">#99FF99</color>
    <color name="blue">#0000FF</color>
    <color name="light_blue">#0099FF</color>
    <color name="white">#FFFFFF</color>
    <color name="black">#000000</color>

</resources>
```

5.1.4.3 - menu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="90" >
    </ListView>

    <TextView
        android:id="@+id/textViewMessage"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:background="@color/pink"
        android:text="Message"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/red" />

</LinearLayout>
```

5.2 - LA GESTION PERSONNALISÉE DES PRÉFÉRENCES

5.2.1 - Présentation

Le système des préférences permet de gérer des données sous forme de paires clé/valeur. Ce sont des données permanentes. Les types de données sont des types primitifs (seulement les 4 suivant : int, long, float, boolean) ou des Strings.

Par défaut les préférences sont partageables par toutes les activités de l'application.

Exemple de fichier de préférences : autres_preferences.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="IP">10.3.16.217</string>
</map>
```

Le fichier est stocké dans /data/data/nom.du.package/shared_prefs/.

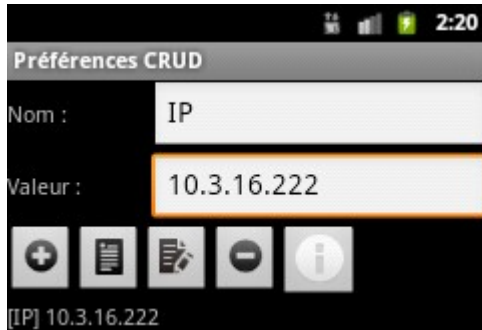
▲	fr.pb.preferences		2013-06-20	06:08	drwxr-x--x
▶	lib		2013-06-20	06:08	drwxr-xr-x
▲	shared_prefs		2013-06-20	03:42	drwxrwx--x
	PreferencesCRUD.xml	107	2013-06-20	03:42	-rw-rw----
	autres_preferences.xml	145	2013-06-20	02:40	-rw-rw-r--
	fr.pb.preferences_preferences.xml	200	2013-06-20	03:03	-rw-rw----

Le nom du fichier de préférences dépend de son mode de création :

- ✓ nom de l'activité qui crée les préférences (préférence non partageable),
- ✓ nom donné si l'on utilise la méthode getSharedPreferences(),
- ✓ nom.du.package_preferences.xml si celui-ci a été créé avec le framework Android.

5.2.2 - Objectif et écran

Gérer (CRUD) une préférence de l'utilisateur (son IP) via une interface personnalisée.



Note : les quatre premiers boutons représente le CRUD.

5.2.3 - Syntaxes

La création d'une instance de préférences.

Il faut créer une instance de préférences pour pouvoir ensuite travailler sur une préférence (faire du CRUD).

Récupérer une instance de préférences propre à l'activité

```
SharedPreferences sp = getPreferences(MODE);
```

C'est une méthode de la classe Activity.

MODE : Context.MODE_PRIVATE, Context.MODE_WORLD_READABLE,

Context.MODE_WORLD_WRITEABLE, Context.MODE_MULTI_PROCESS.

Les modes barrés sont obsolètes depuis l'API 17 (Jelly Bean version 4.2).

Le nom du fichier de stockage des préférences est NomDeClasse.xml.

Récupérer une instance de préférences partagées (entre les activités de l'application)

```
SharedPreferences sp = getSharedPreferences("nom_du_fichier", MODE);
```

Le nom du fichier de stockage des préférences est nom_du_fichier.xml.

L'extension .xml est ajoutée automatiquement par Android au nom du fichier.

C'est une méthode de la classe abstraite Context.

Récupérer une instance de préférences par défaut de l'application (références partagées entre les activités de l'application).

```
SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(this);
```

Le nom du fichier de stockage des préférences est nom_du_package_preferences.xml.

La mise à jour des préférences.

Il faut créer un éditeur de préférences pour pouvoir les créer et/ou les modifier et/ou les supprimer

```
SharedPreferences.Editor editeur = sp.edit();
```

Ajouter ou modifier une préférence

```
editeur.putXXX("preference", valeur);
```

putXXX() : putString(), putInt(), putLong(), putFloat(), putBoolean().

Valider les préférences

```
editeur.commit();
```

Supprimer toutes les préférences

```
editeur.clear();
```

Supprimer une préférence

```
editeur.remove("clé");
```

La récupération des valeurs des préférences.

Récupérer toutes les préférences. La clé est nécessairement une String cependant la valeur peut être une String ou un type primitif d'où le ? Pour gérer la généricité.

```
Map<String, ?> prefs = sp.getAll();
```

Récupérer la valeur d'une préférence

```
String pref = sp.getString("preference", "valeur par défaut");
```

Il existe aussi les méthodes `getInt()`, `getLong()`, `getFloat()`, `getBoolean()`.

Pour aller plus loin :

cf <http://developer.android.com/reference/android/preference/package-summary.html>

Le package `android.preference` fournit des classes qui gèrent les préférences et implémentent des préférences UI (sous forme de `ListView`, de cases à cocher, etc). Cf la section 4.3.

5.2.4 - Codes

5.2.4.1 - preferences_crud.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="30"
            android:text="@string/nom"
            android:textColor="@color/black" />

        <EditText
            android:id="@+id/editTextNomPreference"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="70"
            android:textColor="@color/black" >

            <requestFocus />
        </EditText>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="30"
            android:text="@string/valeur"
            android:textColor="@color/black" />

        <EditText
            android:id="@+id/editTextValeurPreference"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="70"
            android:textColor="@color/black" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:orientation="horizontal" >

        <ImageButton
            android:id="@+id/buttonAjouter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/ajouter"
            android:src="@drawable/insert_noir_24_24" />

        <ImageButton
            android:id="@+id/buttonVoir"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/voir"
            android:src="@drawable/select_noir_24_24" />

        <ImageButton
            android:id="@+id/buttonModifier"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/modifier"
            android:src="@drawable/update_noir_24_24" />

        <ImageButton
            android:id="@+id/buttonSupprimer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/supprimer"
            android:src="@drawable/delete_noir_24_24" />

        <ImageButton
            android:id="@+id/buttonAide"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/aide"
            android:src="@android:drawable/ic_dialog_info" />
    </LinearLayout>

    <TextView
        android:id="@+id/textViewPreferences"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black" />

</LinearLayout>
```

5.2.4.2 - PreferencesCRUD.java

```
package fr.pb.preferences;

import java.util.Map;
import java.util.Set;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

/*
 * L'utilisation des SharedPreferences
 */
public class PreferencesCRUD extends Activity implements OnClickListener {

    private EditText editTextNomPreference;
    private EditText editTextValeurPreference;
    private ImageButton buttonAjouter;
    private ImageButton buttonModifier;
    private ImageButton buttonSupprimer;
    private ImageButton buttonVoir;
    private ImageButton buttonAide;
    private TextView textViewPreferences;

    private SharedPreferences sp;
    private SharedPreferences.Editor editeur;

    // -----
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.preferences_crud);

        initInterface();

        // Pour tests
        editTextNomPreference.setText("IPDuJour");
        editTextValeurPreference.setText("192.168.0.1");

        // Cree des preferences avec comme nom : NomDeClasse.xml
        // Disponible pour cette seule activite
        // sp = getPreferences(Context.MODE_PRIVATE);

        // Cree un fichier crud.xml
        // Partageable entre activites
        sp = getSharedPreferences("crud", Context.MODE_PRIVATE);

        // --- Editeur de preferences (pour en ajouter, en
        // supprimer, ...)
```

```

        editeur = sp.edit();

    } // / onCreate()

    // -----
    private void initInterface() {
        editTextNomPreference =
        findViewById(R.id.editTextNomPreference);
        editTextValeurPreference =
        findViewById(R.id.editTextValeurPreference);
        textViewPreferences = findViewById(R.id.textViewPreferences);

        buttonAjouter = (ImageButton) findViewById(R.id.buttonAjouter);
        buttonModifier = (ImageButton)
        findViewById(R.id.buttonModifier);
        buttonSupprimer = (ImageButton)
        findViewById(R.id.buttonSupprimer);
        buttonVoir = (ImageButton) findViewById(R.id.buttonVoir);
        buttonAide = (ImageButton) findViewById(R.id.buttonAide);

        buttonAjouter.setOnClickListener(this);
        buttonModifier.setOnClickListener(this);
        buttonSupprimer.setOnClickListener(this);
        buttonVoir.setOnClickListener(this);
        buttonAide.setOnClickListener(this);
    } // / initInterface

    @Override
    public void onClick(View v) {
        // -----
        if (v == buttonAjouter) {
            // --- Ajout d'une preference (cle, valeur)

            editeur.putString(editTextNomPreference.getText().toString(),
            editTextValeurPreference.getText().toString());
            // --- Validation
            editeur.commit();
        } // / if buttonAjouter

        // -----
        if (v == buttonModifier) {
            // --- Modification d'une preference (cle, valeur)

            editeur.putString(editTextNomPreference.getText().toString(),
            editTextValeurPreference.getText().toString());
            // --- Validation
            editeur.commit();
        } // / if buttonModifier

        // -----
        if (v == buttonSupprimer) {
            // Toutes
            if
            (editTextNomPreference.getText().toString().equals("")) {
                // --- Suppression de toutes les preferences
                editeur.clear();
            }
            // Une
            else {
                // --- Suppression d'une preference (cle)

```

```

        editeur.remove(editTextNomPreference.getText().toString());
    }

    // --- Validation
    editeur.commit();

} // / if buttonSupprimer

// -----
if (v == buttonVoir) {
    StringBuilder lsbContenu = new StringBuilder();

    // Toutes
    if
(editTextNomPreference.getText().toString().equals("")) {
        // --- Recuperation des preferences
        Map<String, ?> preferences = sp.getAll();

        // --- Nombre de preferences
        int liPreferences = preferences.size();

        if (liPreferences > 0) {
            Set<String> cles = preferences.keySet();
            for (String cle : cles) {
                lsbContenu.append("[");
                lsbContenu.append(cle);
                lsbContenu.append("] ");

                lsbContenu.append(preferences.get(cle).toString());
                lsbContenu.append("\n");
            }
        } else {
            lsbContenu.append("Aucune préférence pour
cette activité !");
        }
    }
    // Une
    else {
        if
(sp.contains(editTextNomPreference.getText().toString())) {
            // --- Recuperation d'une preference
            // --- sp.getString("nomPreference", "valeur
par default");

            String lsValeur =
sp.getString(editTextNomPreference.getText().toString(), "Pas de valeur");
            lsbContenu.append(lsValeur);
        } else {
            lsbContenu.append("Aucune préférence avec
cette clé !");
        }
    }
    // --- Affichage
    textViewPreferences.setText(lsbContenu.toString());
} // / if buttonVoir

// -----
if (v == buttonAide) {
} // / if buttonAide

} // / onClick

```



```
} // / classe
```

5.2.4.3 - L'activité MenuPrincipal.java

```
package fr.pb.preferences;

import java.util.Map;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.TextView;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;

/**
 *
 * @author pascal
 */
public class MenuPrincipal extends ListActivity {

    private Intent intention;
    private TextView textViewMessage;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menu_principal);

        textViewMessage = findViewById(R.id.textViewMessage);

        String[] tItemsMenu = getResources().getStringArray(
            R.array.items_menu_principal);
        ListView liste = this.getListView();
        ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, tItemsMenu);
        liste.setAdapter(aaListe);

    } // / onCreate

    @Override
    // -----
    public void onItemClick(ListView parent, View v, int position,
        long id) {
        String lsSelection =
            parent.getItemAtPosition(position).toString();

        if (lsSelection.equals("Préférences (Sans Framework)")) {
            intention = new Intent(this.getContext(),
                PreferencesCRUD.class);
            startActivityForResult(intention, 1);
        }
        if (lsSelection.equals("Préférences réseau et couleurs
            (Framework)")) {
        }
    }
}
```

```
        if (lsSelection.equals("Divers")) {
        }

    } // / onItemClick()

    /*
     * Au retour ...
     */
    public void onActivityResult(int requestCode, int resultCode, Intent
data) {

        SharedPreferences sp;

        switch (requestCode) {
        case 1: // Preference personnalisée

            // Recuperation de l'IP ...
            sp = getSharedPreferences("crud", Context.MODE_PRIVATE);
            String lsIPDuJour = sp.getString("IPDuJour", "Aucune
IP");

            textViewMessage.setText(lsIPDuJour);
            return;

        } // / switch (requestCode)

    } // / onActivityResult

} // / classe
```

5.2.5 - Fichiers générés

PreferencesCRUD.xml (MODE_PRIVATE)

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="IPDuJour">192.168.0.1</string>
</map>
```

crud.xml (partageable)

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="IPDuJour">192.168.0.1</string>
</map>
```

5.2.6 - Exercice : terminer le CRUD

5.2.6.1 - First

Ajoutez la suppression de toutes les préférences, la visualisation de la valeur d'une préférence et la modification d'une préférence.

5.2.6.2 - Second

Utilisez le système de préférences partagées pour stocker l'IP du jour !

5.3 - LE FRAMEWORK ANDROID POUR GÉRER LES PRÉFÉRENCES

5.3.1 - Objectif

Saisir des préférences via le framework fourni par le SDK Android.

L'avantage du framework c'est qu'il propose des interfaces graphiques standards et stocke automatiquement les valeurs.

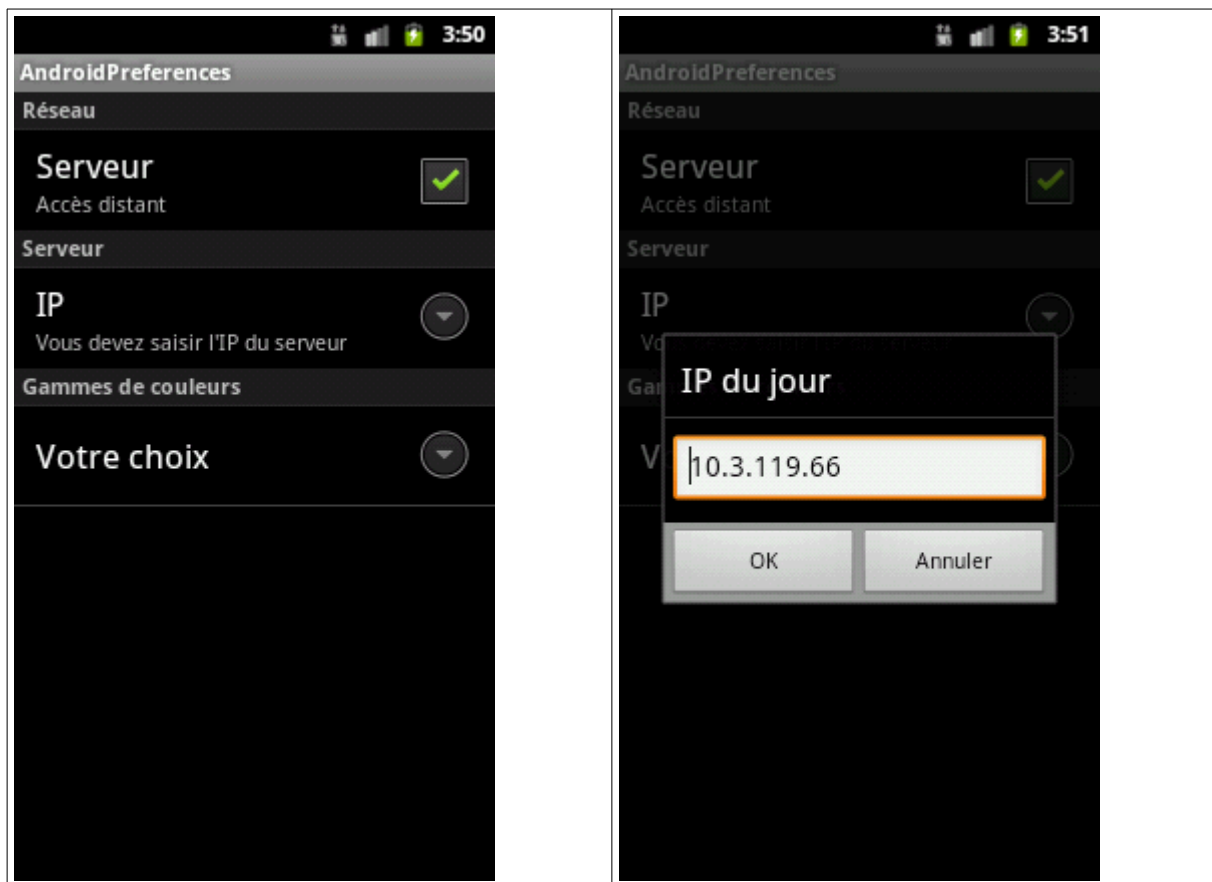
Il est possible de choisir une préférence sous forme de case à cocher et/ou sous forme de boîte de dialogue de saisie et/ou sous forme de liste.

Dans cet exemple l'affichage de la boîte de dialogue est dépendant de l'état de la case à cocher. Si la case à cocher est cochée alors le bouton IP est accessible.

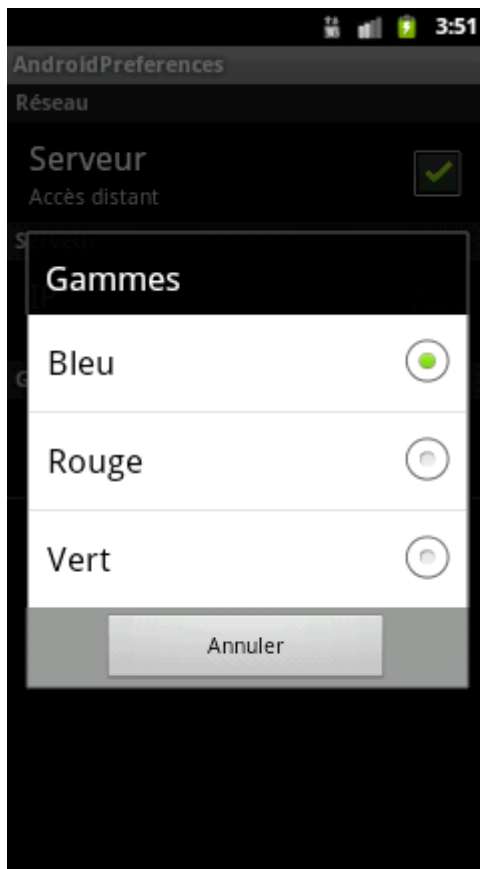
Le résultat attendu ... la zone de message change de couleur et le texte affiché aussi ...



La saisie d'une préférence sous forme de boîte de dialogue.



Le choix d'une préférence sous forme de liste.



Le résultat des choix est enregistré dans le fichier suivant :

/data/data/fr.pb.preferences/shared_prefs/fr.pb.preferences_preferences.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="checkBoxReseau" value="true" />
<string name="gamme_couleurs">blue</string>
<string name="IP">10.3.119.66</string>
</map>
```


5.3.2 - Démarche

Création de la vue statique pour la classe activité

Un layout suivant les canons du framework.

Création de la classe Activité

La classe hérite de **PreferenceActivity** et la méthode onCreate fait appel à la méthode **addPreferencesFromResource()**.

Autres

Les fichiers strings.xml, arrays.xml, colors.xml, ...

5.3.3 - Syntaxes

Classe héritant de PreferenceActivity

```
public class NomDeClasse extends PreferenceActivity
```

Affichage de la vue statique

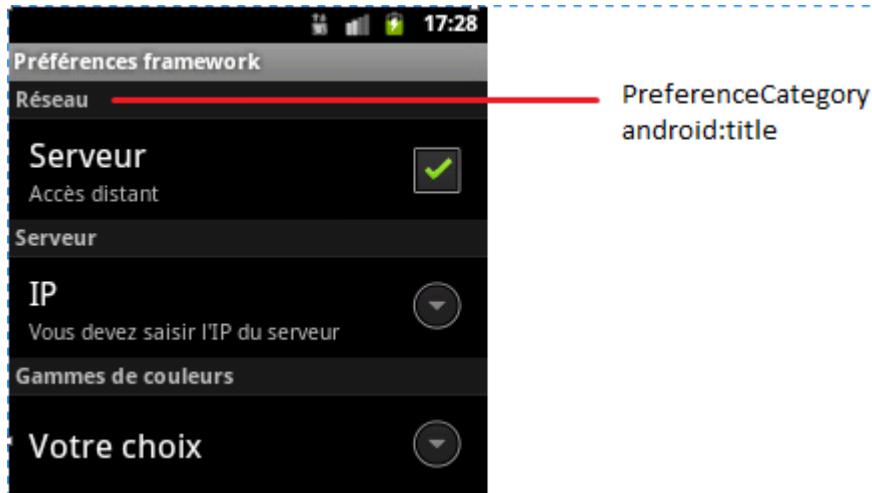
```
addPreferencesFromResource(R.layout.fichier_xml);
```

Note : obsolète depuis API 11 (3.0). cf inflate.

Le layout

L'élément parent du layout est un `<PreferenceScreen>`.

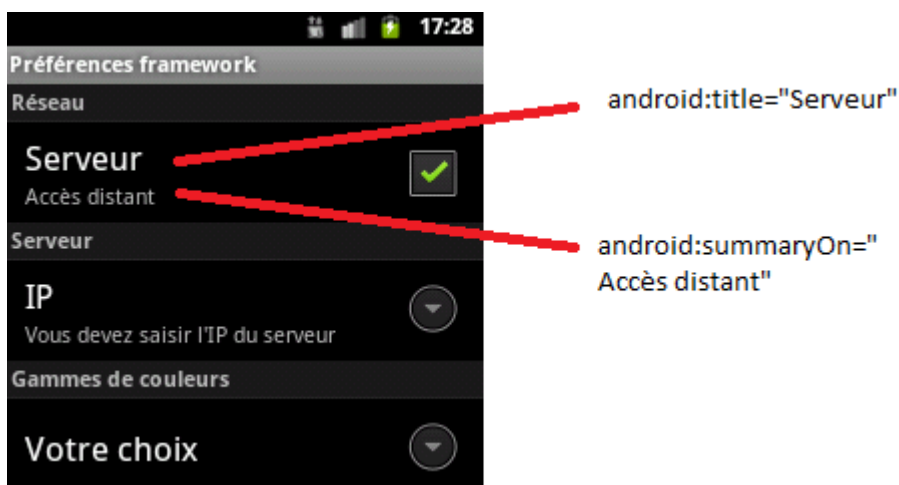
Chaque élément est un `<PreferenceCategory>` qui possède l'attribut `android:title` pour renseigner le titre du bloc.



Les éléments enfants peuvent être :

Widget	Balise	Attributs spécifiques
Case à cocher	<code><CheckBoxPreference></code>	<code>android:summaryOff=""</code> <code>android:summaryOn=""</code>
Zone de saisie	<code><EditTextPreference></code>	<code>android:dialogTitle=""</code> <code>android:negativeButtonText=""</code> <code>android:positiveButtonText=""</code> <code>android:dependency=""</code>
Liste	<code><ListPreference></code>	<code>android:dialogTitle=""</code> <code>android:entries="@array/"</code> <code>android:entryValues="@array/"</code>

Tous les éléments ont les attributs suivants : `android:title` et `android:key`.



5.3.4 - Codes

5.3.4.1 - preferences_framework.xml stocké dans /res/layout/

L'écran statique pour le choix des préférences via le framework.

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

    <PreferenceCategory android:title="Réseau" >
        <CheckBoxPreference
            android:defaultValue="true"
            android:key="checkBoxReseau"
            android:summaryOff="Pas d'accès distant"
            android:summaryOn="Accès distant"
            android:title="Serveur" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Serveur" >
        <EditTextPreference
            android:dependency="checkBoxReseau"
            android:dialogTitle="IP du jour"
            android:key="IP"
            android:negativeButtonText="Annuler"
            android:positiveButtonText="OK"
            android:summary="Vous devez saisir l'IP du serveur"
            android:title="IP" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Gammes de couleurs" >
        <ListPreference
            android:dialogTitle="Gammes"
            android:entries="@array/liste_couleurs_fr"
            android:entryValues="@array/liste_couleurs"
            android:key="gamme_couleurs"
            android:title="Votre choix" />
    </PreferenceCategory>
</PreferenceScreen>
```

5.3.4.2 - PreferencesFramework.java

L'activité pour le choix des préférences via le framework.

```
public class PreferencesFramework extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.layout.preferences_frame);
    } /// onCreate
} /// classe
```

5.3.4.3 - MenuPrincipal.java

```
package fr.pb.preferences;

import java.util.Map;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.TextView;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;

/**
 *
 * @author pascal
 *
 */
public class MenuPrincipal extends ListActivity {

    private Intent intention;
    private TextView textViewMessage;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menu_principal);
        textViewMessage = findViewById(R.id.textViewMessage);

        String[] tItemsMenu = getResources().getStringArray(
            R.array.items_menu_principal);
        ListView liste = this.getListView();
        ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
            R.layout.preferences_ligne, R.id.textViewLigne,
tItemsMenu);
        liste.setAdapter(aaListe);

    } // / onCreate

    @Override
    // -----
    public void onItemClick(ListView parent, View v, int position,
long id) {
        String lsSelection =
parent.getAdapter().getItem(position).toString();

        if (lsSelection.equals("Préférences (Sans Framework)")) {
            intention = new Intent(this.getContext(),
PreferencesCRUD.class);
            startActivityForResult(intention, 1);
        }
        if (lsSelection.equals("Préférences réseau et couleurs
(Framework)")) {
```

```
        intention = new Intent(this.getBaseContext(),
                                PreferencesFramework.class);
        startActivityForResult(intention, 2);
    }

    if (lsSelection.equals("Divers")) {
        intention = new Intent(this.getBaseContext(),
Divers.class);
        startActivityForResult(intention, 3);
    }

    } // / onItemClick()

    /*
     * Au retour ...
     */
    public void onActivityResult(int requestCode, int resultCode, Intent
data) {

        SharedPreferences sp;

        switch (requestCode) {
            case 1: // Preference personnalisée

                // Recuperation de l'IP ...
                sp = getSharedPreferences("crud", Context.MODE_PRIVATE);
                String lsIPDuJour = sp.getString("IPDuJour", "Aucune
IP");

                textViewMessage.setText(lsIPDuJour);
                return;

            case 2: // Preferences reseau et couleurs
                // Recuperation de l'etat du reseau
                // Recuperation de l'IP eventuellement
                // Recuperation de la couleur et action en consequence

                sp = PreferenceManager.getDefaultSharedPreferences(this);

                StringBuilder lsbContenu = new StringBuilder();

                // --- Recuperation des preferences
                Map<String, ?> preferences = sp.getAll();

                // --- Nombre de preferences
                int liPreferences = preferences.size();

                if (liPreferences > 0) {

                    // getString("cle", "valeur par default")
                    String lsIP = sp.getString("IP", "Aucune");
                    textViewMessage.setText(lsIP);

                    String gammeCouleurs =
sp.getString("gamme_couleurs", "noir");

                    if (gammeCouleurs.equals("red")) {

                        textViewMessage.setBackgroundColor(getResources().getColor(
R.color.red));

                        textViewMessage.setTextColor(getResources().getColor(
```

```
                R.color.pink));
            }
            if (gammeCouleurs.equals("green")) {

textViewMessage.setBackgroundColor(getResources().getColor(
                R.color.green));

textViewMessage.setTextColor(getResources().getColor(
                R.color.light_green));
            }
            if (gammeCouleurs.equals("blue")) {

textViewMessage.setBackgroundColor(getResources().getColor(
                R.color.blue));

textViewMessage.setTextColor(getResources().getColor(
                R.color.light_blue));
            }

        } else {
            lsbContenu.append("Aucune préférence pour cette
activité !");
        }
        return;

        case 3: // Divers

            // Recuperation du jour ...
            // KO par definition
            // SharedPreferences sp =
getPreferences (Context.MODE_PRIVATE);
            // OK
            sp = PreferenceManager.getDefaultSharedPreferences(this);
            // OK
            // sp = getSharedPreferences("jour", MODE_PRIVATE);

            String lsJour = sp.getString("preferenceJour", "Aucun
jour");

            textViewMessage.setText(lsJour);

            return;
        } // / switch (requestCode)

    } // / onActivityResult

} // / classe
```

5.3.4.4 - La modification du fichier Manifest

L'ajout des activités secondaires.

```
<activity
    android:name="fr.pb.preferences.MenuPrincipal"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name="fr.pb.preferences.PreferencesCRUD"
    android:label="Préférences (CRUD sans framework)" >
</activity>

<activity
    android:name="fr.pb.preferences.PreferencesFramework"
    android:label="Préférences framework" >
</activity>

<activity
    android:name="fr.pb.preferences.Divers"
    android:label="Préférences (Divers)" >
</activity>
```


5.3.5 - Fichier généré

fr.pb.preferences_preferences.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="checkBoxReseau" value="true" />
<string name="IP">192.168.00.07</string>
<string name="gamme_couleurs">blue</string>
</map>
```

5.3.6 - Exercice

5.4 - LES PRÉFÉRENCES SYSTÈME

<http://developer.android.com/reference/android/provider/Settings.System.html>

5.4.1 - Objectifs

Récupérer une préférence système.

Créer une préférence de niveau système.

5.4.2 - Récupérer une préférence système

```
Settings.System.getString(contentResolver, "préférence");
```

```
ContentResolver cr = getApplicationContext().getContentResolver();

try {
    String lsPreference = "";
    //lsPreference = Settings.System.getString(cr,
    Settings.System.ANDROID_ID);
    //lsPreference = Settings.System.getString(cr,
    Settings.System.BLUETOOTH_ON);
    lsPreference = Settings.System.getString(cr,
    Settings.System.VOLUME_RING);
    textViewMessage.setText(lsPreference);
} catch (Exception e) {
    Log.e("Exception-PreferencesSysteme", e.getMessage());
}
```

5.4.3 - Créer une préférence système

```
Settings.System.putString(contentResolver, "préférence", "valeur");
```

A ajouter dans le fichier Manifest comme nœud du manifeste.

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

```
// --- Créer une preference systeme
ContentResolver cr = getApplicationContext().getContentResolver();
// ---- Creation
Settings.System.putString(cr, "MaPreferenceAMoi", "U2");
// --- Recuperation
lsPreference = Settings.System.getString(cr, "MaPreferenceAMoi");

textViewMessage.setText(lsPreference);
```

CHAPITRE 6 - FILEPROVIDER

Un file provider permet de partager des fichiers stockés ??? à d'autres applications qui créeront des FileResolver.

<https://developer.android.com/reference/android/support/v4/content/FileProvider.html>

Dans l'application propriétaire

/res/xml/file_paths.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="textes" path="txt"/>
  </paths>
</resources>
```

AndroidManifest.xml

```
<application
  android:allowBackup="true"
  android:icon="@mipmap/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >

  <provider
    android:name="android.support.v4.content.FileProvider"
    android:authorities="fr.pb.datadata"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
      android:name="android.support.FILE_PROVIDER_PATHS"
      android:resource="@xml/file_paths" />
  </provider>
```

Dans l'application consommatrice

CHAPITRE 7 - ANNEXES

7.1 - HISTOIRE DE CONTEXTE

getApplicationContext() Application context is associated with the Application and will always be the same throughout the life cycle.

getBaseContext() should not be used, just use Context instead of it which is associated with the activity and could possibly be destroyed when the activity is destroyed.

getApplication() is available to Activity and Services only. Although in current Android Activity and Service implementations, `getApplication()` and `getApplicationContext()` return the same object, there is no guarantee that this will always be the case (for example, in a specific vendor implementation). So if you want the Application class you registered in the Manifest, you should never call `getApplicationContext()` and cast it to your application, because it may not be the application instance (which you obviously experienced with the test framework).

getParent() returns object of the activity if the current view is a child..In other words returns the activity object hosting the child view when called within the child.

7.2 - LIRE UN FICHIER IMAGE STOCKÉ SUR LA SD ET L'AFFICHER DANS UNE IMAGEVIEW

```
String lsChemin =  
Environment.getExternalStorageDirectory().getAbsolutePath() +  
File.separator + "carla_gugino21.jpg";  
  
InputStream is = new FileInputStream(lsChemin);  
Bitmap bitmap = BitmapFactory.decodeStream(is);  
  
imageView1.setImageBitmap(bitmap);
```

Note :

A ajouter dans le fichier Manifest pour pouvoir lire la SD (depuis l'API 19) :

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

A ajouter dans le fichier Manifest pour pouvoir écrire sur la SD (depuis Android 4.0) :

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

7.3 - LIRE UN FICHIER ASCII STOCKÉ DANS LES ASSETS

assets = Actifs.

```
InputStream is = this.getAssets().open("asterix.txt");
```

7.4 - LIRE UN FICHIER STOCKÉ DANS UN SOUS-DOSSIER DE /DATA/DATA/.../FILES/

```
try {
    File fileDataFiles = this.getFilesDir();
    URI uriFichier = new URI("file://" + fileDataFiles.getAbsolutePath()
+ "/" + "sous dossier" + "/" + "1.txt");
    File file = new File(uriFichier);
    FileReader fr = new FileReader(file);
    BufferedReader br = new BufferedReader(fr);
    String lsLigne;
    while ((lsLigne = br.readLine()) != null) {
        lsb.append(lsLigne);
        lsb.append("\n");
    }
    br.close();
    fr.close();
} catch (Exception e) {
    lsb.append(e.getMessage());
}
```

7.5 - TRANSFÉRER UN FICHIER DE /RES/ VERS /DATA/DATA/FILES

7.5.1 - Transférer un fichier de /res/raw vers /data/data/files

```
String lsFichier = getFilesDir().getAbsolutePath() + "/" + PAYS_CSV;

File fichier = new File(lsFichier);
// Si le fichier existe
if (fichier.exists()) {
    Toast.makeText(this, lsFichier+ " existe", Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(this, lsFichier + " n'existe pas",
Toast.LENGTH_SHORT).show();
    /*
    Transfert fichier pays.csv vers /data/data/files
    */
    try {
        InputStream isIN = getResources().openRawResource(R.raw.pays);
        int tailleIN = isIN.available();
        byte[] tOctets = new byte[tailleIN];
        isIN.read(tOctets, 0, tailleIN);

        OutputStream osOUT = openFileOutput(PAYS_CSV,
Context.MODE_PRIVATE);
        osOUT.write(tOctets);
    } catch (Exception e) {
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
    }
}
```

7.5.2 - Transférer un fichier de /res/xml vers /data/data/files

Le plus plus simple de stocker le fichier .xml dans /res/raw/

7.6 - LES FICHIERS DE DONNÉES UTILISÉS DANS CE MODULE

7.6.1 - repertoire.txt de /res/raw

repertoire.txt

```
Renseignements;12
Réclamations;13
Samu social;115
Pompiers;17
Police;18
```

7.6.2 - repertoire.csv du File System

repertoire.csv

```
Nom;Téléphone
Tintin;+3311111111111
Milou;+3322222222222
Dupont;+3333333333333
Dupond;+3344444444444
Casta;+3355555555555
Tournesol;+3366666666666
Rasta;+3377777777777
Obelix;+3388888888888
Asterix;+3399999999999
Jean;+3300000000000
Pierre;+3300000000001
Paul;+3300000000002
```

7.6.3 - villes_data.xml : document de type DATA

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- villes_data.xml -->

<!-- <villes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="villesDonnes.xsd"> -->

<villes>
  <ville code_pays="250" nom_ville="Paris" />
  <ville code_pays="380" nom_ville="Rome" />
  <ville code_pays="724" nom_ville="Madrid" />
  <ville code_pays="826" nom_ville="Londres" />
  <ville code_pays="840" nom_ville="Washington" />
</villes>
```

7.6.4 - villes_doc.xml : document de type DOCUMENT

```
<?xml version="1.0" encoding="utf-8"?>
<villes>
  <ville>
    <cp>
      75000
    </cp>
    <nom_ville>
      Paris
    </nom_ville>
  </ville>
  <ville>
    <cp>
      69000
    </cp>
    <nom_ville>
      Lyon
    </nom_ville>
  </ville>
  <ville>
    <cp>
      13000
    </cp>
    <nom_ville>
      Marseille
    </nom_ville>
  </ville>
</villes>
```

7.6.5 - communes_mixte.xml : document de type mixte

```
<?xml version="1.0" encoding="utf-8"?>
<communes>
  <cp code="75000">
    <commune>
      Paris
    </commune>
  </cp>
  <cp code="24200">
    <commune>
      Sarlat
    </commune>
    <commune>
      Vitrac
    </commune>
    <commune>
      Saint-Natalene
    </commune>
  </cp>
  <cp code="13000">
    <commune>
      Marseille
    </commune>
  </cp>
</communes>
```


7.6.6 - rss.xml

rss.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
  <channel>

    <title>Photos</title>
    <description>Les meilleurs photos d'acteurs du jour</description>
    <link>http://localhost/rssFournisseur</link>
    <image>
      <url>actrices.jpg</url>
      <link>http://localhost/rssFournisseur/actrices.jpg</link>
    </image>

    <item>
      <title>Angelina Jolie</title>
      <description>Une actrice au top</description>
      <pubDate>21 May 2009 18:00:00 +0000</pubDate>
      <link>http://localhost/rssFournisseur/angelina_jolie.html</link>
      <enclosure
url="http://localhost/rssFournisseur/angelina_jolie.jpg" type="image/jpeg"
/>
    </item>

    <item>
      <title>Penelope Cruz</title>
      <description>La star de Pedro Almodovar</description>
      <pubDate>21 May 2009 21:00:00 +0000</pubDate>
      <link>http://localhost/rssFournisseur/penelope_cruz.html</link>
      <enclosure url="http://localhost/rssFournisseur/penelope_cruz.jpg"
type="image/jpeg" />
    </item>

    <item>
      <title>Julia Roberts</title>
      <description>La star</description>
      <pubDate>21 May 2009 22:00:00 +0000</pubDate>
      <link>http://localhost/rssFournisseur/julia_roberts.html</link>
      <enclosure url="http://localhost/rssFournisseur/julia_roberts.jpg"
type="image/jpeg" />
    </item>

  </channel>
</rss>
```

7.6.7 - villes.json

```
[
  {
    "cp": "75000",
    "nom": "Paris"
  },
  {
    "cp": "69000",
    "nom": "Lyon"
  },
  {
    "cp": "13000",
    "nom": "Marseille"
  }
]
```

7.6.8 - pays.json

```
[
  {
    "iso2": "FR",
    "nom": "France"
  },
  {
    "iso2": "IT",
    "nom": "Italie"
  },
  {
    "iso2": "SP",
    "nom": "Espagne"
  }
]
```

7.6.9 - Autres fichiers .txt

7.6.9.1 - tintin.txt

```
id;nom;prenom;description;image
1;Tintin;Augustin;Jeune reporter;tintin.jpg
2;Milou;;Fox Terrier de Tintin;milou.jpg
3;Haddock;Archibald;Capitaine de marine marchande et meilleur ami de
4;Tintin;haddock.jpg
4;Castafiore;Bianca;Cantatrice italienne de renommée internationale,
surnommée Rossignol milanais;castafiore.jpg
```

7.6.9.2 - daltons.txt

```
id;nom
1;Robert (Bob)
2;Grattan (Grat)
4;William (Bill)
5;Emmett Dalton
```

7.6.9.3 - pieds_nickeles.txt

```
id;nom;allure;deguisement;image
1;Ribouldingue;gros barbu sympathique;casquette verte, assortie au
chandail vert;
2;Filochard;petit borgne, au caractère sanguin; béret, un col roulé rouge
et une veste noire;
3;Croquignol;grand et assez fin;monocle, un petit chapeau, et une veste
jaune;
```

7.6.9.4 - asterix.txt

```
id;nom
1;Astérix
2;Obélix
3;Idéfix
```

7.7 - QUELQUES CLASSES UTILITAIRES

7.7.1 — La classe **Tableau**

La classe contient, pour l'instant, une méthode statique - `xml2Tableau(XPP, Element, Attribut)` – qui renvoie un tableau de String contenant les valeurs d'un attribut d'un élément avec l'API `org.xmlpull.v1`.

Utile pour remplir un Spinner ou une ListView à partir d'un document XML de type Data.

Tableau
(+) <code>xml2Tableau(xpp, élément, attribut) : String[]</code>

```
package fr.pb.utilitaires;

import java.util.ArrayList;
import java.util.List;

import org.xmlpull.v1.XmlPullParser;

/**
 * Classe "utilitaire" utilisée par la classe ListeXML.java
 *
 * Méthodes
 *
 * xml2Tableau()
 *
 * @author pascal
 */
public class Tableau {

    /**
     * L'attribut de l'élément dans un tableau
     *
     * @param xpp
     * @param asNomElement
     * @param asNomAttribut
     * @return
     */
    public static String[] xml2Tableau(XmlPullParser xpp, String
asNomElement,
        String asNomAttribut) {

        List<String> lal = new ArrayList<String>();
        String[] tableau = null;
        String lsValeurAttribut;

        try {
```

```

// --- Tant que le document n'a pas ete analyse jusqu'a
fin
while (xpp.getEventType() != XmlPullParser.END_DOCUMENT)
{
    // --- Si c'est une balise ouvrante
    if (xpp.getEventType() == XmlPullParser.START_TAG)
    {
        // --- Si c'est la bonne balise
        if (xpp.getName().equals(asNomElement)) {
            // --- Recuperation d'un attribut de la
            // --- getAttributeValue("namespace",
            // --- lsValeurAttribut =
            // --- xpp.getAttributeValue(null,
            // --- asNomAttribut);
            if (lsValeurAttribut != null) {
                lal.add(lsValeurAttribut);
            }
        } // / IF balise asNomElement trouvee

        // --- IF START_TAG
        // --- On passe a la balise suivante
        xpp.next();

        // --- WHILE parse

        // --- Transfert de l'ArrayList dans le String[]
        tableau = lal.toArray(new String[lal.size()]);

        // --- try

        catch (Exception e) {

        } // / catch

        return tableau;
    } // / xml2tableau()
} // / class
```

7.7.2 - La classe Fichier

Cf aussi 3.10.5

Fichier
(+) <i>listerDossier</i> (chemin) : String !!! (+) <i>creerDossier</i> (chemin, nom du dossier) : String (+) <i>supprimerFichier</i> (dossier, fichier) : String (+) <i>getContenuFichierASCII</i> (chemin, fichier) : String (+) <i>creerFichierASCII</i> (chemin, fichier, contenu) : String (+) <i>getDateOfFileData</i> (Context contexte, String psFile) : long (+) <i>fileExistInData</i> (Context contexte, String psFile) : boolean (+) <i>writeFileToData</i> (Context contexte, String psFile, String psContent) : boolean (+) <i>readFileFromResources</i> (Context contexte, int resource) : String (+) <i>readFileFromData</i> (Context contexte, String psFile) : String

Note : il faut ajouter csv2Map, ..., csv2List, ...

```
package fr.pb.utilitaires;

import java.io.*;
import java.net.URI;
import java.net.URISyntaxException;
import android.content.Context;
import android.util.Log;
import java.util.Date;

/**
 *
 * @author pascal
 *
 */
public class Fichier {

    /**
     *
     * @param asCheminDossier
     * @return
     */
    public static String listerDossier(String asCheminDossier) {
        StringBuilder lsbContenu = new StringBuilder();

        // via une URI
        URI uri = null;
        File cheminDossier;
        File[] entreesDossier;
        try {
            uri = new URI("file://" + asCheminDossier);
            cheminDossier = new File(uri);
            entreesDossier = cheminDossier.listFiles();
            for (int i = 0; i < entreesDossier.length; i++) {
```

```

        lsbContenu.append(entreesDossier[i].getAbsolutePath());
        lsbContenu.append("\n");
    }
    } catch (URISyntaxException e) {
        lsbContenu.append("-1");
    }

    catch (Exception e) {
        lsbContenu.append("-1");
    }

    return lsbContenu.toString();
} // / listerDossier

/**
 *
 * @param asCheminDossier
 * @param asNomDuNouveauDossier
 * @return
 */
public static String creerDossier(String asCheminDossier,
    String asNomDuNouveauDossier) {
    StringBuilder lsbMessage = new StringBuilder("");

    try {
        File chemin = new File(asCheminDossier);
        File nouveauDossier = new File(chemin,
asNomDuNouveauDossier);
        boolean lbOK = nouveauDossier.mkdir();
        if (lbOK) {
            lsbMessage.append("Dossier créé !");
        } else {
            lsbMessage.append("Dossier non créé, il existe déjà
!");
        }
    } catch (Exception e) {
        lsbMessage.append(e.getMessage());
    }

    return lsbMessage.toString();
} // / creerDossier

/**
 *
 * Supprime un FICHER ou un DOSSIER
 *
 * @param asCheminDossier
 * @param asNomFichier
 * @return
 */
public static String supprimerFichier(String asCheminDossier,
    String asNomFichier) {
    StringBuilder lsbMessage = new StringBuilder("");

    // Suppression fichier ou dossier
    try {
        File chemin = new File(asCheminDossier + asNomFichier);
        boolean lbOK = false;
        if (chemin.isDirectory()) {
            lsbMessage.append("Le dossier ");

```



```

        }
        if (chemin.isFile()) {
            lsbMessage.append("Le fichier ");
        }
        lbOK = chemin.delete();
        if (lbOK) {
            lsbMessage.append(" a été supprimé !");
        } else {
            lsbMessage.append(" n'existe pas ou n'a pu être
supprimé !");
        }
    } catch (Exception e) {
        lsbMessage.append(e.getMessage());
    }

    return lsbMessage.toString();
} // / supprimerFichier

/**
 *
 * @param asChemin
 * @param asNomFichier
 * @return
 */
public static String getContenuFichierASCII(String asChemin,
String asNomFichier) {
    StringBuilder lsbContenu = new StringBuilder();

    try {
        File f = new File(asChemin + asNomFichier);
        if (f.exists()) {
            // --- Pour lire un fichier en UTF8 :
            FileInputStream fis = new FileInputStream(f);
            InputStreamReader isr = new InputStreamReader(fis,
"utf8");

            BufferedReader br = new BufferedReader(isr);
            String lsLigne = "";
            while ((lsLigne = br.readLine()) != null) {
                lsbContenu.append(lsLigne);
                lsbContenu.append("\n");
            }
            br.close();
            isr.close();
            fis.close();
        } else {
            lsbContenu.append(f.getAbsolutePath());
            lsbContenu.append(" n'existe pas !!!");
        }

    } catch (Exception e) {
        lsbContenu.append(e.getMessage());
    }

    return lsbContenu.toString();
} // / getContenuFichierASCII

/**
 *
 * @param asChemin
 * @param asNomFichier
 * @param asContenu

```

```
* @return
*/
public static String creerFichierASCII(String asChemin,
                                     String asNomFichier, String asContenu) {

    FileOutputStream fos;
    OutputStreamWriter osw;
    StringBuilder lsbMessage = new StringBuilder("");

    try {
        fos = new FileOutputStream(asChemin + asNomFichier);
        osw = new OutputStreamWriter(fos);
        osw.write(asContenu);
        osw.close();
        fos.close();
        lsbMessage.append("Le fichier suivant a été créé : ");
        lsbMessage.append(asNomFichier);
    } catch (Exception e) {
        lsbMessage.append(e.getMessage());
    }

    return lsbMessage.toString();
} // / creerFichierASCII

/**
 * Created by pascal on 19/07/16.
 */

/**
 * @param contexte
 * @param resource
 * @return
 */
public static String readFileFromResources(Context contexte, int
resource) {

    StringBuilder lsbContenu = new StringBuilder("");
    String lsLigne = "";
    InputStream is;
    InputStreamReader isr;
    BufferedReader br;

    try {
        // --- Avec un buffer
        is = contexte.getResources().openRawResource(resource);
        isr = new InputStreamReader(is);
        br = new BufferedReader(isr);
        while ((lsLigne = br.readLine()) != null) {
            lsbContenu.append(lsLigne);
            lsbContenu.append("\n");
        }
        br.close();
        isr.close();
        is.close();
    } catch (Exception e) {
        lsbContenu.append("-1");
    }

    return lsbContenu.toString();
} /// readFileFromResources
```

```
/**
 *
 * @param contexte
 * @param psFile
 * @return
 */
public static String readFileFromData(Context contexte, String psFile)
{
    StringBuilder lsbContenu = new StringBuilder("");
    String lsLigne = "";
    FileInputStream fis;
    InputStreamReader isr;
    BufferedReader br;

    try {
        // --- Avec un buffer
        fis = contexte.openFileInput(psFile);
        isr = new InputStreamReader(fis);
        br = new BufferedReader(isr);
        while ((lsLigne = br.readLine()) != null) {
            lsbContenu.append(lsLigne);
            lsbContenu.append("\n");
        }
        br.close();
        isr.close();
        fis.close();
    } catch (Exception e) {
        lsbContenu.append("-1");
    }

    return lsbContenu.toString();
} /// readFileFromData

/**
 *
 * @param contexte
 * @param psFile
 * @param psContent
 * @return
 */
public static boolean writeFileToData(Context contexte, String psFile,
String psContent) {
    boolean lbOK = true;

    FileOutputStream fos;
    OutputStreamWriter osw;
    BufferedWriter bw;

    try {
        fos = contexte.openFileOutput(psFile, Context.MODE_PRIVATE);
        osw = new OutputStreamWriter(fos);
        bw = new BufferedWriter(osw);
        bw.write(psContent);

        bw.close();
        osw.close();
        fos.close();

    } catch (Exception e) {
        lbOK = false;
    }
}
```

```
        }
        return lbOK;
    } /// writeFileToData

    /**
     * @param psFile
     * @return
     */
    public static boolean fileExistInData(Context contexte, String psFile)
    {
        boolean lbExist = false;

        try {
            File dir = contexte.getFilesDir();

            String lsChemin = dir.getAbsolutePath();
            File f = new File(lsChemin + "/" + psFile);
            if (f.exists()) {
                lbExist = true;
            }
        } catch (Exception e) {
            Log.e("Erreur", e.getMessage());
        }

        return lbExist;
    } /// fileExistInData

    /**
     * @param contexte
     * @param psFile
     * @return
     */
    public static long getDateOfFileData(Context contexte, String psFile)
    {
        long date = 0;

        try {
            File dir = contexte.getFilesDir();
            String lsChemin = dir.getAbsolutePath();
            File f = new File(lsChemin + "/" + psFile);
            date = f.lastModified();
        } catch (Exception e) {
            Log.e("Erreur", e.getMessage());
        }

        return date;
    } /// getDateOfFileData
} // / class Fichier
```

7.7.3 - La classe XMLXPP

XMLXPP
(+) <i>getXPPFromResource</i> (id) : xpp (+) <i>getXPPFromData</i> (chemin) : xpp (+) <i>getXPPFromSD</i> (chemin) : xpp (+) <i>xmlData2Map</i> (xpp, fichier, balise, attributs) : Map (+) <i>xml2DocumentMap</i> (xpp, fichier, balise, enfants) : Map (+) <i>map2XMLData</i> () : boolean (+) <i>map2XMLDocument</i> () : boolean

Notes :

Pour récupérer un XPP from resource c'est une ligne de code !

```
XmlPullParser xpp = getResources().getXml(R.xml.id_ressource);
```

7.7.4 - La classe XML

Au-delà de XPP !

Avec un InputStream ?

7.8 - TP : L'APPLICATION LEXIQUE

Objectif : créer son propre dictionnaire ou plutôt lexique.

Exemples de lexiques :

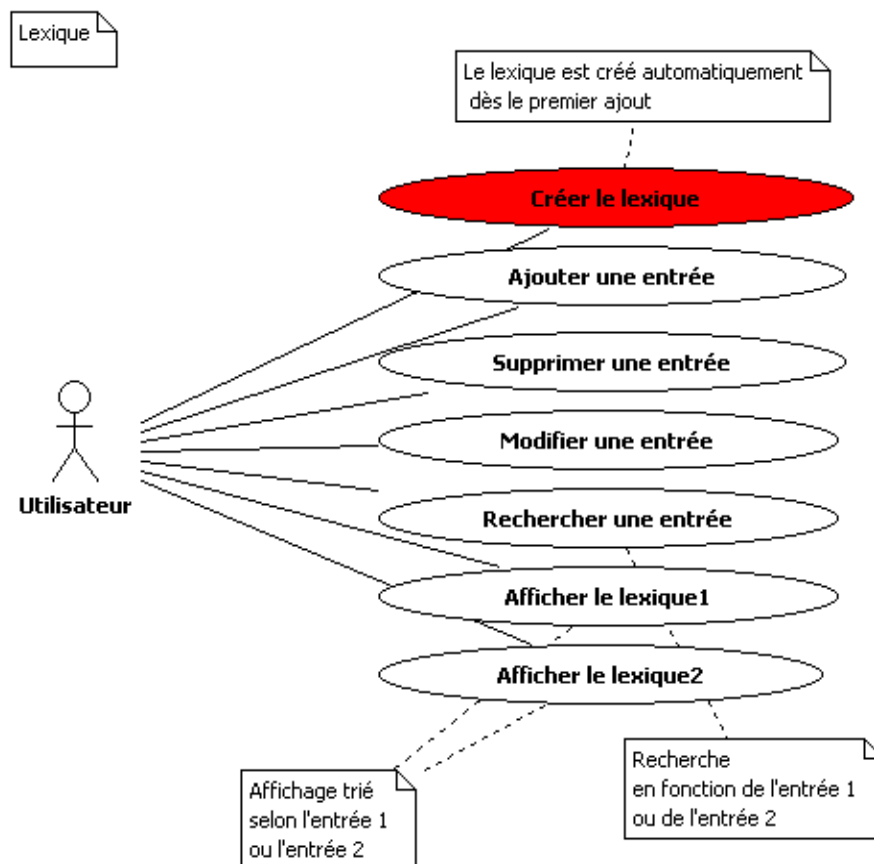
lexiqueFRSP.txt

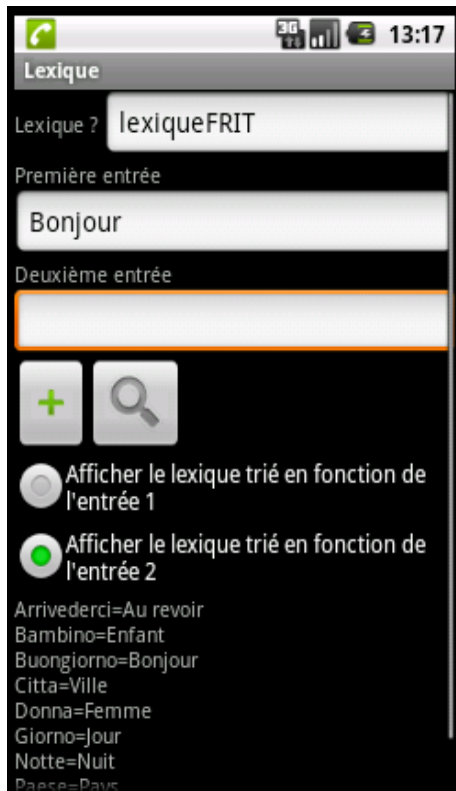
```
Bonjour;Buenas dias
Au revoir;Adios
```

lexiqueFRIT.txt

```
Bonjour;Buongiorno
Au revoir;Arrivederci
```

Fonctionnalités



Interface simplifiée (Application AndroidLexique)**Corrigé**

Cf l'application AndroidLexique ou AndroidLexiqueV2.

7.9 - INPUTSTREAM/FILEINPUTSTREAM (COMPARATIF)

En fonction du lieu de stockage la méthode à utiliser est différente.

Lieu de stockage	Méthode d'activité
/assets/	InputStream is = this.getAssets().open("asterix.txt");
/res/raw/	InputStream is = getBaseContext().getResources().openRawResource(R.drawable.img);
/data/data/.../files/	InputStream is = getBaseContext(). openFileInput (FICHIER_TXT); Renvoie aussi un FileInputStream.
/data/data/.../files/sousDossier/	
SD	String lsChemin = Environment.getExternalStorageDirectory() .getAbsolutePath() + File.separator + "xxx.jpg"; InputStream is = new FileInputStream(lsChemin);
Ailleurs	URI uri = new URI("file:///dossier/ressource"); URL url = uri.toURL(); InputStream is = url.openStream();
http	String lsURL = "http://172.60.12.180/xxx.xml"; URL url = new URL(lsURL); InputStream is = url.openStream();

1) is -> isr -> br -> ligne/enr

```
InputStream is = new FileInputStream("/home/pascal/tempo/statuts.txt");
InputStreamReader isr = new InputStreamReader(is);
```

```
BufferedReader br = new BufferedReader(isr);
String lsLigne = br.readLine();
```

2) fr -> br -> ligne/enr

```
FileReader fr = new FileReader("/home/pascal/tempo/statuts.txt");
```

```
BufferedReader br = new BufferedReader(fr);
String lsLigne = br.readLine();
```

7.10 - FILEOUTPUTSTREAM (COMPARATIF)

Lieu de stockage	Méthode d'activité
/data/data/.../files/	
/data/data/.../files/sousDossier/	
SD	
Ailleurs	URI("file:///dossier/ressource);
HTTP	

7.11 - ADB SHELL

Allez dans le dossier de adb.exe :

cd "\\Program Files\\Android\\android-sdk\\platform-tools"

ou dans

cd "\\Program Files (x86)\\Android\\android-sdk\\platform-tools"

Lancez adb.exe avec l'option shell :

shell>**adb shell**

et les commandes cd, ls, rmdir, ...

7.12 - L'APPLICATION TERMINAL

Gratuite sur Google Play ou même pré-installée.

Détruire un dossier

Ouvrir un terminal.

Aller dans le dossier en question (`cd /dossier/dossier`)

Taper "`rm -r dossier`"

Si accès refusé, taper "`su`" puis les commandes précédentes (mais il faut avoir "rooter" le terminal).

Ou avec `adb shell` (cf plus loin).

Note : sur les versions récentes des terminaux une application Terminal est fournie.

7.13 - SYNTHÈSE

7.13.1.1 - Le monde RES

A vous !!!

Quoi ?	Où ?	RO ?	RW ?	Partageable	API
Text/CSV	/res/raw/	Yes	No	No	Java.io
XML					
JSON					
SQL					

7.13.1.2 - Le monde /data/data/n.d.p/files/

A vous !!!

Quoi ?	Où ?	RO ?	RW ?	Privé	API
Text/CSV	/data/data/n.d.p/files				
XML					
JSON					
SQL					

Et sur la SD ?

7.13.1.3 - Fichier dans /data/data/n.d.p/files et lecture

Basique : avec openFileInput

URI → URL → FileInputStream → InputStreamReader → BufferedReader

```
String lsChemin = contexte.getFilesDir().getAbsolutePath() + "/" + psFichier;  
FileInputStream fis = contexte.openFileInput(psFichier);  
InputStreamReader isr = new InputStreamReader(fis);  
BufferedReader br = new BufferedReader(isr);
```

Avec URI et URL :

URI → URL → InputStream → InputStreamReader → BufferedReader

```
URI uri = new URI("file:/// " + contexte.getFilesDir().getAbsolutePath() + "/" + psFichier);  
URL url = uri.toURL();  
InputStream is = url.openStream();  
InputStreamReader isr = new InputStreamReader(is);  
BufferedReader br = new BufferedReader(isr);
```